



浙江大学
ZHEJIANG UNIVERSITY

科研经验分享

报告人：彭思达

GitHub Repo: https://github.com/pengsida/learning_research

Agenda

1. 一个research project包含哪些流程，怎么做一篇论文
 - a. 如何想Idea
 - b. 如何做实验
 - c. 如何写论文
2. 总结博士生应该具有的能力

一个research project包含哪些流程

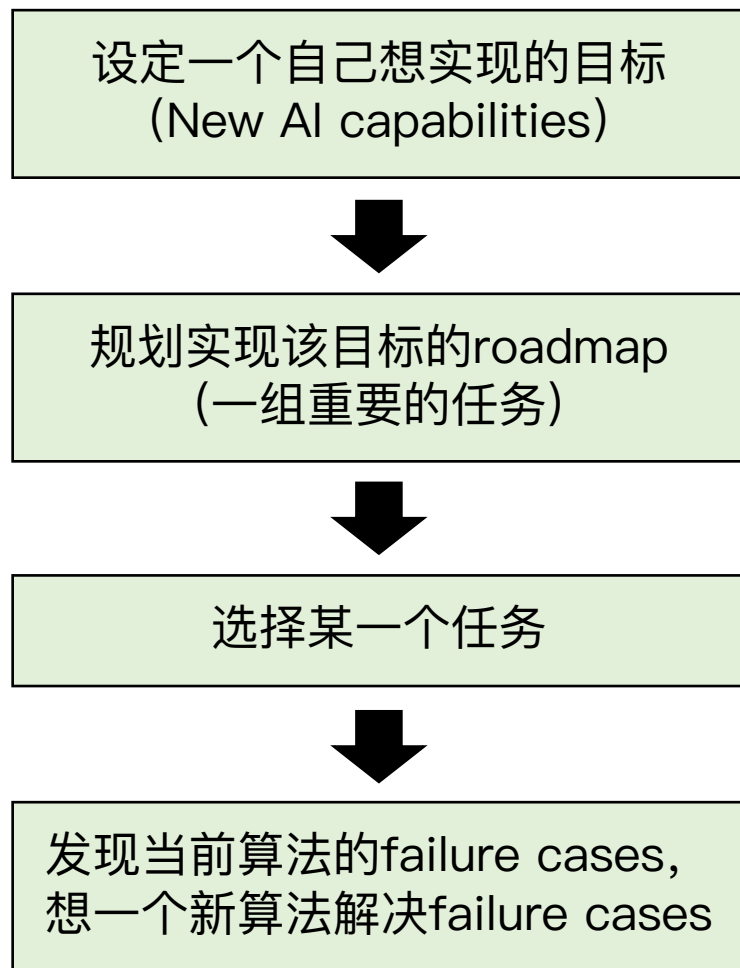
1. 如何想Idea	1.1 规划重要的任务
	1.2 发现任务中需要解决的technical challenge、failure cases
	1.3 构思解决failure cases的方法
2. 如何做实验	2.1 设计简单实验快速验证解法的正确性
	2.2 在简单数据上，改进解法、畅想更多解法
	2.3 在真实数据上，做实验把解法调work
3. 如何写论文	3.1 根据论文的截止日期设置自己论文重要的几个截止时间
	3.2 写论文
	3.3 Review自己的论文
	3.4 改论文

1. 如何想Idea

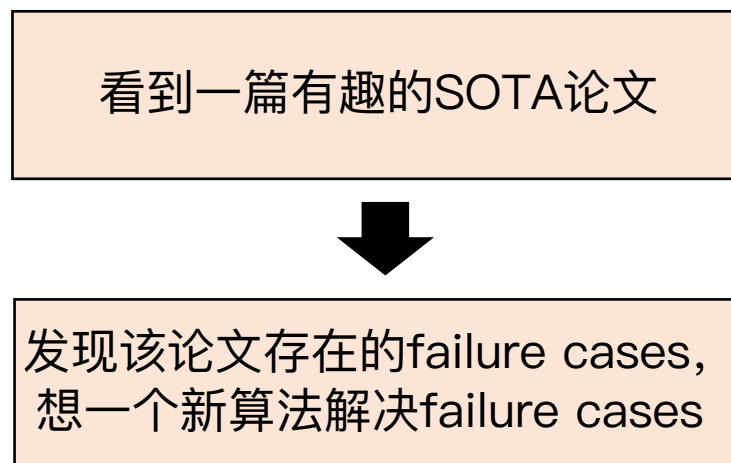
1. 如何想Idea	1.1 规划重要的任务
	1.2 发现任务中需要解决的technical challenge、failure cases
	1.3 构思解决failure cases的方法

我把这样的研究风格称为Goal-driven research，与之相对的还有Idea-driven research。

Goal-driven research vs. Idea-driven research



Goal-driven research



Idea-driven research

Idea-driven research的优缺点

- 优点：容易上手。这应该是大部分新手的科研方式。
- 缺点：
 1. 因为是follow别人，所以除非效果好一大截，不然论文影响力有限。
 2. 容易和别人撞idea，因为其他人也可能在改进这篇论文。
 3. 因为已经有一些解法了，所以解法上的创新空间有限。需要很聪明、有很深刻的见解，才能为这个task提出突破性的new technique。

看到一篇有趣的SOTA论文



发现该论文存在的failure cases,
想一个新算法解决failure cases

Idea-driven research

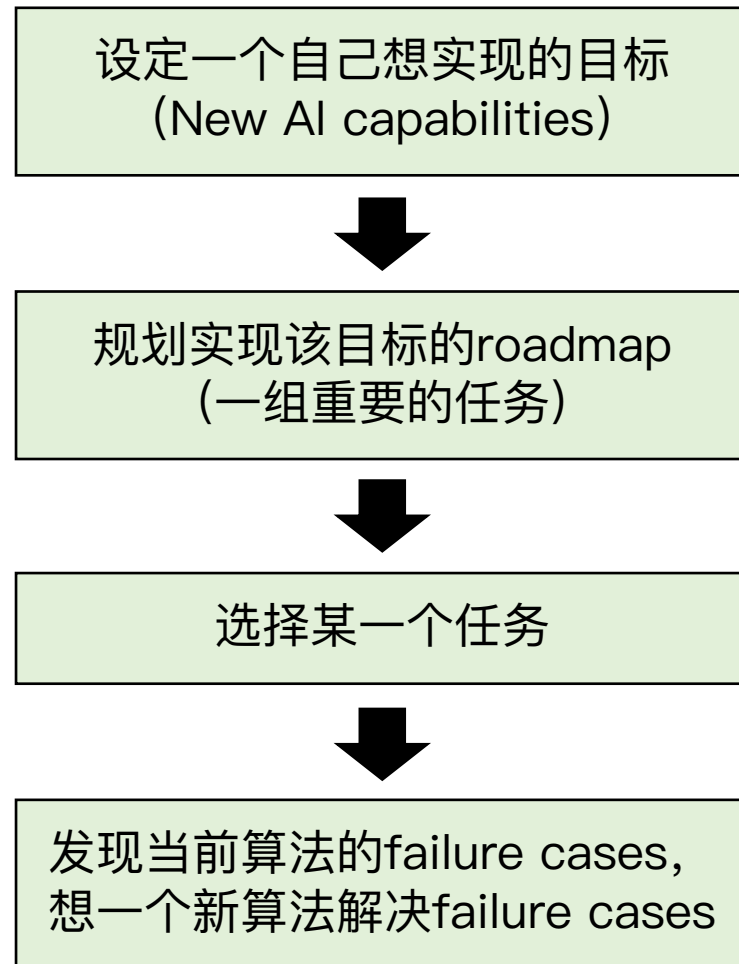
Goal-driven research的优缺点

- 优点:

1. 容易做出突破性的创新, 因为我们不再是follow-up, 而是引领者。
2. 自己的科研更有motivation, 有大的picture, 容易吸引他人。
3. 科研更具有连续性, 成体系。

- 缺点:

1. 规划一个好的roadmap是一个很难的事情。



Goal-driven research

1.1 如何规划重要的任务（如何进行goal-driven research）

1. 如何想Idea	1.1 规划重要的任务
	1.2 发现任务中需要解决的technical challenge、failure cases
	1.3 构思解决failure cases的解法

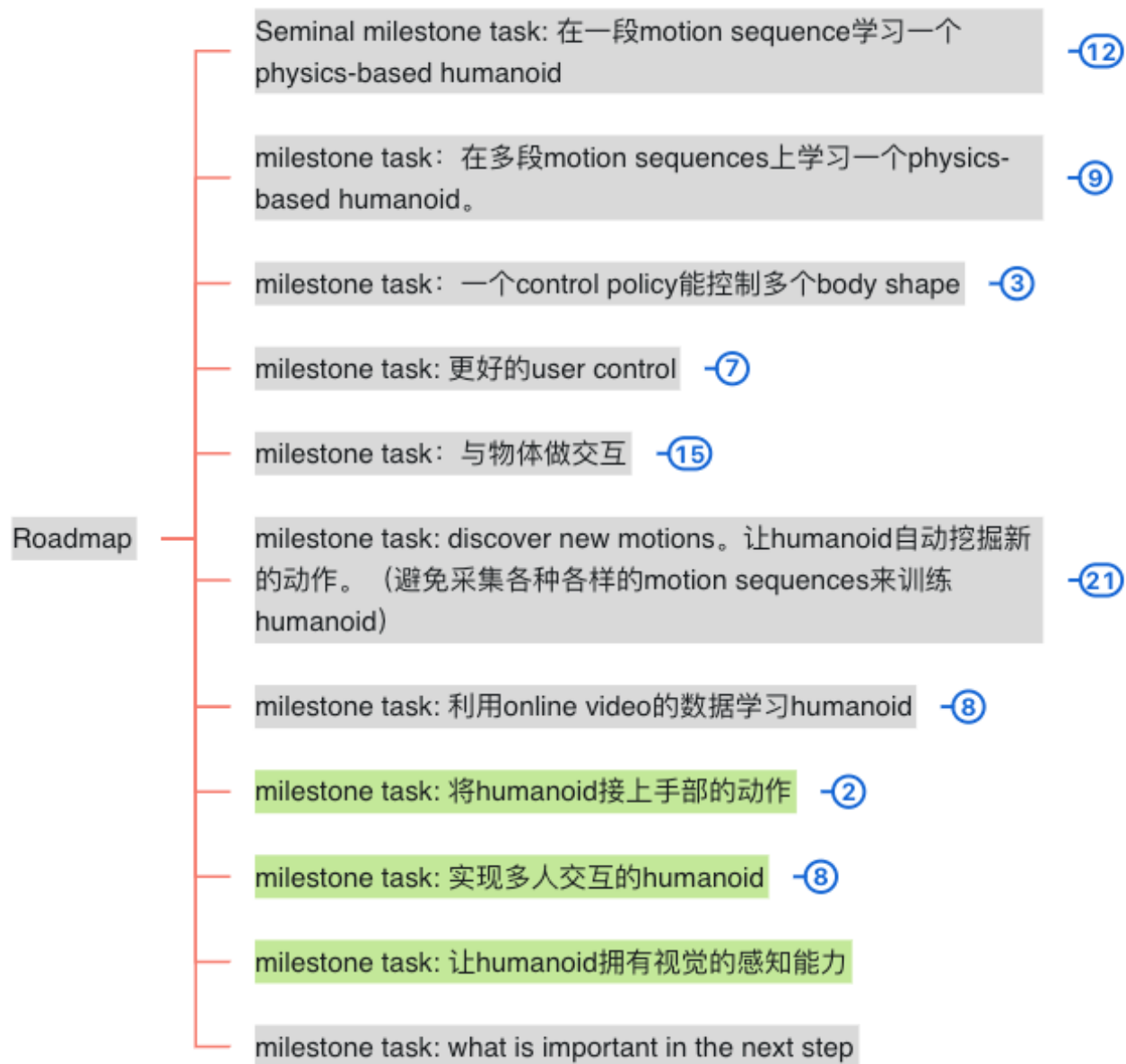
1.1 如何规划重要的任务（如何进行goal-driven research）

- 首先，设定一个自己的长期科研目标。

比如，让ChatGPT拥有身躯，实现一个有自己身躯的AI agent。

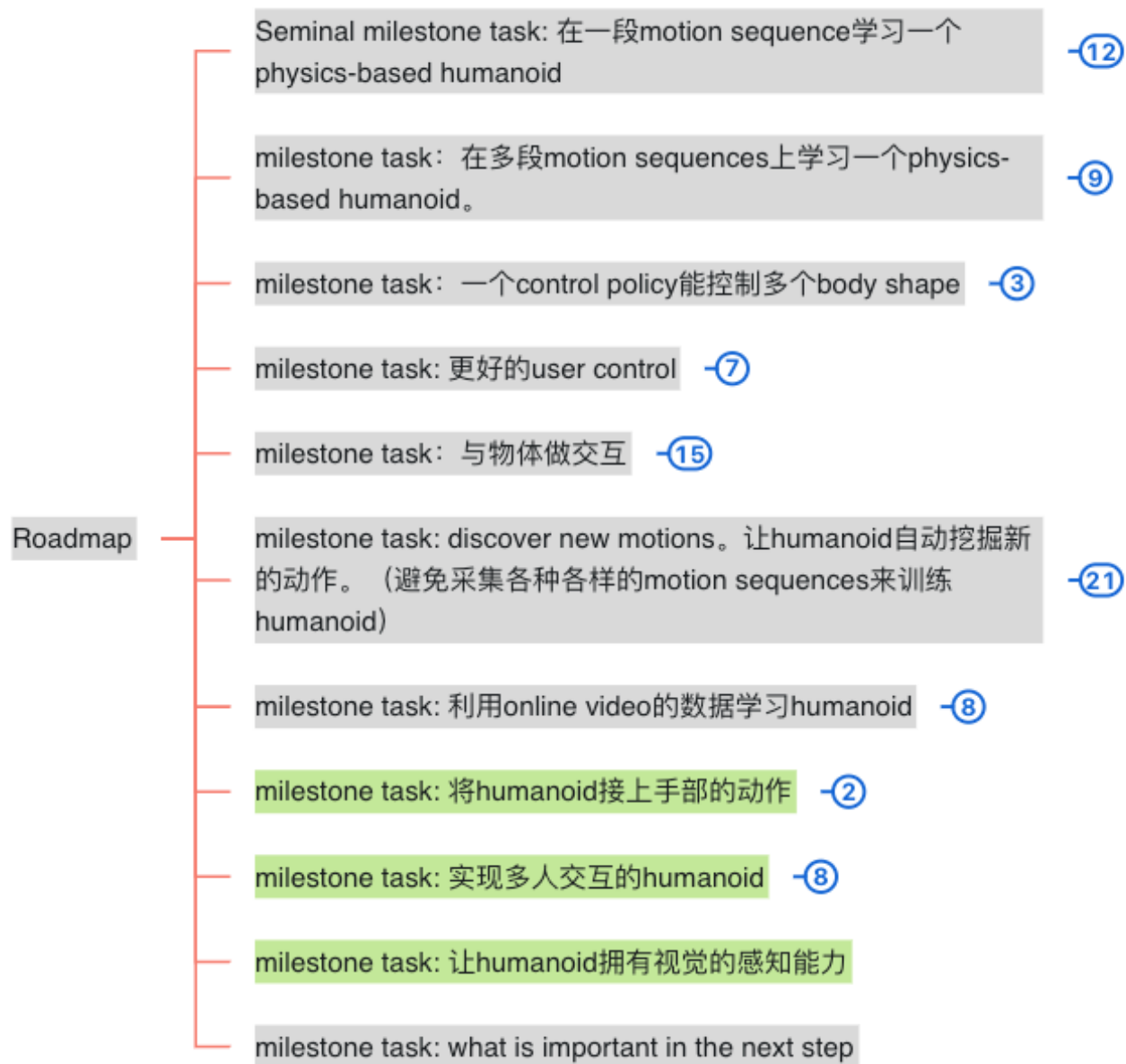
1.1 如何规划重要的任务（如何进行goal-driven research）

- 然后，规划实现该目标的roadmap，制定一组重要的任务。



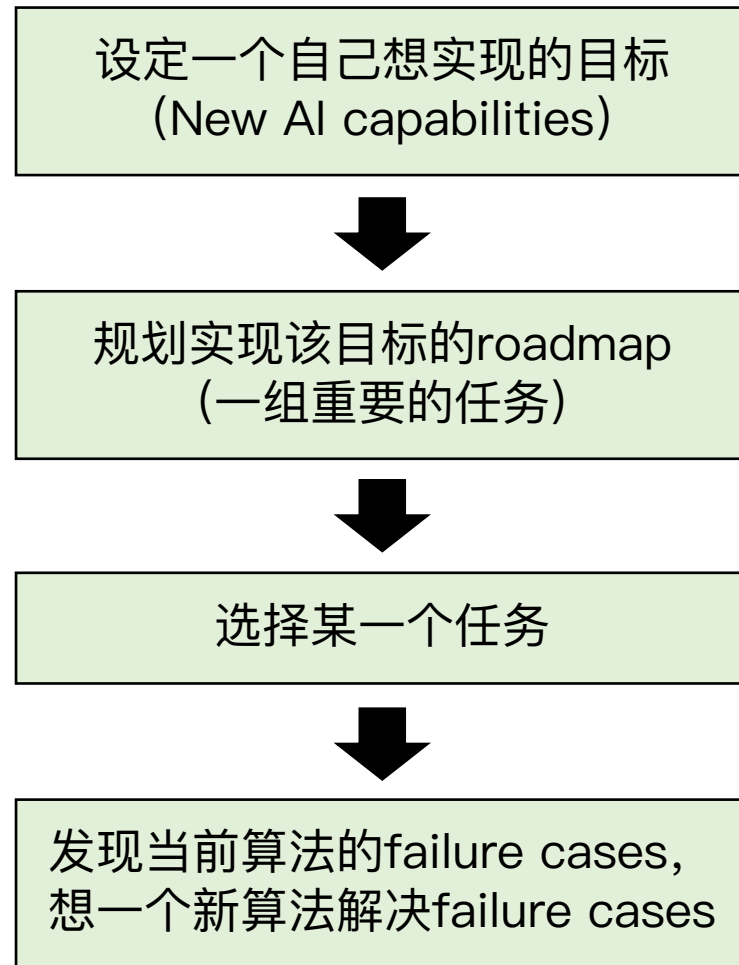
1.1 如何规划重要的任务（如何进行goal-driven research）

- 最后，根据研究空间与当前学术界的技术发展情况，选择合适的任务。



1.1 如何规划重要的任务（如何进行goal-driven research）

规划Roadmap是最重要的一步，也是最难的一步。



Goal-driven research

1.1.1 如何制定roadmap

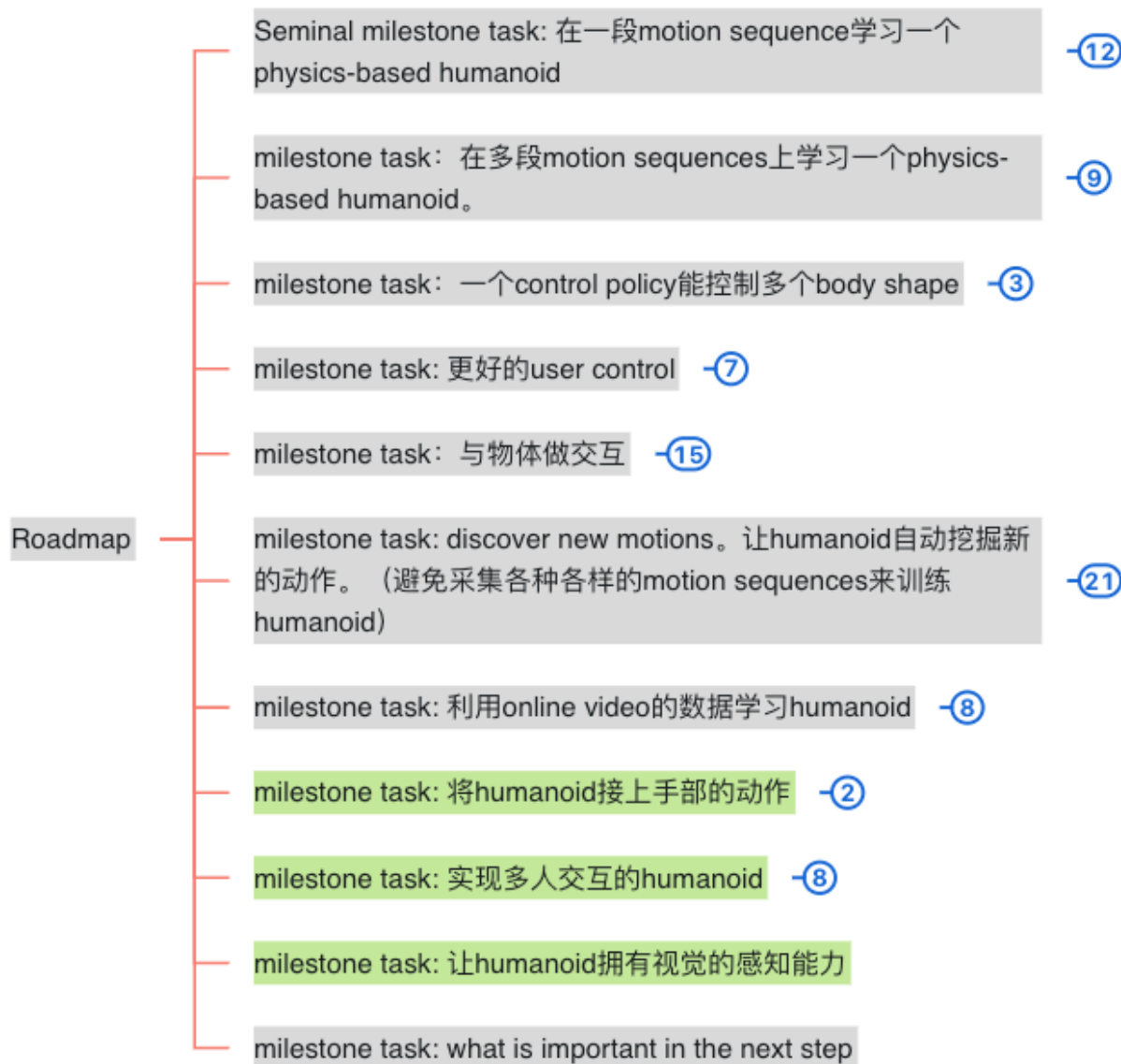
通过literature tree帮助自己：

1. 列出自己科研方向的大多数论文。

1.1.1 如何制定roadmap

通过literature tree帮助自己：

2. 通过阅读论文，梳理出当前方向已有的milestone tasks，并标记提出该task的第一篇论文（1类novelty）。



1.1.1 如何制定roadmap

通过literature tree帮助自己：

3. 将论文根据milestone tasks进行归类。梳理出有代表性的 pipelines/representations，并标记提出该pipeline/representation的第一篇论文（2类novelty）。

milestone task: 在多段motion sequences上学习一个physics-based humanoid。

pipeline: 先学习一个low-level motion controller。再学习一个latent-conditioned controller，输出latent，让low-level motion controller输出action。

pipeline: 学习expert motion policy，再组合到一起实现多sequence

ASE ①

Physics-based Character Controllers Using Conditional VAEs

controlVAE: Model-Based Learning of Generative Controllers for Physics-Based Characters

和conditional VAE是同期工作。用differentiable world model学习motio...

A Scalable Approach to Control Diverse Behaviors for Physically Simulated Characters

1.1.1 如何制定roadmap

通过literature tree帮助自己：

4. 根据pipeline/representation再细分，归类论文（3类novelty）。

1.1.1 如何制定roadmap

通过literature tree帮助自己制定Roadmap

Roadmap

Seminal milestone task: 在一段motion sequence学习一个physics-based humanoid -12

milestone task: 在多段motion sequences上学习一个physics-based humanoid。 -9

milestone task: 一个control policy能控制多个body shape -3

milestone task: 更好的user control -7

milestone task: 与物体做交互 -15

milestone task: discover new motions。让humanoid自动挖掘新的动作。（避免采集各种各样的motion sequences来训练humanoid） -21

milestone task: 利用online video的数据学习humanoid -8

milestone task: 将humanoid接上手部的动作 -2

milestone task: 实现多人交互的humanoid -8

milestone task: 让humanoid拥有视觉的感知能力

milestone task: what is important in the next step

1.2 如何发现任务中需要解决的failure cases

1. 如何想Idea	1.1 规划重要的任务
	1.2 发现任务中需要解决的technical challenge、failure cases
	1.3 构思解决failure cases的方法

1.2 如何发现任务中需要解决的failure cases

- 探索算法的上限，尝试更具有挑战性的cases（通常是数据）。
- 在新的task setting或者新的数据上容易发现新的failure cases。

1.2 如何发现任务中需要解决的failure cases

- 探索算法的上限，尝试更具有挑战性的cases（通常是数据）。
- 在新的task setting或者新的数据上容易发现新的failure cases。
- 注意：在新数据上探索方法的可能性，让大家看到新的实验结论，这是很大的贡献。

1.3 如何构思解决failure cases的方法

1. 如何想Idea	1.1 规划重要的任务
	1.2 发现任务中需要解决的technical challenge、failure cases
	1.3 构思解决failure cases的方法

1.3 如何构思解决failure cases的方法

怎么去提出有效的解法：**积累自己的武器库**。（背后的道理：很多任务中存在相似的technical challenge，而解决这些technical challenge的技术是通用的）

1.3 如何构思解决failure cases的方法

怎么去提出有效的解法：**积累自己的武器库**。（背后的道理：很多任务中存在相似的technical challenge，而解决这些technical challenge的技术是通用的）

怎么积累自己的武器库：**我的做法是构建challenge-insight tree**。帮助自己知道这个研究方向存在哪些technical challenges，而有哪些techniques/insights在尝试解决这些technical challenges。

举个challenge-insight tree的例子 (简化版)

Challenge-insight tree
这个领域的技术挑战有哪些，常用的技术有哪些

challenge: 很难在training motion上学习physics-based model的 control policy

insight: sampling strategy

DeepMimic

insight: 保证training data的质量

insight: 把target pose作为control policy的输入, 减小学习难度。
(但这个策略也导致相应的方法只能复现training motion)

1. Residual Force Control for Agile Human Behavior Imitation and Ext...

challenge: 基于physics-based model学习generative model

insight: 使用GAN训练latent space

1. AMP 2. ASE

①

insight: 使用VAE训练latent space

1. conditional VAE 2. ControlVAE

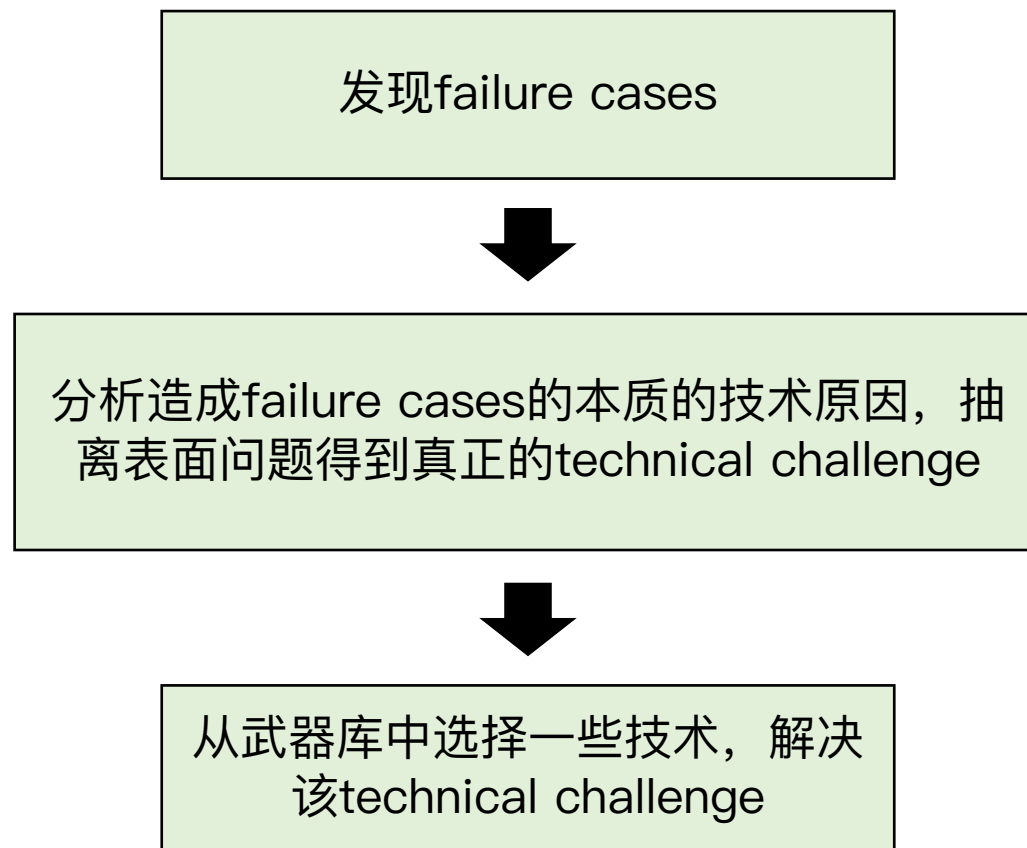
insight: 使用diffusion model

challenge: simulator的reward提供的梯度不够好, 导致训练不稳定

insight: 使用differentiable world model

world model的相比较于reward可以提供更好的梯度。 1. conditi...

1.3 如何构思解决failure cases的方法

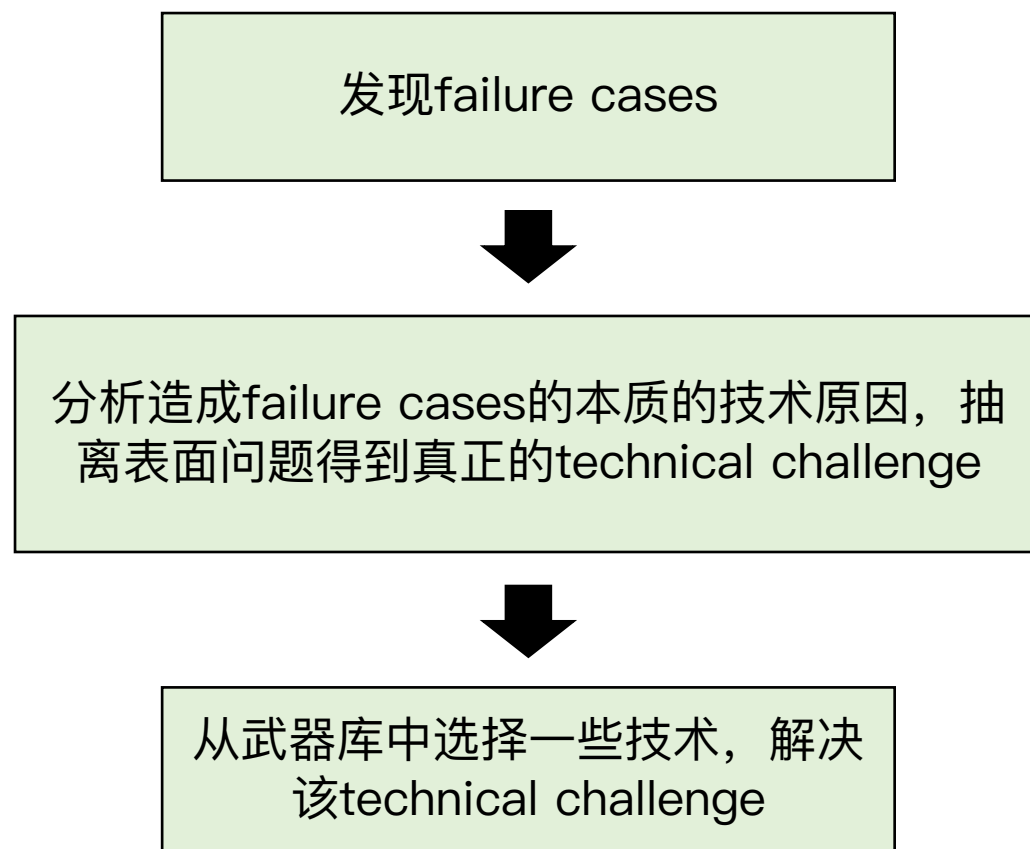


1.3.1 解决重要failure cases的方法一定是novel的

- 大胆的结论：只要failure cases是重要的，那么解决该failure cases的技术一定是新的。
- 证明：如果简单的技术组合即可解决该failure cases，那么该failure cases并不重要。

重要的failure cases会push我们提出novel technique。

需要注意：我们提出的技术需要适用于general cases，不可以只能用于该failure cases。



1.4 一些额外的经验

1. 如何想Idea	1.1 规划重要的任务
	1.2 发现任务中需要解决的technical challenge、failure cases
	1.3 构思解决failure cases的方法

1.4.1 特殊的想idea的情况

当出现新锤子的时候，非常值得拿新锤子来做自己roadmap上的某一个milestone task，这样容易做出有影响力的工作。

例子：

1. NeRF出来的时候，出现了VolSDF、NeuS。
2. Stable diffusion出来的时候，出现了DreamFusion。

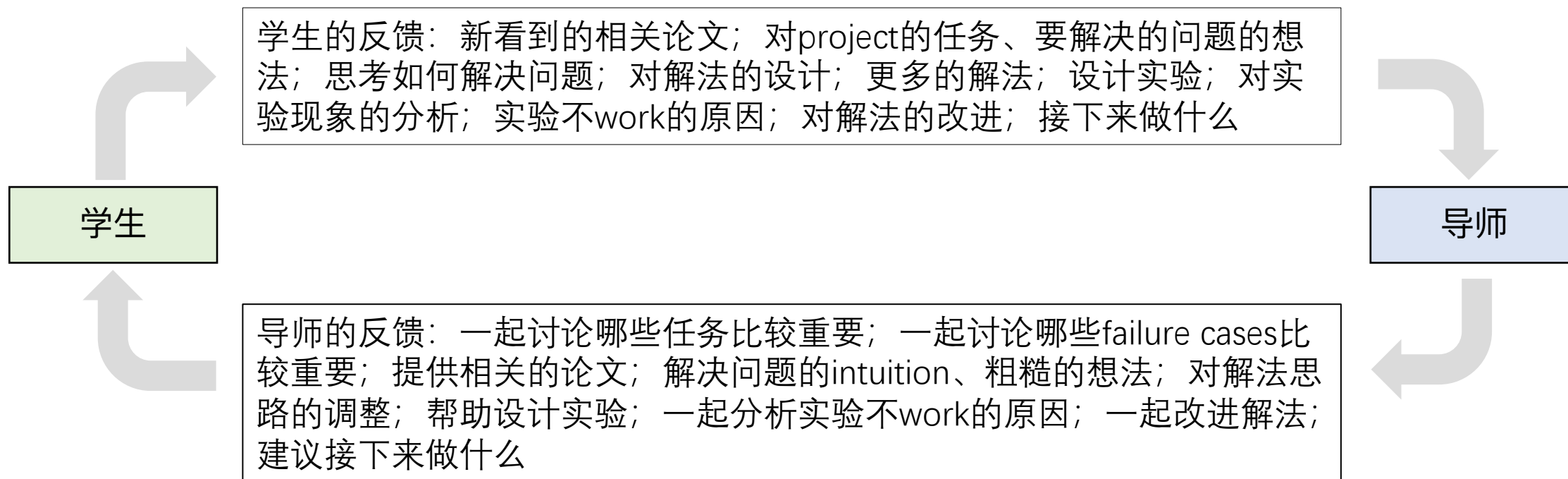
1.4.2 想解决方法时要有的心态：**独立自主**

请勿依赖导师。原因如下：

- 导师不一定比自己懂。毕竟自己是在此前沿问题上做得最多的人。
- 导师没有时间设计详细的解法。

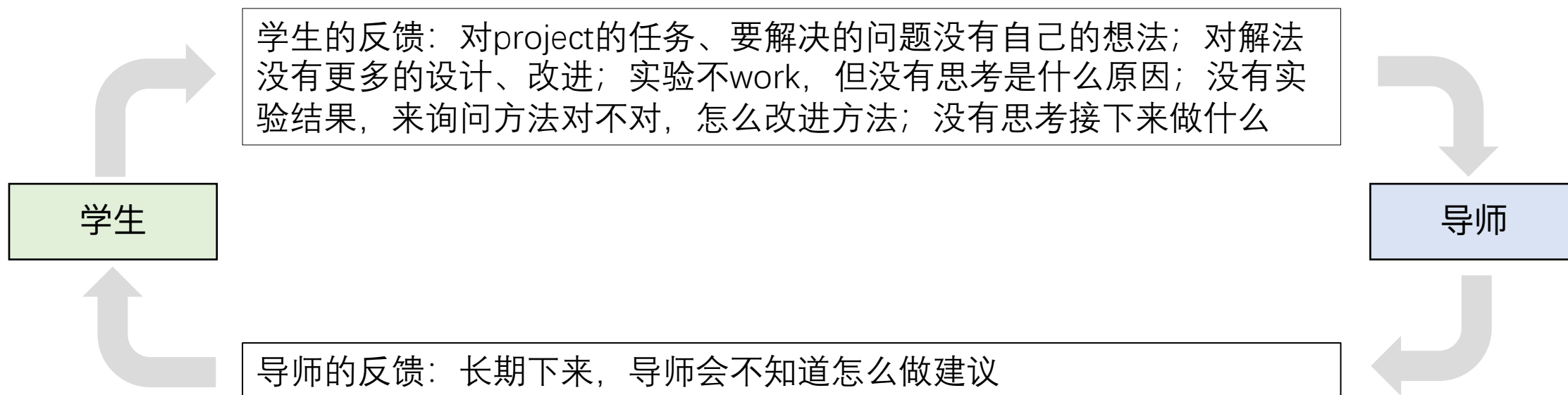
1.4.3 导师和学生是互相促进的，警惕某一方的单向输出

正面例子：导师和学生互相促进



1.4.3 导师和学生是互相促进的，警惕某一方的单向输出

反面例子：学生没有有效的反馈



2. 如何做实验

1. 如何想Idea	1.1 规划重要的任务
	1.2 发现任务中需要解决的technical challenge、failure cases
	1.3 构思解决failure cases的方法
2. 如何做实验	2.1 设计简单实验快速验证解法的正确性
	2.2 在简单数据上，改进解法、畅想更多解法
	2.3 在真实数据上，做实验把解法调work

2.1 如何设计简单实验快速验证解法的正确性

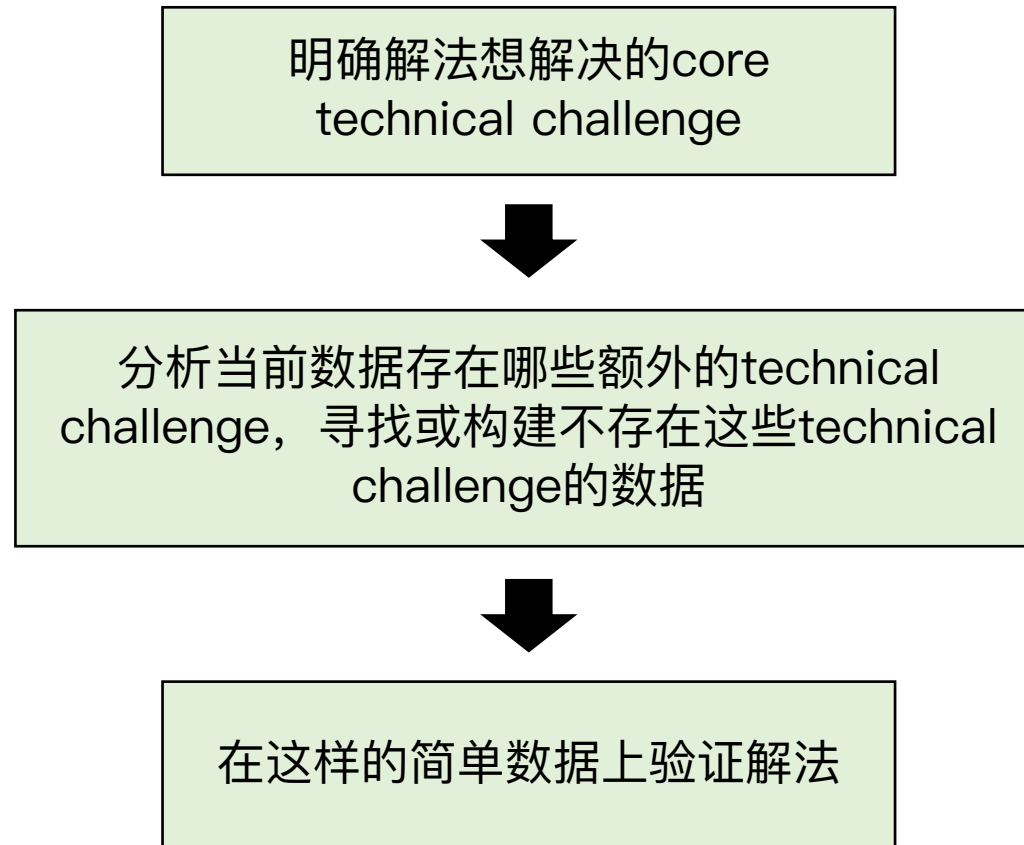
- 什么是简单实验：只存在core technical challenge的实验

2.1 如何设计简单实验快速验证解法的正确性

- 什么是简单实验：只存在core technical challenge的实验
- 通过简单实验验证解法的好处：
 1. 实验成本低
 2. 不存在其他technical challenges的干扰，可以更好地确定当前的解法是否work

2.1 如何设计简单实验快速验证解法的正确性

- 什么是简单实验：只存在core technical challenge的实验
- 如何设计简单实验

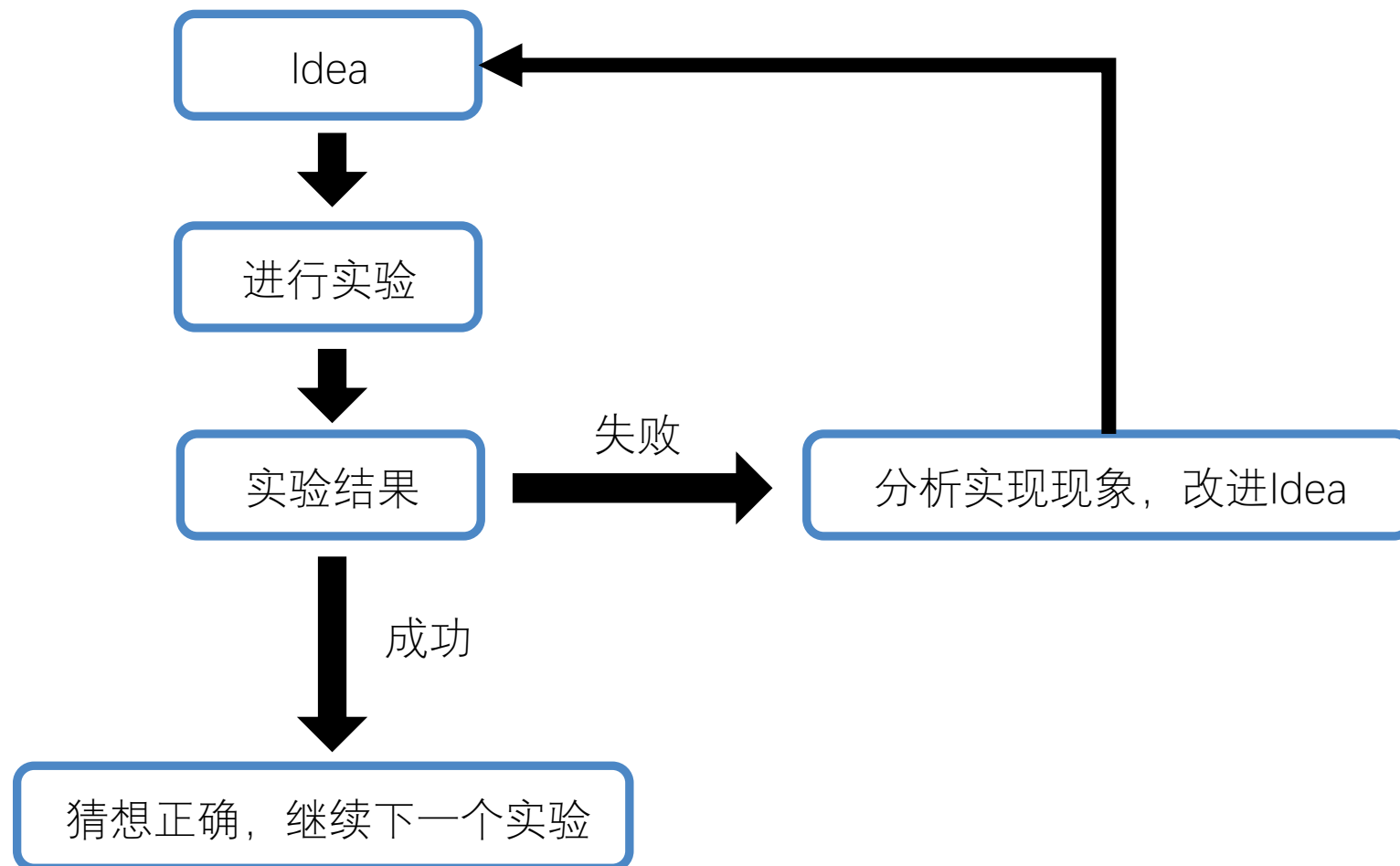


2.2 在简单数据上，改进解法、畅想更多解法

1. 如何想Idea	1.1 规划重要的任务
	1.2 发现任务中需要解决的technical challenge、failure cases
	1.3 构思解决failure cases的方法
2. 如何做实验	2.1 设计简单实验快速验证解法的正确性
	2.2 在简单数据上，改进解法、畅想更多解法
	2.3 在真实数据上，做实验把解法调work

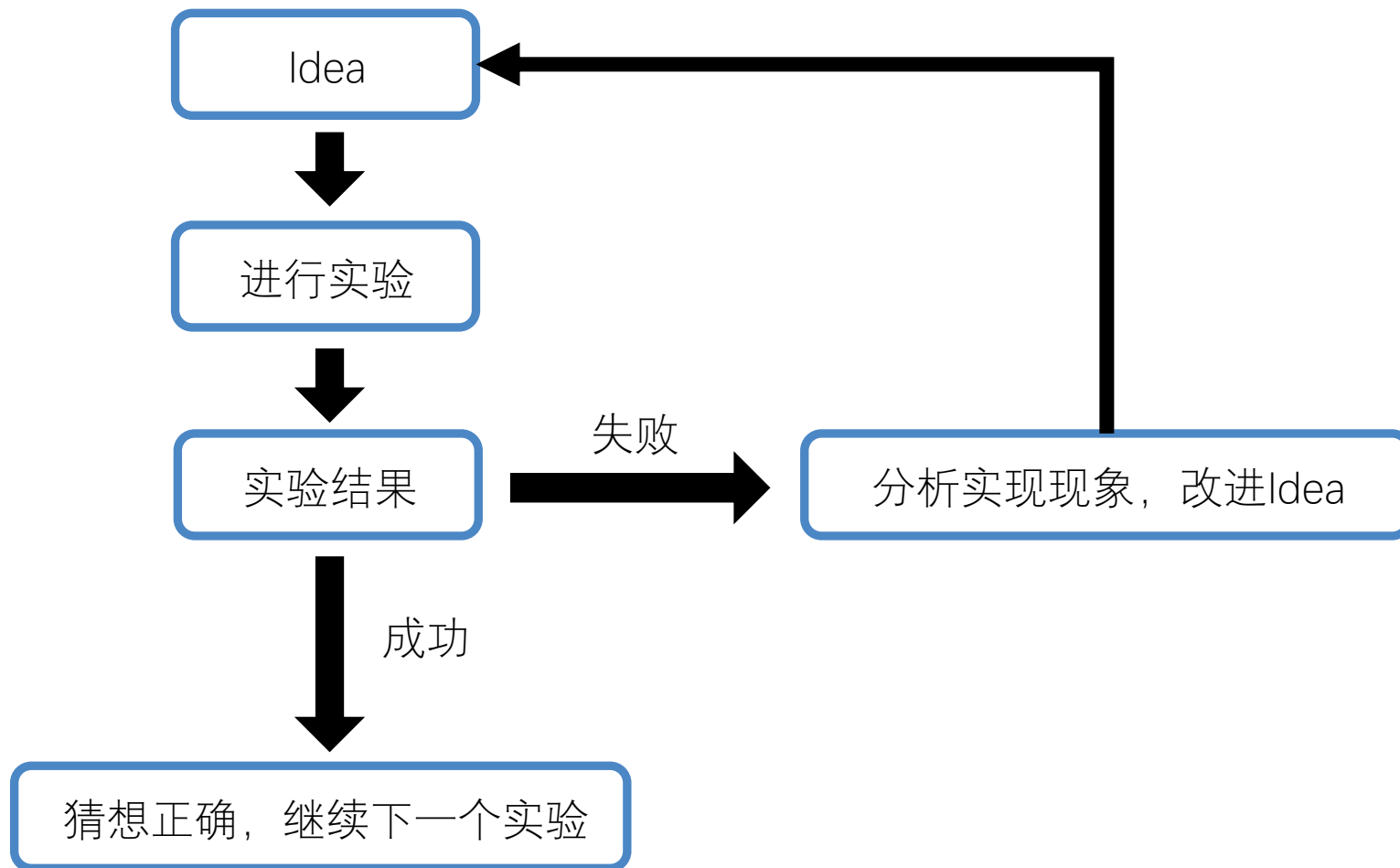
2.2 改进解法、畅想更多解法

- 常见的实验流程：



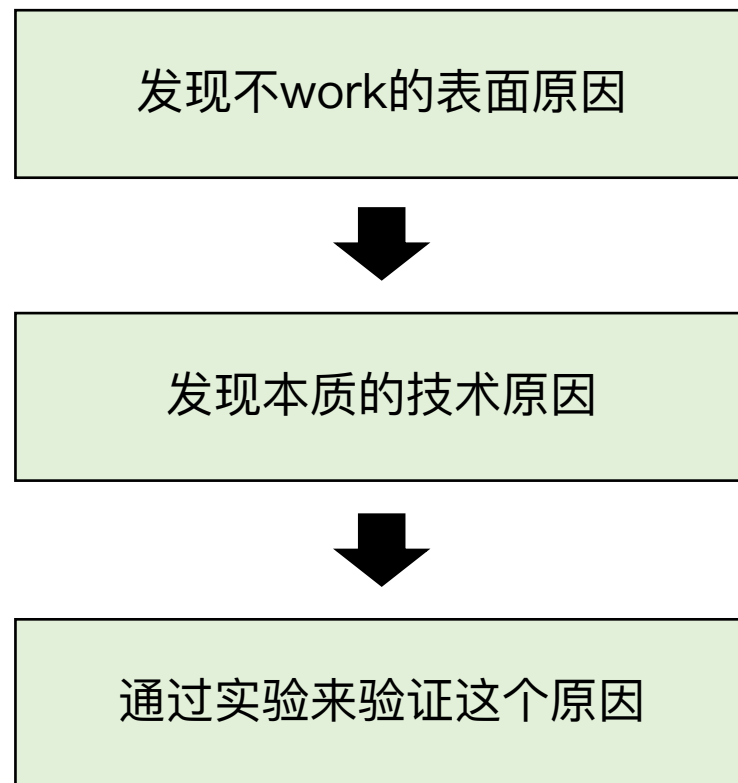
2.2 改进解法、畅想更多解法

- 常见的实验流程 → 核心步骤：**分析实验不work的原因**



2.2.1 如何分析实验不work的原因

- 表面原因有哪些：
 - 换到某个数据后，效果变得不好了
 - 加了某个module后，效果变得不好了
 - 改了某个参数后，效果变得不好了
 - ...
- 本质的技术原因有哪些：
 - 为什么换到某个数据后，效果变得不好了
 - 为什么加了某个module后，效果变得不好了
 - 为什么改了某个参数后，效果变得不好了
 - ...

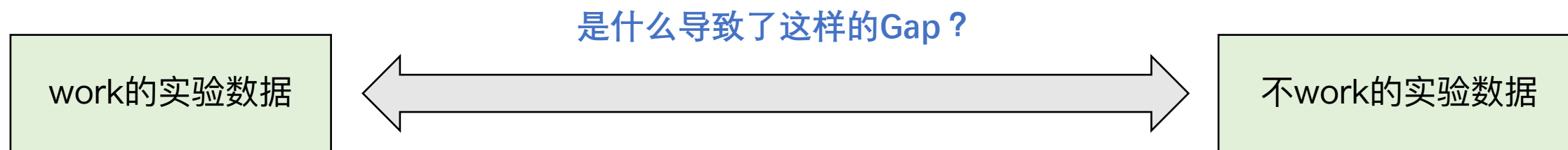


2.2.1.1 如何发现实验不work的表面原因

- 找到work的数据/算法，再对比不work的数据/算法，看看它们有什么不同。

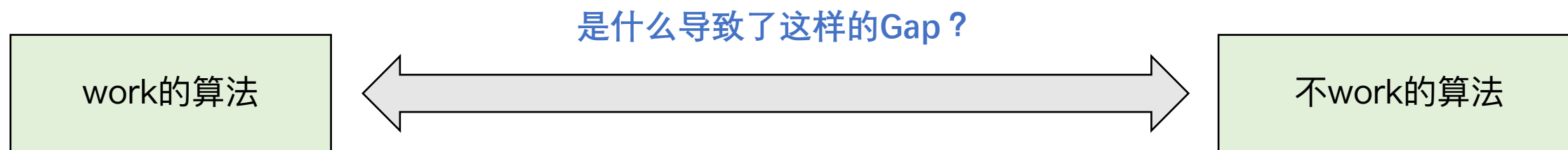


2.2.1.1 如何发现实验不work的表面原因



- 可能是数据造成了实验不work。两种情况：
 - 同一组实验中，存在good cases和failure cases，直接对比两组数据的差异。
 - 一个实验中，所有数据全部fail。切换到更简单的数据做实验，寻找可以work的实验数据。然后再对比两组数据的差异。

2.2.1.1 如何发现实验不work的表面原因



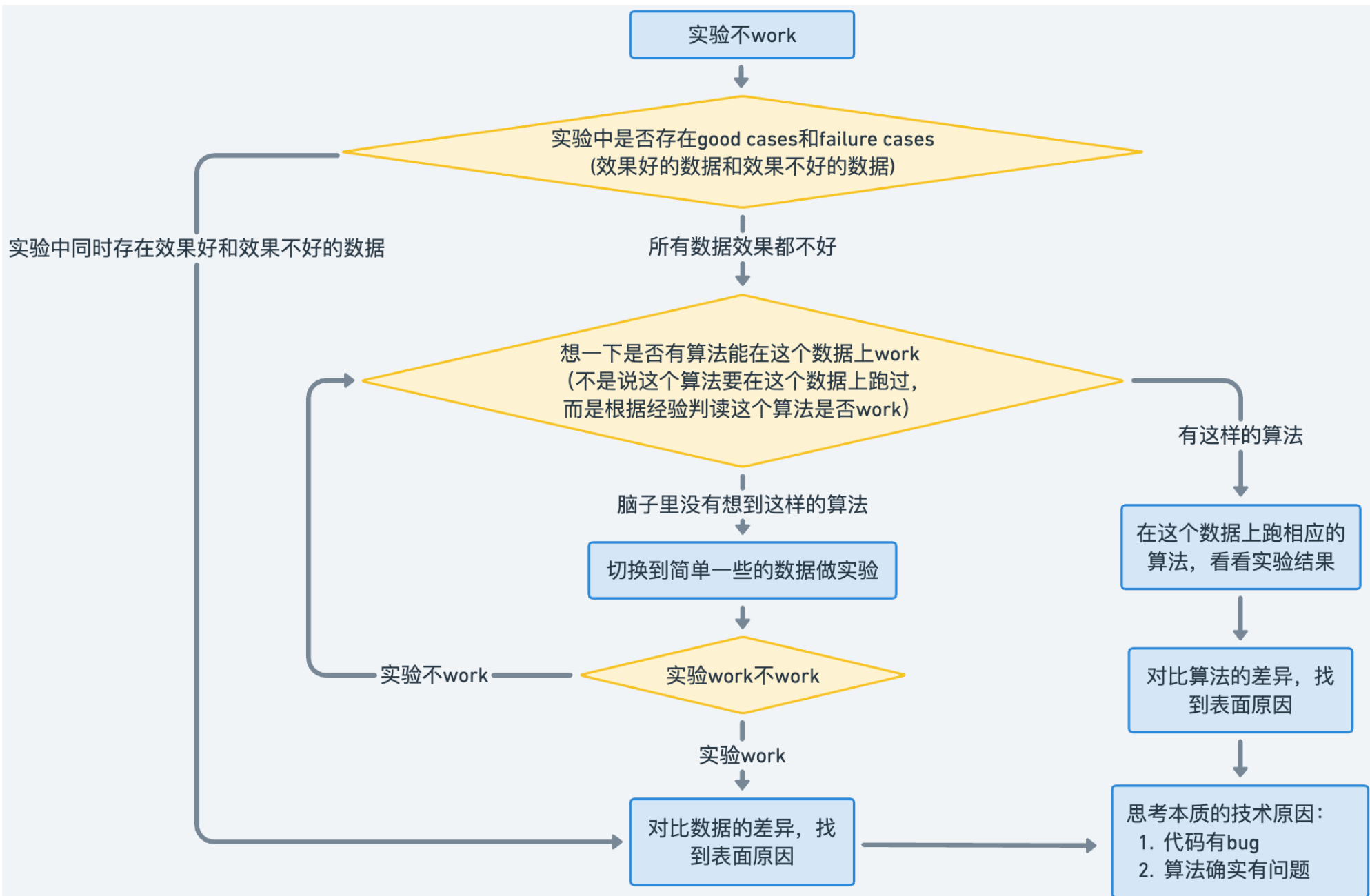
- 对比两个算法的差异，并尽量减少两个算法的差异，从而找到导致实验不work的关键的pipeline module

2.2.1.1 如何发现实验不work的表面原因



- 同一个算法，对比“效果好的数据”和“效果不好的数据”
- 同一个数据，对比“效果好的算法”和“效果不好的算法”

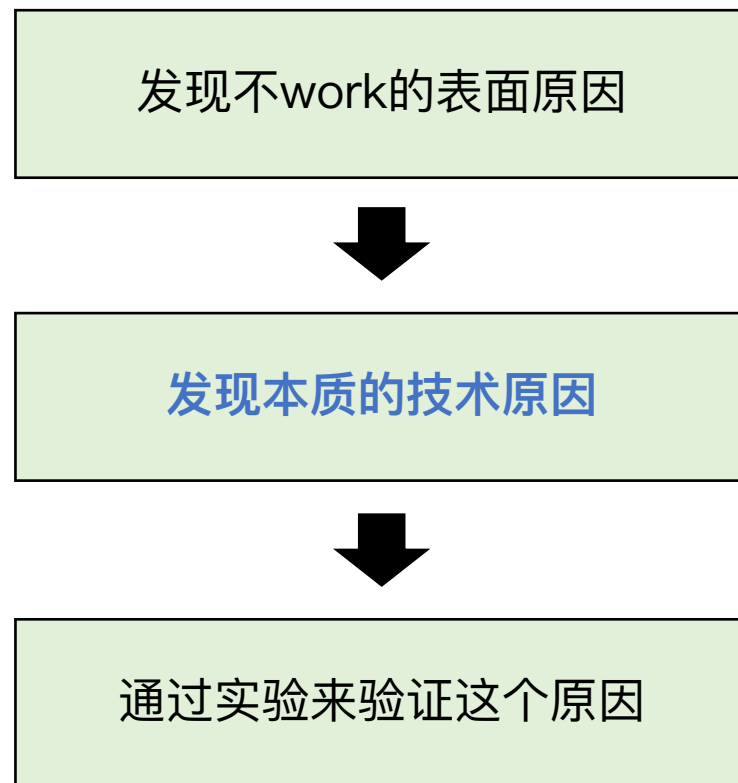
2.2.1.1 如何发现实验不work的表面原因（切勿完全按照这个流程执行，要灵活一些）



2.2.1.2 如何发现实验不work的本质技术原因

列出尽量多的可能的技术原因

- 本质的技术原因有哪些（思考为什么）：
 - 为什么换到某个数据后，效果变得不好了
 - 为什么加了某个module后，效果变得不好了
 - 为什么改了某个参数后，效果变得不好了
 - ...



2.2.1.2 如何发现实验不work的本质技术原因

一般有两个技术原因：

1. 可能是代码有bug。
2. 可能是“xx算法”在“xx数据”上确实有问题。

算法有问题的三种可能：超参没设置对、算法缺了几个tricks、算法本身确实不行。

2.2.1.2.1 怎么找代码的bug

1. 可以逐行检查代码的输出，验证输出的结果和自己的预期是否一样。
可以检查数据的shape，还有可视化代码输出的结果来验证。
2. 这个bug可能是个人对算法的理解不到位，此时需要去看论文或者原理性的东西去理解透彻了再回去检查代码。

2.2.1.2.2 怎么找算法的问题

一个有效的方法是看相关的论文为什么可以work，看他们使用了什么tricks。

给相关的论文算法做ablation study，看看算法的哪些部分是关键。

2.2.1.2.2 怎么找算法的问题

一个有效的方法是看相关的论文为什么可以work，看他们使用了什么tricks。

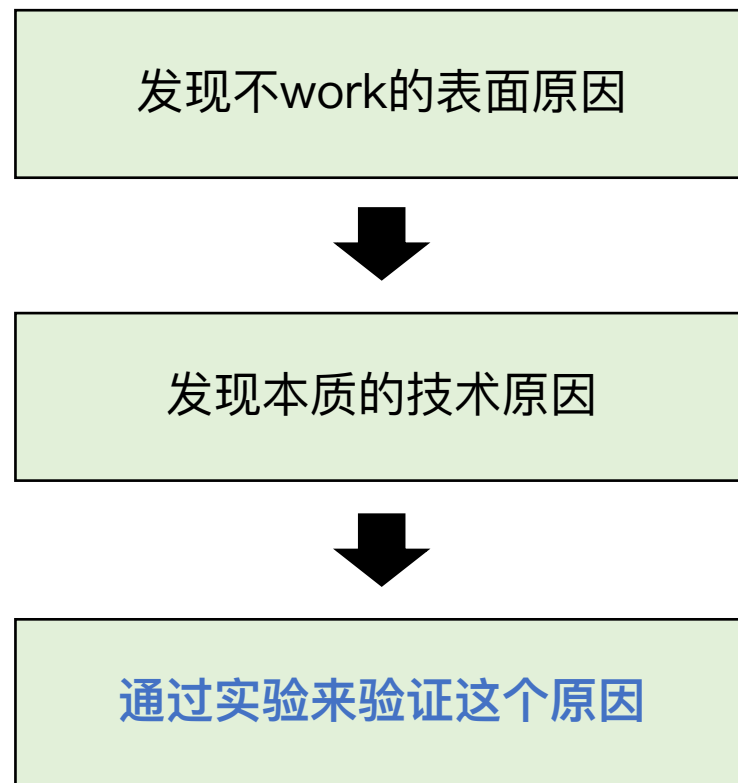
给相关的论文算法做ablation study，看看算法的哪些部分是关键。

有时候，一些算法确实是work的，只是少了点tricks。也就是说，这些牛逼的ideas，单独自己的时候不work，需要加一些tricks才work。

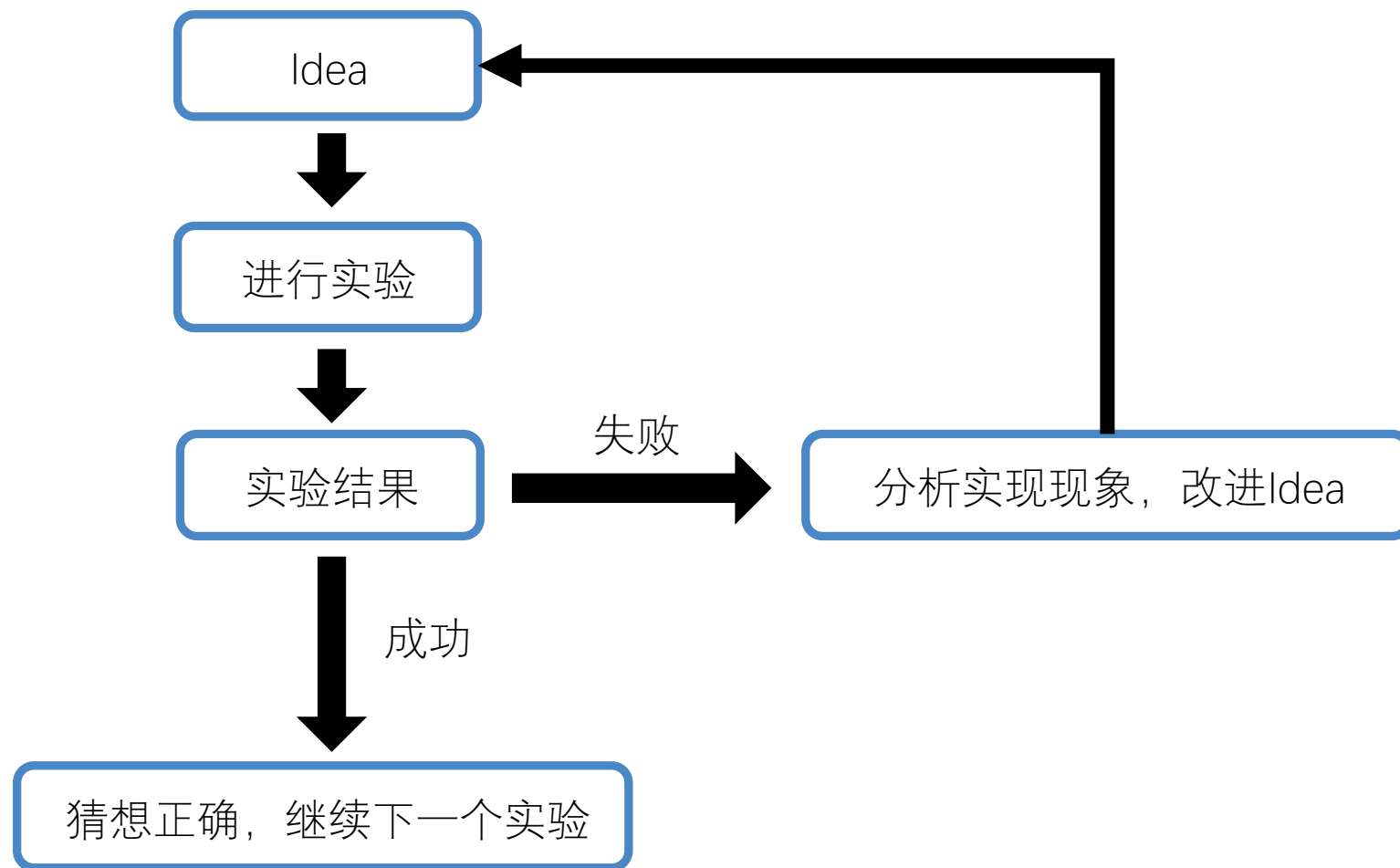
(比如，NeRF + positional encoding)

2.2.1.3 通过实验来验证找到的原因

- 列出尽量多的可能的技术原因以后，
通过实验来验证找到的原因。
- 需要先确定真的是“技术原因”，才能有效地改进算法。



2.2 针对找到的**技术原因**，改进解法、畅想更多解法



2.3 在真实的数据上，把算法调work

1. 如何想Idea	1.1 规划重要的任务
	1.2 发现任务中需要解决的technical challenge、failure cases
	1.3 构思解决failure cases的方法
2. 如何做实验	2.1 设计简单实验快速验证解法的正确性
	2.2 在简单数据上，改进解法、畅想更多解法
	2.3 在真实数据上，做实验把解法调work

3. 如何写论文

1. 如何想Idea	1.1 规划重要的任务
	1.2 发现任务中需要解决的technical challenge、failure cases
	1.3 构思解决failure cases的方法
2. 如何做实验	2.1 设计简单实验快速验证解法的正确性
	2.2 在简单数据上，改进解法、畅想更多解法
	2.3 在真实数据上，做实验把解法调work
3. 如何写论文	3.1 根据论文的截止日期设置自己论文重要的几个截止时间
	3.2 写论文
	3.3 Review自己的论文
	3.4 改论文

3.1 根据论文的截止日期设置自己论文重要的几个截止时间

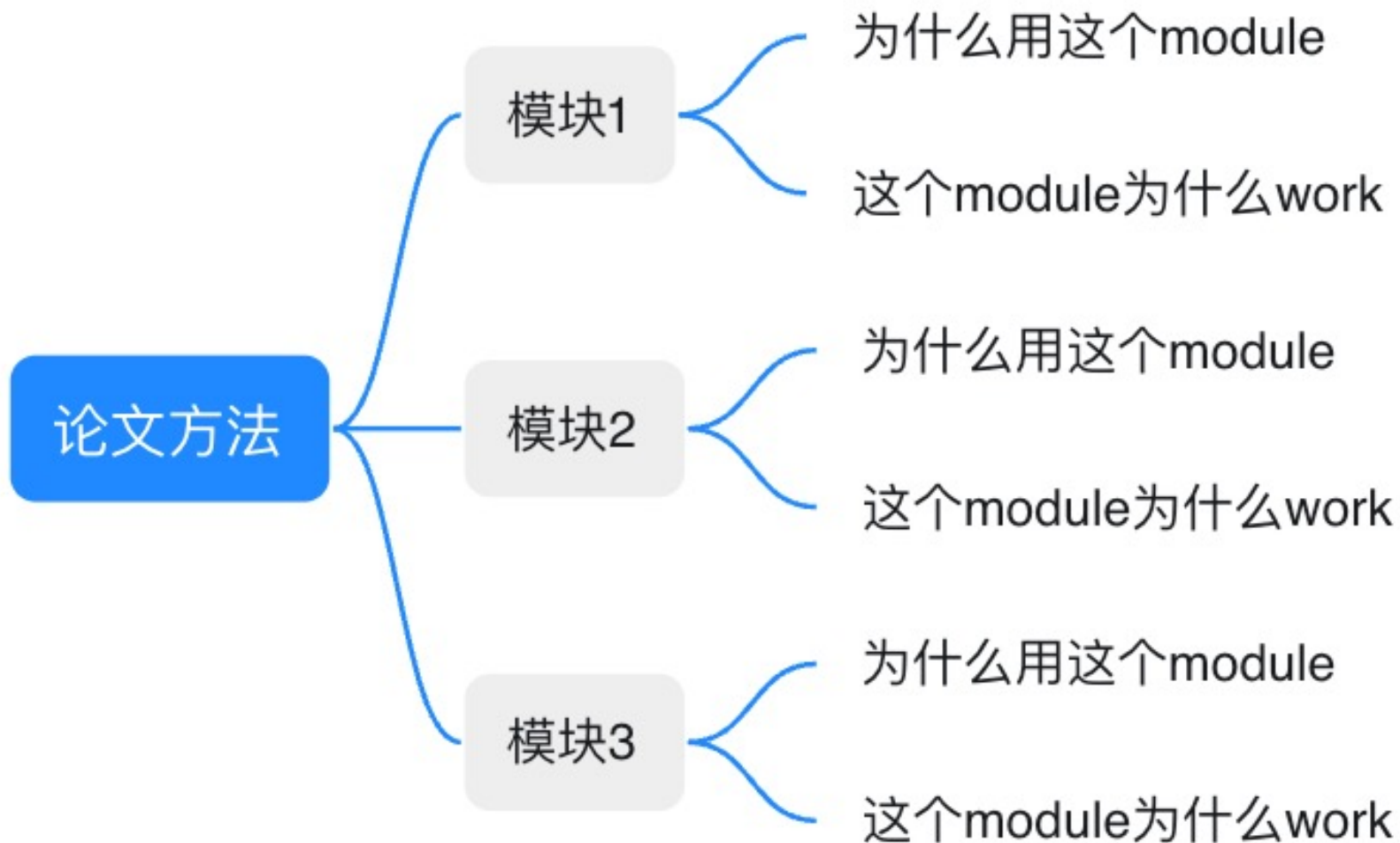
时间点	论文要写的内容
截稿时间四周前	<ol style="list-style-type: none">1. 整理现有的story, 包括core contribution、方法的各个模块及其motivation (推荐用脑图整理)。2. 列出要做的comparison experiments和ablation studies (推荐用脑图整理)。3. 这一周写一个introduction的初稿。
截稿时间三周前	<p>这一周最好能把方法定下来。</p> <ol style="list-style-type: none">1. 这一周写一个method的初稿。这一周至少method框架定下来了, 所以可以把method开始写起来。如果方法的细节还没定下来, 就在相应的地方写给\{todo\}, 先空着, 至少把method的框架写出来。 <p>这周截止, 必须把introduction和method的初稿给导师改, 不然导师很可能改不完论文 (想象一下, 导师最后几天开始改十篇非常不完整的论文, 是什么样的地狱体验。如果自己面对这种情况, 会是什么心情)。</p>
截稿时间两周前	这一周把experiments, abstract, related work写一个初稿。
截稿最后一周	改论文、画图、做demo

3.2 如何写论文

写论文和做饭有点像，具体有两步：

1. **准备食材**：列出当前方法的所有模块。对于每个模块，回答两个问题：为什么用这个module、这个module为什么work。记录为一个脑图，称为method tree。
2. **炒菜**：根据method tree，梳理论文的story，整理要做的实验。

3.2.1 如何梳理方法的模块：method tree



3.2.2 如何梳理论文的story：倒推，然后正推

倒推：

1. 首先确定论文中想突出的contributions。根据method tree，我们容易知道论文的contributions。
2. 然后，明确我们contributions的好处是什么，解决了什么technical challenge。
3. 最后，根据这个technical challenge，思考怎么通过讨论之前的方法引出我们解决了的technical challenge。

3.2.3 如何梳理论文的story：倒推，然后正推

正推：

1. 介绍论文的Task
2. 通过讨论之前的方法引出我们解决了的technical challenge
3. 为了解决这个technical challenge，我们提出了xx contributions。
4. 我们contributions的技术优势是什么。

3.2.4 论文一般要做什么实验

1. 对比实验。
2. Ablation studies。
3. Applications、demos。

3.2.4.1 要做什么ablation studies

一篇论文包含了一些core contributions和每个pipeline module中的一些design choices。

读者通常会很在意core contributions对performance的影响，并且会好奇这些design choices是否真的有用。

3.2.4.1 要做什么ablation studies

- **一个大表和相应的可视化对比图**：列出论文的core contributions以及一些重要的components对论文方法performance的影响。
- **一些小表和相应的可视化对比图**：每个小表分别列出一个pipeline module中design choices对论文方法performance的影响（方法对超参的敏感性，方法对input data质量的敏感性，不采用某个design choice对performance的影响）。

3.2.4.2 要做什么applications/demos

- 好的demo和论文的影响力有非常大的关系，增加论文阅读量。

当把算法在目标数据上做work以后，尽量在更多新的更具挑战性的数据上做实验，看看算法的极限，挖掘算法的潜力。大家都很想看看一个算法能做到什么样的程度，这是一个很大的贡献。

3.3 如何review自己的论文

1. 如何想Idea	1.1 规划重要的任务
	1.2 发现任务中需要解决的technical challenge、failure cases
	1.3 构思解决failure cases的方法
2. 如何做实验	2.1 设计简单实验快速验证解法的正确性
	2.2 在简单数据上，改进解法、畅想更多解法
	2.3 在真实数据上，做实验把解法调work
3. 如何写论文	3.1 根据论文的截止日期设置自己论文重要的几个截止时间
	3.2 写论文
	3.3 Review自己的论文
	3.4 改论文

3.3 如何review自己的论文：仔细检查论文是否存在被拒的因素

1. Contribution不够 (论文没有给读者带来新的知识)	1.1 想解决的failure cases很常见
	1.2 提出的技术已经被well-explored了，该技术带来的performance improvement是可预见的/well-known的
2. 写作不清楚	2.1 缺少技术细节，不可复现
	2.2 某个方法模块缺少motivation
3. 实验效果不够好	3.1 只比之前的方法效果好了一点
	3.2 虽然比之前的方法效果好，但效果仍然不够好
4. 实验测试不充分	4.1 缺少ablation studies
	4.2 缺少重要的baselines、缺少重要的evaluation metric
	4.3 数据太简单，无法证明方法是否真的work
5. 方法设计有问题	5.1 实验的setting不实际
	5.2 方法存在技术缺陷，看起来不合理
	5.3 方法不鲁棒，需要每个场景上调超参
	5.4 新的方法设计在带来benefit的同时，引入了更强的limitation，导致新方法的收益为负

一个research project包含哪些流程

1. 如何想Idea	1.1 规划重要的任务
	1.2 发现任务中需要解决的technical challenge、failure cases
	1.3 构思解决failure cases的方法
2. 如何做实验	2.1 设计简单实验快速验证解法的正确性
	2.2 在简单数据上，改进解法、畅想更多解法
	2.3 在真实数据上，做实验把解法调work
3. 如何写论文	3.1 根据论文的截止日期设置自己论文重要的几个截止时间
	3.2 写论文
	3.3 Review自己的论文
	3.4 改论文

要完成一个research project, 一个博士生应该具有哪些能力

1. 阅读	1.1 阅读大量论文, 进行literature review, 构建literature tree的能力
	1.2 追踪积累最新论文/技术的能力
2. 创新	2.1 Goal-driven research的能力
	2.2 Search的能力
	2.3 设计解决问题的方法的能力
3. 实践	3.1 代码能力、实现能力
	3.2 分析实验不work的原因的能力
4. 展示	4.1 论文写作能力
	4.2 把论文做得漂亮、吸引人, 做demo的能力
	4.3 做Slides, 讲Talk的能力
5. 交流	5.1 和导师、同学交流讨论的能力



浙江大学
ZHEJIANG UNIVERSITY

谢谢!

报告人：彭思达

GitHub Repo: https://github.com/pengsida/learning_research