

Gemini - Research Proposal Translation Request

 gemini.google.com/share/f213a46aa25e



通过其计算图验证思维链推理

翻译： 通过其计算图验证思维链推理 Zheng Zhao^{1,2,*}, Yeskendir Koishekenov¹, Xianjun Yang¹, Naila Murray¹, Nicola Cancedda¹ ¹ FAIR at Meta, ²爱丁堡大学 *工作于FAIR实习期间完成

解读： 这篇论文的标题揭示了其核心思想：研究人员想要通过检查人工智能（AI）进行思考时内部计算过程形成的“图”（Computational Graph），来判断它的“思维链”（Chain-of-Thought）是否正确。让我们来逐一拆解这些概念。

- **思维链 (Chain-of-Thought, CoT)：** 这就像你在解一道复杂的数学题时，不会直接写下答案，而是会写出详细的步骤：“第一步，根据题意列出方程... 第二步，化简方程... 第三步，解得 $x=...$ ”。对于大型语言模型（LLM）——比如你可能用过的那些高级聊天机器人——也是如此。当你让它解决一个复杂问题时，它会一步步地展示它的推理过程，这就是“思维链”。这让AI的思考过程看起来更透明。
- **计算图 (Computational Graph)：** 你可以把AI模型想象成一个极其复杂的电路板。当你给它一个问题，电流（信息）就会流过这个电路板的特定路径，经过成千-上万个计算单元（神经元），最终在输出端得到答案。这个电流流经的路径和所有被激活的计算单元，可以被描绘成一张网络图，这就是“计算图”。它本质上是AI为解决这一个问题所执行的“计算步骤示意图”。
- **验证 (Verifying)：** 意思就是检查、核对。

所以，整篇论文的目标就是开发一种新方法，通过观察AI大脑内部为生成某一步推理所画出的“电路图”，来判断这一步推理到底对不对。作者团队来自Meta的人工智能实验室（FAIR）和爱丁堡大学，这两个都是全球顶尖的人工智能研究机构，说明这项研究具有很高的前沿性。

摘要

翻译：当前的思维链（CoT）验证方法基于输出（黑盒）或激活（灰盒）来预测推理的正确性，但对于计算为何失败提供的洞见有限。我们引入一种白盒方法：基于电路的推理验证（CRV）。我们假设，正确的CoT步骤的归因图，可以被看作是模型潜在推理电路的执行轨迹，它拥有与错误步骤的归因图截然不同的结构性指纹。通过这些图的结构特征上训练一个分类器，我们表明这些轨迹包含了推理错误的强大信号。

解读：这里，研究者们首先指出了现有方法的不足，并用三个非常形象的术语来分类：

- **黑盒 (Black-box) 方法：**这就像老师批改你的试卷只看最终答案。比如答案是“42”，你写的是“42”，就判你对；你写的“43”，就判你错。这种方法很简单，但老师完全不知道你中间哪一步算错了，是审题不清还是计算失误。对于AI来说，就是只看它生成的文字结果。
- **灰盒 (Gray-box) 方法：**这好比老师不仅看你的答案，还能通过一个脑电波扫描仪看到你答题时大脑的活跃程度。老师可能会发现，你算错的时候，大脑某个区域的信号（激活值）特别强或特别弱。这提供了一些线索（比如你可能很困惑），但老师依然不知道你具体的思路错在哪里。对于AI，就是检查它内部神经元的“活跃度”。
- **白盒 (White-box) 方法 (本文提出的CRV)：**这是本文的核心创新。它相当于老师拿到了你的全部草稿纸，上面清清楚楚地写着你的每一步计算和思考过程。老师可以明确地指出：“哦，原来你在第三步把加号误用成了乘号。”对于AI，CRV方法就是去观察它内部计算的完整“草稿”，即“归因图”（attribution graph）。

作者的核心假设是：正确的推理过程在计算图上会留下一种“干净”、“规整”的模式，就像一个优等生清晰的解题步骤，他们称之为“结构性指纹”（structural fingerprints）。而错误的推理则会留下一张“混乱”、“异常”的图。他们训练了一个AI分类器，专门学习识别这两种“指纹”的区别，从而能高效地判断推理的对错。

翻译：我们的白盒方法产生了其他方法无法获得的全新科学见解。(1) 我们证明了错误的结构性签名具有高度的可预测性，从而确立了直接通过其计算图来验证推理的可行性。(2) 我们发现这些签名是高度领域特定的，揭示了不同推理任务中的失败会表现为截然不同的计算模式。(3) 我们提供的证据表明，这些签名不仅仅是相关性的；通过利用我们的分析来指导对单个转码器特征的定向干预，我们成功地纠正了模型的错误推理。我们的工作表明，通过审视模型的计算过程，我们可以从简单的错误检测，迈向对大语言模型推理更深层次的理解。

解读：这部分总结了他们研究的三个重大突破，这也是这篇论文最大的贡献：

1. **证明了可行性：**他们证明了“通过观察计算草稿来判断对错”这个想法是完全行得通的。错误的“指纹”确实非常明显，可以被AI分类器准确地识别出来。

2. **发现了领域特定性**：这是一个非常深刻的发现。它意味着，AI在解决数学题时犯错所留下的“计算痕迹”，和它在解决逻辑题时犯错的痕迹，是完全不同的。这强烈暗示了AI的“大脑”里，并不是只有一个通用的“推理引擎”，而是进化出了许多专门处理特定任务的“小工具”或“子电路”。就像你的大脑里有专门处理语言的区域，也有专门处理视觉的区域一样。
3. **实现了因果干预**：这是最令人兴奋的一点。他们不仅仅是“看到”了错误（发现相关性），比如“每次算错，这个神经元就很活跃”。他们更进一步，能够主动“动手”去修复错误（建立因果性）。这就像一个电路工程师，不仅用万用表检测到某个元件电压异常，还能直接更换或调整这个元件，从而让整个电路恢复正常。他们通过精确调整计算图中的某个“特征”，成功地让原本会犯错的AI走上了正确的推理道路。

总而言之，这项研究的意义从“给AI当裁判，判断对错”，提升到了“给AI当医生，诊断病因，甚至能对症下药”，这对于提升AI的可靠性和安全性至关重要。

1 引言

翻译：思维链（CoT；Wei et al., 2022; Kojima et al., 2022）提示已被证明是提升大型语言模型（LLMs）性能的强大方法。这一能力现在是最新一代推理模型的核心，例如DeepSeek-R1（DeepSeek-AI et al., 2025）和OpenAI的o1（OpenAI et al., 2024）。尽管取得了这样的成功，但在这些系统的整个范围内仍然存在一个根本性的漏洞：推理过程本身有时是有缺陷的（Turpin et al., 2023; Li et al., 2025b; Arcuschin et al., 2025; Lindsey et al., 2025; Chen et al., 2025b）。

解读：引言部分首先肯定了“思维链”（CoT）技术的巨大成功。CoT是当前让AI变得更“聪明”、更会推理的关键技术之一。所有顶尖的AI模型，比如DeepSeek和OpenAI的最新模型，都依赖于它。然而，作者紧接着指出了一个严峻的现实：即使是最先进的AI，在展示其一步步的推理过程时，这些步骤也常常会出错。这个“可靠性差距”（reliability gap）是一个亟待解决的核心问题，因为如果我们希望AI在医疗、金融、科研等关键领域发挥作用，我们就不能容忍它在推理过程中犯低级错误。

翻译：这个可靠性差距推动了对自动化验证的研究。当前的方法主要分为两大类。黑盒方法分析生成的文本或最终的logit分布（Jacovi et al., 2024; Wang et al., 2025b; Baker et al., 2025）。灰盒方法则观察模型的内部状态，使用简单的探针探测原始激活值，或分析隐藏状态的轨迹（Xie et al., 2025; Zhang et al., 2025; Afzal et al., 2025; Bi et al., 2025; Wang et al., 2025a）。虽然这些方法有其洞见，但它们在根本上是受限的；它们能检测到模型的内部状态与错误相关，但无法解释为什么底层的计算会导致错误。

解读：这里再次详细阐述了现有验证方法的局限性，并使用了之前在摘要中提到的分类：

- **黑盒方法**：它们只看最终的输出文本，或者模型在生成最后一个词时的概率分布（logit distribution）。这就像只看学生写的最终答案，完全不关心过程。
- **灰盒方法**：它们会深入一层，去观察模型内部的“神经活动”（激活值或隐藏状态）。这就像通过脑电波监测来判断学生是否在思考上遇到了困难。你能看到“异常信号”，但你不知道这个信号具体对应着哪一个逻辑错误。

作者强调，这两种方法的根本缺陷在于，它们只能建立“相关性”（correlation），而不能揭示“因果性”（causality）。它们能告诉你“当模型犯错时，通常会伴随着某个内部信号的出现”，但无法回答“是不是这个信号的出现 导致了模型的错误”。

翻译：这一局限性激发了一种更深入、更机械化的方法。我们假定，模型通过专门的子图或“电路”（Olah et al., 2020; Elhage et al., 2021）来实现解决特定任务的潜在算法。从这个角度来看，推理失败不仅仅是一个错误的状态，而是潜在算法执行过程中的一个缺陷。要诊断这类缺陷，就需要检查底层的计算过程，类似于在经典软件中检查执行轨迹。我们建议通过构建一个归因图（Dunefsky et al., 2025）来近似这个轨迹——这是一个表示模型组件之间因果信息流的结构化表示。

解读：为了突破“相关性”的瓶颈，作者引入了一个核心理论——**机械可解释性 (Mechanistic Interpretability)**。这个理论的核心思想是，尽管神经网络看起来像一个神秘的“黑箱”，但它内部其实是由许多像小程序一样的、功能明确的“电路”组成的。例如，模型中可能有一个专门负责“加法”的电路，一个负责“识别句子主语”的电路等等。

基于这个理论，一个推理错误就不再是一个抽象的“坏状态”，而是一个非常具体的问题：某个“电路”在执行时出了 bug。这就像调试传统计算机程序一样。要想找到这个 bug，就不能只看最终结果，而必须去查看程序的“执行轨迹”（execution trace），也就是程序一步步运行的日志。在AI模型中，这个“执行轨迹”的近似物就是作者提出的“归因图”（attribution graph），它描绘了信息在模型内部是如何一步步传递并相互影响的。

翻译：为了使这样的图成为一个有意义的轨迹，其组件必须是可解释的。因此，我们首先通过用训练好的转码器（Dunefsky et al., 2025）替换其标准的MLP模块，来创建一个可解释的替代模型。然后，我们构建并分析这个替代模型中稀疏激活特征的归因图（Ameisen et al., 2025）。最后，为了正式检验这些轨迹是否包含可检测的错误信号，我们在它们的结构属性上训练一个诊断分类器。这整套我们称之为基于电路的推理验证（CRV）的方法，因此被设计成一个科学仪器，用以研究我们的中心假设：推理失败会在其计算执行轨迹上表现为可检测的结构性签名，而这些签名可被用于自动化验证。

解读：这里详细描述了实现“白盒”方法的具体技术路线，也就是CRV的三个关键步骤：

- 1. 使模型可解释：**正常AI模型的内部组件（比如MLP模块）是“密集”且难以理解的。作者的第一步是进行“模型手术”，用一种叫做“转码器”（transcoder）的特殊组件来替换它们。转码器的一个关键特性是“稀疏激活”，意味着在任何时候，只有少数几个特征是活跃的，并且这些特征通常代表了某种可以理解的含义。这就像把一个嘈杂的会议室（MLP）变得井然有序，每次只有少数几个人（稀疏特征）发言，让你能清楚地追踪对话的流向。
- 2. 构建归因图：**在这个“可解释”的新模型上，他们就可以构建归因图，追踪那些活跃的、可解释的特征是如何共同作用产生最终结果的。
- 3. 训练分类器：**最后，他们把归因图的“结构”（比如图中节点的数量、连接的紧密程度、活跃特征在哪一层等）转换成一系列数字，然后训练一个简单的AI（诊断分类器）来学习识别代表“错误”的图结构模式。

CRV整套方法被定位为一个“科学仪器”，它的目标不只是为了开发一个工具，更是为了验证一个科学假设：AI的推理错误会在其底层的计算结构上留下可识别的“疤痕”。

翻译： 作为一个科学仪器，CRV需要一个受控的实验环境。虽然先进的推理模型采用了像搜索和回溯这样的复杂机制，但它们复杂的推理路径可能会掩盖单个推理步骤的基本计算。因此，我们的工作专注于标准的、经过指令微调的模型，它们自回归地生成CoT，因为这种范式为我们提供了一个更清晰的窗口，来观察构成涌现推理的原始计算。

解读： 作者在这里解释了为什么他们选择使用相对简单的模型和设置进行研究。更高级的AI模型可能会有非常复杂的策略，比如“思考多种可能性，然后选择最好的一条”（搜索）或者“发现走错了就退回上一步重来”（回溯）。这些复杂的行为会把底层的计算过程搞得非常混乱，就像在一团乱麻中寻找一根特定的线。为了能看得更清楚，他们选择了一个更“纯净”的实验对象：一个标准的、一步接一步进行推理（自回归生成）的模型。这让他们能够专注于研究最基础、最“原始”的计算单元是如何工作的，以及它们是如何出错的。

翻译： 尽管我们的方法有效，但计算成本过高，不适合作为一种实用的、即插即用的验证器，但它产生了其他方法无法获得的全新科学见解。因此，我们的主要贡献不仅仅在于性能，更在于理解：

- 我们引入了基于电路的推理验证，这是一种用于分析推理失败的白盒方法，表明通过其计算图来验证推理是可行的。
- 我们发现错误的结构性签名是高度领域特定的，揭示了在执行不同推理任务时的失败会表现为截然不同的计算模式。
- 我们确立了这些错误签名的因果作用，通过对单个转码器特征进行定向干预，成功地纠正了错误的推理。
- 为了支持未来的研究，我们将发布带有CoT推理步骤级正确性标签的数据集，涵盖合成任务和真实世界任务，以及我们训练好的转码器。

解读： 这段话再次强调了CRV的定位和贡献。作者非常坦诚地指出，这个方法目前还只是一个“实验室工具”，计算量太大，不能直接用到实际产品中。它的价值在于它带来的“科学理解”。这里重申了摘要中提到的三大贡献：

1. **证明可行性：** 验证了通过计算图找错这条路是走得通的。
2. **揭示领域特定性：** 发现了AI内部存在针对不同任务的“专用电路”。
3. **建立因果性：** 成功地通过“手术”修复了推理错误，而不仅仅是诊断。

最后，他们承诺将他们的数据和工具开源，供其他研究者使用。这是科学研究中非常重要的一环，它能推动整个领域的共同进步。

2 问题定义与预备知识

2.1 问题陈述

翻译： 让一个LLM生成一个思维链 $S=(s_1,s_2,...,s_m)$ 来解决一个问题，其中每个步骤 s_i 是一个token序列。在生成步骤 s_i 的过程中，底层模型会产生一个计算状态 M_i 。从这个状态中，我们构建一个归因图 $G_i=(V,E)$ ，其中顶点 V 代表可解释的特征和token，边 E 代表它们之间的因果影响（见3.2节）。从每个图 G_i 中，我们提取一个固定大小的特征向量 $x_i=\phi(G_i)$ ，其中 ϕ 是一个旨在捕捉图结构属性的特征提取函数。我们称这个向量为该步骤的**结构性指纹**。我们的目标是学习一个诊断分类器 f_θ ，它将这个结构性指纹作为输入，来预测该推理步骤的正确性：

$$y^i=f_\theta(x_i)$$

其中 $y^i \in \{\text{correct}, \text{incorrect}\}$ 。

解读： 这部分用数学语言精确地定义了他们要做的事情，可以看作是整个研究任务的“官方说明书”。让我们一步步把它翻译成通俗的语言：

1. **输入：** AI的解题步骤 S ，被拆分成一个个小步骤 s_i (例如，“第一步： $5+9=14$ ”)。
2. **生成轨迹：** 对于每一步 s_i ，他们会捕捉到AI模型内部的“计算快照” M_i 。
3. **构建图：** 这个“快照” M_i 会被转换成一张网络图 G_i 。图中的点 (V) 是构成计算的“零件”（比如输入的数字“5”、“9”，以及被激活的“加法”特征），图中的线 (E) 则表示这些“零件”之间的相互影响。
4. **提取指纹：** 他们用一个函数 ϕ 来“扫描”这张图 G_i ，并把它总结成一串数字 x_i 。这串数字就是“结构性指纹”，它描述了这张图的形状和特性（比如“这张图很稀疏”、“这张图主要用了第10层的零件”等等）。
5. **进行分类：** 他们的最终目标是训练一个分类器（一个简单的AI模型） f_θ 。这个分类器的工作就是接收“指纹” x_i ，然后给出一个判断 y^i ：这个步骤是“正确”的还是“错误”的。

这个公式 $\hat{y} = f(\theta)(\overline{x_i})$ 简洁地概括了整个流程：**根据计算图的结构指纹来预测推理步骤的正确性**。这成功地将一个非常抽象的“判断推理对错”的问题，转化成了一个经典且可解的机器学习分类问题。

2.2 预备知识：Transformer中的电路

翻译： 在机械可解释性领域，“电路”一词指的是神经网络内部一个特定的子图，它实现了一个人类可以理解的算法（Olah et al., 2020）。在Transformer（Vaswani et al., 2017）中，这些电路由注意力头和MLP计算组成。我们的工作概念上受到寻找能够区分推理中健全和有缺陷的电路激活模式的启发。虽然我们的方法不直接观察这些电路，但我们的假设是，它们会在我们构建的归因图上投下可检测的结构性指纹。因此，我们后续分析的一个主要目标是，将那些对预测失败最有用的基于图的特征，解释为这些底层错误模式的签名。

解读： 这部分解释了他们研究背后的核心理论基础。

- **什么是“电路”？** 在AI研究中，“电路”不是指物理的电线，而是一个比喻。它指的是AI庞大神经网络中，负责执行某个特定、可理解任务的一小部分网络结构。比如，可能有一个专门负责“复述事实”的电路，或者一个负责“执行乘法”的电路。这个概念让我们不再把AI看作一个无法理解的整体，而是看作由许多功能明确的“模块”组成的系统。
- **CRV与电路的关系：** 作者坦言，他们的方法（CRV）还不能像电路图一样精确地“看到”这些电路本身。但是，他们相信，当一个电路正常工作或出现故障时，它会在模型的整体计算活动中留下痕迹。他们构建的“归因图”就像是地震后地面上留下的裂缝，虽然不是地震本身，但通过分析裂缝的形状和分布（结构性指纹），可以推断出震源的位置和强度（底层电路的状态）。因此，他们的目标就是通过分析这些“痕迹”，来反向推断出是哪个“电路”以及如何出的问题。

2.3 预备知识：用于可解释特征的转码器

翻译： 分析模型激活值的一个重大挑战是它们的高维度和缺乏直接的可解释性。应对这一挑战的一个强有力的方法是，使用稀疏自编码器（SAE; Cunningham et al., 2023）为这些激活值学习一个稀疏的、过完备的基。一个SAE被训练用来从一个维度高得多但大部分为零的特征向量 $f \in \mathbb{R}^D$ (其中 $D \gg d$) 重构一个激活向量 $x \in \mathbb{R}^d$ 。 f 的元素对应于一组学习到的、可解释的、被输入稀疏激活的特征。虽然标准的SAE目标是重构其自身的输入 ($f(x) \approx x$)，但我们的工作利用了一种被称为转码器（transcoder; Dunefsky et al., 2025）的变体，它被训练来近似一个目标组件（如MLP）的输入-输出函数 ($f(x) \approx \text{MLP}(x)$)。这种方法使得转码器成为原始模块真正意义上的功能替代品。它的目标不仅仅是重构，而是在一个可解释的、稀疏激活的基上模拟一个计算步骤。通过用训练好的转码器替换模型的标准MLP模块，我们迫使它的中间计算不再由一个密集的向量表示，而是由这些有意义的特征的稀疏组合来表示。

解读： 这部分解释了他们实现“白盒”方法的关键技术工具——**转码器 (transcoder)**。

- **问题：** AI的内部计算（激活值）通常是一个由成千上万个数字组成的“密集向量”（dense vector），比如 $[0.12, -0.87, 0.54, \dots]$ 。这些数字本身没有任何直观含义，就像一串天书，这就是“缺乏可解释性”。
- **解决方案：稀疏表示。** 稀疏自编码器（SAE）或转码器的目标，就是把这个密集、难懂的向量，转换成一个“稀疏”且“可解释”的向量。这个新向量维度可能更高，但绝大部分元素都是0，只有少数几个元素有值，例如 $[0, 0, 0.98 \text{ (代表'加法')}, 0, 0, \dots]$ 。这里的 0.98 就代表“加法”这个特征被激活了。
- **转码器 vs. 标准SAE：** 一个标准的SAE只是学习如何编码和解码信息（像一个压缩和解压工具）。而“转码器”更进一步，它不仅要学会表示信息，还要学会**执行与原模块（MLP）相同的计算功能**。它是一个功能性的替代品。
- **核心作用：** 通过用转码器替换掉模型中那些不透明的MLP模块，研究者们就完成了了一次关键的“手术”。这迫使模型在思考时，必须使用这些稀疏的、有意义的特征来进行计算。这使得模型的内部思维过程从一团乱麻的数字，变成了一系列清晰、可追踪的“特征激活事件”，为后续的归因图分析铺平了道路。

3 方法论

翻译：与过程奖励建模（PRM）的目标仅限于判断一个推理步骤的正确性不同，我们站在模型开发者的角度，他们对自己拥有完全访问权限的特定模型中的推理失败进行调试感兴趣。我们引入了基于电路的推理验证（CRV），一种通过分析其结构性指纹来检测有缺陷的推理的方法。

解读：这里作者明确了他们研究的出发点和定位。过程奖励建模（Process Reward Modeling, PRM）是一种常见的方法，它训练一个模型来给AI的每一步推理打分，目标是“判断对错”。而CRV的目标更进一步，它不仅仅是当一个“裁判”，更是要成为一个“调试工具”（debugger）。它面向的是AI模型的开发者，帮助他们不仅知道“哪里错了”，更重要的是理解“为什么错了”，从而能够修复模型。CRV的核心手段就是通过分析计算过程的“结构性指纹”来诊断问题。

图1 CRV流程图

翻译：!(https://i.imgur.com/your_image_url.png) 图1 CRV流程图。(1) LLM的MLP模块被逐层转码器（PLTs）替换，使其变得可解释。(2) 对于一个给定的CoT步骤，我们生成一个归因图，捕捉可解释特征和模型组件之间的因果流。(3) 从这个图中提取结构特征，并(4) 将其输入诊断分类器，以预测该步骤的正确性。

解读：这张图是整篇论文方法论的核心总结，它清晰地展示了CRV的四个步骤：

- 第一步：用转码器替换MLP模块。** 这是“模型手术”阶段。图的左侧展示了一个标准的LLM结构，其中每一层都有一个黑盒的MLP模块。在第一步中，这些MLP被换成了透明的、可解释的“逐层转码器”（Per-Layer Transcoders, PLTs）。从此，信息流必须经过这些由稀疏特征构成的“关卡”。
- 第二步：构建归因图。** 当模型进行一步推理时，研究者会追踪那些被激活的稀疏特征以及它们之间的联系，从而构建出一张网络图，即归因图。这张图就像一张详细的“信息流地图”，显示了从输入到输出，信息是如何在不同层的不同特征之间传递的。
- 第三步：提取图特征。** 这张“信息流地图”本身还是太复杂，需要被量化。第三步就是对这张图进行“测量”，提取出描述其结构特性的关键数据，比如“全局统计量”（图有多大）、“节点统计量”（哪些节点最重要）和“拓扑特征”（图的连接方式是怎样的）。这些测量数据组合起来，就是该步骤的“结构性指纹”。
- 第四步：诊断分类器。** 最后，将这串代表“指纹”的数字输入一个预先训练好的分类器。这个分类器就像一个经验丰富的诊断医生，它看过成千上万个“正确”和“错误”的指纹，能够根据你提供的这个新指纹，给出一个诊断结果：“正确”或“不正确”。

这个流程图完美地将一个复杂的AI调试问题，分解成了一个清晰、有序的工程流程。

3.1 数据集构建和步骤级标注

翻译：开发我们方法的一个先决条件是一个拥有可靠的步骤级正确性标签的数据集。此外，我们的白盒方法论施加了一个关键要求，这使得我们的数据需求与先前的工作有所不同。由于CRV分析的是产生一个推理步骤的因果计算图，我们必须在生成过程中捕捉我们特定模型的完整内部状态。因此，现有的纯文本数据集，如PRM800K (Lightman et al., 2024) 和REVEAL (Jacovi et al., 2024)，它们提供静态的‘(文本, 标签)’对，并且是为训练黑盒验证器而设计的，与我们的机械化方法不兼容。我们必须生成并标注我们自己模型的CoT输出，以创建必要的‘(文本, 标签, 计算轨迹)’元组用于分析。因此，我们创建了一个新的基准，涵盖了受控的合成任务和真实的GSM8K数据集 (Cobbe et al., 2021)。

解读：这部分解释了为什么他们不能使用现成的数据集，而必须自己从头开始构建。

- **现有数据集的局限：**以前的数据集（比如PRM800K）只包含了“AI写的推理步骤文本”和“这个步骤是对是错的标签”。这对于训练一个只看文本的“黑盒”裁判是足够的。
- **CRV的特殊需求：**CRV方法需要的信息远不止于此。它不仅需要知道某一步错了，还需要知道在模型生成这一错误步骤的那一瞬间，它大脑内部的完整计算状态（即“计算轨迹”）。这就好比要诊断一种疾病，你不仅需要病人的症状描述（文本），还需要那一刻的CT扫描图（计算轨迹）。
- **解决方案：**他们只能自己动手，让他们的模型（一个经过“手术”的可解释模型）去做题，然后记录下每一道题、每一步推理的“文本”、“标签”以及最重要的“计算轨迹”，三者打包成一个数据单元。他们为此构建了两类数据集，以确保研究的全面性。

翻译：合成数据集（布尔和算术）。 为了在受控环境中研究推理失败，我们生成了两个数据集。第一个涉及评估复杂的布尔表达式，第二个涉及多步算术问题。这些数据集的动机是存在明确无误的基准真相：推理链中任何一步的正确性（例如，“ $15+7=22$ ”）都可以通过一个简单的解析器和评估器自动验证。这使我们能够生成一个大型、带标签的数据集用于初步的训练和分析。此外，这些任务本质上是组合性的，样本的复杂性可以完全控制。更多细节在附录A中提供。**GSM8K的步骤级标注。** 标注像GSM8K这样的真实世界数据集是具有挑战性的。为了规模化，我们使用了一个半自动化的过程，让一个更强的LLM（例如，Llama 3.3 70B Instruct）作为专家裁判。对于每个CoT，裁判会根据完整的题目背景评估步骤的正确性。我们通过对一个相当大的子集进行人工审查来验证这些标签，从而产生了一个用于真实世界推理的高保真度数据集。更多细节在附录A中提供。

解读：这里详细介绍了他们构建的两类数据集，体现了科学研究的严谨性：

1. **合成数据集 (布尔逻辑和算术)：**这就像在“无菌实验室”里做实验。

优点：对错一目了然，可以用程序自动、大规模、100%准确地进行标注。比如，程序可以轻易判断 $\$15+7=22\$$ 是对的，而 $\$15+7=21\$$ 是错的。这为训练模型提供了大量高质量、无歧义的数据。同时，他们可以精确控制题目的难度（比如，是3个运算符还是10个运算符），便于后续分析。

2. **真实世界数据集 (GSM8K)**：这是在“野外丛林”里做实验。GSM8K是一个著名的小学数学应用题数据集，它的推理步骤是用自然语言写的，更加复杂和模糊。

- **挑战**：无法用简单的程序来判断对错。比如，“首先，计算小明有的苹果数量”这一步，本身没有计算，但逻辑上可能是正确的。
- **解决方案**：他们采用“以AI治AI”的方法。他们请了一个更强大、更聪明的AI (Llama 3.3 70B) 来充当“专家裁判”，负责批改他们实验模型的作业。为了确保这个“AI裁判”靠谱，他们还找了人类专家对“AI裁判”的批改结果进行了抽查和验证，确保了标注质量。

通过结合这两种数据集，他们的研究既有在理想条件下的精确性，又经受了真实世界复杂性的考验。

3.2 基于电路的推理验证 (CRV)

翻译：CRV是一个四阶段的流程，旨在通过分析一个修改过的、可解释的LLM的计算图，来对CoT步骤的正确性进行分类。图1展示了其概览。

解读：这部分是对整个CRV方法的正式介绍，再次强调了其核心思想：通过分析一个经过改造的、透明化的AI模型的内部计算图，来判断其推理步骤是否正确。接下来的四个小节将详细拆解图1中展示四个步骤。

3.2.1 步骤1：用可解释的转码器替换MLP

翻译：CRV的基础是一项架构上的修改，使得目标LLM变得可解释。对于模型中的每个MLP模块，我们在从原始LLM中收集的大量、多样化的激活值数据集上训练一个相应的转码器。训练目标结合了L2重构损失和一个TopK激活函数，后者通过仅保留k个最大的特征激活来强制实现稀疏性。一旦训练完成，我们就用其对应的转码器替换LLM中每一层的MLP模块。模型的正向传播现在被迫流经这些稀疏的、可解释的瓶颈。所有后续的分析都在这个修改过的、可解释的替代模型上进行。转码器架构和训练的全部细节在附录B中提供。

解读：这是CRV的第一步，也是最关键的一步：“模型手术”。

- **目标**：把不透明的LLM变得透明。
- **手段**：逐个替换掉模型内部叫做MLP（多层感知器）的核心计算模块。
- **替代品**：转码器 (Transcoder) 。
- **转码器如何训练**：他们先让原始的LLM处理大量数据，并记录下每个MLP模块的输入和输出。然后，他们训练一个转码器，让它学习去模仿这个MLP的行为——给定相同的输入，产生尽可能接近的输出。

- **关键机制：稀疏性。** 在训练转码器时，他们加入了一个特殊规则——TopK激活函数。这个函数强制要求转码器在计算时，内部只能有k个（比如128个）最活跃的特征单元工作，其余的都必须“沉默”（值为0）。这就好比把一个嘈杂的百人讨论组，变成了一个每次只有少数几位关键人物发言的有序会议。
- **结果：** 经过替换后，模型在进行计算时，信息流必须通过这些由少数可解释特征构成的“瓶颈”。这使得追踪信息流向、理解计算过程成为可能。

3.2.2 步骤2：构建步骤级归因图

翻译： 有了我们注入了转码器的替代模型，我们需要一种有原则的方法来追踪信息流并构建计算的因果图。为此，我们改编了Dunefsky等人（2025）最近的电路分析方法。应用他们的贪婪寻路算法，使我们能够从最终的logits反向追踪高归因的连接，为每个推理步骤 s_i 产生一个稀疏的、加权的、有向的图 $G_i=(V,E)$ 。这个图代表了核心的计算子图，其中节点 V 是输入token、活跃的转码器特征和输出logits的不交并集。有向边 E 代表了这些组件之间的高归因因果路径（例如，从一个早期层的特征到一个后期层的特征，或者从一个特征到一个logit），其权重则量化了它们影响的强度。关于电路寻找算法的完整推导和描述，我们建议读者参考原始工作（Dunefsky et al., 2025）。

解读： 在完成了“模型手术”后，第二步是利用这个透明的模型来生成“计算草稿”——归因图。

- **方法：** 他们使用了一种先进的“反向追踪”算法。这个算法从模型最终的输出（logits，即模型预测下一个词的概率分布）开始，像侦探一样反向提问：“是哪些内部特征和输入词对这个输出的贡献最大？”
- **过程：** 算法会沿着贡献度最高的路径，一层一层地向后追溯，直到输入端。这个过程就像在复杂的电路板上，从亮起的灯泡（输出）出发，沿着最亮的电线（高贡献连接）往回找，直到找到源头的开关（输入）和沿途所有关键的元器件（活跃特征）。
- **结果：** 最终得到的是一张稀疏的网络图 G_i 。这张图非常关键，因为它不是模型全部计算的无差别记录，而是只包含了那些对当前推理步骤**最重要、最具因果关系的**计算单元和连接。图中的节点（ V ）包括了输入的文字、中间被激活的关键特征、以及最终的输出。图中的带箭头的线（ E ）则表示影响的方向和强度。

3.2.3 步骤3：提取可解释的图特征

翻译： 从每个归因图 G_i 中，我们提取一个固定大小的特征向量 x_i ，作为计算的结构性指纹。我们对图进行剪枝，只保留其最有影响力的组件，即保留那些对最终logits贡献达到一个阈值（例如，80%）的节点和边。特征集是在这个剪枝后的子图上计算的（除非另有说明），并被组织成三个层级。**全局图统计量：** 这些特征捕捉了计算子图的高层概括，包括剪枝后活跃特征节点的数量，以及最终的logit概率和熵。它们提供了对计算复杂度和不确定性的粗略度量。**节点影响力和激活统计量：** 这组特征量化了可解释特征节点的属性。我们计算它们激活值和影响力得分的统计数据（均值、最大值、标准差）。这有助于区分是由少数几个高度活跃、决定性的特征驱动的计算，还是由许多微弱特征的弥散组合驱动的计算。我们还包括了一个按层级划分的活跃特征直方图，它描述了推理步骤的计算深度。**拓扑和基于路径的特征：** 为了分析信息流的结构，我们在剪枝后的子图上计算了一套丰富

的拓扑特征。这些包括图密度、中心性度量（度、介数）以识别计算枢纽，以及连通性指标。这套全面的特征集为我们的诊断分类器提供了基础。每个特征的完整列表和详细动机在附录C.1中提供。

解读：有了归因图这张“计算草稿”，第三步就是把它转化成可供机器学习模型使用的数字“指纹”。

1. **剪枝 (Pruning)：**首先，他们会进一步简化归因图，只保留最重要的80%的影响力路径，把一些次要的“噪音”连接去掉，让分析聚焦于核心计算。

2. **特征提取：**然后，他们从三个维度来“测量”这张剪枝后的图：

- **全局统计量：**就像从宏观上看一张城市地图，记录下它的总面积、总人口等。这里就是测量图里总共有多少个节点、模型对自己的答案有多自信（logit概率和熵）等。
- **节点统计量：**就像分析城市里的各个区，看哪个区最繁华、影响力最大。这里就是分析图中的特征节点，比如它们的平均“活跃度”是多少？有没有一个“超级活跃”的节点？这些活跃的节点主要分布在模型的浅层还是深层？
- **拓扑特征：**就像研究城市的交通网络，看它是四通八达还是拥堵不堪。这里就是分析图的连接结构，比如图的“密度”（连接有多密集）、哪些节点是“交通枢纽”（中心性）、信息从输入到输出的平均路径有多长等。

这三类特征组合在一起，就形成了一个全面的、多维度的“结构性指纹”向量 $\mathbf{x}_i$ ，它从不同层面描绘了这一次计算过程的形态。

3.2.4 步骤4：诊断分类器

翻译：对于最后的分类步骤，我们使用一个在提取的特征向量上训练的梯度提升分类器（GBC）： $f_{\theta}(\mathbf{x}_i) = y^i$ 。GBC适合我们这种异构的、表格化的特征，并能提供鲁棒的特征重要性度量，我们利用这一点来识别错误电路中最具预测性的结构属性。我们还在附录C.3中与其他几个备选分类器进行了基准比较。

解读：这是CRV流程的最后一步，也是将分析结果转化为最终判断的一步。

- **工具：**他们选择了一种叫做“梯度提升分类器”（Gradient Boosting Classifier, GBC）的机器学习模型。这种模型非常擅长处理像他们提取的“结构性指纹”这样的表格类数据（即一堆不同类型的数字）。
- **任务：**这个GBC分类器的任务很简单：输入一个步骤的“指纹”向量 $\mathbf{x}_i$ ，输出一个预测结果 y^i ，即“正确”或“不正确”。
- **额外好处：**GBC模型还有一个非常有用的特性，就是它能告诉我们，在所有输入的特征中，哪些特征对于做出最终判断最重要。这就好比医生诊断后，不仅告诉你“你生病了”，还能告诉你“主要是因为你的A指标和B指标异常”。这对于后续深入理解错误原因（即第4.3节的“机械化分析”）至关重要。

4 实验

翻译： 我们进行了一系列实验，旨在验证我们工作的中心假设：推理步骤的归因图包含着关于其正确性的丰富、结构化的信号。我们的评估围绕三个主要研究问题展开。首先，我们研究CRV的白盒方法在验证准确性上是否显著优于一套全面的灰盒和黑盒基线，并测试其在领域转换和任务难度增加时的鲁棒性（RQ1）。接下来，我们分析我们训练好的模型，以识别图中最能预测失败的具体计算结构，从而从检测转向机械化理解（RQ2）。最后，我们进行探索性研究，以评估这些机械化见解是否可以用于执行有针对性的、因果性的干预，从而纠正错误的推理（RQ3）。

解读： 这一章是论文的核心——用实验数据来证明他们的想法。整个实验设计得非常有条理，围绕着三个递进的研究问题（Research Questions, RQ）展开：

1. RQ1：它好用吗？（性能和鲁棒性）

- **核心问题：** CRV方法是不是真的比现有的黑盒、灰盒方法更准？
- **测试内容：** 在不同数据集上进行准确率大比拼。同时，还会测试它在面对新类型问题（领域转换）和更难问题时，表现是否依然稳定。

2. RQ2：它为什么好用？（机理分析）

- **核心问题：** 在那些“结构性指纹”中，究竟是哪些特征在“报警”？错误的计算图到底长什么样？
- **测试内容：** 深入分析分类器模型，找出最重要的错误指示特征，并尝试将这些特征可视化，让我们能直观地看到“正确”和“错误”的计算图在结构上的差异。

3. RQ3：我们能用它做什么？（因果干预）

- **核心问题：** 既然我们知道了错误的原因，我们能“动手”修复它吗？
- **测试内容：** 进行案例研究，尝试通过手动干预模型内部某个被CRV识别为“有问题”的特征，看看是否能让模型“改邪归正”，做出正确的推理。这是检验他们是否真正理解了因果关系的关键一步。

这三个问题从“是什么”（What）到“为什么”（Why）再到“怎么办”（How），构成了一个完整而有力的科学论证链条。

4.1 实验设置

翻译： **模型和数据集。** 我们的实验在Llama 3.1 8B Instruct模型（AI@Meta, 2024）上进行。我们选择了指令微调变体，因为其遵循提示的优化对于可靠地引出我们分析所需的CoT推理轨迹至关重要。然后，如第3节所述，用我们训练的转码器对该模型进行修改。我们在我们的三个数据集上评估性能：合成（布尔）、合成（算术）和标注过的GSM8K基准。**基线。** 我们将CRV与两类基线进行比较。首先，是使用最终logit分布的黑盒方法：最大Softmax概率（MaxProb）、困惑度（PPL）、熵、温度缩放（Temp. Scaling）和能量（Energy）。其次，是操作内部状态的灰盒方法。这包括分析跨层隐藏状态动态的基于轨

迹的方法，如嵌入链（及其真实空间CoE-R和复数空间CoE-C变体）和CoT动力学（CoT-Kinetics），以及一个在步骤的平均隐藏状态上训练的标准逻辑回归探针（LR Probe）。虽然CoE和CoT-Kinetics最初是为完整的CoT评估设计的，但它们被证明是强大的步骤级基线。所有实现细节都推迟到附录C.2。 **评估指标。** 我们使用AUROC、FPR@95和AUPR来评估验证器的性能。由于我们的目标是检测推理失败，因此在所有指标计算中，我们将“不正确”标签视为正类。AUROC评估该方法在不同阈值下对正确与不正确步骤的排序能力。AUPR捕捉了正类（不正确）的精确率-召回率权衡。FPR@95衡量在95%的正类被正确识别时，假阳性率是多少，反映了在严格条件下的可靠性；较低的分數表示验证器可以在产生最少错误警报的情况下检测到大多数错误。这些指标共同提供了对性能的互补视角。

解读： 这部分详细说明了实验的“游戏规则”，确保实验是公平和可复现的。

- **实验对象：** 他们选用了Meta的Llama 3.1 8B Instruct模型，这是一个经过“指令微调”的模型，意味着它很擅长听从指令、按部就班地进行CoT推理，这正是研究所需的。当然，这个模型已经被他们用转码器“改造”过了。
- **参赛选手：**
 - **主角：** CRV（白盒方法）。
 - **对手1（黑盒队）：** 一系列只看最终输出概率的传统方法，如MaxProb（看模型对最可能的词有多自信）、PPL（看模型对生成的句子有多困惑）等。
 - **对手2（灰盒队）：** 一系列观察模型内部状态的方法，如CoE和CoT-Kinetics（分析思考过程中“状态”的变化轨迹），以及一个简单的LR Probe（直接在平均“神经活动”上训练一个分类器）。
- **评分标准（评估指标）：** 他们用了三个专业的指标来评判谁更优秀。这里，他们把“发现错误”定义为目标（正类）。
 - **AUROC：** 综合判断能力。分数越高，说明越能准确地区分“对”和“错”。
 - **AUPR：** 查准率和查全率的综合体现。在错误样本很少的情况下，这个指标尤其重要。分数越高越好。
 - **FPR@95 (95%召回率下的假阳性率)：** 这是一个非常严格的指标，衡量“可靠性”。它问的是：“当我们需要你找出95%的真正错误时，你会把多少个正确的步骤误报为错误？”这个值越低，说明这个方法越可靠，“误报”或“冤枉好人”的情况越少。

4.2 验证性能和鲁棒性 (RQ1)

翻译： 我们首先通过评估CRV与所有基线在推理步骤验证任务上的表现来解决RQ1，然后探究其在更具挑战性条件下的鲁棒性。 **主要验证性能。** 结果呈现在表1中，为我们的中心假设提供了强有力的经验支持：推理步骤的计算轨迹中存在的结构性签名，包含了一个可以直接验证其正确性的信号。CRV在每个数据集和每个指标上都持续优于所有的黑盒和灰

盒基线。这种结构性信号的强度在合成数据集上尤其明显。例如，在算术任务上，CRV达到了92.47的AUROC，显著超越了最强基线的76.45分。这种可靠性优势在FPR@95指标上得到了进一步的强调，CRV将假阳性率从基线的63.33%降低到了37.09%。

解读：这是对RQ1的第一个回答，结果非常惊人。

- **核心发现：** CRV在所有比赛项目（所有数据集、所有指标）中，都取得了压倒性的胜利，完胜了所有黑盒和灰盒对手。这强有力地证明了论文的核心观点：分析计算图的“结构指纹”是一种非常有效且优越的错误检测方法。
- **亮点数据：** 在算术任务中，CRV的综合判断能力（AUROC 92.47） 远超第二名（76.45）。更重要的是，在可靠性方面（FPR@95），当要求找出95%的错误时，最强的基线方法会有高达63.33%的误报率（即超过一半的警报都是假的），而CRV的误报率仅为37.09%，可靠性大大提高。

表格 1：验证性能

翻译：

范 式	方法	合成 (布尔)			合成 (算术)			GSM8K		
		AUROC↑	AUPR↑	FPR@95↓	AUROC↑	AUPR↑	FPR@95↓	AUROC↑	AUPR↑	FPR@95↓
黑 盒	MaxProb	58.81	0.34	95.20	61.87	1.81	84.98	54.91	7.99	91.86
	PPL	57.37	0.29	91.02	60.19	1.68	85.52	55.46	8.12	90.69
	Entropy	53.56	0.24	97.55	60.03	1.52	85.40	56.67	7.29	87.08
	Temp. Scaling	58.77	0.36	91.41	59.67	1.66	86.96	54.42	8.24	92.28
	Energy	95.11	0.28	51.08	76.45	5.59	73.86	62.55	9.11	86.34
灰 盒	CoE-R	53.17	0.33	92.85	58.47	1.93	76.68	52.38	8.34	96.20
	CoE-C	51.03	0.38	92.07	69.39	3.03	63.33	53.57	10.80	96.33
	CoT- Kinetics	53.62	0.24	97.13	60.83	1.58	85.09	56.54	7.35	86.83
	LR Probe	88.42	0.25	52.91	54.22	1.50	91.90	55.86	7.99	90.32
白 盒	CRV (Ours)	75.87	0.97	79.17	92.47	28.92	37.09	70.17	14.3	79.61

注：箭头表示优劣方向（↑越高越好，↓越低越好）。每个指标的最佳和次佳结果已高亮显示。布尔数据集上的AUPR较低反映了极端的标签不平衡，不正确标签仅占0.2%（附录A.5）。

解读： 这张表格是论文最有力的证据。

- **整体趋势：** 无论看哪个数据集（布尔、算术、GSM8K），也无论看哪个指标（AUROC, AUPR, FPR@95），最下面一行的CRV（白盒）方法的性能都显著优于上面所有的黑盒和灰盒方法。加粗的数字清晰地显示了这一点。
- **合成数据集上的巨大优势：** 在规则清晰的算术任务上，CRV的优势尤其明显。它的AUROC（92.47）和AUPR（28.92）都比其他方法高出一个数量级。这说明在算法性强的任务中，正确推理的“计算图”模式非常规整，任何偏离这个规整模式的错误都极易被CRV捕捉到。
- **可靠性（FPR@95↓）：** 这是一个关键指标。在算术任务中，CRV的FPR@95是37.09%，而表现最好的基线是63.33%。这意味着CRV在找出绝大多数错误的同时，产生的“误报”要少得多，这在实际应用中至关重要。
- **真实世界任务（GSM8K）：** 即使在更复杂的GSM8K数学应用题上，CRV的优势依然存在，所有指标都名列第一。
- **布尔数据集的AUPR：** 作者特意注释了布尔数据集上AUPR值普遍很低的原因。因为在这个任务中模型犯错的概率极低（只有0.2%），导致数据极度不平衡，AUPR指标会受到影响。但即便如此，CRV的AUPR（0.97）也远高于其他方法（都在0.3左右）。

翻译： 我们假设，算法推理的结构化特性会为有效解决方案引导出高度一致的执行轨迹。因此，错误的结构性签名表现为与此基线的更统一的偏离，从而使其高度可检测。**跨域泛化分析。** CRV与大多数基线的一个关键区别在于其诊断分类器需要训练。因此，一个关键问题是，CRV学到的是领域特定的相关性，还是更基础、可泛化的有缺陷推理的签名。为了测试这一点，我们进行了一项全面的跨域评估。我们在我们的三个数据集上分别训练一个CRV分类器，并评估其在另外两个未见过的领域上的零样本性能。

解读： 在庆祝了CRV的胜利之后，研究者们提出了一个更深刻、更具科学性的问题：CRV学到的到底是什么？它学到的是一种通用的“错误推理模式”，还是说它只是记住了“在算术任务中，这种图是错的；在逻辑任务中，那种图是错的”？为了回答这个问题，他们设计了“跨域泛化”实验，这是一种压力测试。就好比，一个在中国考驾照学得很好的司机（在“中文路牌”这个领域训练），把他直接扔到英国的道路上（在“英文路牌、靠左行驶”这个新领域测试），看他还能不能开好车。如果他依然开得很好，说明他学到的是通用的“驾驶原理”；如果他开得一塌糊涂，说明他学到的只是针对“中文路牌”的特定规则。

表格 2：跨域泛化性能

翻译：

测试集	方法 (训练集)	指标		
		AUROC↑	AUPR↑	FPR@95↓
布尔	基线 (MaxProb)	58.81	0.34	95.20
	CRV (GSM8K)	45.77	0.21	97.28
	CRV (算术)	61.58	0.51	87.55
	CRV (布尔)	75.87	0.97	79.17
算术	基线 (Energy)	76.45	5.59	73.86
	CRV (GSM8K)	55.11	1.50	91.91
	CRV (布尔)	69.59	2.64	72.87
	CRV (算术)	92.47	28.92	37.09
GSM8K	基线 (Energy)	62.55	9.11	86.34
	CRV (布尔)	44.37	6.33	95.71
	CRV (算术)	57.04	7.85	94.37
	CRV (GSM8K)	70.17	14.3	79.16

注：对于每个测试数据集，我们将最强的基线（基于AUROC）与在域内和域外训练的CRV进行比较。

解读：这张表格揭示了本文最深刻的科学发现之一。

- **结果：**表格清楚地显示，当CRV的训练和测试在不同领域进行时（例如，用“算术”数据训练，去测试“GSM8K”），其性能会大幅下降，甚至常常低于不需要训练的基线方法。例如，在GSM8K测试集上，“域内”训练的CRV能达到70.17的AUROC，而从“算术”任务迁移过来的CRV只有57.04，远低于基线的62.55。
- **结论：**错误的“指纹”是领域特定的。这就像那个去英国开车的中国司机，他彻底“蒙圈”了。这说明CRV学到的不是一个通用的“错误模式”，而是非常具体的、针对特定任务的“错误模式”。
- **深层含义：**这个发现为“AI内部存在专门化的功能电路”这一假说提供了强有力的证据。模型在进行布尔逻辑推理、算术计算和自然语言数学题时，所使用的内部计算路径（电路）是不同的。因此，当这些不同的电路出故障时，它们在计算图上留下的“结构性疤痕”也是完全不同的。这虽然限制了当前这个分类器的直接泛化能力，但从科学探索的角度看，它揭示了LLM内部认知结构的复杂性和模块化，这是一个非常重要的发现。

翻译： 难度增加下的性能。 为了进一步探究CRV的鲁棒性，我们分析了它在合成算术数据集上随问题复杂性（由运算符数量 $n \in \{5, 7, 10\}$ 控制）变化的性能。图2绘制了CRV与关键基线在这些难度级别上的性能。虽然大多数方法的AUROC和FPR@95表现稳定，但CRV在所有难度级别上都保持着一致的优势。随着难度的增加，所有方法的AUPR普遍提高，因为更难的问题会增加不正确示例的比例（AUPR对这种情况很敏感）。重要的是，尽管存在这些变化，CRV的优势依然存在，凸显了其结构性信号在任务难度和类别平衡变化中的鲁棒性。

解读： 这是对RQ1的最后一个测试：压力测试。他们通过增加算术题的计算步骤（运算符数量从5个增加到10个）来提升难度，观察CRV的表现是否会下滑。

- **结果：** 从图2可以看出，虽然所有方法的性能都受到了一些影响，但CRV始终保持着对其他方法的明显领先优势。
- **AUPR的上升：** 作者解释了为什么随着题目变难，AUPR指标反而上升了。这是因为难题更容易出错，导致数据集中“错误”样本的比例增加，数据变得更平衡，AUPR指标的数值自然会提高。但这并不影响结论。
- **结论：** CRV是鲁棒的。它所依赖的“结构性信号”是推理过程中的一个根本性特征，无论问题是简单还是复杂，这种信号都稳定存在。

图2 难度增加下的性能

翻译： 图2 在合成算术任务上，步骤正确性预测器随难度（运算符数量）变化的性能。随着复杂性增加，CRV保持了明显的优势。

解读： 这张图由三个子图组成，分别展示了AUROC、AUPR和FPR@95三个指标随题目难度（运算符数量 $n=5, 7, 10$ ）的变化情况。

- **AUROC Score (左图)：** 代表综合判断能力。可以看到，最上面的那条线代表CRV，它在所有难度级别上都远远高于其他所有方法，并且基本保持稳定。
- **AUPR Score (中图)：** 代表查准/查全能力。同样，代表CRV的线遥遥领先。所有方法的AUPR都随着难度增加而上升，原因已在前面解释过。
- **False Positive Rate (%) (右图)：** 代表可靠性（误报率），这个指标越低越好。可以看到，最下面的线代表CRV，它的误报率在所有难度下都是最低的，且非常稳定。

总而言之，这组图表直观地证明了CRV方法的优越性和稳定性。无论题目多难，它都能比其他方法更准确、更可靠地检测出推理错误。

4.3 错误计算的机械化分析 (RQ2)

翻译： 在证明了CRV的预测能力之后，我们现在转向其关键优势：可解释性。为了解决RQ2，我们剖析我们的图表示，以识别错误的结构性“指纹”，从高层的特征消融到对最具预测性结构的细粒度分析。

解读：在第一部分（RQ1）证明了“CRV很好用”之后，这一部分开始回答更深层次的问题“CRV为什么好用？”（RQ2）。他们不再仅仅关注性能分数，而是要打开CRV的“引擎盖”，看看里面到底是什么在起作用。他们将通过两种方式进行分析：

1. **特征消融 (Ablation)：** 像做控制变量实验一样，逐一拿掉某一组特征，看性能会下降多少，从而判断哪组特征最重要。
2. **细粒度分析和可视化：** 直接观察那些被证明最重要的特征，看看在“正确”和“错误”的计算中，这些特征的分布到底有什么不同。

表格 3：特征家族的消融研究

翻译：

特征集	算术			
		AUROC↑	AUPR↑	FPR@95↓
CRV (全部三族)	92.47	28.92	37.09	
消融:				
w/o 全局统计量	89.62	24.35	44.54	
w/o 节点统计量	88.31	23.25	49.07	
w/o 拓扑统计量	90.89	26.83	39.19	

注：在合成（算术）数据集上的留一法消融研究。

解读：这张表格展示了“控制变量实验”的结果。他们把之前提取的三类图特征（全局、节点、拓扑）每次去掉一类，观察性能的变化。

- **最重要的特征：** 当去掉“节点影响力和激活统计量”（Node Stats）时，性能下降得最厉害。特别是FPR@95（误报率）从37.09%急剧上升到49.07%，增加了超过12个百分点。这说明，**错误信号最主要地体现在计算图中具体节点的属性上**，比如是哪些特征被激活了，它们有多活跃，分布在哪一层等等。
- **次要但仍重要的特征：** 去掉“全局图统计量”（Global Stats）也会导致明显的性能下降。去掉“拓扑和基于路径的特征”（Topological Stats）影响相对最小。
- **结论：** 尽管“节点统计量”是核心，但三类特征共同作用才能达到最佳性能。这说明错误的信号是多维度的，需要从不同层面去捕捉。对于算术任务来说，计算过程中“谁”参与了（节点特征）比它们“如何连接”（拓扑特征）更重要。

翻译： **错误结构性签名的可视化。** 为了为我们的假设提供定性证据，我们可视化了我们的分类器学到的“结构性指纹”。图4显示了在GSM8K推理步骤中，五个高预测性特征在正确与不正确步骤上的分布。从图拓扑（如图密度）到节点统计（如总活跃特征数）等不同类型的

的特征，其分布都清晰可辨。在我们的合成数据集上也观察到了类似的模式（见附录 C.4），证实了图表示捕捉到了有效和有缺陷计算之间可分离的结构差异。

解读：文字不如图像直观。这部分，他们将那些被证明很重要的特征，用统计图的方式画出来，让我们能亲眼看到“正确”和“错误”的计算指纹到底有何不同。

图4 GSM8K上错误的拓扑指纹

翻译：图4 GSM8K上错误的拓扑指纹。五个选定的图特征在正确（蓝色）与不正确（红色）推理步骤上的分布。图中显示的每个特征的视觉分离在统计上都是显著的（独立t检验, $p < 0.001$ ），并且代表了中到大的效应量（Cohen's d）。这为归因图包含清晰、可分离的计算完整性结构信号提供了定量证据。

解读：这张图非常直观地展示了“错误指纹”的存在。图中是五个不同的特征在GSM8K数据集上的表现：

- **蓝色分布：**代表所有“正确”推理步骤的特征值分布。
- **红色分布：**代表所有“错误”推理步骤的特征值分布。

我们可以清晰地看到，在每一个特征上，蓝色和红色的分布都有明显的错位。例如：

- **总活跃特征数 (Total Active Features)：**错误的步骤（红色）倾向于激活更多的特征。
- **剪枝后特征节点数 (Pruned Feature Node Count)：**错误的步骤（红色）在核心计算图中也保留了更多的节点。
- **平均节点影响力 (Mean Node Influence)：**错误的步骤（红色）的平均节点影响力似乎更低。
- **图密度 (Graph Density) 和 平均边权重 (Mean Edge Weights)：**错误的步骤（红色）在这两个指标上也表现出与正确步骤（蓝色）不同的分布中心。

这些图有力地证明了，正确的计算和错误的计算在底层的结构上确实遵循着不同的统计规律。CRV的分类器正是学习到了如何区分这些蓝色和红色的分布。

翻译：虽然单个特征具有预测性，但CRV的优势在于它们的组合。为了说明这一点，我们通过主成分分析（PCA）将完整的高维特征向量投影到二维空间。图3揭示了不正确的步骤在更广泛的正确步骤分布中形成了一个密集的子集。至关重要的是，正确的步骤也占据了一个不正确的计算所不共享的独特区域。这表明许多推理失败是计算上的“功亏一篑”，在结构上与有效步骤相似，这解释了重叠部分。然而，存在一个计算完整性的区域，一个由只有正确推理才能达到的结构属性定义的区域。

解读：单个特征的区别已经很明显，但CRV是综合了所有特征来进行判断的。为了将这种高维度的综合判断可视化，他们使用了PCA技术，这是一种数据降维方法，可以把一个由几十个数字组成的“指纹”压缩到二维平面上的一个点，便于我们观察。

图3 PCA降维后的特征分布

翻译：图3 正确（蓝色）与不正确（红色）推理步骤的特征经过PCA降维后的分布。(a) 布尔 (b) 算术 (c) GSM8K

解读：这三张图分别展示了三个数据集上所有步骤“指纹”的二维投影。

- **蓝色点：**代表正确的步骤。
- **红色点：**代表错误的步骤。

从图中可以观察到一个非常有趣的现象：

- **重叠与分离：**红色点（错误）大部分都聚集在一起，形成一个相对密集的区域，并且这个区域与蓝色点（正确）有很大的重叠。这说明，很多时候，错误的计算和正确的计算在结构上非常相似，是“差之毫厘”的“近失误”（near misses）。这解释了为什么错误检测有时会很困难。
- **“安全区”的存在：**更重要的是，图中存在大片的纯蓝色区域，那里几乎没有任何红色点。这暗示着存在一个“计算完整性区域”（zone of computational integrity）。也就是说，正确的推理可以达到某些特定的、非常“健康”的计算结构状态，而错误的推理则无法进入这个“安全区”。

CRV分类器的工作，本质上就是在高维空间中学习划出这个“安全区”的复杂边界。

4.4 从诊断到因果干预 (RQ3)

翻译：最后，我们进行了一项探索性研究，以测试CRV的诊断性见解是否能指导规范性干预。我们白盒方法的一个关键优势是，预测失败的图特征可以追溯到模型计算的特定组件。我们通过一个关于算术任务的案例研究来说明这一点，纠正一个运算顺序错误。给定表达式 $(7 * ((5 + 9) + 7))$ ，模型在第二步错误地计算了 $7 * 14 = 98$ （表4）。CRV将此步骤标记为不正确，并且特征重要性分析突出显示了一个后期层转码器特征的异常高激活。将信号追溯回图，我们发现一个高度活跃的最后层转码器特征（ID 91814）与乘法密切相关。我们假设是这种过早的激活导致了错误，并进行了一次直接干预。我们在生成失败点之前重新运行生成过程，并使用一个前向钩子手动将这个特定的乘法特征的激活钳制为零。效果是立竿见影的：如表4右栏所示，随着过早的乘法脉冲被抑制，模型正确地生成了下一步 $14 + 7 = 21$ ，并继续得到了正确的最终答案。

解读：这是整篇论文最激动人心的部分，真正展示了从“诊断”到“治疗”的飞跃，回答了RQ3。

- **病例：**模型在计算 $(7 * ((5 + 9) + 7))$ 时，本应先算括号里的 $14 + 7$ ，但它却错误地先算了外面的 $7 * 14$ 。这是一个典型的运算顺序（PEMDAS）错误。
- **诊断 (CRV)：**CRV成功地捕捉到了这个错误，并通过分析“指纹”，指出“罪魁祸首”是一个与“乘法”相关的特征（ID 91814）在不该活跃的时候“跳”了出来（过早激活）。

- **治疗 (因果干预)：** 研究者们像做外科手术一样，在模型即将犯错的前一刻暂停了计算，然后用一个技术手段（forward hook）强行将那个“捣乱”的乘法特征的活性压制为零。
- **疗效：** “手术”效果立竿见影。没有了那个错误的“乘法”信号的干扰，模型立刻回到了正确的计算轨道上，先计算了 $14+7=21$ ，最终得到了正确的答案147。

这个案例雄辩地证明了CRV找到的“结构性指纹”不仅仅是与错误相关的“症状”，而是真正导致错误的“病因”。

表格 4：因果干预前后对比

翻译：

干预前 (错误)	干预后 (正确)
评估下面的算术表达式。	评估下面的算术表达式。
$(7 * ((5+9)+7))$	$(7 * ((5+9)+7))$
为了评估这个表达式，我们需要遵循运算顺序 (PEMDAS):	为了评估这个表达式，我们需要遵循运算顺序 (PEMDAS):
1. 评估最内层括号内的表达式: $5+9=14$	1. 评估最内层括号内的表达式: $5+9=14$
2. 将7与结果相乘: $7*14=98$	2. 将7加到结果上: $14+7=21$
3. 将7加到结果上: $98+7=105$	3. 将7与结果相乘: $7*21=147$
因此，该表达式的计算结果为105。	因此，该表达式的值为147。

注：高亮部分显示了分歧点，在此处抑制一个单一的乘法转码器特征纠正了模型的计算路径。

解读： 这张表格并排展示了“治疗”前后的推理过程，差异一目了然。

- **左栏 (干预前)：** 展示了模型的原始错误路径。在第一步正确计算出 $5+9=14$ 后，第二步它错误地执行了乘法 $7*14$ 。
- **右栏 (干预后)：** 展示了在抑制了那个“捣乱”的乘法特征后的新路径。在同样的第一步之后，模型现在正确地执行了加法 $14+7=21$ ，随后完成了正确的计算。
- **分歧点：** 高亮显示的第二步是整个故事的关键转折点。仅仅通过压制一个内部特征，就彻底改变了模型的行为，使其从错误走向正确。

翻译： 虽然这不是一个通用的解决方案，但这个概念验证是至关重要的一步。除了抑制错误的特征，我们还通过放大一个不活跃的特征来纠正了一个错误（附录C.4）。两种干预的成功提供了闭环证据，表明CRV的结构性签名在因果上与错误有关，为有针对性的模型干预开辟了一个有希望的方向。

解读：作者强调，虽然他们目前只展示了几个成功的案例，还不能算是一个能自动修复所有错误的通用工具，但这个“概念验证”（proof-of-concept）的意义非常重大。它证明了CRV不仅能“看病”，还能“治病”。他们还提到，除了“抑制”过强的信号，他们也能通过“增强”过弱的信号来纠正错误（详见附录），这说明他们的干预手段是灵活的。这为未来开发更智能、更自动化的AI“调试和修复”工具指明了方向。

5 相关工作

翻译：CoT提示以改进推理。思维链（CoT）提示由Wei等人（2022）引入，通过展示逐步解决问题的少样本示例，来引发LLM更复杂的推理。随后的工作在几个方向上扩展了这一思想。Kojima等人（2022）表明，即使是像“让我们一步步思考”这样简单的零样本指令也能触发连贯的推理轨迹。**验证和改进CoT推理。**CoT的透明性也使其成为模型可解释性和可靠性研究的焦点。虽然一些工作假设推理轨迹在某种程度上是模型内部过程的忠实表示，但大量证据凸显了它们的不可靠性。这催生了一个致力于验证和改进CoT轨迹的丰富研究领域。**CoT推理的机械可解释性。**我们的工作最直接地处于机械可解释性领域，该领域旨在逆向工程神经网络学到的算法，超越对CoT轨迹的表层分析。该领域的一个核心信条是，模型会发展出专门的子图，或称电路，来执行特定的计算。最近的工作已开始将这一视角应用于推理，不仅用于解释，还用于通过引出或引导行为电路来提高性能。该领域一个特别强大且日益流行的工具是使用稀疏自编码器（SAE），它学习将模型的密集激活向量分解为可解释特征的稀疏基。我们的工作直接建立在其变体——转码器（transcoder）之上。虽然先前的工作已使用基于转码器的归因图来定性分析CoT推理的忠实性，但我们的工作第一个将这种方法操作化用于自动化验证的。我们超越了视觉检查，通过系统地从这些图中提取定量的、结构性的特征，并证明它们可以用来诊断计算失败。

解读：这部分内容将本文的研究放置在更广阔的学术背景中，相当于说明“我们站在哪些巨人的肩膀上”以及“我们的工作与他人的不同之处”。

- **CoT提示的发展：**作者首先回顾了“思维链”技术的起源和发展，承认这是他们研究的基础。
- **CoT验证的现状：**接着，他们梳理了其他研究者在“验证和改进CoT”方面所做的工作。这些工作大多是从文本层面（黑盒）或隐藏状态层面（灰盒）进行的，比如训练一个“奖励模型”来给每一步打分。作者指出，他们的工作与这些方法的目标一致，但手段完全不同。
- **本文的独特定位：**作者明确指出，他们的研究属于“机械可解释性”这个更前沿、更深入的领域。这个领域的目标是彻底搞清楚AI内部的“算法”和“电路”。在这一领域内，虽然已经有人使用SAE或转码器来“可视化”AI的思考过程，但那些工作大多停留在“定性分析”和“视觉检查”的层面，就像医生看着CT图说“这里看起来有点不对劲”。而本文的开创性在于，他们首次将这个过程**“量化”和“自动化”**了。他们不是简单地“看”图，而是系统地“测量”图的各种结构属性，并用这些测量值来训练一个能自动做出判断的验证器。这是从“艺术”到“科学”的转变。

6 结论

翻译：在这项工作中，我们介绍了CRV，一种用于研究推理失败计算结构的白盒方法。通过将归因图视为潜在电路的执行轨迹，我们表明正确和不正确的推理会留下截然不同的结构性指纹。CRV揭示了这些错误签名不仅能实现准确的验证，而且是领域特定的，不同推理任务中的失败会表现为不同的模式。此外，对转码器特征的定向干预表明，这些签名在因果上是有牵连的，使我们能够纠正错误的推理。总而言之，这些发现确立了CRV作为一种机械化分析的概念验证，表明从不透明的激活转向可解释的计算结构，能够实现对LLM如何以及为何不能正确推理的因果性理解。

解读：结论部分是对整篇论文工作的高度概括和升华。

- **总结方法：**再次强调CRV是一种“白盒”方法，它把AI的思考过程（归因图）看作是内部“电路”运行的“日志”。
- **重申发现：**
 1. **指纹存在：**正确和错误的推理确实会留下不同的“结构性指纹”。
 2. **指纹可用于验证：**这些指纹可以被用来准确地判断推理的对错。
 3. **指纹是领域特定的：**不同任务的错误指纹不同，暗示了AI内部有专门化的电路。
 4. **指纹具有因果性：**通过干预指纹对应的特征，可以修复错误。
- **提升意义：**作者最后指出，他们的工作最重要的意义在于实现了一个范式转换。研究者们不再需要把AI的内部状态看作是无法理解的、不透明的数字，而是可以将其看作是结构清晰、可解释、甚至可干预的“计算过程”。这为我们真正从根本上理解和控制AI的“心智”活动，打开了一扇新的大门。

附录

附录A 数据集附加细节

翻译：A.1 合成数据集构建 为了创建一个受控的环境来研究推理失败，我们程序化地生成了两个合成数据集：布尔和算术。对于每一个，我们首先生成一个基准真相表达式，然后提示我们的基础模型（Llama 3.1 8B Instruct）生成一个解决该表达式的思维链解决方案。

解读：附录A详细介绍了他们如何精心制作实验所用的数据集。这部分展示了AI研究中“数据工程”的重要性。对于合成数据集，他们使用程序自动生成大量的布尔逻辑题和算术题，这样做的好处是每道题都有一个100%正确的标准答案，便于后续的自动标注。

表格 5：用于CoT生成的提示

数据集	Llama 3.1 8B Instruct 提示模板
布尔	`<
算术	`<

注：斜体表示动态内容的占位符。

解读： 这张表格展示了他们用来引导Llama 3.1模型生成思维链的具体指令（Prompt）。这些指令非常简洁明了，通过设定一个“系统”角色和“用户”角色，来模拟对话，并要求“助手”（即Llama模型）开始它的解题过程。这是当前与大型语言模型交互的标准方式。

表格 6：合成数据集示例

任务	示例
布尔 (n=5)	表达式: (((True or True) and (True and True)) or (True and False)) 生成的CoT: 为了评估布尔表达式，我们需要遵循运算顺序(PEMDAS): 1. 评估最内层括号内的表达式: * (True or True) -> True * (True and True) -> True 2. 现在表达式变为: (True and True) or (True and False) ... (后续步骤省略)... 最终答案是True。
算术 (n=7)	表达式: ((((-3)+(-6))*(9*6))+(-4)) 生成的CoT: 为了评估这个表达式，我们需要遵循运算顺序(PEMDAS): 1. 评估最内层括号内的表达式: (-3)+(-6) = -9 2. 乘以9和6: 9*6=54 3. 乘以-9和54: -9*54 = -486 4. 将-4加到-486上: -486+(-4) = -490 最终答案是-490。

解读： 这张表格给出了两个合成数据集中的具体例子，让我们能直观地看到模型是如何一步步解决布尔逻辑和算术问题的。这些生成的CoT就是他们后续分析和标注的对象。

翻译：A.2 标注协议 我们的主要目标是创建最高质量的标签。对于合成数据集，我们制定了一个严格的双管齐下的策略，同时利用强大的LLM裁判和直接的程序化验证，只保留两种方法都同意的标注。**方法1：LLM作为裁判。** 我们的第一个标注方法使用一个强大的外部LLM——Llama 3.3 70B Instruct——作为专家裁判。**方法2：程序化状态验证（仅限合成数据集）。** 对于合成任务，我们开发了一种程序化方法来验证整个推理轨迹的逻辑完整性。

解读： 这里揭示了他们为确保标注质量所采用的“双重保险”策略，尤其是在合成数据集上：

1. **AI裁判：** 请一个更强大的AI（Llama 3.3 70B）来批改作业。
2. **程序裁判：** 编写一个程序来自动检查每一步计算的对错。只有当“AI裁判”和“程序裁判”都认为某一步是正确的（或错误的），他们才会采纳这个标签。这种极其严格的交叉验证确保了他们训练数据的“纯净度”，排除了任何可能的模棱两可或标注错误。

翻译：A.4 处理错误传播 标注CoT的一个关键挑战是处理在第一次错误之后发生的步骤。一个模型可能会基于一个先前错误的数字或表达式进行一次技术上正确的计算。这就产生了一个根本性的模糊性：一个步骤应该根据其局部正确性来判断，还是根据其对最终答案

的全局贡献来判断？为了避免这种模糊性并创建一个清晰、定义明确的验证任务，我们采用了一个严格的截断策略。对于任何给定的推理链，我们只保留直到并包括第一个不正确步骤在内的所有步骤。所有后续步骤都从我们的数据集中丢弃。

解读：这是一个非常重要的细节，体现了研究设计的严谨性。他们指出了一个难题：如果AI在第二步算错了（比如把14算成了15），但在第三步基于这个错误的15做了一次完全正确的加法（比如 $15+7=22$ ），那么第三步到底算对还是算错？

- 从局部看，计算是正确的。
- 从全局看，它建立在错误的基础上，对最终答案是无益的。

为了避免这种定义上的混乱，他们采取了一个简单而有效的策略：“**一错即停**”。在一个推理链中，只要出现了第一个错误，他们就只把这个错误步骤本身纳入数据集，而它之后的所有步骤，无论“看起来”对错，都直接丢弃。这确保了他们研究的每个“错误”样本都是问题链条的“最初源头”，使得分析更加清晰和有意义。

表格 12：最终数据集统计

数据集	总步骤数	正确率	错误率
合成 (布尔)	126,624	99.8%	0.2%
合成 (算术)	155,434	98.8%	1.2%
GSM8K	8,737	93.4%	6.6%

解读：这张表展示了他们最终清理、标注完成的数据集的规模和类别分布。一个非常显著的特点是**数据极度不平衡**。例如，在布尔数据集中，99.8%的步骤都是正确的，只有0.2%是错误的。这意味着模型在这些任务上表现已经相当不错，但也给后续的分类任务带来了挑战，因为分类器需要从海量正确样本中准确地“揪出”那极少数的错误样本。

附录B 转码器训练细节

翻译：我们的方法依赖于高保真的、稀疏激活的转码器来创建一个可解释的模型。为此，我们为Llama 3.1 8B Instruct模型中的每个目标MLP模块训练了一个TopK-转码器。我们的训练协议旨在实现鲁棒性，并遵循了近期文献中建立的几个最佳实践。

解读：附录B深入到了CRV方法的技术核心——转码器的训练过程。他们强调，转码器的质量直接决定了后续所有分析的成败，因此他们采用了非常细致和成熟的训练方法。

图5 转码器训练损失曲线 图5 转码器训练损失曲线。x轴代表训练步数。在所有情况下，损失都高效地收敛，通常在大约4000步后饱和。

解读：这张图展示了转码器训练过程中的“学习曲线”（损失函数的变化）。损失（Loss）可以理解为转码器的“犯错程度”，这个值越低，说明它学得越好。从图中可以看到，不同层（layers.0, layers.15,...）的转码器在训练开始后，损失都迅速下降，并在大约4000步之后就趋于平缓（饱和），这表明训练过程非常高效和成功。

翻译： **B.2 归因图计算 关于归因位置的消融。** 归因图是相对于一个特定的token位置来计算的。这个位置的选择是一个关键的方法论决定，因为它决定了我们分析的是哪个计算时刻。我们研究了两种假设：分析一个步骤生成之前的状态（“预计算”轨迹）与它完成之后的状态（“后计算”轨迹）。为了测试这一点，我们比较了两种设置：（1）之前：在前一个推理步骤的最后一个token位置计算图。（2）之后：在当前推理步骤的最后一个token位置计算图，这是我们主要实验的默认设置。

解读： 这里他们探讨了一个非常微妙但重要的问题：在生成归因图时，应该选择哪个时间点进行“快照”？

- **选项1 (“之前”)：** 在模型即将生成当前步骤之前的一刻进行快照。这可以理解为分析模型的“意图”或“计划”。
- **选项2 (“之后”)：** 在模型已经生成完当前步骤之后的一刻进行快照。这可以理解为分析模型“已完成的行动”的轨迹。

表格 15：不同归因位置的性能比较

归因位置	合成 (布尔)			合成 (算术)			GSM8K		
	AUROC↑	AUPR↑	FPR@95↓	AUROC↑	AUPR↑	FPR@95↓	AUROC↑	AUPR↑	FPR@95↓
之前	68.66	1.80	77.44	85.95	12.05	47.89	70.32	16.19	85.29
之后	75.87	0.97	79.17	92.47	28.92	37.09	70.17	14.3	79.61

解读： 实验结果非常明确。表格显示，在所有数据集上，“之后”设置的性能都全面优于“之前”设置。这说明，一个错误的计算所留下的结构性痕迹，在计算完成的那一刻最为清晰和完整。预计算状态可能包含了犯错的“苗头”，但后计算状态则包含了错误的“既成事实”和最明确的证据。因此，他们所有的主要实验都采用了“之后”这个设置。

附录C 附加分类细节

翻译： **C.1 归因图特征** 这里我们详细介绍我们用于分类器的归因图提取特征。特征集被组织成三个层级：

1. 全局图统计量：...
2. 节点影响力和激活统计量：...
3. 拓扑和基于路径的特征：...

解读： 这部分详细列出了他们在3.2.3节中提到的三类特征的具体内容，比如“活跃转码器特征的总数”、“最终logit的熵”、“剪枝后节点的平均影响力”、“活跃特征的层级直方图”、“图密度”、“平均最短路径长度”等等。这为其他研究者复现他们的工作提供了完整的技术细节。

表格 16：不同诊断分类器的性能比较

方法	合成 (布尔)	合成 (算术)	GSM8K
	AUROC↑ AUPR↑ FPR@95↓	AUROC↑ AUPR↑ FPR@95↓	AUROC↑ AUPR↑ FPR@95↓
Dummy	50.8 0.25 100	49.84 1.20 100	48.06 6.46 100
Logistic Regression	76.4 0.75 68.91	89.5 11.46 41.56	73.8 18.70 78.69
Random Forest	61.71 4.49 100	92.99 30.56 43.68	71.7 17.65 76.18
Gradient Boosting	75.87 0.97 79.17	92.47 28.92 37.09	70.17 14.3 79.61

解读： 为了证明他们在主实验中选择梯度提升分类器（Gradient Boosting）是合理的，他们在这里进行了一场“分类器选拔赛”。

- **Dummy**：一个只会瞎猜的基线，性能接近50%，证明了任务并非易事。
- **逻辑回归 (Logistic Regression)**：一个简单的线性模型，表现出人意料地好，尤其是在GSM8K上甚至超过了GBC。这说明特征本身质量很高，即使是简单的模型也能学到很多东西。
- **随机森林 (Random Forest)**：另一种强大的集成模型，在算术任务上表现不错。
- **梯度提升 (Gradient Boosting)**：综合来看，GBC在不同任务和指标上表现最为稳定和强大，尤其是在作者最看重的合成任务上优势明显。因此，选择它作为主要实验的分类器是合理的。

翻译：C.4 RQ的附加结果 我们通过一个具体的算术任务案例研究，展示了另一次因果干预，这次我们不是通过抑制一个错误的特征，而是通过放大一个正确的特征来纠正一个微妙的数学解析错误。

解读： 这部分提供了支持主文论点的更多证据。最引人注目的是第二个因果干预的案例，它展示了干预的另一种形式：“扶强”而非“抑弱”。

表格 17：通过特征放大实现的成功因果干预

干预前 (错误)	干预后 (正确)
评估下面的算术表达式。	评估下面的算术表达式。
$-(5+(4*9))$	$-(5+(4*9))$
...	...
1. 乘以4和9: $4*9=36$	1. 乘以4和9: $4*9=36$

干预前 (错误)	干预后 (正确)
2. 将5和36相加: $5+36=41$	2. 将5和36相加: $5+36=41$
3. 从-1中减去41: $-1-41 = -42$	3. 将-1乘以41: $-1*41 = -41$
最终答案是-42。	最终答案是: -41。

解读：这个案例非常精彩。

- **病例：**模型在计算 $-(5+4*9)$ 时，错误地将开头的负号理解为了 $(-1) - \dots$ ，而不是正确的 $(-1) * \dots$ 。这是一个解析错误。
- **诊断 (CRV)：**CRV发现，在犯错时，一个代表“乘法”的特征信号强度不足。
- **治疗 (因果干预)：**研究者们反其道而行之，不是去抑制某个错误信号，而是手动增强了那个本该出现但却很微弱的“乘法”特征的激活值。
- **疗效：**效果同样立竿见影。得到了增强的“乘法”信号后，模型立刻正确地将运算理解为乘法，得出了正确答案-41。

这个案例与主文中抑制特征的案例一起，构成了一个强有力的证明：CRV不仅能识别出“不该有的信号”，还能识别出“该有却没有的信号”，并且两种情况都能通过精准干预来修复。这极大地增强了他们关于因果理解的论断。

附录E 局限性

翻译：我们的工作引入了一种用于推理科学分析的新方法，其局限性与其作为一种白盒、机械化工具的设计内在相关。 **计算强度。** CRV的一个主要局限是其计算强度。这个过程...比黑盒或灰盒验证方法要多出几个数量级的资源消耗。 **聚合性与特征级分析。** CRV使用的特征集主要是聚合性的...它尚未完全利用构成图节点的单个转码器特征的语义内容。 **错误签名的泛化性。** 我们的经验结果基于单一模型家族 (Llama 3.1) 的8B规模...错误签名是高度领域特定的。 **可解释性工具的保真度。** 我们分析的有效性取决于底层可解释性工具的质量和保真度。

解读：在附录的最后，作者非常坦诚地列出了他们工作的局限性，这是严谨科学态度的体现。

1. **太贵太慢：**CRV目前计算成本极高，只能作为实验室研究工具，无法大规模实用。
2. **分析粒度不够细：**目前的CRV分析的是图的“宏观”统计特征（比如节点总数），而不是每个节点的具体“语义”（比如“这个加法特征用得对不对”）。未来的方向是发展能理解节点语义的、更智能的验证器。
3. **泛化性有限：**实验只在一个模型上进行，且已证明错误指纹是领域特定的，能否推广到其他模型和任务还是个未知数。
4. **依赖于工具：**CRV的效果好坏，完全取决于底层的转码器和归因算法的好坏。如果这些基础工具有缺陷，CRV的分析结果也会不准确。

这些局限性不仅展示了作者的科学诚信，也为未来的研究指明了方向。

This content was created by another person. It may be inaccurate or unsafe. [Report unsafe content](#)[Opens in a new window](#)
