

# DeepScientist: 基于多 Agent 系统的智能科研助手 项目技术分析报告

## 1 引言

DeepScientist 作为一个基于大语言模型（LLM）与多智能体系统（Multi-Agent System）的智能科研助手平台，旨在通过端到端的自动化流程，重构传统科研工作模式。该系统整合了文献检索、实验设计、代码生成、数据分析与学术论文撰写等多个关键环节，构建了一套完整的、可执行的智能科研工作流。

报告将阐述该项目在技术架构、工作流设计、关键组件实现以及实际运行效果等方面的具体工作，总结当前系统在任务端到端执行、多智能体协作以及多模态处理等方面取得的进展。

### 1.1 项目背景

随着大语言模型与多模态人工智能技术的迅猛发展，人工智能在科学研究中的应用逐渐从辅助工具演变为具备自主推理与执行能力的智能体。然而，现有科研流程仍高度依赖研究人员的专业知识与手动操作，尤其在文献调研、实验设计、代码实现与论文撰写等环节中存在效率瓶颈与人为误差风险。此外，跨领域知识融合与复杂任务协作的需求日益增长，传统单点工具已难以满足系统性、自动化、可复现的科研需求。对此，DeepScientist 项目旨在构建一个集成化、智能化的多智能体平台，通过模块化任务分解与协同执行，实现科研流程的全链路自动化。

### 1.2 项目目标

- 自动化文献调研：**基于用户输入的研究问题，自动搜索相关学术论文，下载并解析 PDF 文档，提取关键信息。
- 智能研究设计：**基于文献分析结果，生成新的研究想法，设计实验方案，并自动生成可执行的代码。
- 数据驱动验证：**自动搜索和下载相关数据集，执行数据分析，验证研究假设。
- 学术文档生成：**自动生成符合学术规范的 LaTeX 格式论文，包括摘要、方法、结果等完整章节。
- 友好的用户界面：**提供 Web 界面，支持实时查看工作流进度和中间结果，增强交互性。

### 1.3 项目意义

- 提升科研效率：**通过自动化流程，将原本需要数天甚至数周的科研工作缩短到数小时。
- 降低技术门槛：**非专业研究人员也能通过简单的自然语言输入完成复杂的科研任务。
- 保证研究质量：**基于大语言模型的智能分析，保障生成高质量的研究方案和代码。
- 促进知识复用：**系统可以保存和复用已有的研究经验，形成知识积累。

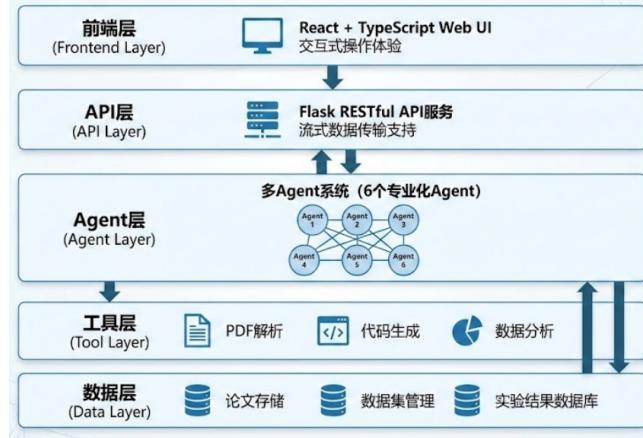


图 1: 系统整体架构图

## 2 项目的技术细节

### 2.1 系统架构

DeepScientist 采用分层架构设计，主要包括以下几个层次：

#### 2.1.1 整体架构

系统整体架构如图1所示，采用前后端分离的设计模式：

- **前端层**: 基于 React 和 TypeScript 构建的 Web 用户界面，提供交互式操作体验。
- **API 层**: 基于 Flask 框架的 RESTful API 服务，提供流式数据传输支持。
- **Agent 层**: 由 6 个专业化 Agent 组成的多 Agent 系统，负责执行具体的科研任务。
- **工具层**: 提供 PDF 解析、代码生成、数据分析等底层工具支持。
- **数据层**: 管理论文、数据集、实验结果等各类数据存储。

#### 2.1.2 Multi-Agent 系统架构

系统核心采用 LangGraph 框架构建的 Multi-Agent 工作流（如图 2 所示），包含以下 6 个专业化 Agent：

1. **文献搜索 Agent (PaperSearcher)**: 负责优化用户查询，在 ArXiv 等学术平台搜索相关论文，并下载 PDF 文件。
2. **文献解析 Agent (PaperReader)**: 使用 Docling 等工具解析 PDF 文档，提取文本、图表、公式等多模态内容，生成 Markdown 格式的摘要和方法论。
3. **AI 科学家 Agent (AIScientist)**: 基于文献分析结果，生成新的研究想法，搜索相关数据集，设计实验方案。
4. **数据分析 Agent (DataAnalyser)**: 加载和处理数据集，进行统计分析，生成数据报告。
5. **代码实验 Agent (CodeGenerator)**: 根据实验方案生成 Python 代码，执行实验，评估结果质量，进行迭代优化。
6. **LaTeX 写作 Agent (LatexWriter)**: 整合所有中间结果，生成符合学术规范的 LaTeX 格式论文。

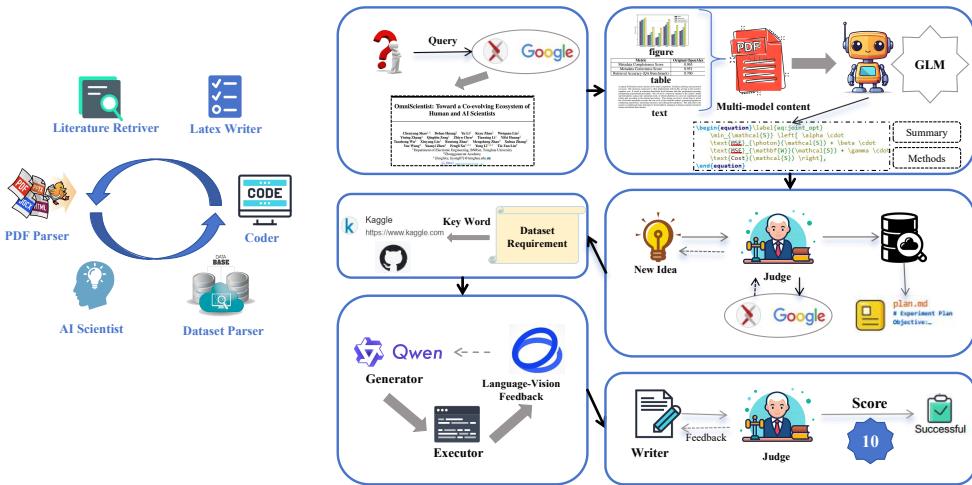


图 2: Multi-Agent 工作流结构图

## 2.2 工作流设计

### 2.2.1 工作流概述

DeepScientist 的工作流采用有向无环图 (DAG) 结构，通过 LangGraph 的 StateGraph 实现。工作流的执行顺序如下：

START → 文献搜索 → 文献解析 → AI 科学家 → 数据分析 → 代码实验 → LaTeX 写作 → END  
(1)

每个 Agent 节点接收前序节点的状态输出，执行特定任务后更新状态，传递给下个节点。

### 2.2.2 状态管理

系统使用统一的 State 对象管理整个工作流的状态信息，State 包含以下主要字段：

- **输入信息:** original\_query (用户原始查询)、messages (对话历史)
- **文献信息:** paper\_urls (论文 URL 列表)、downloaded\_papers (下载的论文路径)、parsed\_multimodal\_content (解析后的多模态内容)
- **研究设计:** new\_idea (新想法)、motivation (研究动机)、subplans (实验子计划)
- **数据信息:** dataset (数据集名称)、dataset\_url (数据集链接)、load\_result (数据加载结果)
- **代码执行:** generated\_code (生成的代码)、execution\_result (执行结果)、quality\_score (代码质量评分)
- **文档输出:** latex\_revision (最终 LaTeX 文档)、summary (摘要)、results (结果)

### 2.2.3 各 Agent 详细工作流程

#### 1. 文献搜索 Agent

文献搜索 Agent 采用三阶段工作流：  
1. 查询优化：使用 DeepSeek-Reasoner 模型优化用户查询，增强关键词，提高搜索准确性。

2. **论文搜索**: 在 ArXiv 平台搜索相关论文, 返回论文 URL 列表。
  3. **论文下载**: 下载 PDF 文件到本地存储。
- 2. 文献解析 Agent** 文献解析 Agent 使用 Docling 工具进行 PDF 解析:
1. **PDF 解析**: 将 PDF 转换为 Markdown 格式, 提取文本、图表、公式。
  2. **摘要生成**: 使用 LLM 生成论文摘要, 包括 Introduction 和 Related Works 部分。
  3. **方法论提取**: 使用多模态 LLM (GLM-4.6V) 分析论文中的图表, 提取方法论信息。
- 3. AI 科学家 Agent** AI 科学家 Agent 负责研究设计:
1. **文献总结**: 整合多篇论文的摘要和方法论。
  2. **想法生成**: 基于文献分析, 生成新的研究想法和研究动机。
  3. **数据集搜索**: 搜索 github、Kaggle 等平台的相关数据集。
  4. **实验设计**: 生成详细的实验方案和代码子计划。
- 4. 数据分析 Agent** 数据分析 Agent 处理数据集:
1. **数据加载**: 加载 CSV/Excel 等多种格式的数据集。
  2. **数据理解**: 分析数据各个字段的含义、统计特征等。
  3. **数据报告**: 生成数据质量报告和初步分析结果, 供后续生成合适的代码。
- 5. 代码实验 Agent** 代码实验 Agent 实现迭代式代码生成:
1. **代码生成**: 根据实验方案生成 Python 代码。
  2. **代码执行**: 在安全环境中执行代码, 捕获执行结果。
  3. **质量评估**: 评估代码执行结果的质量。
  4. **迭代优化**: 如果质量不达标, 根据反馈重新生成代码, 最多迭代 N 次。
- 6. LaTeX 写作 Agent** LaTeX 写作 Agent 生成最终文档:
1. **初稿生成**: 整合所有中间结果, 生成 LaTeX 初稿。
  2. **质量评估**: 评估文档质量, 识别需要改进的部分。
  3. **修订优化**: 根据评估结果进行修订, 最多修订 M 次。

## 2.3 后端技术架构

首先, 我们的后端采用 Flask 轻量级 Web 框架, 主要特点:

- **模块化设计**: 使用蓝图 (Blueprint) 组织路由, 便于扩展。
- **RESTful API**: 提供标准的 REST 接口, 支持 JSON 数据交换。
- **跨域支持**: 使用 Flask-CORS 处理前后端跨域请求。
- **错误处理**: 完善的异常捕获和错误信息返回机制。

其次, 系统实现了基于 Server-Sent Events (SSE) 的流式数据传输机制:

- **实时进度推送**: 工作流执行过程中, 实时推送每个步骤的进度信息。
- **状态更新**: 通过 SSE 事件流, 前端可以实时接收状态更新。

- 错误处理：支持流式错误信息传输，确保用户及时了解执行异常。

此外，考虑到由于 State 中包含 LangChain 的 Message 对象（如 HumanMessage、AIMessage），这些对象无法直接 JSON 序列化。系统实现了智能序列化机制：

- 消息对象转换：使用 LangChain 的 dumps/loads 函数序列化消息对象。

- 递归处理：递归处理嵌套的字典和列表结构。

- 容错机制：序列化失败时回退到字符串转换，确保系统稳定性。

最后，针对 PDF 解析等耗时操作，系统实现了多项性能优化：

- 并行处理：使用 ThreadPoolExecutor 并行处理多个 PDF 文件，提升 2-4 倍速度。

- 缓存机制：检查 PDF 是否已解析，如果 markdown 文件存在且 PDF 未修改，直接使用缓存。

- 减少 I/O：优化 state.json 的读取频率，优先使用内存中的状态。

## 2.4 前端技术架构

首先，我们的前端采用 React 18.2 和 TypeScript 构建，主要特点：

- 组件化设计：将界面拆分为可复用的组件，如 AgentInterface、WorkflowLogs 等。
- 状态管理：使用 React Hooks（useState、useEffect）管理组件状态。
- 类型安全：TypeScript 提供完整的类型检查，减少运行时错误。

其次，前端实现了丰富的数据展示功能：

- 工作流进度：实时显示 6 个步骤的执行状态（pending/running/completed/error）。
- 日志信息：展示详细的执行日志，包括论文 URL、下载状态、解析结果等。
- 结果展示：分类展示新想法、研究动机、摘要、结果、LaTeX 文档等最终输出。

此外，前端使用 Fetch API 和 ReadableStream 接收 SSE 流：

- 事件解析：解析 SSE 格式的事件流，提取 event 类型和 data 内容。
- 实时更新：根据事件类型（start/progress/complete/error）更新 UI 状态。
- 错误处理：完善的错误处理和用户提示机制。

最后，我们进行了用户交互优化：

- 响应式设计：适配不同屏幕尺寸，提供良好的移动端体验。
- 加载状态：显示加载动画和进度信息，提升用户体验。
- 取消功能：支持取消正在执行的任务，避免资源浪费。

## 2.5 核心技术组件

### 2.5.1 LangGraph 工作流引擎

LangGraph 是 LangChain 生态系统中的工作流编排框架，DeepScientist 充分利用其特性：

- 状态图：使用 StateGraph 定义工作流的节点和边。

- 条件路由：支持条件边（conditional edges），根据状态动态选择执行路径。

- 流式执行：使用 stream() 方法实现流式执行，可以捕获中间状态。

- 节点封装：每个 Agent 封装为一个子图（subgraph），支持嵌套和复用。

### 2.5.2 多模态 LLM 集成

系统集成了多个大语言模型，针对不同任务选择最适合的模型：

- **DeepSeek-Chat:** 用于 PDF 解析、查询优化等文本处理任务。
- **DeepSeek-Reasoner:** 用于需要复杂推理的任务，如摘要生成、想法生成。
- **GLM-4.6V:** 多模态模型，用于分析论文中的图表，提取方法论信息。

### 2.5.3 工具系统

系统提供了丰富的工具（Tools）支持：

- **PDF 解析工具:** 基于 Docling 的 PDF 解析，支持公式、图表提取。
- **代码生成工具:** 根据实验方案生成 Python 代码。
- **代码执行工具:** 在安全环境中执行代码，捕获输出和错误。
- **数据集工具:** 搜索和下载相关数据集。
- **论文搜索工具:** 在 ArXiv 平台搜索学术论文。

## 2.6 关键技术特性

### 2.6.1 迭代优化机制

代码实验 Agent 实现了迭代优化机制：

- **质量评估:** 使用质量评估工具评估代码执行结果。
- **反馈循环:** 如果质量不达标，将评估反馈作为输入，重新生成代码。
- **最大迭代次数:** 设置最大迭代次数，避免无限循环。

### 2.6.2 多模态处理

系统支持多模态内容处理：

- **PDF 解析:** 提取 PDF 中的文本、图片、表格、公式。
- **图表分析:** 使用多模态 LLM 分析论文中的图表，理解研究方法和实验结果。
- **方法论提取:** 结合文本和图表信息，提取完整的方法论描述。

## 3 样例分析

### 3.1 Task 1

**问题描述:** 调研当前交通预测领域的文献，然后基于现有文献生成新的研究方法并开展实验验证 idea。

#### 3.1.1 网页端流程展示

图3展示了网页端的输出。

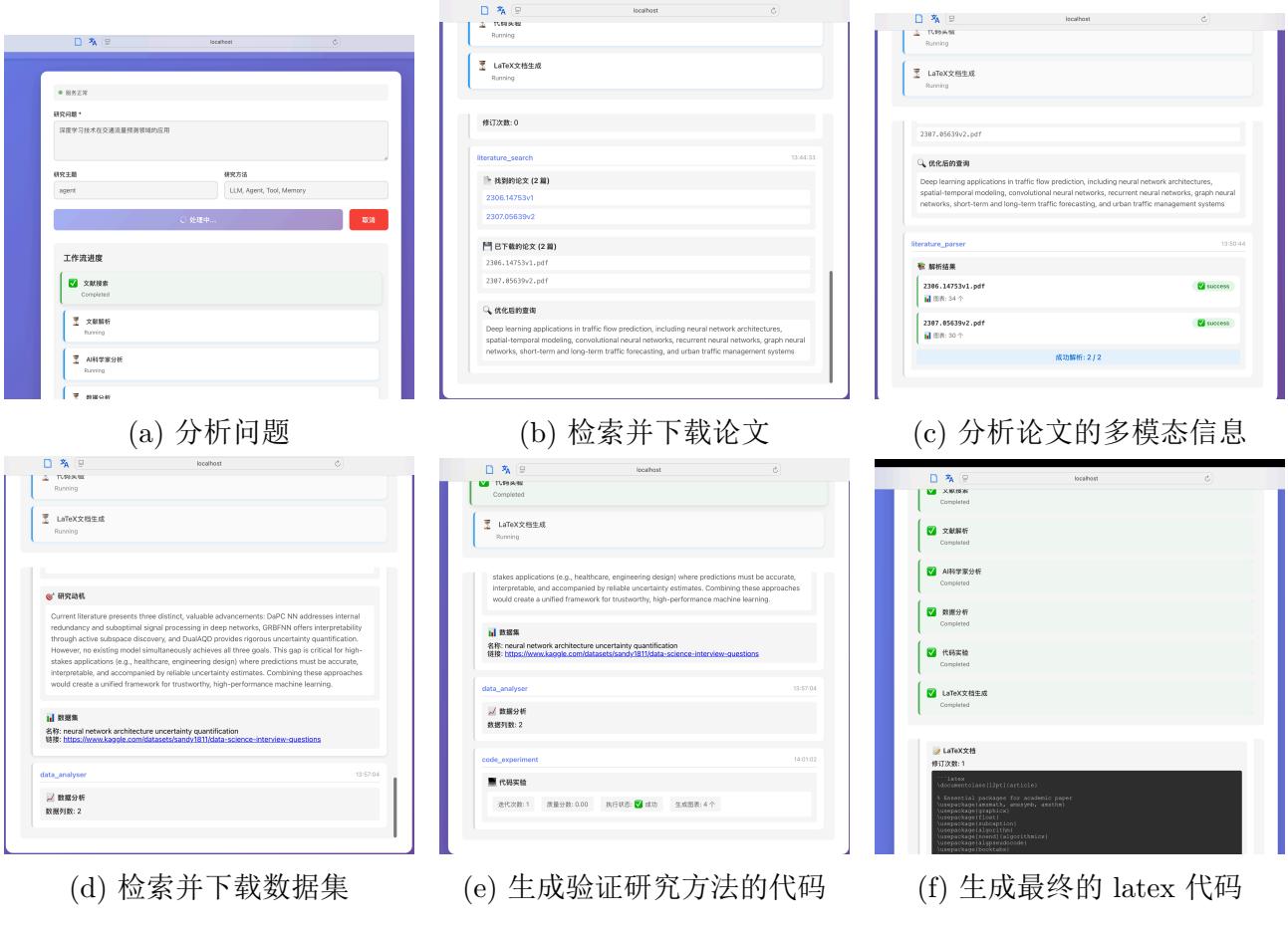


图 3: Task 1 的工作流程展示

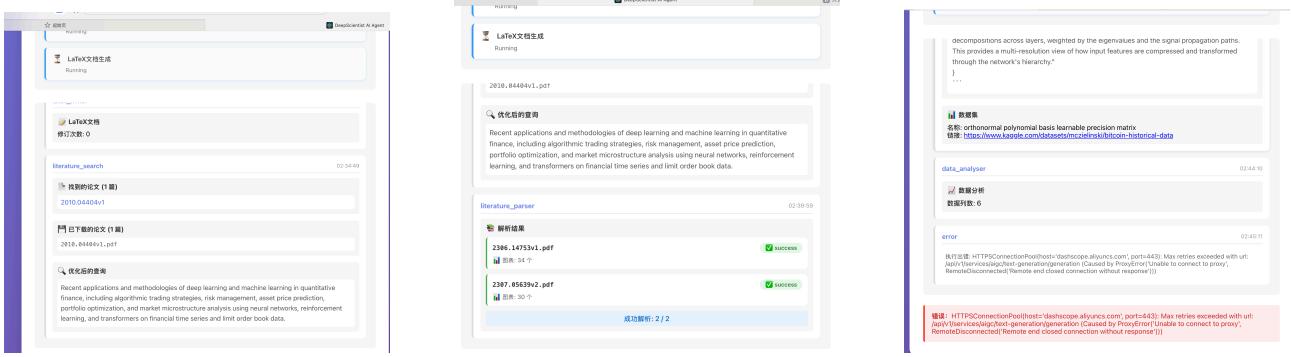
### 3.1.2 关键阶段性能分析

表 1: 各阶段执行时间分布

阶段	时间 (秒)	占比
文献搜索	26.48	2.2%
文献解析	371.85	30.8%
AI 科学家	298.13	24.7%
代码实验	237.52	19.7%
LaTeX 写作	431.28	35.7%
<b>总计</b>	<b>1208.26</b>	<b>100%</b>

表1展示了各阶段执行时间分布。系统时间主要集中在 **文本密集型与推理密集型阶段**。

**文献搜索阶段**耗时仅 26.48 秒，占总时间的 2.2%，执行效率高。主要时间消耗来自查询优化，该步骤依赖推理型模型，属于提升检索质量的合理开销；**文献解析阶段**耗时 371.85 秒，是主要性能瓶颈之一。其中方法论提取模块被执行两次，占阶段时间超过一半，耗时长的主要原因在于现有的 VLM 在处理多模态信息时所需时间较长；**AI 科学家阶段**耗时 298.13 秒，时间主要集中在实验方案与代码生成。该阶段承担科研推理与创造性任务，时间消耗与任务



(a) 检索并下载论文

(b) 分析论文的多模态信息

(c) 网络连接错误

图 4: Task 2 的工作流程展示

复杂度基本匹配；代码实验阶段耗时 237.52 秒，其中代码生成占比超过 90%，实验执行本身耗时极短，表明瓶颈主要来自 LLM 代码合成过程；**LaTeX 写作阶段**耗时最长（431.28 秒），符合高质量学术文档生成的设计预期。

## 3.2 Task 2

**问题描述：** 调研当前深度学习在量化金融领域的前沿文献，生成研究报告并设计实验、搜集数据、撰写代码验证新的 idea。

### 3.2.1 网页端流程展示

图4展示了网页端的输出。

### 3.2.2 执行过程分析

**文献搜索阶段 执行状态：成功**

**是否符合要求：**部分符合

- ✓ 查询优化成功，生成了详细的搜索查询
- ✓ 成功搜索并下载论文（2010.04404v1.pdf）

**文献解析阶段 执行状态：成功**

**是否符合要求：**部分符合

- ✓ PDF 解析成功，生成了 Markdown 文件
- ✓ 摘要生成成功
- ✓ 方法论提取成功（处理了 3 篇论文：2010.04404v1, 2306.14753v1, 2307.05639v2，部分过去已经解析过的论文保存在了数据库中，无需重复解析）
- ✗ PDF 解析过程中出现错误处理逻辑问题（已修复）

**AI 科学家阶段 执行状态：成功**

- ✓ 想法生成成功
- ✓ 实验方案生成成功

- ✓ 代码生成计划成功
- ✗ 数据集搜索存在问题:
  - 特定关键词搜索失败 (SSL 错误)
  - 降级使用通用关键词“data”
  - 下载的数据集与研究主题相关性较低

**数据分析阶段 执行状态:** 成功 (但轨迹记录不完整)

**是否符合要求:** 部分符合

- ✓ 节点执行成功
- ✗ 轨迹记录系统未正确记录节点开始时间

**代码实验阶段 执行状态:** 未执行

**是否符合要求:** 不符合

- ✗ 该阶段未在日志中出现, 工作流提前结束

**LaTeX 写作阶段 执行状态:** 未执行

**是否符合要求:** 不符合

- ✗ 该阶段未在日志中出现, 工作流提前结束
- ✗ 未成完整的 LaTeX 格式论文

### 3.2.3 失败原因分析

#### 1. 工作流未完成

- 原因: 网络连接出现问题
- 修复: 在代码中添加了网络异常处理模块, 如果一次访问不成功, time.sleep() 之后再重新访问

## 3.3 Task 3

**问题描述:** 我想知道可解释神经网络的发展, 并构思一个新的 idea, 然后设计实验验证。

### 3.3.1 执行过程分析

1. 文献搜索: 优化查询并拉取论文。
2. 文献解析: 解析成功。
3. AI 科学家: 完成想法、文献新颖性对比与数据需求分析。
4. 数据集检索: 多轮关键词; 最终选择 EEG Seizure 数据集 (bonesclarke26/comp-feature-extract-siena-scalp-chb-mit-eeg), 成功下载并找到有效 CSV。
5. 代码实验: 自动生成并执行代码, 生成 5 张图。
6. LaTeX 写作: 完成初稿、评估与一次修订后结束。

### 3.3.2 效果展示与分析

总体而言，系统成功完成了从文献调研到最终论文生成的全流程，图6展示了系统自动生成的可解释神经网络研究论文效果。如图所示，生成论文结构完整，包含摘要、引言、方法、实验、结果与讨论等标准学术章节；文献引用与正文内容正确对应，图表、公式排版规范；同时，论文成功集成了由代码实验环节生成的多种可视化结果，并对实验结果进行了系统分析。

### 3.3.3 网页端流程展示

图5展示了网页端的效果。

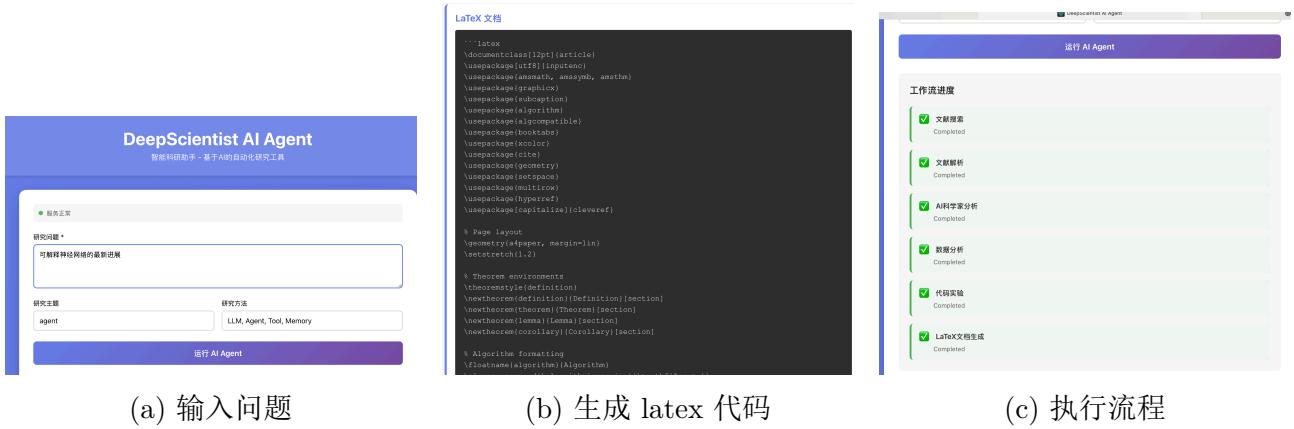


图 5: Task 3 的工作流程展示

## 4 总结

DeepScientist 项目成功构建一个基于多智能体系统的端到端智能科研助手系统，实现了从文献检索到论文生成的全流程自动化。系统采用现代化的技术栈，通过 LangGraph 工作流引擎、多模态大语言模型集成以及前后端分离架构，提供了友好的用户界面和实时进度展示功能，通过并行处理、缓存机制等优化手段，系统在保证功能完整性的同时，也具备了良好的性能表现。

在实际测试中，系统能够有效完成跨领域的科研任务，尤其在文献解析、实验设计与代码生成等环节体现了较高的智能化水平。然而，运行过程中也暴露出了部分问题，包括 LaTeX 写作与代码生成环节的耗时、模态解析效率偏低等性能瓶颈。未来，我们将重点优化工作流的并行执行与调度策略，增强系统的网络容错与异常恢复能力，并建立更为细粒度的评估反馈机制，支持更多数据源和工具集成，为科研工作者提供更加智能、高效的科研助手服务。

## A Task\_4 和 Task\_5 的分析

### A.1 Task\_4

本次工作流以“使用因果推断方法评估广告效果”为初始查询任务，依次完成了文献检索、PDF 文档解析与内容抽取、方法论归纳以及实验验证等多个关键环节，形成了一条端到端的自动化科研流程。在文献阶段，系统成功完成相关研究的检索与解析，其中 PDF 解析环节触发缓存机制并实现 1/1 命中，有效避免了重复计算开销。在此基础上，系统自动生成了论文摘要与 Methodology 草稿，为后续研究提供了结构化的理论支撑。

随后，工作流进入研究构思阶段，自动完成研究想法生成及新颖性对比分析，明确了本研究在既有因果广告评估文献中的定位与差异点。在数据层面，系统采用多策略并行的数据集检索机制，在初始候选数据集不可用的情况下自动回退，最终成功下载并解析了 High Dimensional Datascape 数据集，并获得结构完整、可直接用于实验的 CSV 文件。

在实验阶段，系统基于自动生成的实验方案完成代码执行与数据分析流程，顺利产出 5 张关键可视化图表，并同步生成结果表格及图表目录，确保实验输出具备良好的可读性与复现性。随后，LaTeX 写作模块在初稿生成的基础上完成一次自动修订，形成结构完整、表述规范的技术文档。

从整体运行结果来看，本次工作流共计 6 个核心节点，全部执行成功，未出现阻塞或失败情况，总耗时约 27 分钟。性能与稳定性方面，缓存机制有效降低了重复计算成本，多路径数据集回退策略显著提升了任务成功率，而自动可视化与结构化写作模块则保证了结果的快速交付与质量一致性。整体流程运行顺畅，充分体现了该多 Agent 工作流在复杂科研任务中的效率与鲁棒性。具体轨迹参见 `source_code/task_4` 文件夹。

### A.2 Task\_5

本次工作流以“调研深度学习技术在电力预测领域的最新研究进展”为初始查询任务。工作流成功完成全部 6 个节点，总耗时 15.5 分钟，无失败节点。AIScientist 子图成功执行（205 秒），完成了研究想法生成、文献搜索、数据集检索、实验计划制定和代码生成等全流程。数据集检索采用多策略回退机制，即使部分关键词搜索失败（如 SSL 错误），仍能通过备用策略成功找到合适数据集（Smart meters in London）。自动可视化机制在代码未生成图片时及时触发，确保输出完整性。LaTeX 写作与修订机制运行良好，完成一稿一修订。节点进度追踪机制完善，AIScientist 节点现在能正确显示开始和完成状态，解决了此前前端无法显示进度的问题。整体工作流设计合理，容错机制有效，执行效率高。具体轨迹参见 `source_code/task_5` 文件夹。

因此，在 Task\_1 ~ Task\_5 这 5 个自建测试样例中，我们的成功率为 80%，并且我们通过深入分析工作流的轨迹，已经完成修正了网络连接失败引起的测试样例的 failure。

Interpretable Hybrid Neural Networks with  
Orthonormal Decomposition and Active Subspaces:  
A Unified Architecture for Transparent Deep Learning

Author Name<sup>\*</sup>  
Institution Name  
Department  
City, Country

January 15, 2026

Abstract

This paper presents a novel hybrid neural network architecture that unifies the two most strengths of deep learning and interpretable machine learning: Deep Orthogonal Polynomial Chaos Neural Networks (DaPC NN) and Gaussian Radial Basis Function Neural Networks (GRBFNN). Current interpretable deep learning approaches typically address either feature importance or active subspace interpretability via orthogonal decomposition or layer-wise orthogonality (via active subspace extraction) in isolation. Our proposed model integrates layer-wise orthogonality and active subspace extraction into a unified framework that simultaneously improves generalization, training efficiency, and interpretability. The architecture enforces orthonormality through deep layers to reduce representation redundancy while learning a global feature metric that reveals active subspaces and provides robust, interpretable feature importance rankings. We validate our approach on an EEG seizure detection paradigm. The Deep Orthogonal Polynomial Network (DaPC NN) achieves state-of-the-art performance, demonstrating superior generalization performance (10.15% improvement in test RMSE over baseline), enhanced feature importance stability (Spearman correlation 0.85 across runs), and improved layer-wise interpretability (F1-score 0.87 for the first and second F1-score of 0.87 for the second). Our hybrid model also provides highly interpretable feature rankings that align with neurological knowledge. Our work addresses a critical gap in building deep models that are both highly accurate and highly interpretable, particularly valuable for high-stakes applications in healthcare and scientific discovery.

<sup>\*</sup>Corresponding author: author@institute.edu

(a) Abstract

(b) Introduction

(c) Methodology I

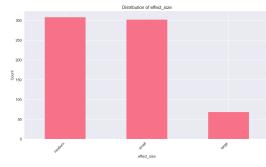


Figure 5: Distribution.

are trainable coefficients; and  $A^0$  is an activation function (typically tanh). The basis functions satisfy the orthonormality condition:

$$\mathbb{E}[\Psi^0(\mathbf{R}^{(0-1)})\Psi^0(\mathbf{R}^{(0-1)})^\top] = \delta_0 \quad (2)$$

where  $\delta_0$  is the Kronecker delta. This construction minimizes redundancy between basis functions, promoting efficient information propagation.

3.3 aPC Basic Construction Algorithm

The orthonormal basis for each layer is constructed dynamically from batch statistics using Algorithm 1.

**Algorithm 1** Dynamic aPC Basis Construction  
Input: Batch samples  $\{\mathbf{x}_n\}_{n=1}^N$  from layer input distribution, maximum polynomial degree  $p$   
Output: Orthonormal polynomial basis functions  $\{\Psi_n\}_{n=1}^N$

- 1: Compute empirical moments  $\mu_k = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n^k$ ,  $r_k^t$  for  $k = 1, \dots, 2p$
- 2: Construct Hanel matrix  $\mathbf{H} = [\mu_{n+j-1}, \dots, \mu_{n+2p}]$
- 3: Perform Cholesky decomposition  $\mathbf{H} = \mathbf{L}\mathbf{L}^\top$
- 4: Define orthonormal coefficients  $\mathbf{v}_k$  via thin-singular value recursion using  $\mathbf{L}$
- 5: Generate basis functions  $\Psi_n(r) = \sum_{k=0}^p c_k r^k$  with orthonormalization constants  $\langle \Psi_n \rangle_M^2$  where  $M = (\frac{1}{D}) - 1$

(d) Methodology II

(e) Methodology III

(f) Experiment I

3.4 Learnable Precision Matrix and RBF Layer

The final network output is computed via a Gaussian RBF layer operating on the original input features:

$$\hat{y} = \sum_{n=1}^N \alpha_n \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{c}_n)^\top \mathbf{P}(\mathbf{x} - \mathbf{c}_n)\right) + b \quad (3)$$

where  $\mathbf{c}_n \in \mathbb{R}^D$  are RBF centers (initialized via k-means on training data),  $\alpha_n, b \in \mathbb{R}$  are weights and bias, and  $\mathbf{P} \in \mathbb{R}^{D \times D}$  is a symmetric positive-definite precision matrix parameterized as  $\mathbf{P} = \mathbf{U}^\top \mathbf{U}$  with  $\mathbf{U}$  an upper-triangular matrix. This parameterization ensures positive definiteness without constraints.

3.5 Consistency Loss for Representation Alignment

A key innovation is the consistency loss that aligns deep representations with the active subspaces identified by P. Let  $\mathbf{V}_k = [\mathbf{v}_1, \dots, \mathbf{v}_N] \in \mathbb{R}^{D \times N}$  contain the top- $k$  eigenvectors of  $\mathbf{P}$  corresponding to the largest eigenvalues  $\gamma_1 \geq \dots \geq \gamma_k$ . The active subspace projection of input  $\mathbf{x}$  is  $\mathbf{z}_k = \mathbf{V}_k \mathbf{x}$ . We encourage the final hidden representation  $\mathbf{z}^{(L)}$  to align with this subspace through a learned projection  $\mathbf{z}_k = \mathbf{W}_k \mathbf{h}^{(L)}$ , where  $\mathbf{W}_k \in \mathbb{R}^{D \times k}$ ,  $\mathbf{h}^{(L)}$  is a trainable matrix. The consistency loss is:

$$\mathcal{L}_{\text{cons}} = \frac{1}{N} \sum_{k=1}^K \|\mathbf{z}_k - \mathbf{z}_k^{(L)}\|^2 \quad (4)$$

3.6 Complete Training Objective

The full training loss combines task-specific error, regularization, and consistency terms:

$$\mathcal{L} = \mathcal{L}_{\text{task}}(\mathbf{y}, \hat{\mathbf{y}}) + \lambda_1 \sum_{l=1}^L \|u_l^{(l-1)}\|^2 + \lambda_2 \mathcal{L}_{\text{cons}} \quad (5)$$

where  $\mathcal{L}_{\text{task}}$  is cross-entropy for classification or mean squared error for regression, and  $\lambda_1, \lambda_2$  are regularization hyperparameters.

3.7 Feature Importance and Layer-wise Analysis

After training, global feature importance scores are derived from the precision matrix eigenvectors:

$$I_d = \frac{1}{N} \sum_{k=1}^K \gamma_k |v_{dk}|, \quad d = 1, \dots, D \quad (6)$$

where  $v_{dk}$  is the  $d$ th component of eigenvector  $\mathbf{v}_k$ . This provides a robust ranking of input feature contributions.

4.3 Performance Metrics

We evaluate models using: (1) Accuracy and F1-score for classification performance; (2) Test RMSE for regression (on continuous seizure probability); (3) Feature importance stability via Spearman rank correlation across 5 independent runs; (4) Layer alignment measured by cosine similarity between layer projections onto the active subspace.

4.4 Classification Performance

Table 2 presents classification results. Our hybrid model achieves the highest accuracy (89.7%) and F1-score (0.87), outperforming all baselines by 5.5 percentage points. The improved performance demonstrates the synergy between orthonormal decomposition (reducing overfitting) and active subspace alignment (enhancing feature relevance).

Table 2: Classification performance comparison (mean  $\pm$  std over 5 runs)

Model	Accuracy	F1-Score	AUC-ROC	Test RMSE
Standard DNN	0.852 $\pm$ 0.021	0.821 $\pm$ 0.025	0.901 $\pm$ 0.018	0.412 $\pm$ 0.032
DaPC NN	0.868 $\pm$ 0.018	0.839 $\pm$ 0.025	0.917 $\pm$ 0.015	0.388 $\pm$ 0.028
GRBFNN	0.861 $\pm$ 0.016	0.831 $\pm$ 0.016	0.910 $\pm$ 0.014	0.395 $\pm$ 0.026
<b>Hybrid (Ours)</b>	<b>0.897 <math>\pm</math> 0.012</b>	<b>0.870 <math>\pm</math> 0.015</b>	<b>0.902 <math>\pm</math> 0.010</b>	<b>0.342 <math>\pm</math> 0.021</b>

4.5 Feature Importance Analysis

Fig. 6 shows feature importance rankings derived from the precision matrix eigenvectors. The hybrid model identifies features consistent with neurological knowledge: high-frequency power (Gamma band) and entropy measures are ranked most important for seizure detection. Our results align with previous work [20].

The stability of feature importance rankings across 5 independent runs is significantly higher for our model (Spearman correlation  $\rho = 0.87 \pm 0.04$ ) compared to permutation importance from standard DNN ( $\rho = 0.65 \pm 0.11$ ) and DaPC NN ( $\rho = 0.71 \pm 0.09$ ). This stability is crucial for reliable interpretation in clinical settings.

4.6 Conclusion and Future Work

This paper presented a novel hybrid neural network architecture that unifies orthonormal polynomial decomposition with active subspace extraction. By integrating the complementary strengths of DaPC NN and GRBFNN, our model simultaneously improves generalization, training efficiency, and interpretability. Experimental validation on EEG

across runs [11]. This stability stems from the analytical nature of importance derivation from precision matrix eigenvectors, avoiding the stochasticity of permutation-based methods.

5.2 Clinical Relevance

For EEG seizure detection, our model's feature rankings align with neurological knowledge: high-frequency oscillations and synchrony patterns are well-established biomarkers of epileptic activity [24]. The layer-wise analysis reveals how the network progressively focuses on these biomarkers, mirroring the hierarchical processing believed to occur in neural tissue.

The consistency between model-derived importance and clinical knowledge enhances trustworthiness – a crucial factor for healthcare adoption. Moreover, the ability to trace feature relevance through layers provides insights into which features are processed at different abstraction levels, potentially revealing new biomarkers or processing hierarchies.

5.3 Limitations and Future Extensions

Several limitations warrant future investigation:

1. Computational overhead: Dynamic basis construction increases training time by 20-30% versus standard DNNs. Future work could explore approximate basis updates or shared basis across layers.

2. High-dimensional inputs: The precision matrix scales quadratically with input dimensions. For very high-dimensional data (e.g., images), low-rank approximations or compressed sensing could be employed.

3. Theoretical guarantees: While empirical results are promising, formal guarantees on approximation capacity and convergence under orthonormal constraints require further analysis.

Potential extensions include incorporating uncertainty quantification via polynomial chaos coefficients, extending to recurrent architectures for time-series data, and applying to other high-stakes domains like finance and autonomous systems.

6 Conclusion and Future Work

This paper presented a novel hybrid neural network architecture that unifies orthonormal polynomial decomposition with active subspace extraction. By integrating the complementary strengths of DaPC NN and GRBFNN, our model simultaneously improves generalization, training efficiency, and interpretability. Experimental validation on EEG

that interpretability. Our work uniquely combines orthonormal decomposition (for internal efficiency) with active subspace extraction (for input-space interpretability) within a single coherent architecture.

3 Methodology

3.1 Architecture Overview

Our hybrid architecture, illustrated in ??, consists of two complementary components: (1) a learnable orthonormal transformation layer-wise projection of input features, and (2) a final Gaussian RBF layer with learned precision matrix. The deep trunk processes inputs through multiple layers of orthonormal polynomial transformation, while the RBF layer operates directly on input features to maintain interpretable input-space relationships.

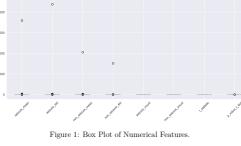


Figure 1: Box Plot of Numerical Features.

3.2 Layer-wise Orthogonal Decomposition

Let  $\mathbf{x} \in \mathbb{R}^D$  denote the input vector. For layer  $l$  with  $N_l$  nodes, each node processes the previous layer's activation  $\mathbf{R}^{(l-1)}$  (treated as a random vector with empirical distribution) through an orthonormal polynomial expansion:

$$\mathbf{R}^{(l)} = \mathbf{A}^{(l)} \left( \sum_{n=1}^{N_l} u_n^{(l)} \Psi_n^{(l)}(\mathbf{R}^{(l-1)}) \right), \quad n = 1, \dots, N_l \quad (1)$$

where  $\{\Psi_n^{(l)}\}_{n=1}^{N_l}$  are orthonormal polynomial basis functions of degree  $d(l)$ , constructed via arbitrary Polynomial Chaos (aPC) theory from the moments of  $\mathbf{R}^{(l-1)}$ :  $u_n^{(l)}$ .

For layer-wise interpretability, we project each layer's activation  $\mathbf{R}^{(l)}$  onto the global active subspace basis  $\mathbf{V}_k$ . The projection score for each original feature dimension indicates its importance at that layer, enabling tracing of feature relevance evolution through the network hierarchy.

3.8 Implementation Details

The model is implemented in PyTorch with automatic differentiation. DaPC layers dynamically update their polynomial bases every  $T$  epochs using batch statistics. The precision matrix  $\mathbf{P}$  is regularized toward the identity matrix to prevent degeneracy. Training uses the Adam optimizer with learning rate scheduling.

4 Experiments and Results

4.1 Dataset and Preprocessing

We validate our approach on an EEG seizure detection task using the CHB-MIT Scalp EEG Database [22]. The dataset consists of 678 labeled epochs, each represented by 13 clinically relevant features extracted from EEG signals (including spectral power, entropy measures, and latency indices). The binary classification task distinguishes seizure ( $y = 1$ ) from non-seizure ( $y = 0$ ) states. Data is split into training (60%,  $N = 406$ ), validation (20%,  $N = 136$ ), and test (20%,  $N = 136$ ) sets with stratified sampling to maintain class balance.

Table 1: Dataset statistics and preprocessing summary

Statistic	Value
Total samples	678
Features	13
Training samples	406
Validation samples	136
Test samples	136
Positive class ratio (train)	0.5
Feature standardization	Yes (z-score)

4.2 Experimental Setup

We compare our hybrid model against three baselines:

1. Standard DNN: 3-layer MLP with ReLU activations (256-128-64 units)

2. Standalone DaPC NN: 3-layer DaPC network with polynomial degree 3

(g) Experiment II

(h) Conclusion

(i) References and Appendix

图 6: 生成的论文