



GAMES 003 科研素养课

第五周：面向实验结果的方案优化



彭思达



高俊

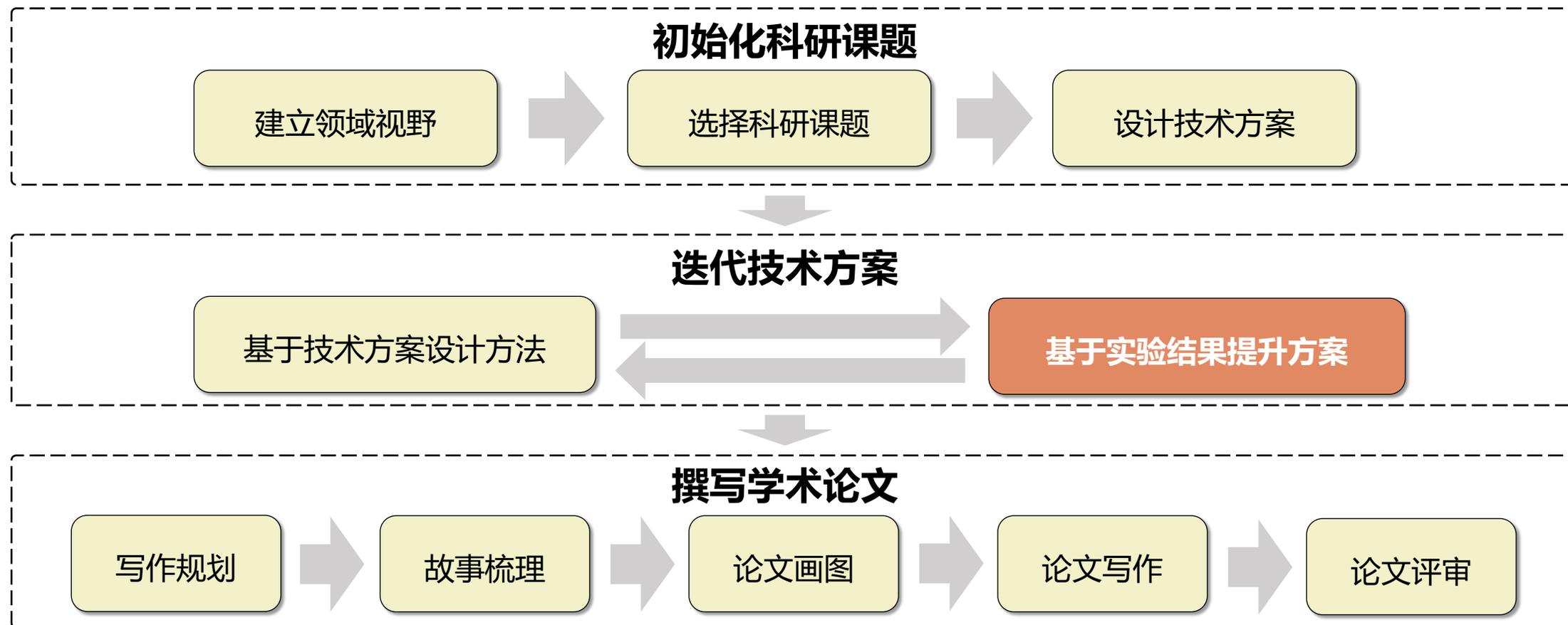


彭崧猷



王倩倩

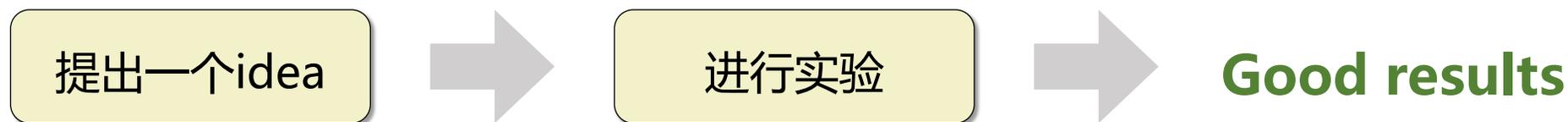
基于实验结果提升方案



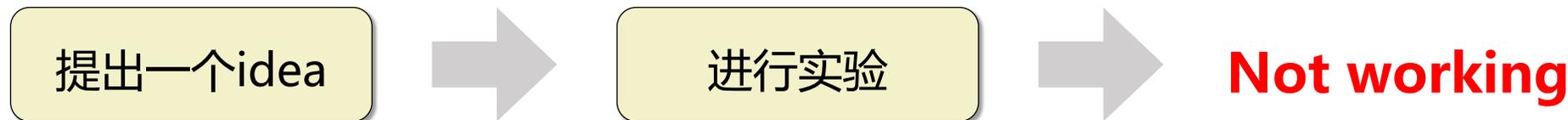
为什么需要改进方法

- 因为初始提出的方法通常会不work。

理想情况：

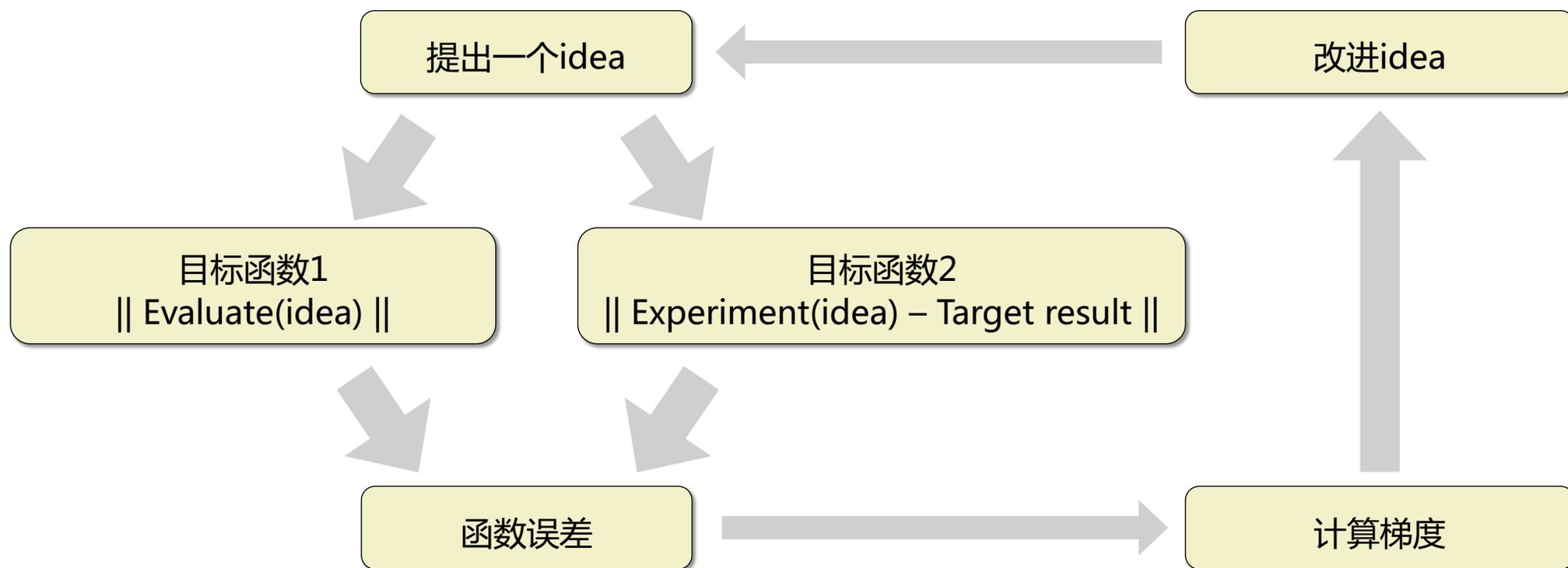


实际情况：

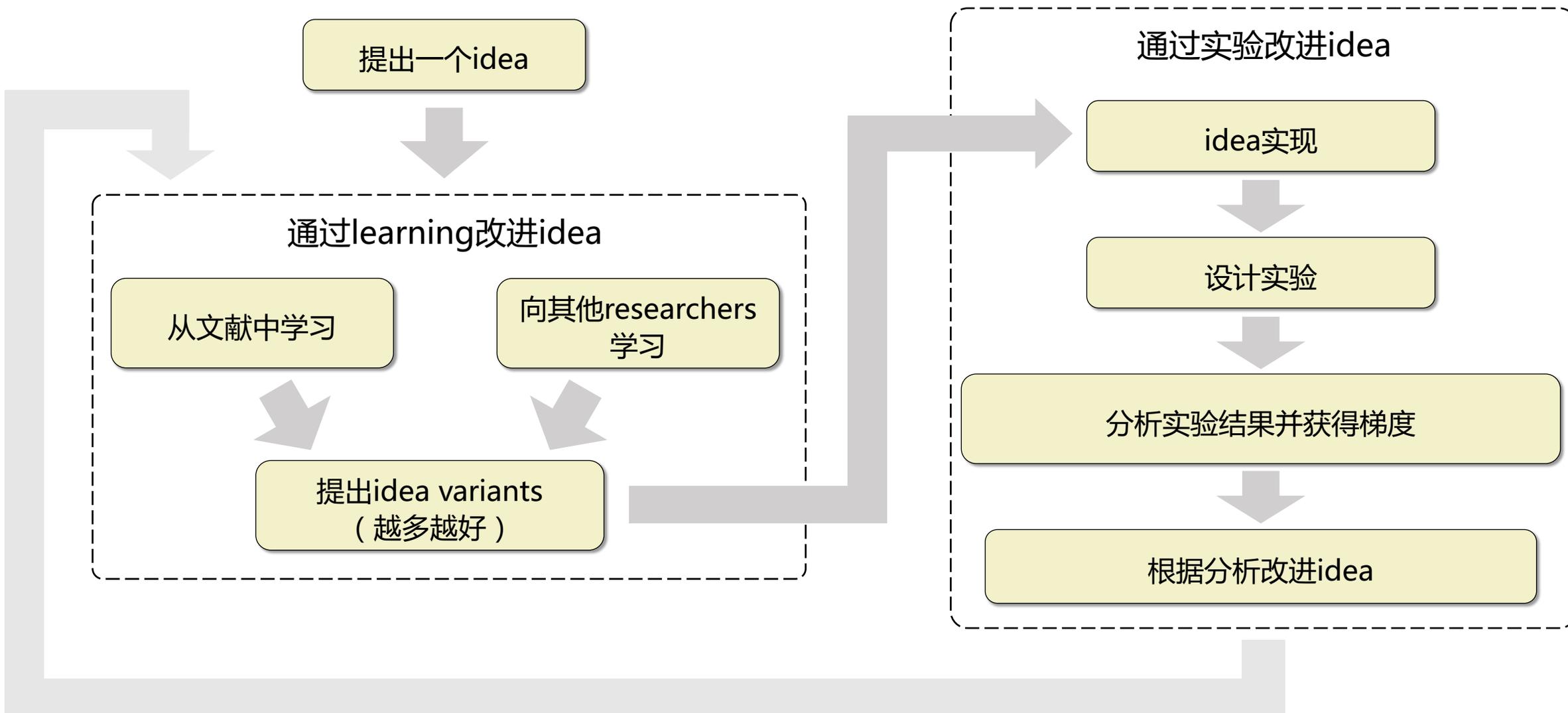


如何改进方法

- 可以把改进方法的过程当作SGD优化过程。



如何改进方法



步骤一：从文献中学习

提出一个idea

通过learning改进idea

从文献中学习

向其他researchers
学习

提出idea variants (越多
越好)

通过实验改进idea

idea实现

设计实验

分析实验结果并获得梯度

根据分析改进idea

为什么要学习related works

寻找相关技术论文的三个目的

检查novelty

是否有类似的文章？

获得技术insights

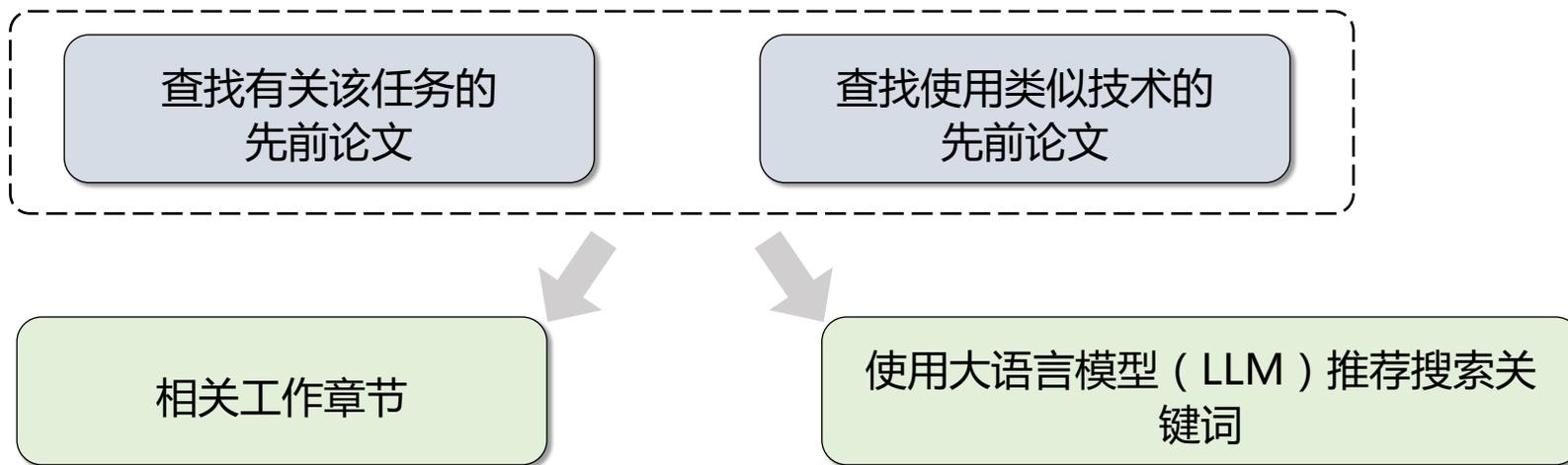
相关论文如何分析相关技术的insights和技术优势？

学习关键tricks

相关技术方法wrok的关键tricks是什么？

如何从相关工作中学习

- 首先，我们需要寻找相关技术的论文。



2. Related Work

Neural scene representations. In the domain of novel view synthesis, various approaches have been proposed to address this challenging problem, including multi-view image-based methods [5, 7, 16, 27, 63, 97], multi-plane image representations [41, 50, 58, 78, 80, 80], light-field techniques [13, 19, 35] as well as explicit surface or voxel-based methods [12, 14, 54, 55, 94]. [12] utilizes depth sensors and multi-view stereo techniques to consolidate per-view depth information into a coherent scene geometry, producing high-quality volumetric video. These methods require intricate hardware setups and studio arrangements, thus constraining their accessibility and applicability. Recently, implicit neural scene representations [3, 21, 26, 28, 45, 46, 51, 71, 74–76, 79, 86] have attracted significant interest among researchers. NeRF [51] encodes the radiance fields of static scenes using coordinate-based Multi-Layer Perceptrons (MLP), achieving exceptional novel view synthesis quality.

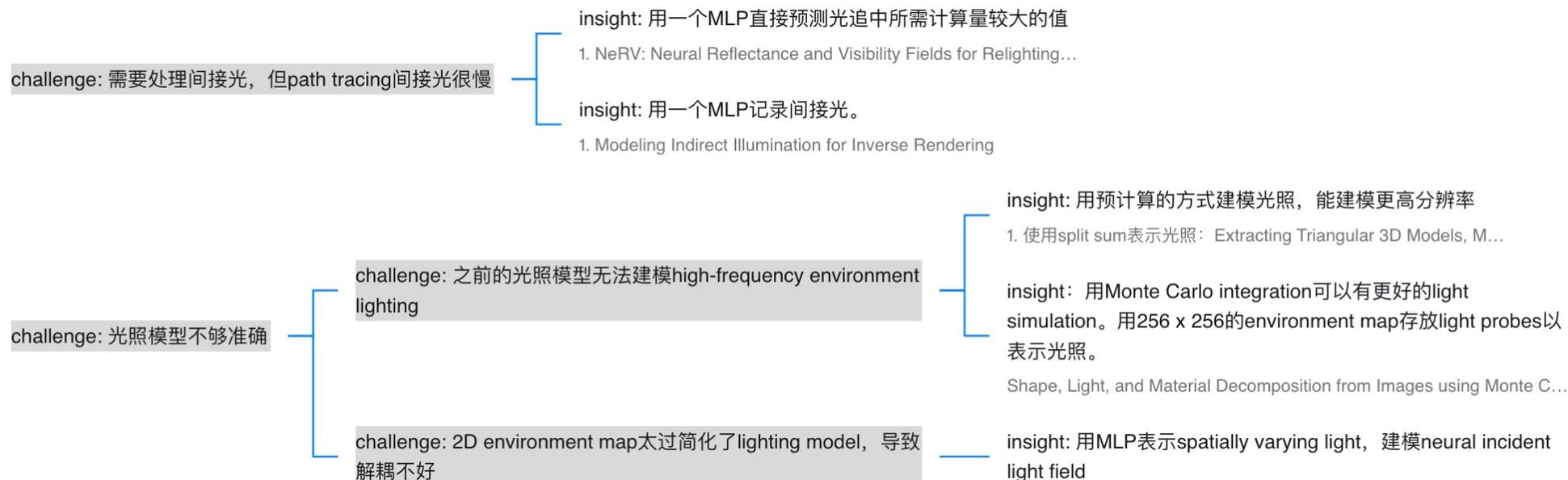
The screenshot shows a search interface with a blue header bar containing the text '如果想寻找使用3DGS技术的论文，应该使用什么关键词进行搜索' (If you want to find papers using 3DGS technology, what keywords should you use for search?). Below the header, a search bar contains the text '找到了 10 篇资料作为参考：' (Found 10 pieces of information for reference:). The search results are listed as follows:

1. "3D Gaussian Splatting"
2. "3DGS"
3. "3DGS technology"
4. "3D Gaussian distribution in rendering"
5. "3DGS for 3D reconstruction"
6. "3DGS in computer vision"
7. "3DGS in computer graphics"
8. "3D Gaussian Splatting papers"
9. "3DGS research"
10. "Gaussian Splatting 3D models"

Below the list, there is a note: '这些关键词可以帮助您在学术数据库、搜索引擎或会议论文集中找到相关的研究论文和技术文档。' (These keywords can help you find related research papers and technical documents in academic databases, search engines, or conference proceedings.) At the bottom, there are navigation options: '复制' (Copy), '再试一次' (Try again), and '分享' (Share).

如何从相关工作中学习

- 首先，我们需要查找相关技术论文。
- 然后我们阅读这些论文以了解相关技术的动机和技术优势。
- 一种有效的学习方法是通过challenge-insight tree总结这些论文。



如何从相关工作中学习

- 首先，我们需要查找相关技术论文。
- 然后我们阅读这些论文以了解相关技术的动机和技术优势。
- 一种有效的学习方法是通过对challenge-insight tree总结这些论文。

challenge: 需要处理间接光，但path tracing间接光很慢

insight: 用一个MLP直接预测光追中所需计算量较大的值

1. NeRV: Neural Reflectance and Visibility Fields for Relighting...

insight: 用一个MLP记录间接光。

1. Modeling Indirect Illumination for Inverse Rendering

如何总结论文：请参阅第一节课。

challenge: 光照模型不够准确

simulation。用256 x 256的environment map存放light probes以表示光照。

Shape, Light, and Material Decomposition from Images using Monte C...

challenge: 2D environment map太过简化了lighting model，导致解耦不好

insight: 用MLP表示spatially varying light，建模neural incident light field

如何从相关工作中学习

- 首先，我们需要查找相关技术论文。
- 然后我们阅读这些论文以了解相关技术的动机和技术优势。
- 一种有效的学习方法是通过对challenge-insight tree总结这些论文。

challenge: 需要处理间接光，但path tracing间接光很慢

insight: 用一个MLP直接预测光追中所需计算量较大的值

1. NeRV: Neural Reflectance and Visibility Fields for Relighting...

insight: 用一个MLP记录间接光。

1. Modeling Indirect Illumination for Inverse Rendering

我们通常不会阅读论文的代码，除非它与我们研究的内容非常相关。

challenge: 光照模型不够准确

simulation。用256 x 256的environment map存放light probes以表示光照。

Shape, Light, and Material Decomposition from Images using Monte C...

challenge: 2D environment map太过简化了lighting model，导致解耦不好

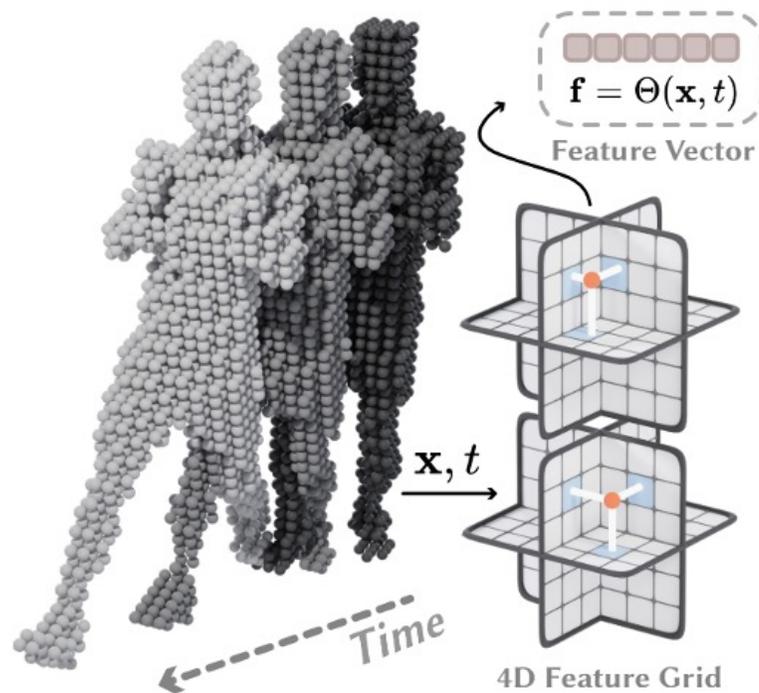
insight: 用MLP表示spatially varying light，建模neural incident light field

案例：4K4D – 论文任务



案例：4K4D

- 初始的Idea：发现在动态场景，Point Rendering还未被较多探索
- 设计方法：Point Cloud Sequence



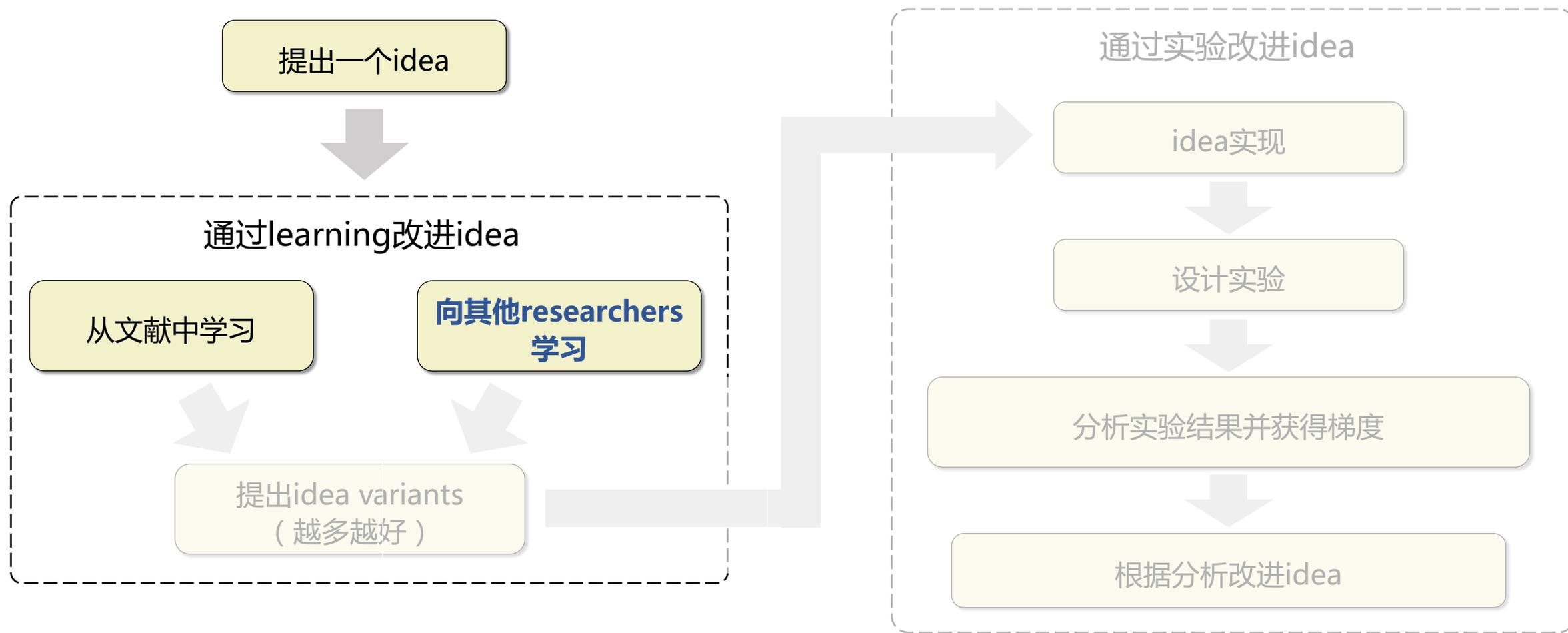
(a) Point Cloud Sequence



案例：4K4D

- 学习volumetric videos里面实时渲染的pipelines
- 学习static view synthesis里面实时渲染的pipelines
- 学习领域现有工作怎么用点云的、学习他们的技巧

步骤二：向其他researchers学习



与他人讨论的动机

与他人讨论的三个好处

找到机会组织你的idea

在与他人讨论时，必须清晰地组织你的思路。

获得对自己idea的反馈

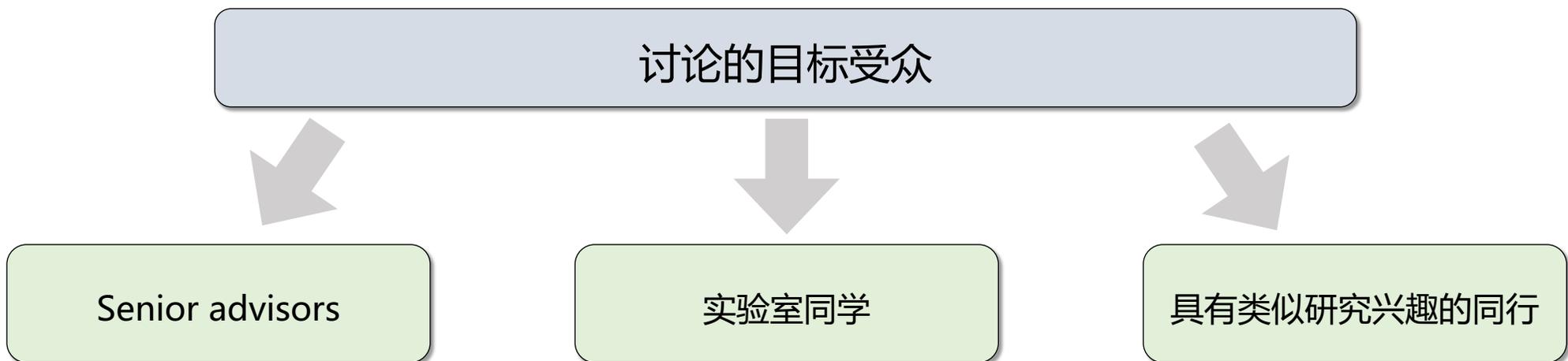
与他人讨论可以帮助你获得对自己idea的反馈，这包括潜在的问题和改进建议。

激发新的idea variants

有时，一次简单的对话可以引发新的问题或解决方案。

如何与他人讨论

- 首先，找到一个你可以讨论的群体。



如何与他人讨论

- 首先，找到一个你可以讨论的群体。
- 然后，通过报告来介绍你的问题和方法

讲清楚以下**五个要点**，实现有效且高效的讨论

任务设定

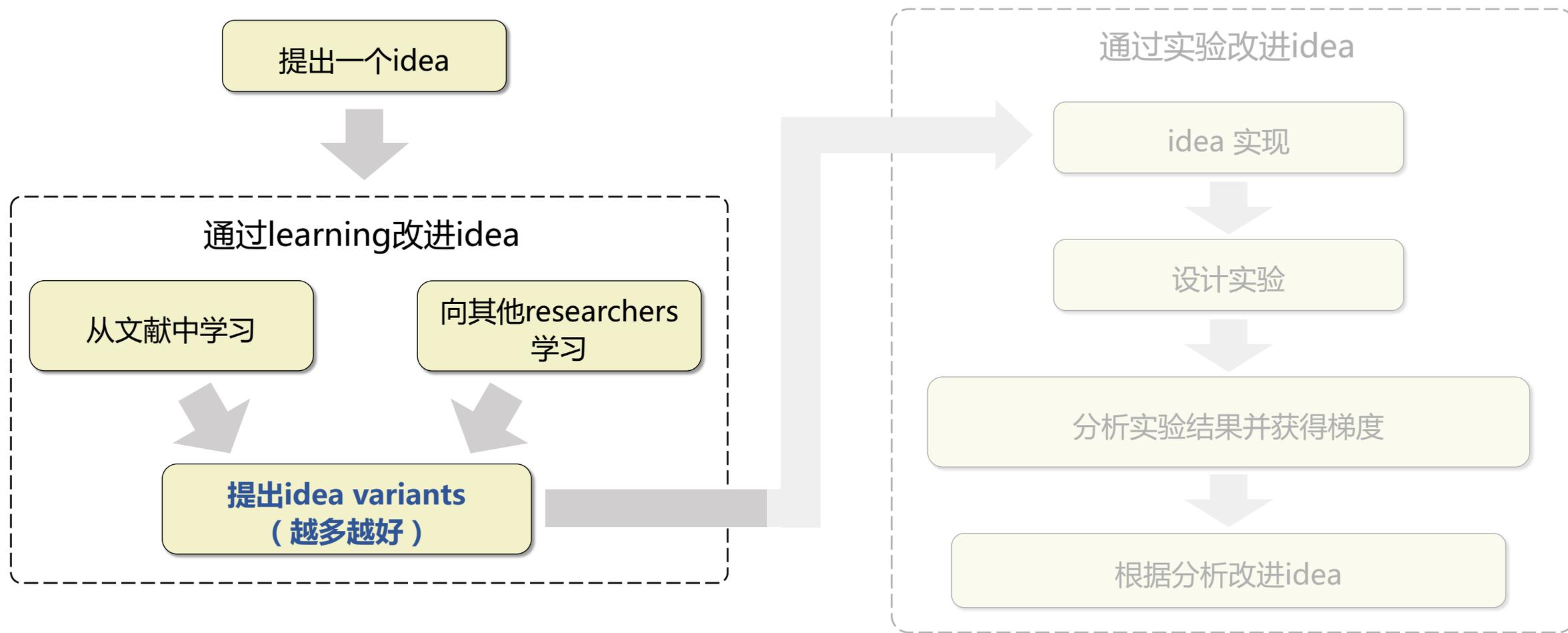
Failure case &
本质的技术原因

解决问题的
直觉、动机

粗略的
pipeline

想讨论的问题

步骤三：提出idea variants



思考更多idea variants的动机

思考更多idea variants的两个好处

避免陷入local minima

类似于优化过程，我们设定几个seed以防止陷入local minima，避免陷入惯性思维。

培养技术创造力

培养技术洞察力、技术创新能力和创造力。

思考更多idea variants的动机

思考更多idea variants的两个好处

避免陷入local minima

类似于优化过程，我们设定几个seed以防止陷入local minima，避免陷入惯性思维。

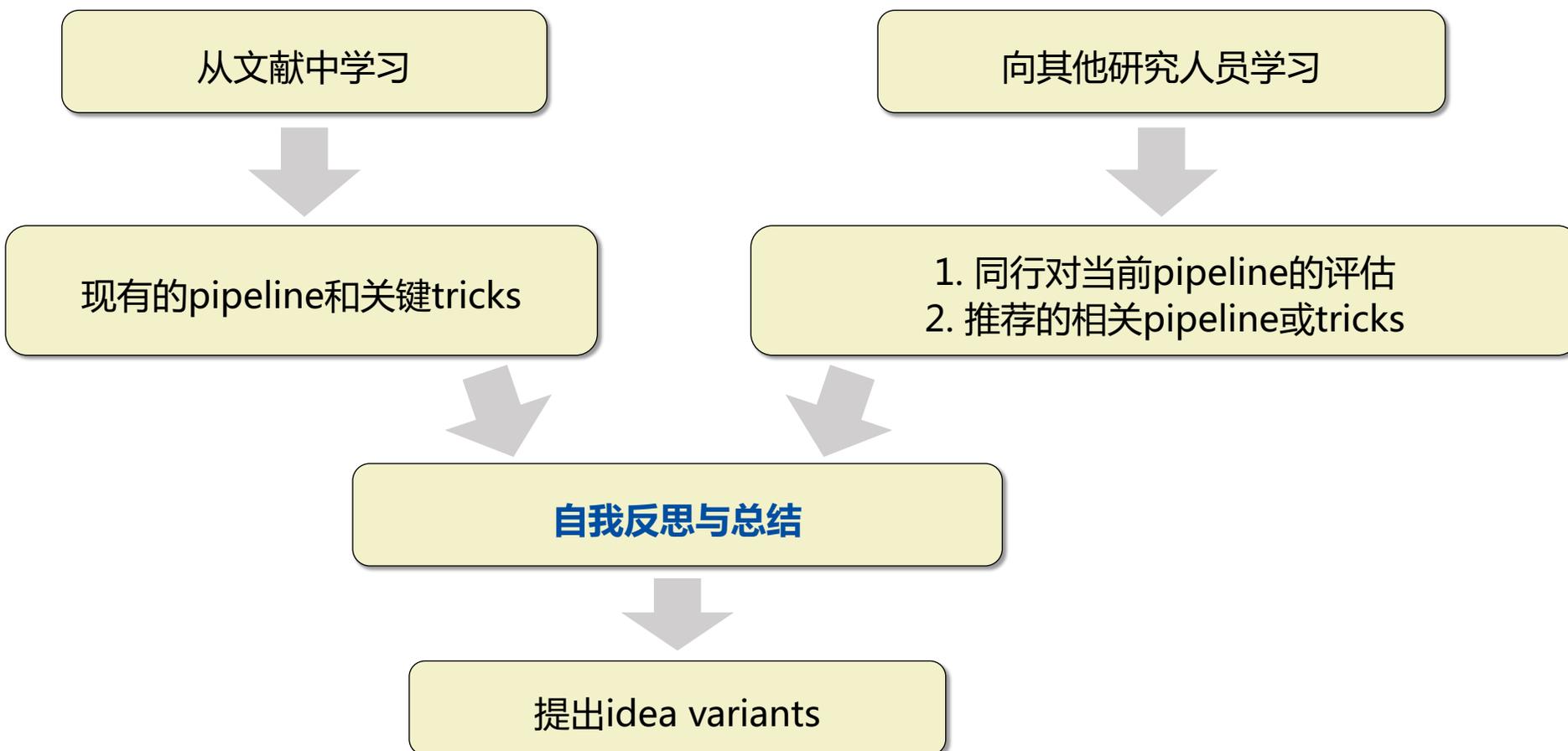
培养技术创造力

培养技术洞察力、技术创新能力和创造力。

一个常见的误区是直接使用导师提供的coarse pipeline进行实验，而不进行深入思考改进。

如何得到更多的idea variants

- 如何进行：根据文献和同行的反馈提出一些idea variants。



步骤四：idea实现

提出一个idea

通过learning改进idea

从文献中学习

向其他researchers
学习

提出idea variants
(越多越好)

通过实验改进idea

idea 实现

设计实验

分析实验结果并获得梯度

根据分析改进idea

如何高效地实现idea

基本做法：
在优秀的代码框架上实现你的idea

更好的做法：
最好在自己的代码框架内重构他人的代码

上述做法的三个好处

节约时间

与从头实现相比，重构他人的代码可以降低实现难度并避免陷入陷阱。

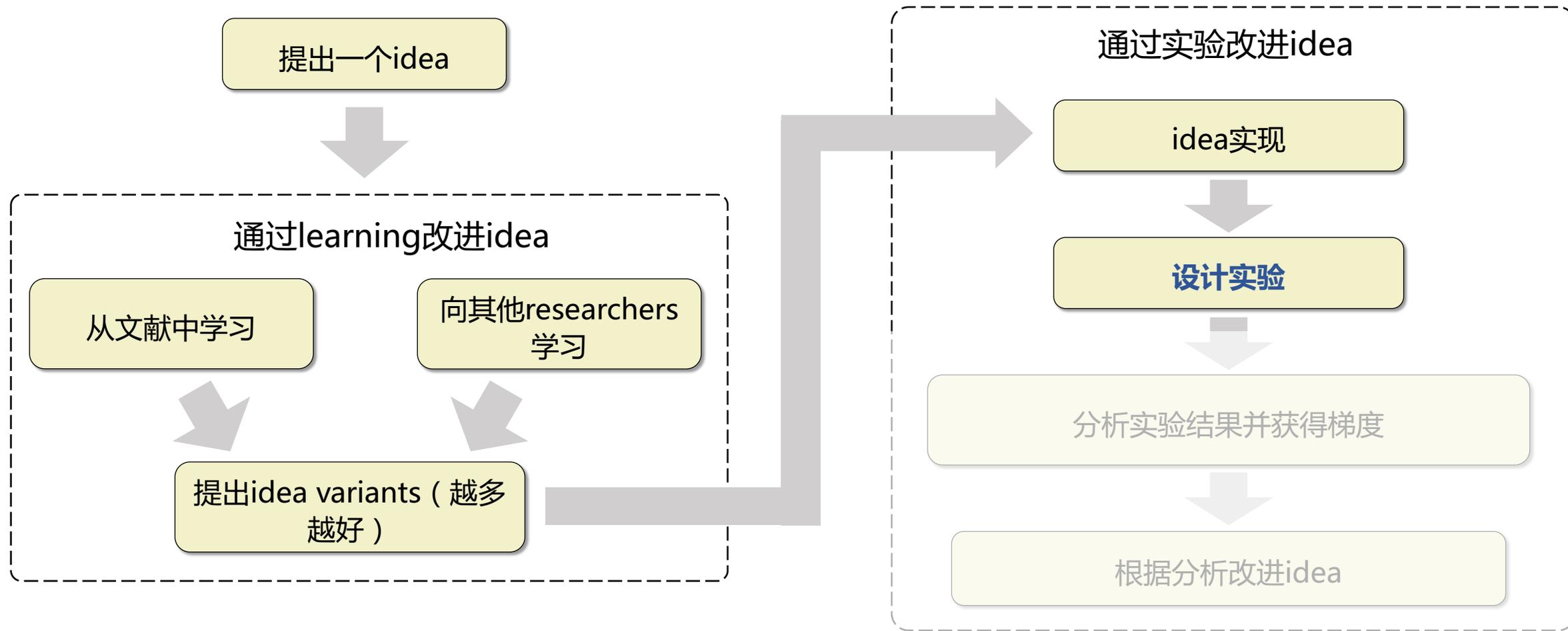
了解关键的tricks

在重构代码的过程中，你会加深对算法的理解，提高技术 insights，并学习到有效的 tricks。

为后续实验做好准备

使用自己的代码框架，便于未来的idea改进。

步骤五：设计实验



设计实验的动机

设计实验的两个好处

让research更容易

明确实验目的和科研方法。

让实验更容易

降低实验难度，提高实验效率。

如何设计实验（简单版）

- 核心原则：减少实验中包含的探索点的数量。

如何减少exploration points的数量

分解pipeline

将idea分解为不同的组件，从可控的idea开始，不断添加探索性和创新性的框架/模块。

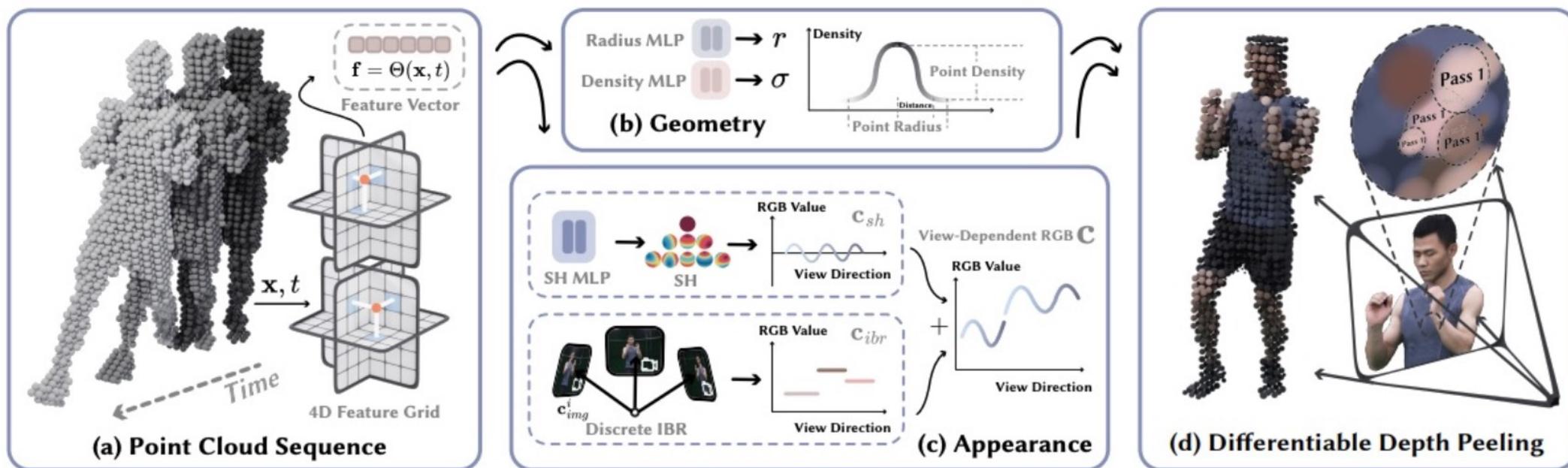
分解实验设置

从一个简单的setting开始探索，然后逐渐增加难度，之后进入到真正的setting。

我们还应该考虑exploration points的重要性，进行实验优先级排序

案例：4K4D

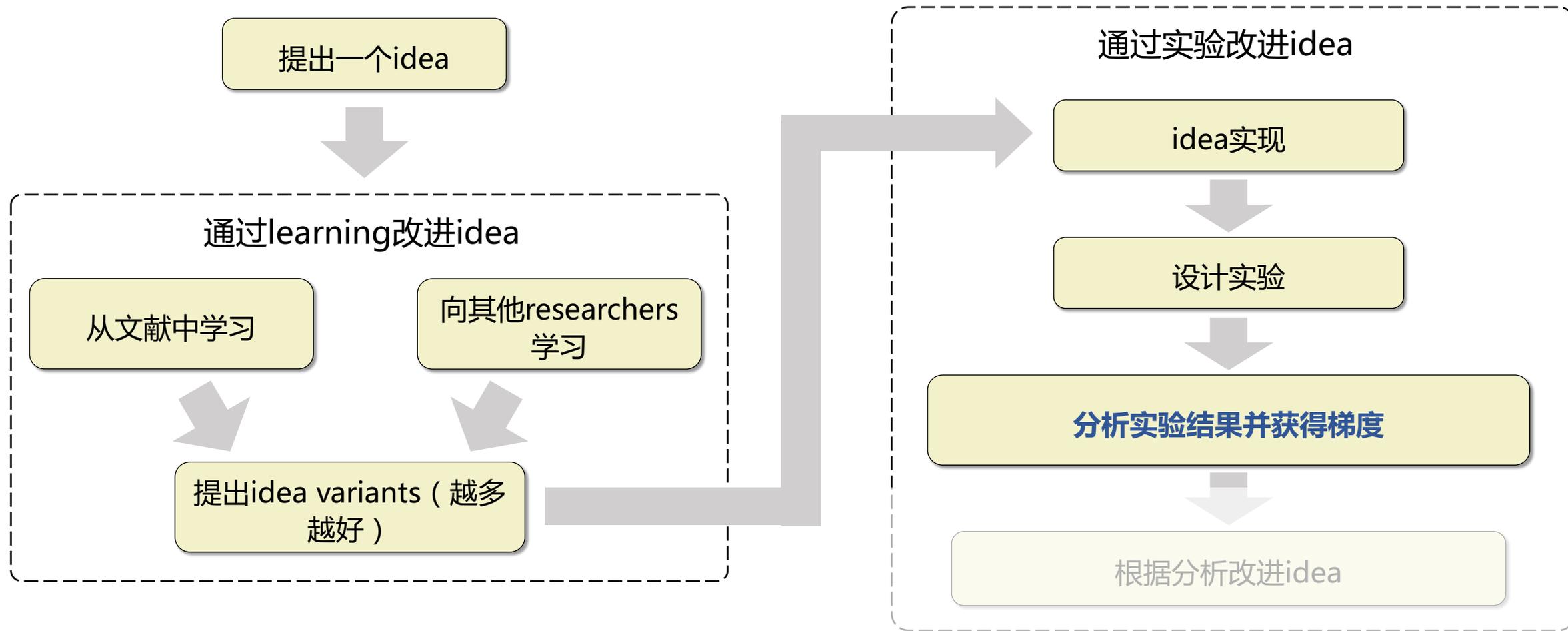
- 直接实现整个pipeline v.s. 先验证核心的一部分内容



问题拆解:

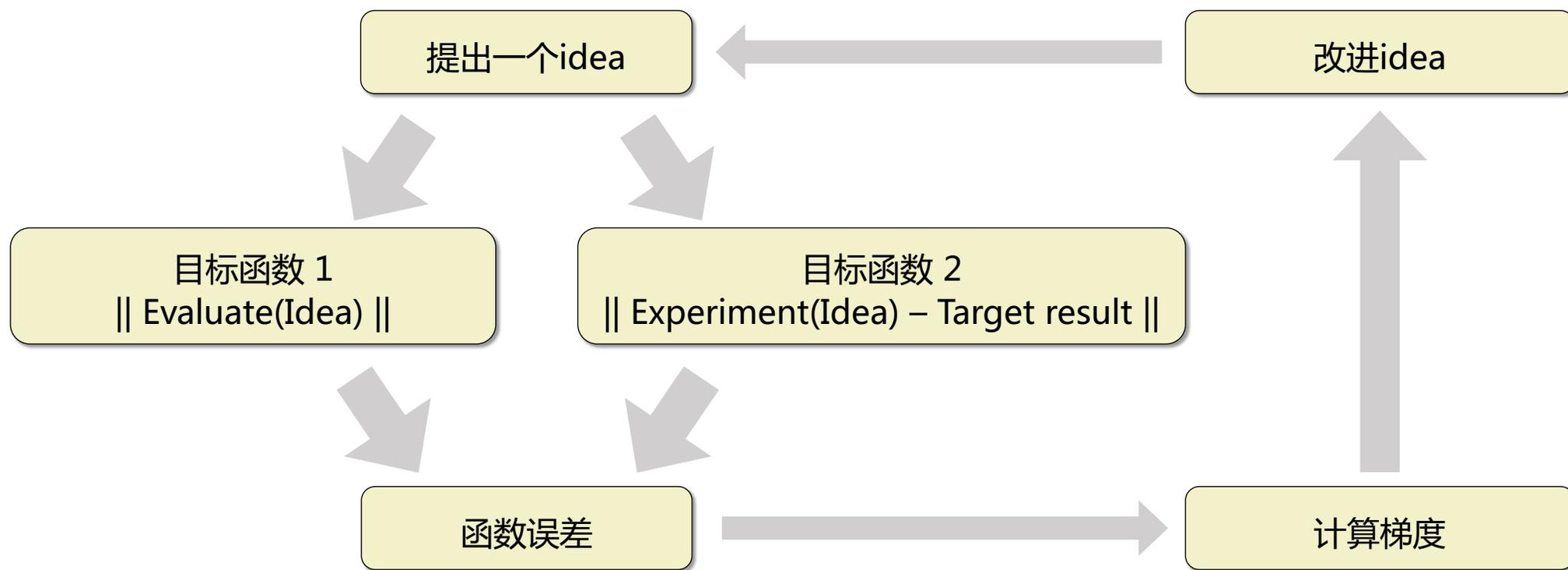
1. 4D feature grid
2. Geometry modeling
3. Appearance modeling
4. Depth peeling

步骤六：分析实验结果

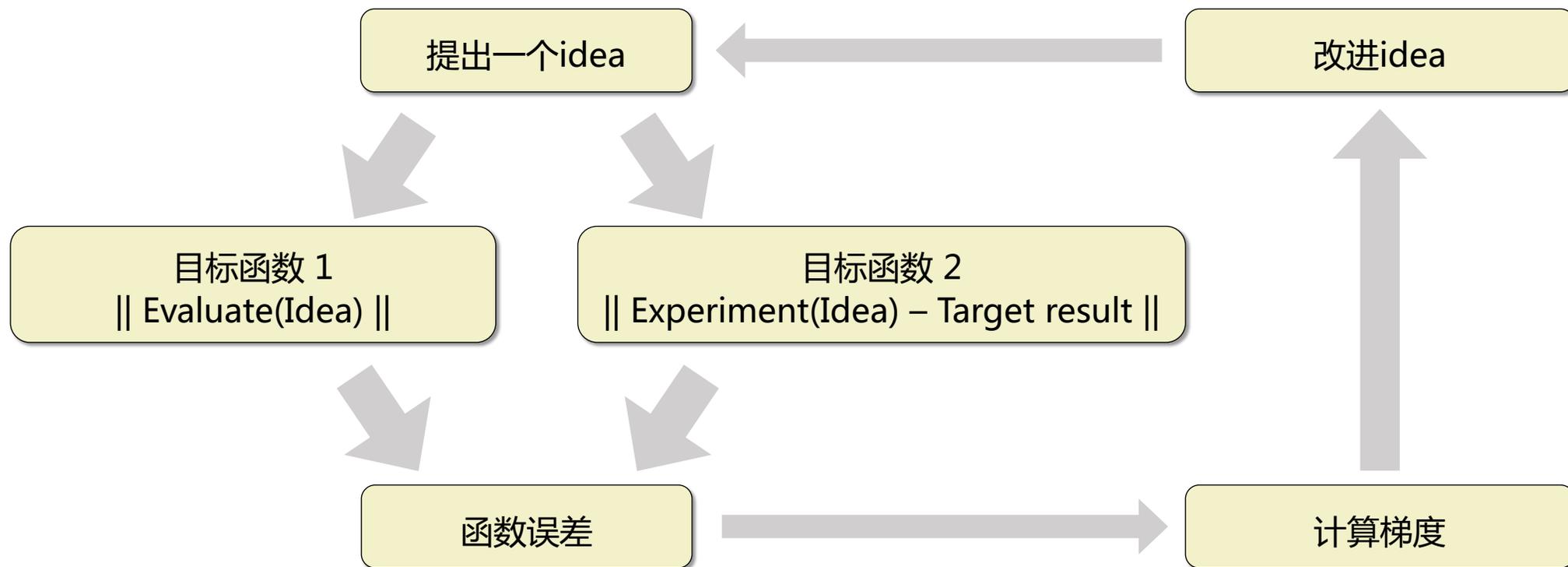


为什么需要分析结果

- 加快idea迭代改进的关键：
 - (1) 更好的迭代梯度。(2) 减少实验所花费的时间。



为什么需要加快idea更新过程



核心动机：避免项目被scooped

如何获得更好的梯度

- 首先，保持良好的实验记录习惯。

做好实验记录的五个要点

实验目的

描述为什么要做这个实验，同时你希望通过这个实验获得什么。

实验设置

实验中用了什么样的数据，同时对算法做出了哪些改变。

实验结果

记录成功和不成功的结果，其中包括量化结果和可视化的结果。

实验结果分析

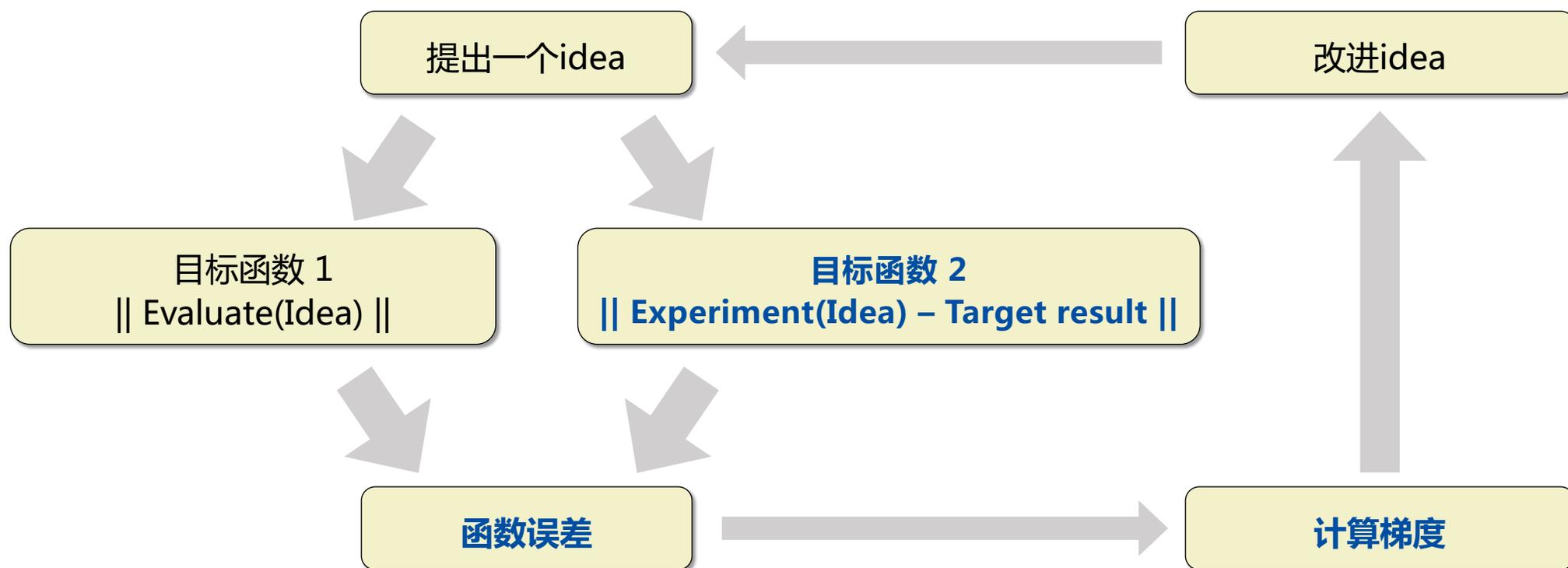
分析实验为什么work或者不work。

下一步骤

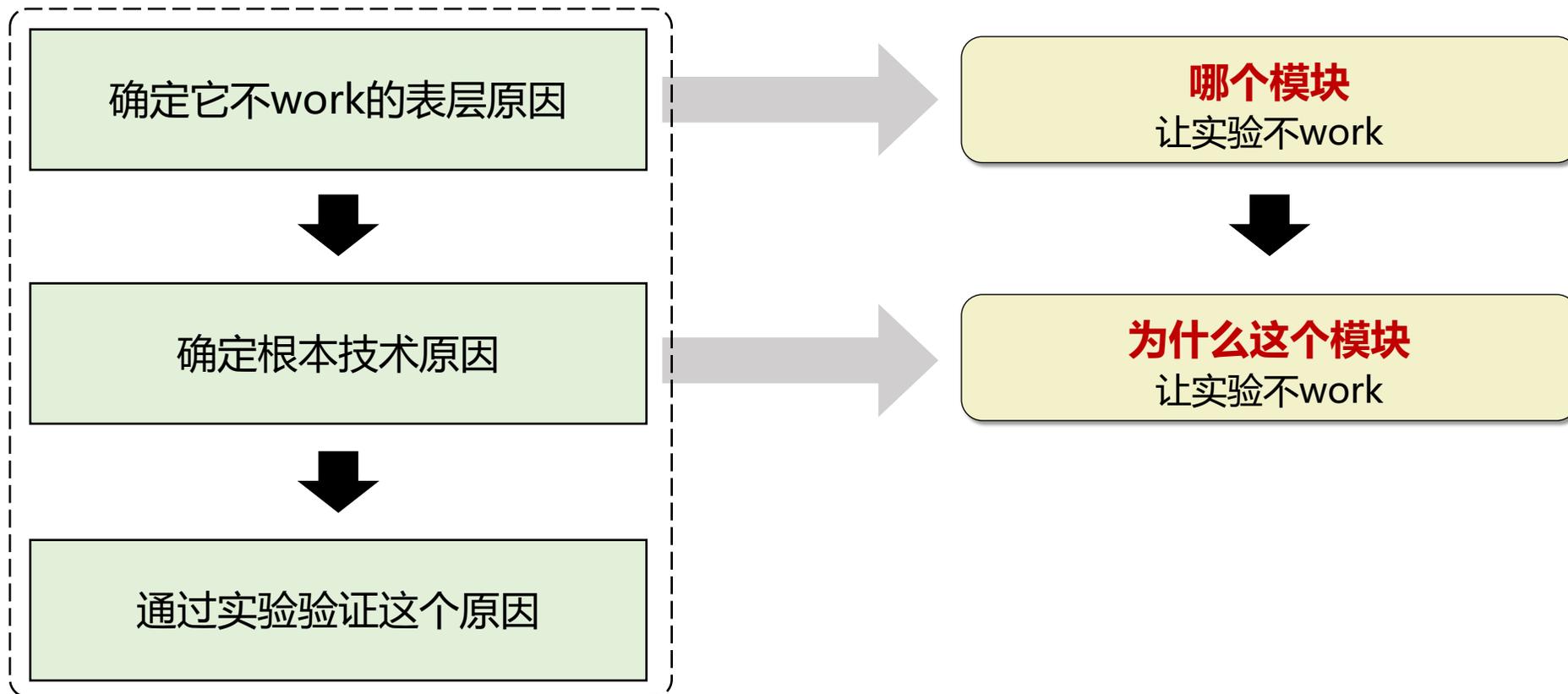
思考如何进行下一步并列出具体的实验。

如何获得更好的梯度

- 首先，保持良好的实验记录习惯。
- 然后，对实验结果进行分析。通常，需要分析为什么实验结果不满意的原因。

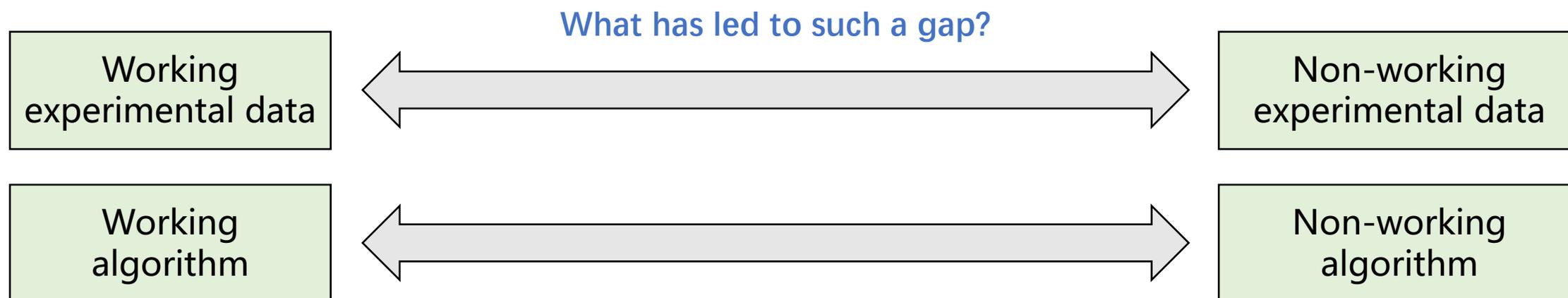


如何分析为什么实验不work的原因

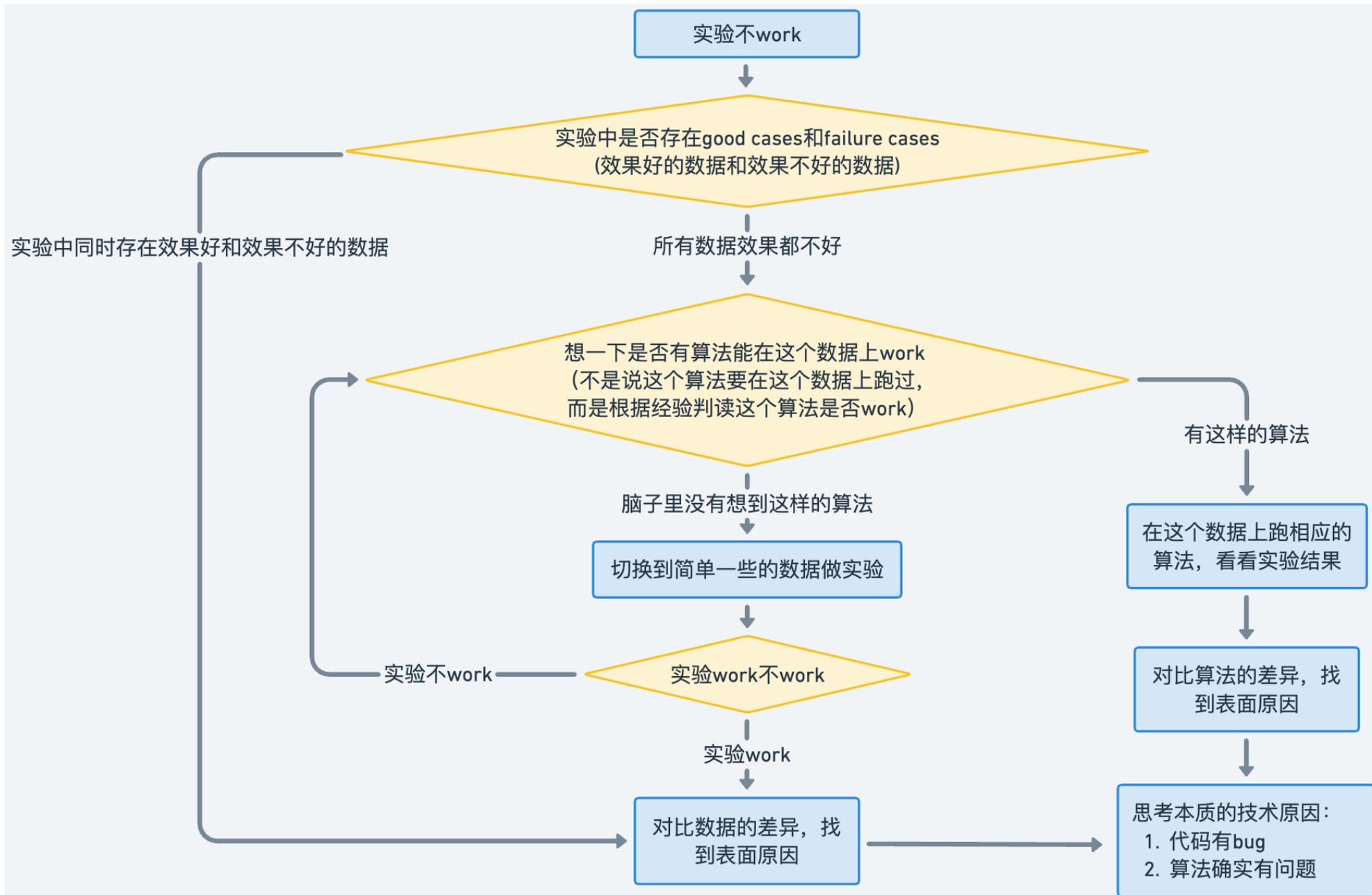


如何发现表层原因：二分查找方法

- 首先确认那些能够work的数据或算法, 然后将其与不work的数据或算法进行比较, 查看他们之间的差异。



如何发现表层原因：二分查找方法



如何发现根本技术原因

确定它不work的表层原因



确定根本技术原因



通过实验验证这个原因

为什么这个模块
让实验不work

为什么切换到某个数据集后
性能会下降？

为什么添加了某个模块后
性能会下降？

为什么改变了某个参数后
性能会下降？

尽可能地列出可能的技术原因



如何发现根本技术原因

这通常有两个技术原因：

1. 代码中有bug。
2. “xx 算法” 应用在 “xx 数据” 确实存在问题。

如何调试代码

步骤1：彻底理解算法/代码（调试的基础）



步骤2：确定能够持续重现bug的输入和设置



步骤3：基于二分查找法缩小bug的范围

TURING 图灵程序设计丛书

AMACOM

Debugging

The 9 Indispensable Rules for Finding

Even the Most Elusive Software and Hardware Problems

调试九法 软硬件错误的排查之道

[美] David J. Agans 著
赵俐 译



人民邮电出版社
POSTS & TELECOM PRESS

如何调试代码

在将bug锁定在适当的范围内后：

步骤4：逐行调试

- 观察操作的细节。
- 使用控制变量法最终确定bug。

TURING 图灵程序设计丛书

AMACOM

Debugging

The 9 Indispensable Rules for Finding

Even the Most Elusive Software and Hardware Problems

调试九法 软硬件错误的排查之道

[美] David J. Agans 著
赵俐 译

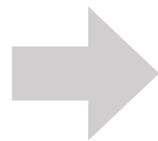


人民邮电出版社
POSTS & TELECOM PRESS

如何调试算法

步骤1：查明导致问题的模块

根据二元搜索方法缩小
可能存在问题的模块范围

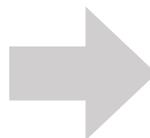


根据变量控制方法确定
导致问题的特定模块（消融实验）

如何调试算法

步骤1：查明导致问题的模块

根据二元搜索方法缩小
可能存在问题的模块范围



根据变量控制方法确定
导致问题的特定模块（消融实验）



步骤2：分析根本技术原因

依靠你积累的技术技能和经验
来推断原因

和他人讨论推断原因

如何与他人讨论实验结果

有效且高效讨论的五个要点

实验目的

描述为什么要做这个实验，同时你希望通过这个实验获得什么。

实验设置

实验中用了什么样的数据，同时对算法做出了哪些改变。

实验现象

能确定的
当前问题的范围

猜测的
可能导致的原因

询问是否有其他想法

分析实验结果的典型问题列表

代码调试检查表

各模块的中间输出结果

每个模块代码的中间输出结果是否满足预期？

代码bug的典型原因

积累一些导致代码bug的典型原因（如NaN值，段错误等）。

分析实验结果的典型问题列表

算法调试检查表（与数据相关的问题）

在简单数据上的表现

1. 在training data上的表现如何？
2. 在 xx data上的表现如何 (simpler one)?

两个数据集之间的差异

我们在 xx data有更好的结果。
为什么在这个数据上不能work？
这两个数据集之间有什么差异？

分析实验结果的典型问题列表

算法调试检查表（算法相关问题）

找到一个能够work的算法

我记得xx算法在这个数据集上的表现并不差。为什么我们的方法不work？这需要进行消融实验来检查哪个模块有问题。

检查各模块输出是否正确

我们是否可视化了xx模块的输出？检查输出结果是否满足预期或者是否有任何奇怪的现象发生。

分析实验结果的典型问题列表

算法调试检查表



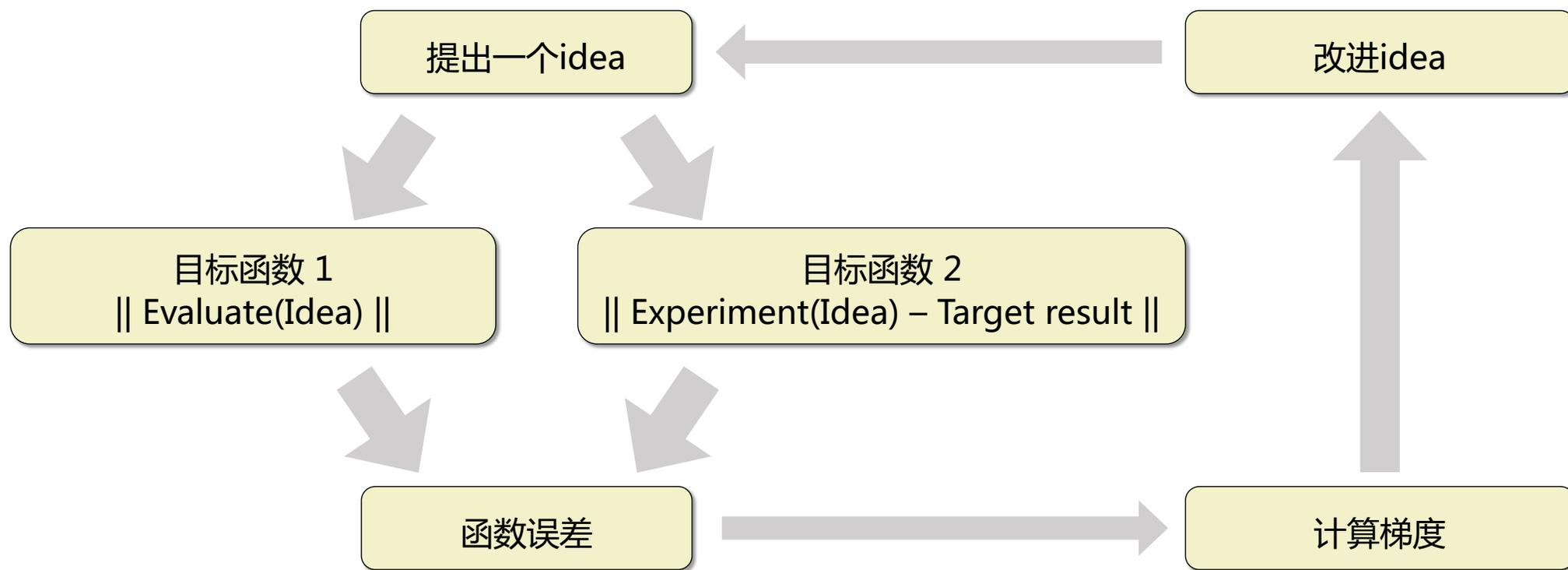
观察failure cases中是否存在特定的pattern ?



使用3D可视化工具进行调试 (如Viser、Wis3D)

如何减少实验时间

- 加速idea迭代改进的关键
 - (1) 更好的梯度。 (2) **减少实验时间。**



如何减少实验时间

减少实验时间的三个tricks

```
graph TD; A[减少实验时间的三个tricks] --> B[可扩展的代码框架]; A --> C[提高编程能力]; A --> D[并行进行实验];
```

可扩展的代码框架

提高编程能力

并行进行实验

步骤七：改进idea

提出一个idea

通过learning改进idea

从文献中学习

向其他researchers
学习

提出idea variants (越多
越好)

通过实验改进idea

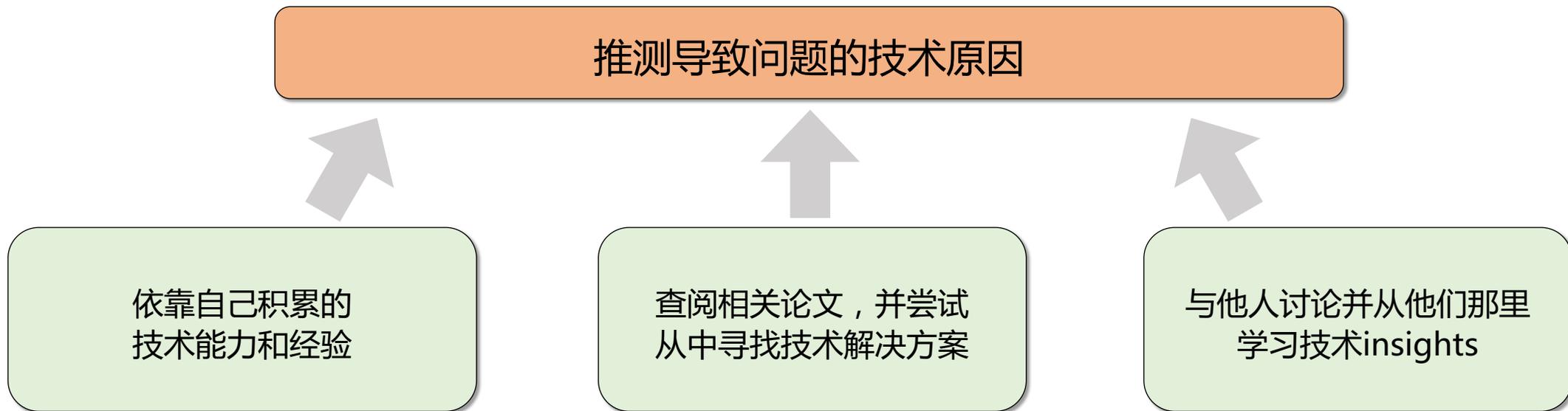
idea实现

设计实验

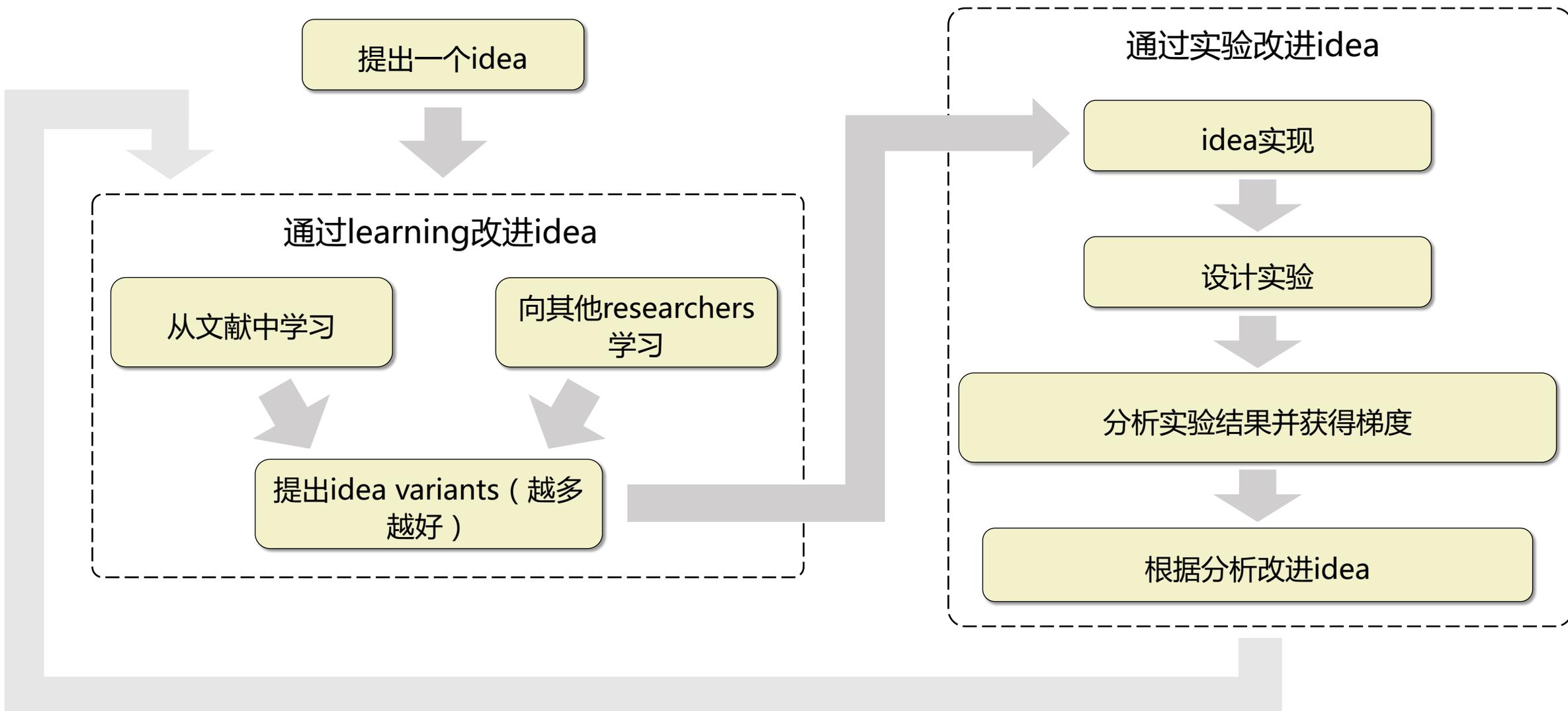
分析实验结果并获得梯度

根据分析改进idea

如何基于分析改进idea



总结如何改进idea



改进idea时需要关注的事情

需要记住三件重要的事情

不要轻易放弃当前的idea

不断质疑这个idea
是否存在根本缺陷

知道什么时候可以
放弃这个idea

为什么不轻易放弃一个idea

- 即使一个idea最初可能行不通，一个人也不应该轻易放弃当前的idea。

不轻易放弃idea的三个原因

最初的尝试通常不会work

第一次尝试可能不会成功，但这并不意味着这个idea本身是不可行的。这可能需要更多的实验和调整来找到正确的方法。

Research是一个迭代的过程

科学研究往往是一个反复迭代的过程。每一次失败都是一次学习和进步的机会。

这是一次获得technical insights的机会

在努力解决问题的过程中，你可能会对问题有更深入的了解，这有助于你完善idea或找到新的解决方案。

为什么要不断质疑idea，看看是否有根本缺陷

质疑idea的三个理由

避免陷入local minima

怀疑能够帮助我们避免盲目自信，保持批判性思维，从而更客观的评估自己的idea。

帮助我们确定问题

怀疑可以帮助我们发现潜在的问题或缺陷，允许在早期阶段进行纠正，并防止以后出现更大的损失。

推动我们的创新pipeline

质疑现有的想法可能会激发新的思考和创新，鼓励我们探索更有效或更先进的解决方案。

何时放弃一个idea

- 什么时候我们有足够的理由放弃一个idea

两个常见的理由

我们发现了当前idea的根本缺陷。

我们发现资源不足以支持这个idea
(计算资源、数据资源)



QA : Idea实现

- 构建自己的代码框架时，能不能用开源的、之前读论文看到的很好的框架，照搬过来作为自己的代码框架？



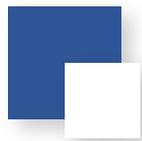
QA : Idea实现

- 如何整理、形成、和维护自己的代码框架？



QA : Idea实现

- 做实验时的代码与最终开源的代码在写的时候有什么区别吗？



QA：实验记录

- 是否有实验管理工具推荐？比如结合wandb、notion等进行实验记录



QA：实验记录

- 实验控制变量过多时如何保持记录的有序性？以我个人为例，有些变量很可能是中途加入的，导致较早的实验结果缺乏此变量的记录，回看记录时有时对其文件命名不知所云，对此是否有一些优化建议？



QA：实验记录

- 按照您的个人习惯，会丢弃一部分废弃或错误的实验结果吗？还是说这些结果未来也可能产生一定价值？



QA：实验效果调优

- 调参的大致思路是什么？



QA：实验效果调优

- 加入idea的实验结果已经跑出来了，和预期差不多。但看了实验结果，知道还有提升的空间，还需要继续改进吗？还是把继续改进的点，放到下一篇工作里？



QA：实验效果调优

- 分析实验结果、做ablation study时，有没有必要分析训练收敛的快慢？还是只需要看结果的一些指标？



谢谢！



彭思达



高俊



彭崧猷



王倩倩