Department of Electrical
and Computer Engineering
Cullen College of Engineering

GPT for Research

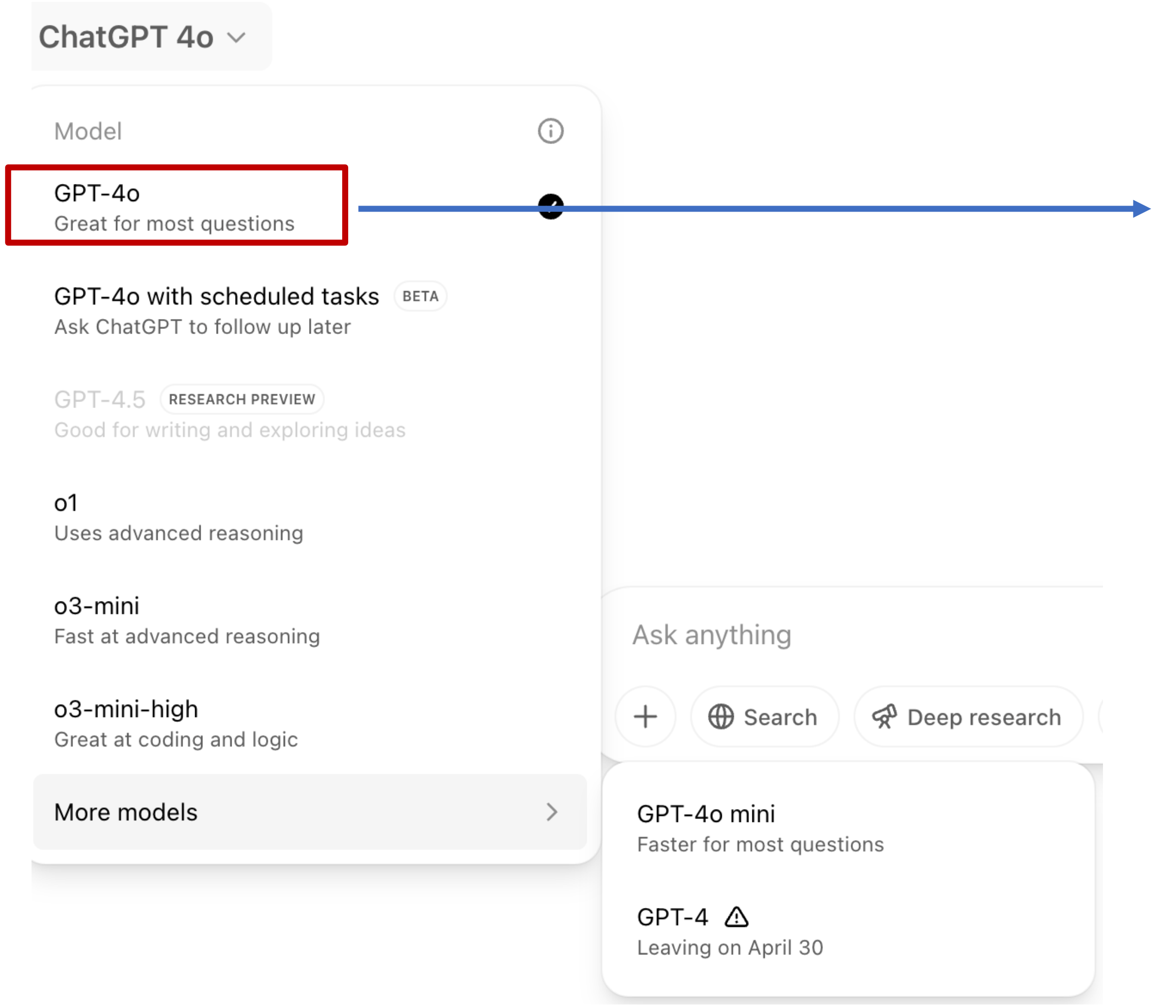
Literature searching, problem solving, and coding

Presented by: Weimin Yuan & Zijun Zhan

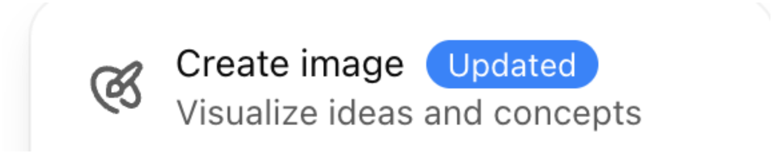
Outline

- Introduce the current mainstream LLMs.
- LLM for research
- LLM for problem solving
- LLM for coding

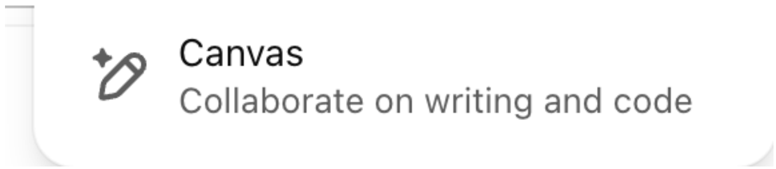
Current mainstream LLMs



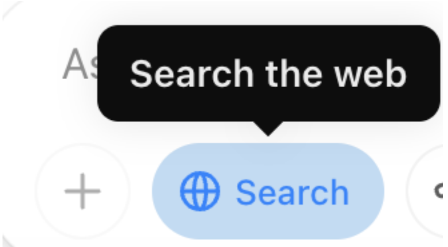
- ➢ Suitable for **everyday queries**, including translation, key point extraction, and concept explanation
- ➢ Supports multimodal input, including images, audio, documents, etc.
- ➢ Equipped with powerful drawing and Photoshop capabilities



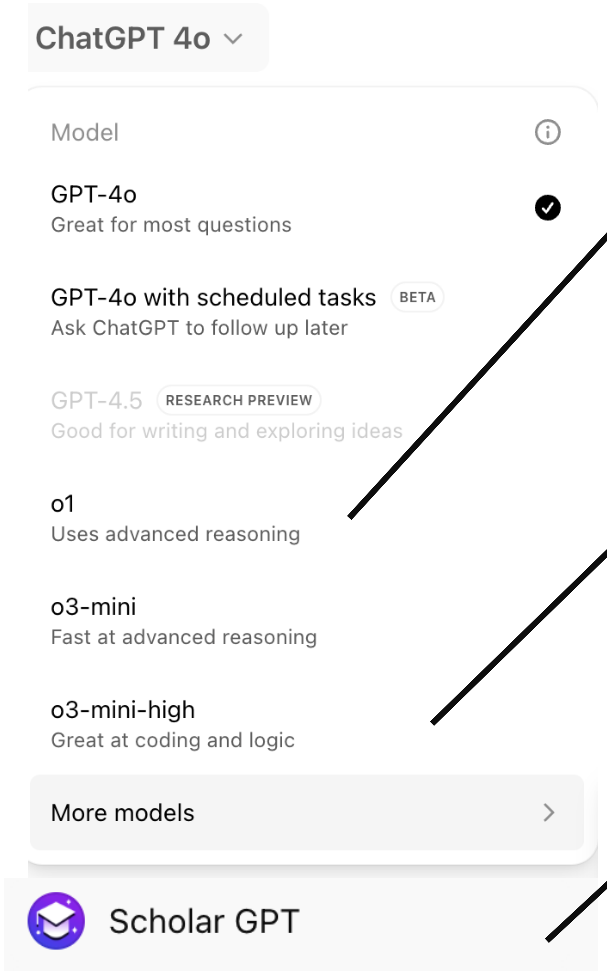
- ➢ Suitable for **writing simple code** and supporting **revision** suggestions **directly on the canvas**.



- ➢ Search the web



Current mainstream LLMs


ChatGPT 4o
Model
GPT-4o
Great for most questions
GPT-4o with scheduled tasks
BETA
Ask ChatGPT to follow up later
GPT-4.5
RESEARCH PREVIEW
Good for writing and exploring ideas
o1
Uses advanced reasoning
o3-mini
Fast at advanced reasoning
o3-mini-high
Great at coding and logic
More models
Scholar GPT

GPT-o1: Leverages advanced reasoning capabilities

- Excels at complex reasoning tasks and can **handle sophisticated problem-solving** scenarios
- Capable of generating **longer** and more **intricate code** implementations
- Supports multimodal input, including text, images, audio, and more

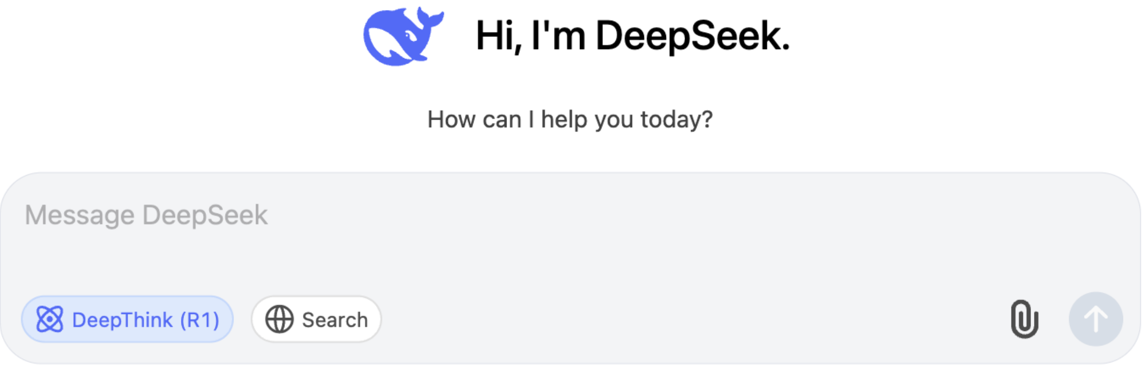
GPT-o3: Fast and capable in advanced reasoning

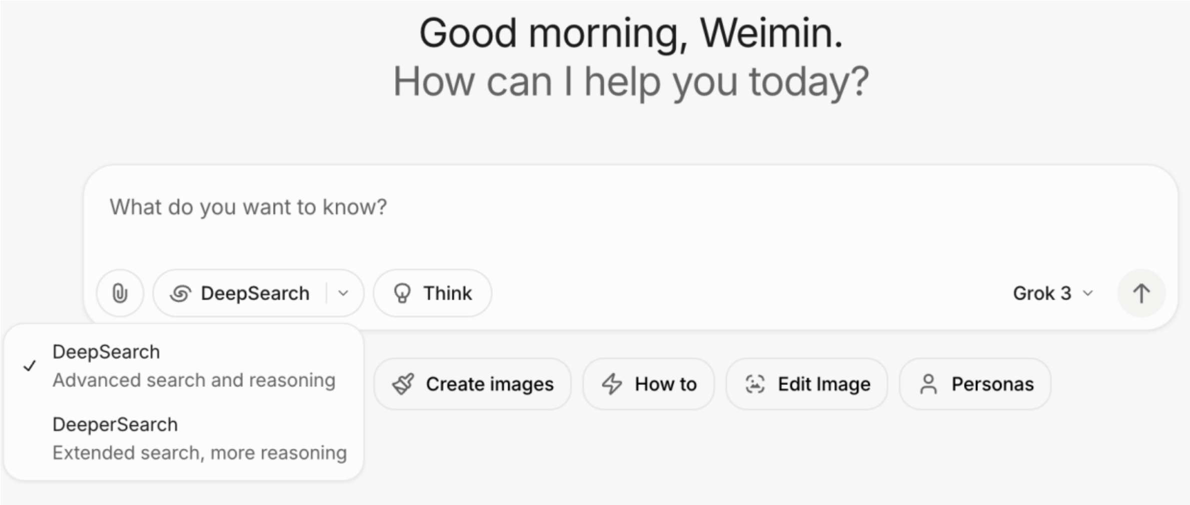
- Enables visibility into its reasoning process
- Strong logical inference capabilities
- Excellent at programming and solving logic problems
- **Deep research**, quickly retrieving relevant academic literature

Scholar GPT: Enhance research with **200M+ resources** and built-in **critical reading skills**. Access Google Scholar, PubMed, bioRxiv, arXiv, and more, effortlessly.

- Ideal for academic inquiries, such as **explaining technical terms**
- Capable of retrieving and analyzing scholarly papers
- Assists in reading and **summarizing research literature**

Current mainstream LLMs




Good morning, Weimin.
How can I help you today?
What do you want to know?
DeepSearch
Think
Grok 3
DeepSearch
Advanced search and reasoning
DeeperSearch
Extended search, more reasoning
Create images
How to
Edit Image
Personas

DeepSeek

Link: https://chat.deepseek.com/

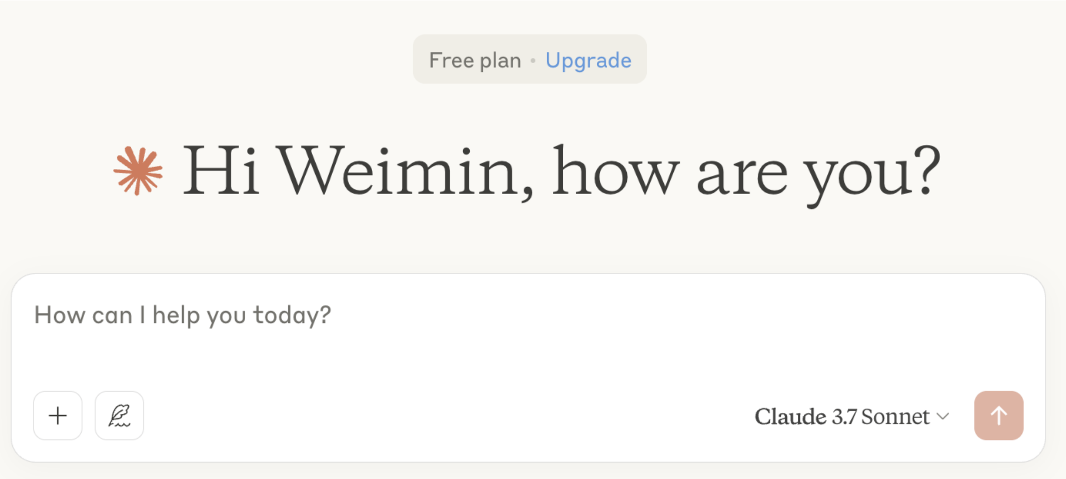
Deepseek-R1 **excels in reasoning** and consistently ranks among the top across various benchmarks, **particularly for logic-intensive tasks**. However, it has a relatively high hallucination rate and may fabricate false content.

Grok

Link: https://grok.com/

You can use the **Deepsearch** for free, which is similar to ChatGPT's **Deep Research**. The **Think** is also free, but comes with a daily quota and does not require a Plus subscription.

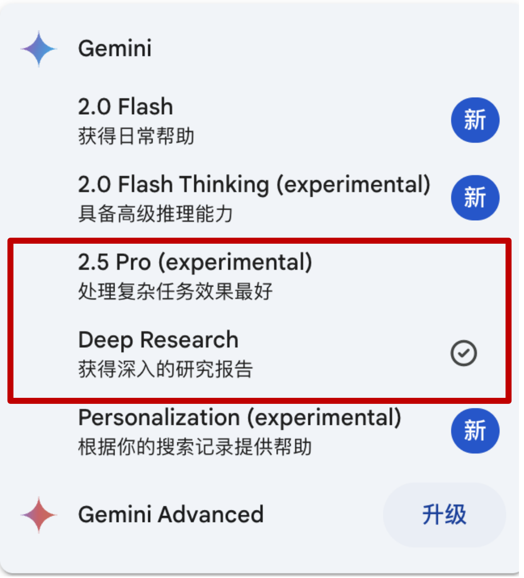
Current mainstream LLMs



Free plan · Upgrade

Hi Weimin, how are you?

How can I help you today?



2.0 Flash
获得日常帮助

2.0 Flash Thinking (experimental)
具备高级推理能力

2.5 Pro (experimental)
处理复杂任务效果最好

Deep Research
获得深入的研究报告

Personalization (experimental)
根据你的搜索记录提供帮助

Claude

Link: https://claude.ai/new

Claude 3.7 Sonnet offers a daily quota for free usage. It has **exceptionally strong coding capabilities**, including the ability to generate **long and complex code**, making it **highly recommended**. However, **only version 3.7 Sonnet is recommended** for this purpose.

Gemini

Link: https://gemini.google.com/app

- **Deep Research**: Leveraging the powerful capabilities of Google Search and integration with Google Scholar, it offers **robust retrieval abilities** across a wide range of sources, including websites and blogs.
- **2.5 Pro**: A newly released model from last week, it **outperforms Claude 3.7 Sonnet in coding tasks**.

Only these two models are recommended for use.

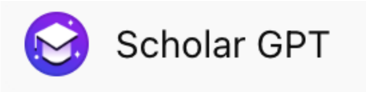
Outline

- Introduce the current mainstream LLMs.
- LLM for research
- LLM for problem solving
- LLM for coding

LLM for Research — High-Level Overview

When I have some idea, here is my workflow

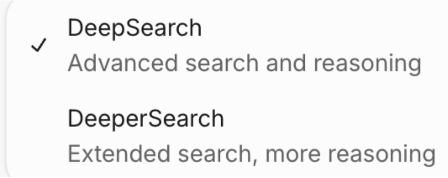
Step 1



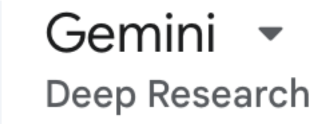
Help me **quickly understand the relevant concepts** and the **existing work** in this field; also, let it initially help me **find** some **related papers**.

Step 2

Grok


✓ DeepSearch
Advanced search and reasoning
DeeperSearch
Extended search, more reasoning

ChatGPT



For Plus users, **GPT Deep Research** offers 10 free uses per month, and it's personally the most powerful tool in my opinion.

Grok and **Gemini** provide free daily quotas, helping me quickly search for literature and summarize it.

Step 3

Scholar GPT or ChatGPT-o1 or o3

For **details in a paper**, you can input the PDF into Scholar GPT and let it interpret it.

For **equation derivations**, I first use GPT-4o to convert the formulas into LaTeX code, and then input them into GPT-o1 or GPT-o3 for explanation.

LLM for Research — Literature Search

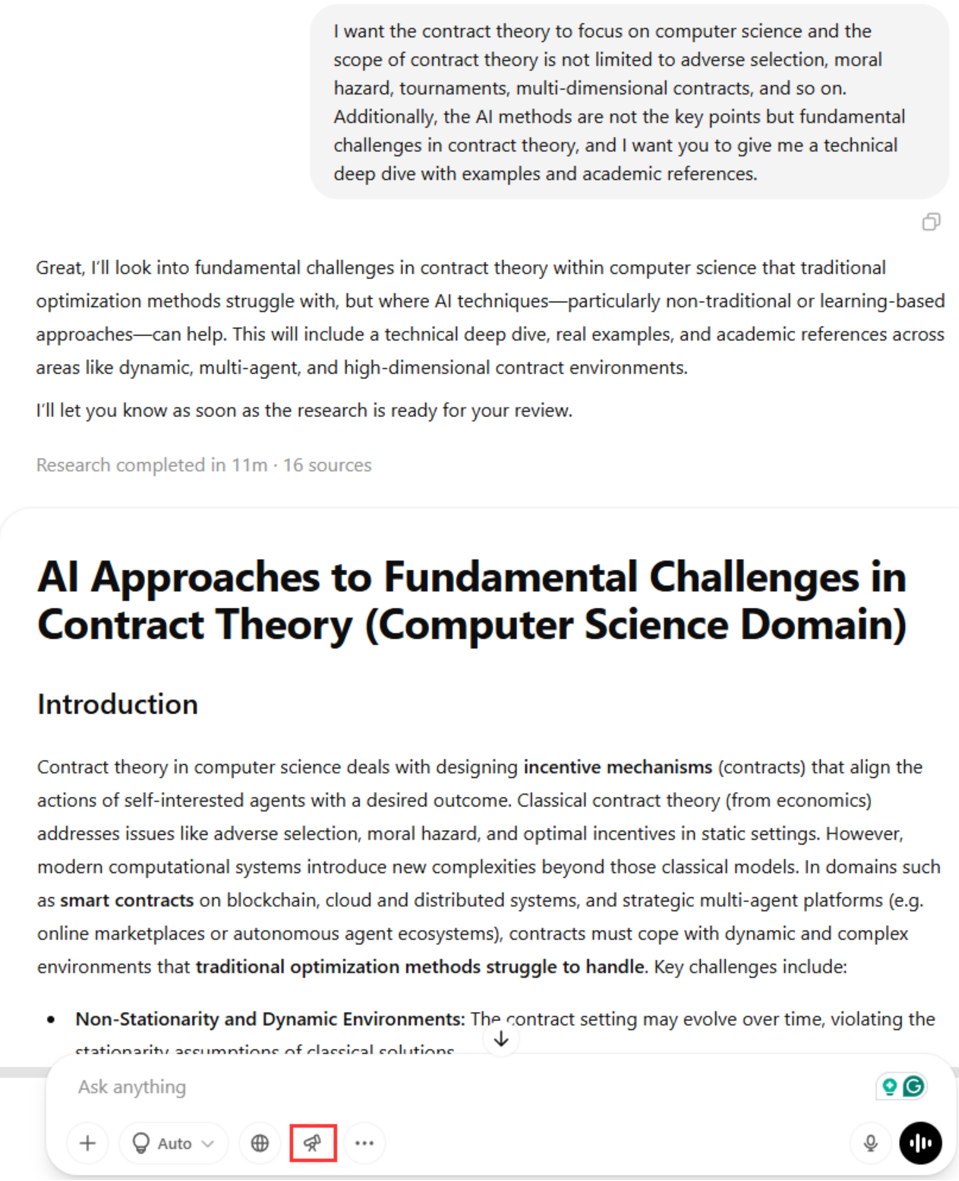
Deep Research of ChatGPT

- **Generate** up-to-date literature review
- **Explore** research direction
- **Obtain** literature searching key words
- **Fresh** our mind

Prompt Instance

I want the **contract theory** to focus on **computer science** and the scope of contract theory is not limited to **adverse selection, moral hazard, tournaments, multi-dimensional contracts**, and so on. Additionally, the AI methods are not the key points but **fundamental challenges in contract theory**, and I want you to give me a technical deep dive with examples and academic references.

ChatGPT



I want the contract theory to focus on computer science and the scope of contract theory is not limited to adverse selection, moral hazard, tournaments, multi-dimensional contracts, and so on. Additionally, the AI methods are not the key points but fundamental challenges in contract theory, and I want you to give me a technical deep dive with examples and academic references.

Great, I'll look into fundamental challenges in contract theory within computer science that traditional optimization methods struggle with, but where AI techniques—particularly non-traditional or learning-based approaches—can help. This will include a technical deep dive, real examples, and academic references across areas like dynamic, multi-agent, and high-dimensional contract environments.

I'll let you know as soon as the research is ready for your review.

Research completed in 11m · 16 sources

AI Approaches to Fundamental Challenges in Contract Theory (Computer Science Domain)

Introduction

Contract theory in computer science deals with designing **incentive mechanisms** (contracts) that align the actions of self-interested agents with a desired outcome. Classical contract theory (from economics) addresses issues like adverse selection, moral hazard, and optimal incentives in static settings. However, modern computational systems introduce new complexities beyond those classical models. In domains such as **smart contracts** on blockchain, cloud and distributed systems, and strategic multi-agent platforms (e.g. online marketplaces or autonomous agent ecosystems), contracts must cope with dynamic and complex environments that **traditional optimization methods struggle to handle.** Key challenges include:

- **Non-Stationarity and Dynamic Environments:** The contract setting may evolve over time, violating the stationarity assumptions of classical solutions.

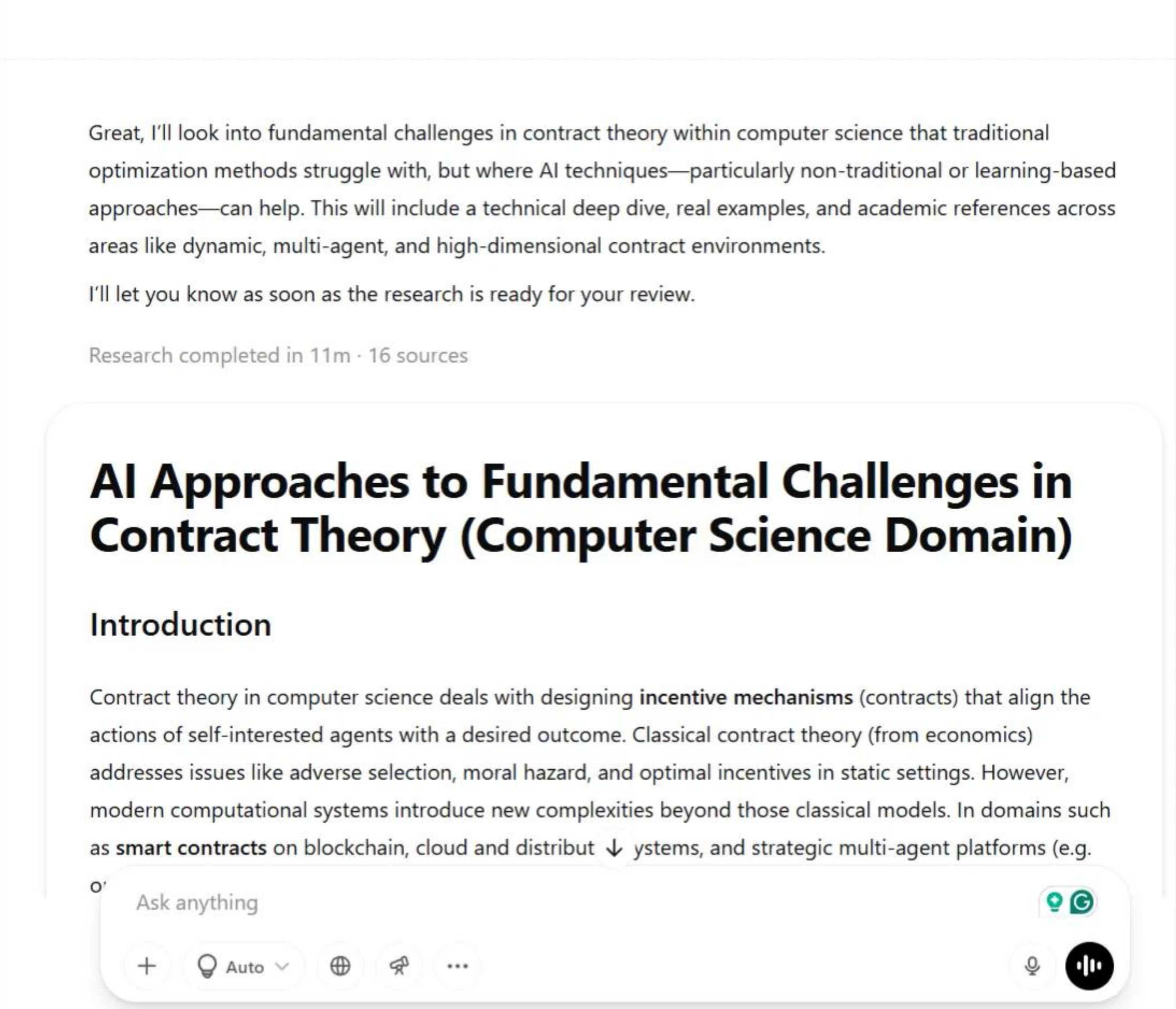
LLM for Research — Literature Search

Deep Research of ChatGPT

- **Generate** up-to-date literature review
- **Explore** research direction
- **Obtain** literature searching key words
- **Fresh** our mind

Prompt Instance

I want the **contract theory** to focus on **computer science** and the scope of contract theory is not limited to **adverse selection, moral hazard, tournaments, multi-dimensional contracts**, and so on. Additionally, the AI methods are not the key points but **fundamental challenges in contract theory**, and I want you to give me a technical deep dive with examples and academic references.



Great, I'll look into fundamental challenges in contract theory within computer science that traditional optimization methods struggle with, but where AI techniques—particularly non-traditional or learning-based approaches—can help. This will include a technical deep dive, real examples, and academic references across areas like dynamic, multi-agent, and high-dimensional contract environments.

I'll let you know as soon as the research is ready for your review.

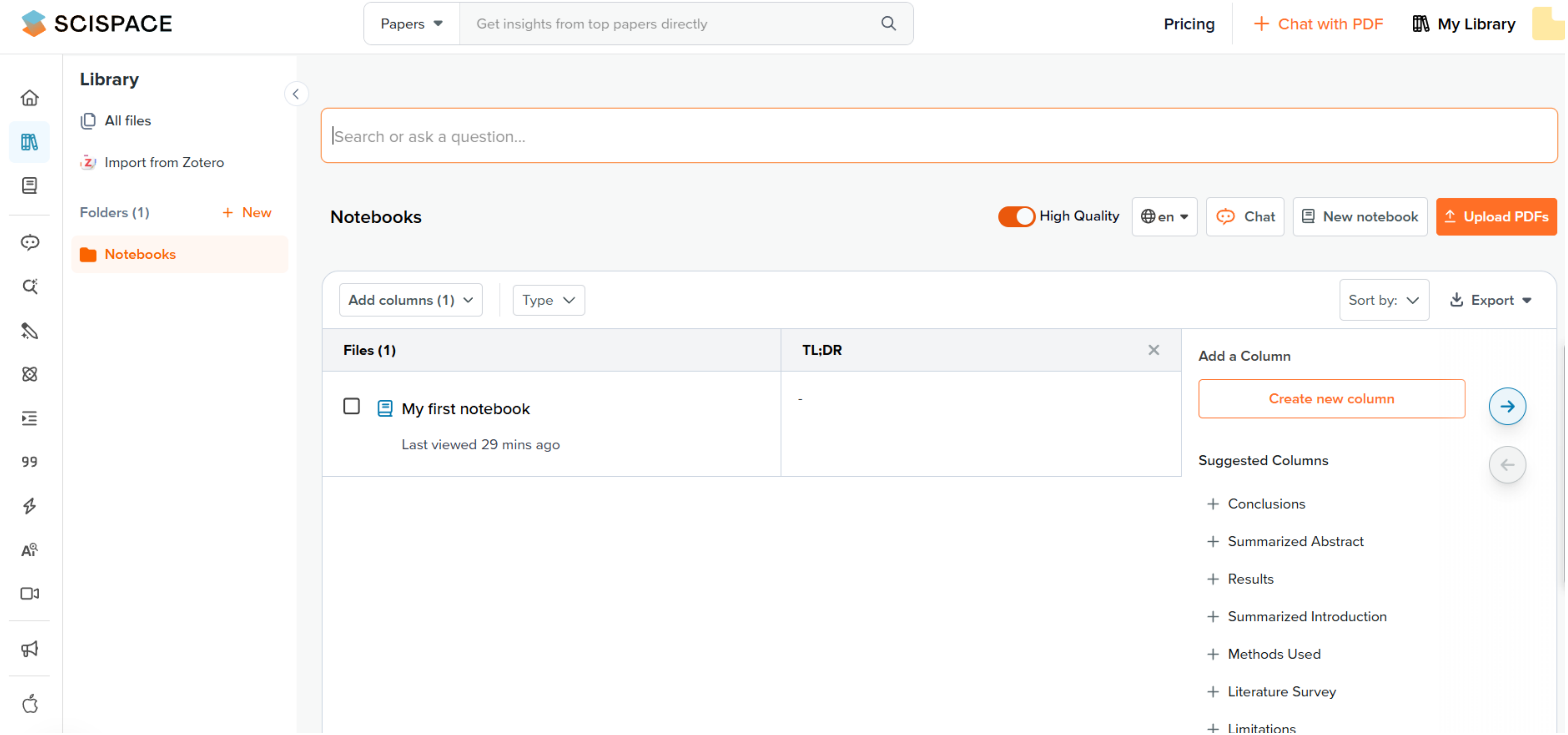
Research completed in 11m · 16 sources

AI Approaches to Fundamental Challenges in Contract Theory (Computer Science Domain)

Introduction

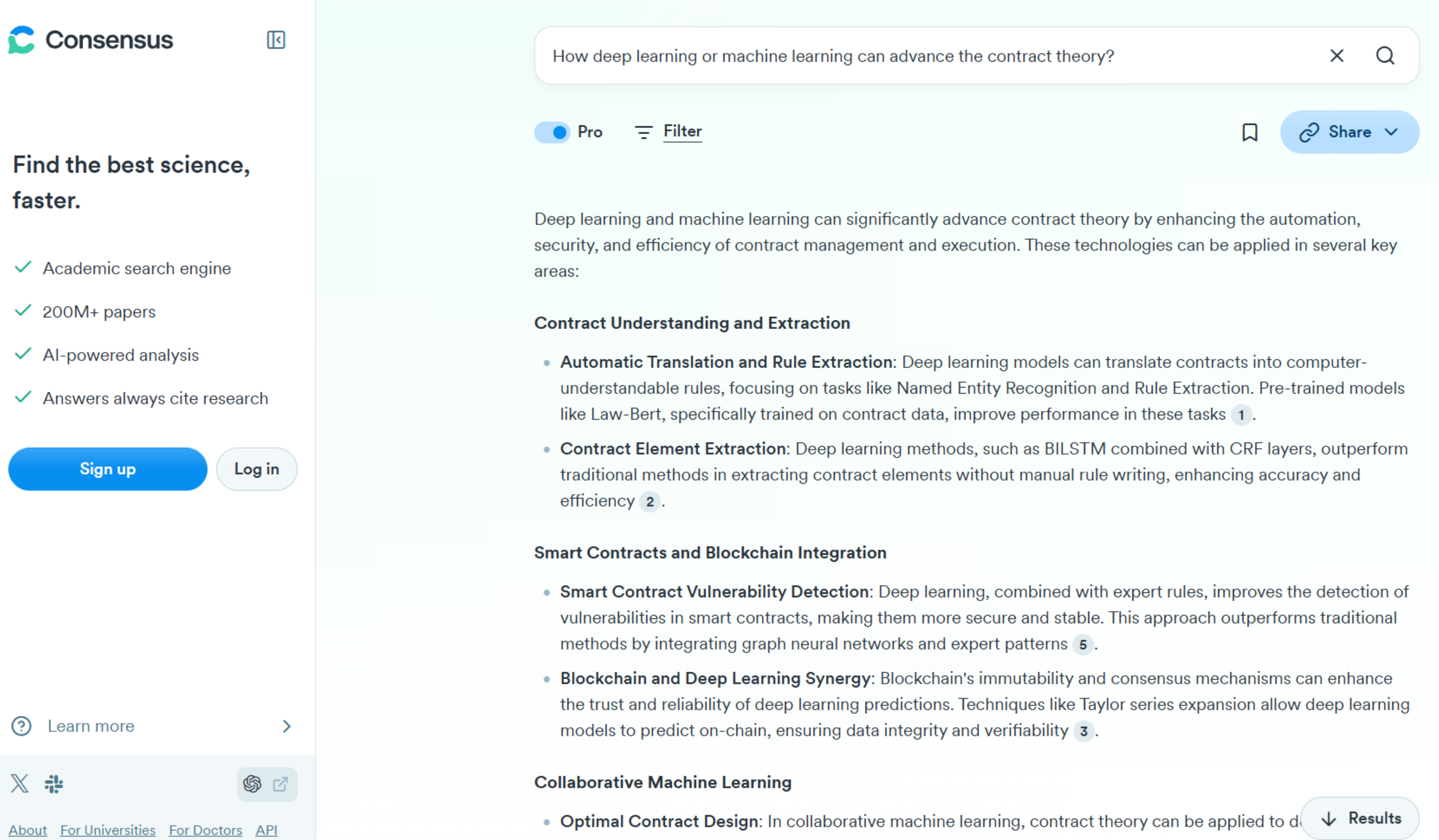
Contract theory in computer science deals with designing **incentive mechanisms** (contracts) that align the actions of self-interested agents with a desired outcome. Classical contract theory (from economics) addresses issues like adverse selection, moral hazard, and optimal incentives in static settings. However, modern computational systems introduce new complexities beyond those classical models. In domains such as **smart contracts** on blockchain, cloud and distribut ↓ ystems, and strategic multi-agent platforms (e.g.

LLM for Research — Literature Search



Files (1)	TL;DR
☐ My first notebook Last viewed 29 mins ago	-

LLM for Research — Literature Search



Find the best science, faster.

- ✓ Academic search engine
- ✓ 200M+ papers
- ✓ AI-powered analysis
- ✓ Answers always cite research

How deep learning or machine learning can advance the contract theory?

Deep learning and machine learning can significantly advance contract theory by enhancing the automation, security, and efficiency of contract management and execution. These technologies can be applied in several key areas:

Contract Understanding and Extraction

- **Automatic Translation and Rule Extraction**: Deep learning models can translate contracts into computer-understandable rules, focusing on tasks like Named Entity Recognition and Rule Extraction. Pre-trained models like Law-Bert, specifically trained on contract data, improve performance in these tasks 1.
- **Contract Element Extraction**: Deep learning methods, such as BILSTM combined with CRF layers, outperform traditional methods in extracting contract elements without manual rule writing, enhancing accuracy and efficiency 2.

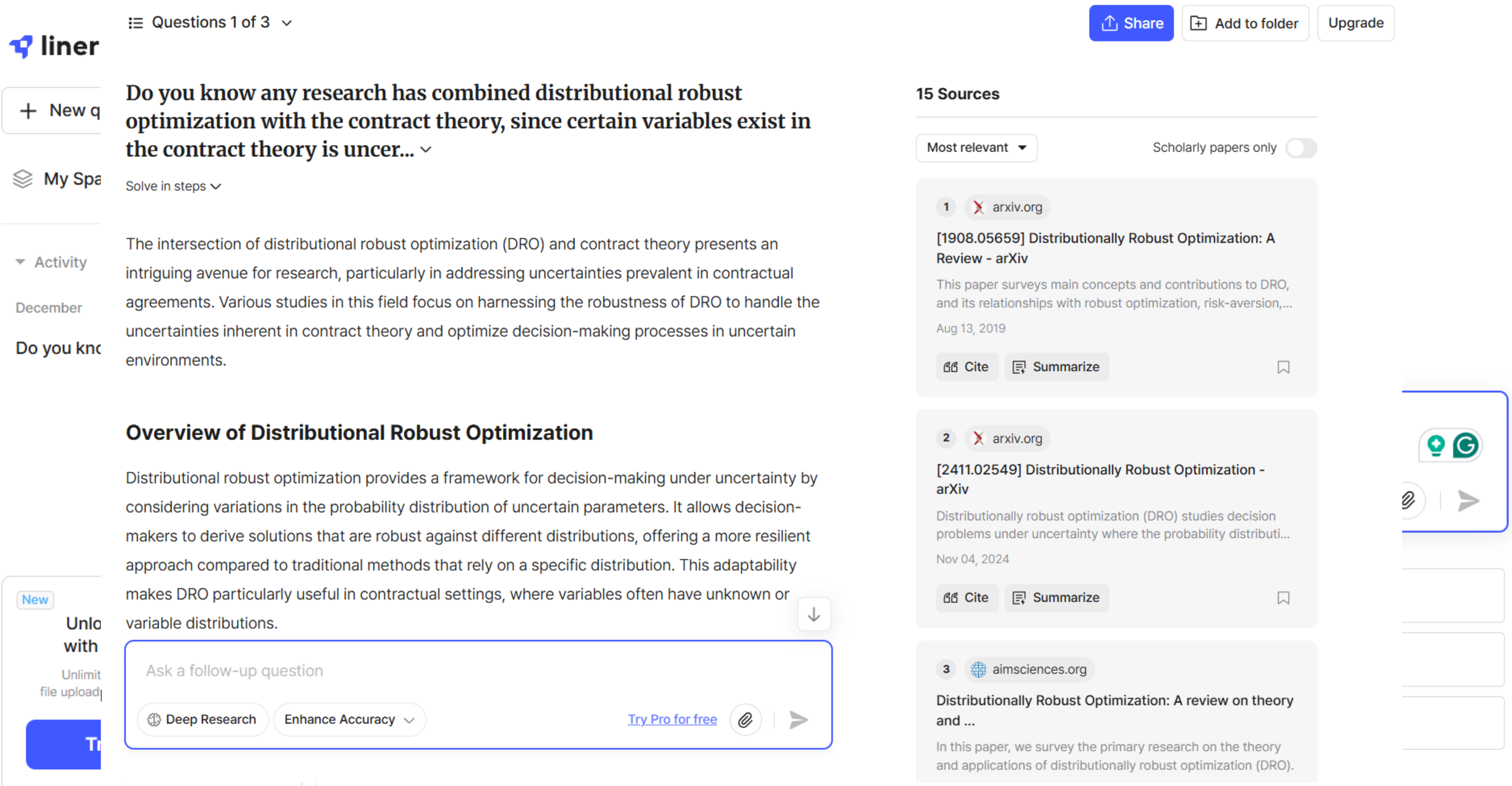
Smart Contracts and Blockchain Integration

- **Smart Contract Vulnerability Detection**: Deep learning, combined with expert rules, improves the detection of vulnerabilities in smart contracts, making them more secure and stable. This approach outperforms traditional methods by integrating graph neural networks and expert patterns 5.
- **Blockchain and Deep Learning Synergy**: Blockchain's immutability and consensus mechanisms can enhance the trust and reliability of deep learning predictions. Techniques like Taylor series expansion allow deep learning models to predict on-chain, ensuring data integrity and verifiability 3.

Collaborative Machine Learning

- **Optimal Contract Design**: In collaborative machine learning, contract theory can be applied to d

LLM for Research — Literature Search



Do you know any research has combined distributional robust optimization with the contract theory, since certain variables exist in the contract theory is uncer...

The intersection of distributional robust optimization (DRO) and contract theory presents an intriguing avenue for research, particularly in addressing uncertainties prevalent in contractual agreements. Various studies in this field focus on harnessing the robustness of DRO to handle the uncertainties inherent in contract theory and optimize decision-making processes in uncertain environments.

Overview of Distributional Robust Optimization

Distributional robust optimization provides a framework for decision-making under uncertainty by considering variations in the probability distribution of uncertain parameters. It allows decision-makers to derive solutions that are robust against different distributions, offering a more resilient approach compared to traditional methods that rely on a specific distribution. This adaptability makes DRO particularly useful in contractual settings, where variables often have unknown or variable distributions.

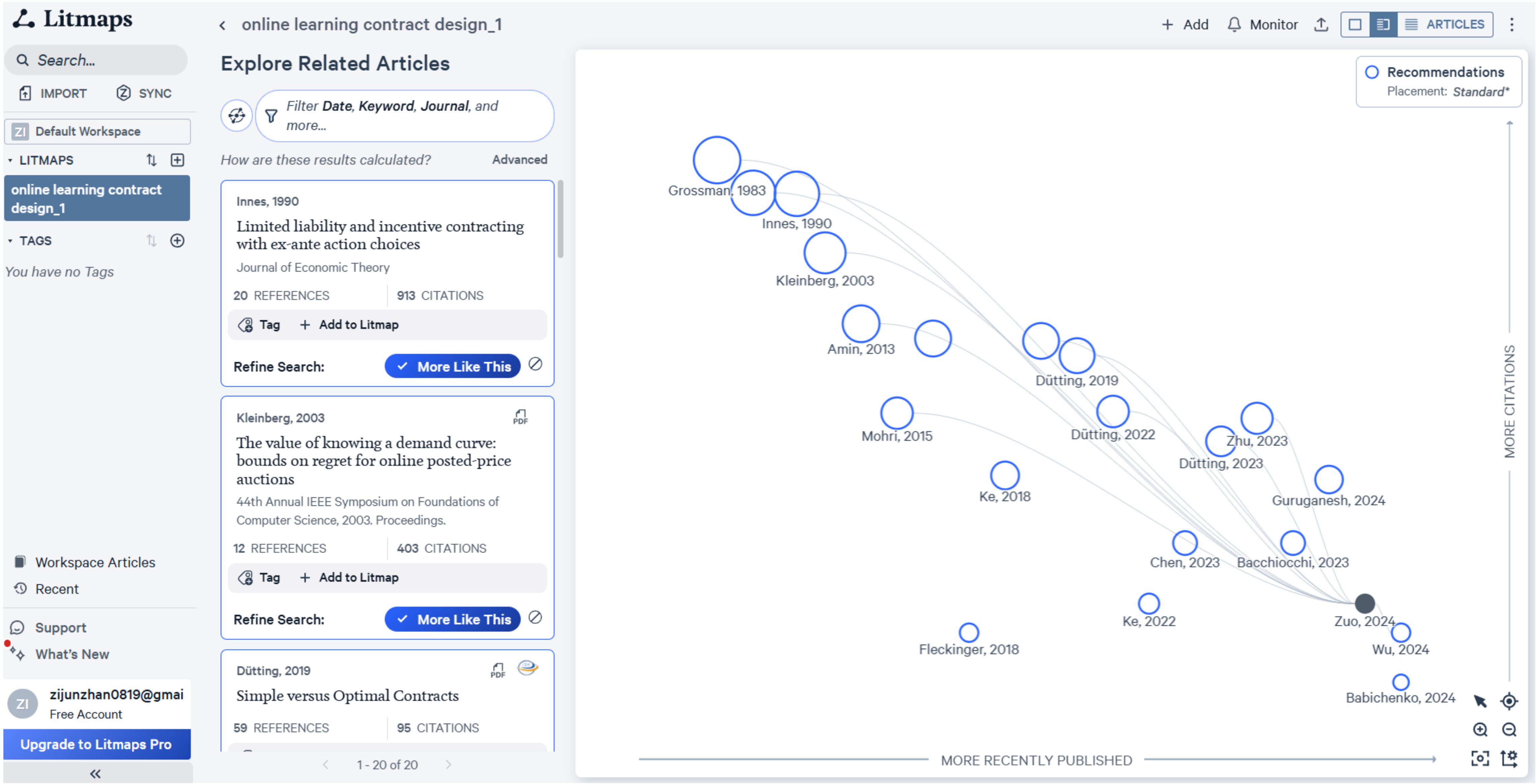
15 Sources

1 arxiv.org
[1908.05659] Distributionally Robust Optimization: A Review - arXiv
This paper surveys main concepts and contributions to DRO, and its relationships with robust optimization, risk-aversion,...
Aug 13, 2019

2 arxiv.org
[2411.02549] Distributionally Robust Optimization - arXiv
Distributionally robust optimization (DRO) studies decision problems under uncertainty where the probability distributi...
Nov 04, 2024

3 aimsciences.org
Distributionally Robust Optimization: A review on theory and ...
In this paper, we survey the primary research on the theory and applications of distributionally robust optimization (DRO).

LLM for Research — Literature Mapping



Explore Related Articles

Filter *Date*, *Keyword*, *Journal*, and more...

How are these results calculated? Advanced

Innes, 1990

Limited liability and incentive contracting with ex-ante action choices

Journal of Economic Theory

20 REFERENCES | 913 CITATIONS

Tag + Add to Litmap

Refine Search: ✓ More Like This

Kleinberg, 2003

The value of knowing a demand curve: bounds on regret for online posted-price auctions

44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.

12 REFERENCES | 403 CITATIONS

Tag + Add to Litmap

Refine Search: ✓ More Like This

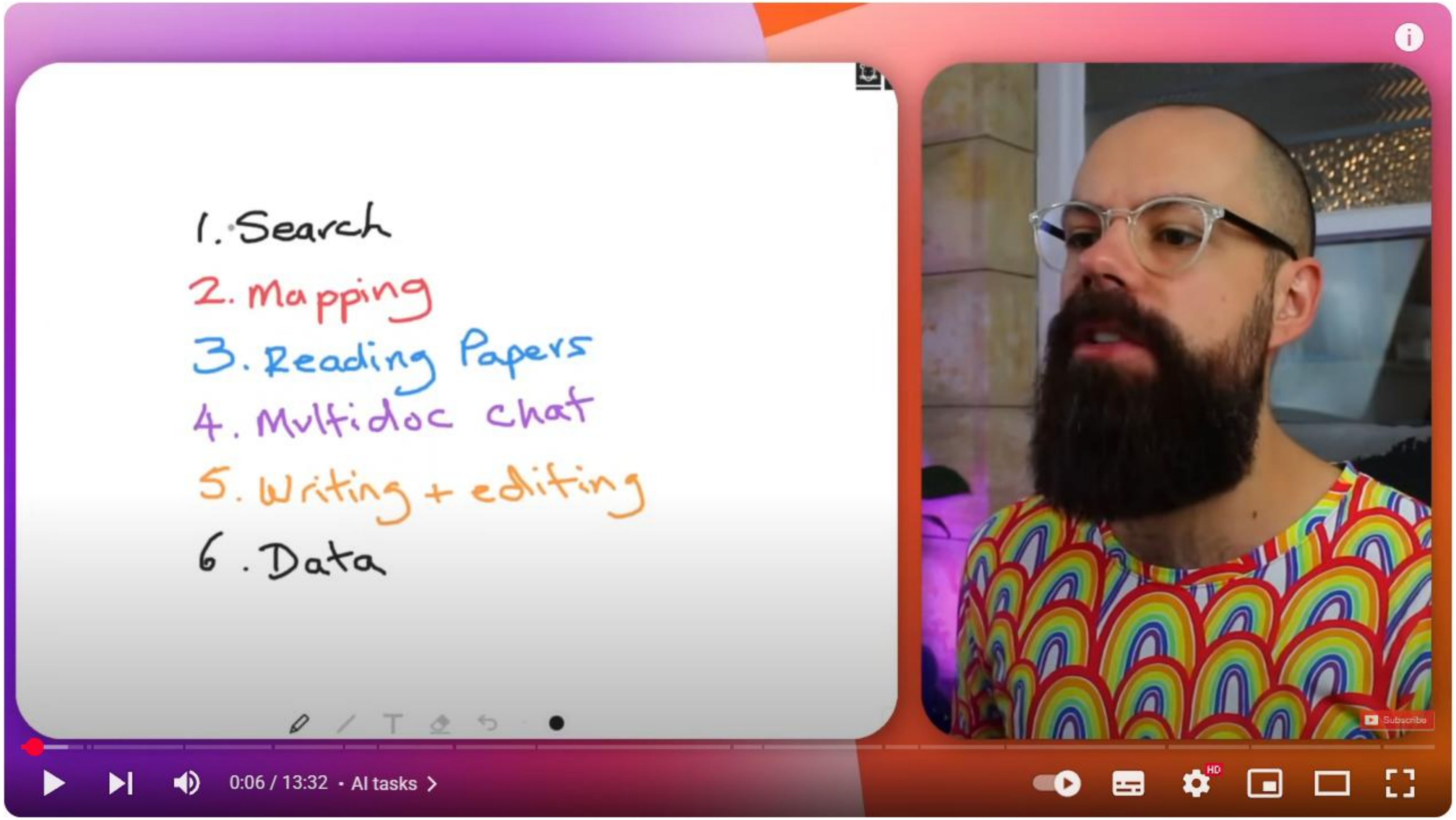
Dütting, 2019

Simple versus Optimal Contracts

59 REFERENCES | 95 CITATIONS

1 - 20 of 20

LLM for Research — Online Video Resource



1. Search
2. Mapping
3. Reading Papers
4. Multidoc chat
5. Writing + editing
6. Data

The Best AI Tools for Academia in 2025 - Stop Searching, Start Using!

https://youtu.be/IBsBixGTh-I

Outline

- Introduce the current mainstream LLMs
- LLM for research
- **LLM for problem solving**
- LLM for coding

LLM for Problem Solving — via ChatGPT

$$\sup_{\mathbf{L},\lambda,\mathbf{s}} -\lambda\varepsilon + \sum_{n=1}^{N} s_n$$

$$\text{s.t.}\quad \inf_{\xi}\Bigg(\sum_{i=1}^{I}\alpha_i\Bigg[\ln(\gamma_2\xi+\gamma_3 L_i) - \frac{\gamma_1 L_1}{\theta_1} -$$

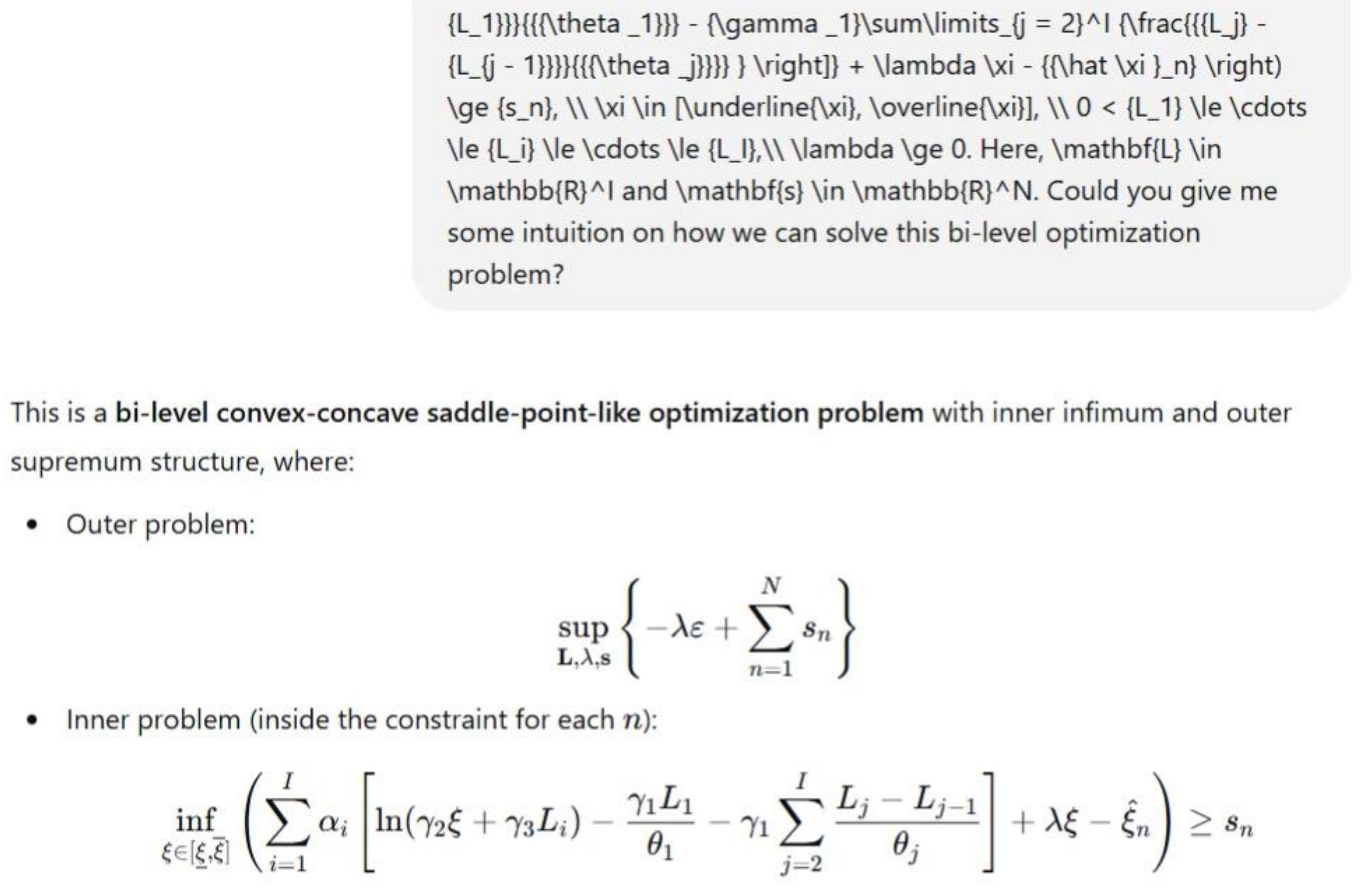
$$\xi \in [\underline{\xi}, \overline{\xi}],$$

$$0 < L_1 \le \cdots \le L_i \le \cdots \le L_I,$$

$$\lambda \ge 0.$$

Prompt In

\mathop {\sup }\limits_{{\mathbf{L}},\lambda
\sum\limits_{n = 1}^N {{s_n}} \\ {\rm{s}}{\rm{
\left(\sum\limits_{i = 1}^I {{\alpha _i}\left[{\l
\frac{{{\gamma _1}{L_1}}}{{{\theta _1}}} - {\ga
{L_{j - 1}}}}{{{\theta _j}}}} } \right]} + \lambda
[\underline{\xi}, \overline{\xi}], \\ 0 < {L_1}
\lambda \ge 0. Here, \mathbf{L} \in \mathb
Could you give me some **intuition on** how
problem?



{L_1}}}{{{\theta _1}}} - {\gamma _1}\sum\limits_{j = 2}^I {\frac{{{L_j} - {L_{j - 1}}}}{{{\theta _j}}}} } \right]} + \lambda \xi - {{\hat \xi }_n} \right) \ge {s_n}, \\ \xi \in [\underline{\xi}, \overline{\xi}], \\ 0 < {L_1} \le \cdots \le {L_i} \le \cdots \le {L_I},\\ \lambda \ge 0. Here, \mathbf{L} \in \mathbb{R}^I and \mathbf{s} \in \mathbb{R}^N. Could you give me some intuition on how we can solve this bi-level optimization problem?

This is a **bi-level convex-concave saddle-point-like optimization problem** with inner infimum and outer supremum structure, where:

- Outer problem:

$$\sup_{\mathbf{L},\lambda,\mathbf{s}}\left\{-\lambda\varepsilon + \sum_{n=1}^{N} s_n\right\}$$

- Inner problem (inside the constraint for each n):

$$\inf_{\xi\in[\underline{\xi},\overline{\xi}]}\left(\sum_{i=1}^{I}\alpha_i\left[\ln(\gamma_2\xi+\gamma_3 L_i) - \frac{\gamma_1 L_1}{\theta_1} - \gamma_1\sum_{j=2}^{I}\frac{L_j - L_{j-1}}{\theta_j}\right] + \lambda\xi - \hat{\xi}_n\right) \ge s_n$$

LLM for Problem Solving — via ChatGPT

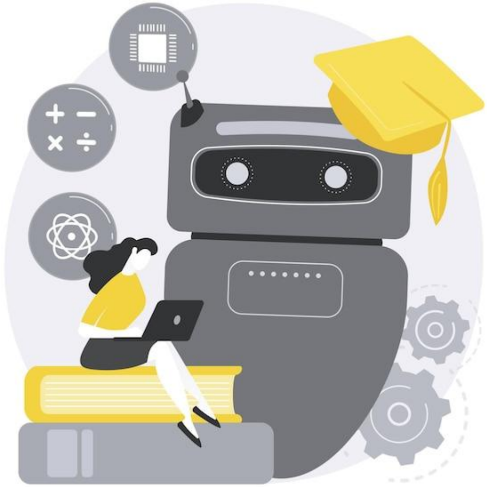
$$\sup_{\mathbf{L},\lambda,\mathbf{s}} -\lambda\varepsilon + \sum_{n=1}^{N} s_n$$

$$\text{s.t.} \quad \inf_{\xi} \left(\sum_{i=1}^{I} \alpha_i \left[\ln(\gamma_2\xi + \gamma_3 L_i) - \frac{\gamma_1 L_1}{\theta_1} - \gamma_1 \sum_{j=2}^{I} \frac{L_j - L_{j-1}}{\theta_j} \right] + \lambda\xi - \hat{\xi}_n \right) \ge s_n,$$

$$\xi \in [\underline{\xi}, \overline{\xi}],$$

$$0 < L_1 \le \cdots \le L_i \le \cdots \le L_I,$$

$$\lambda \ge 0.$$



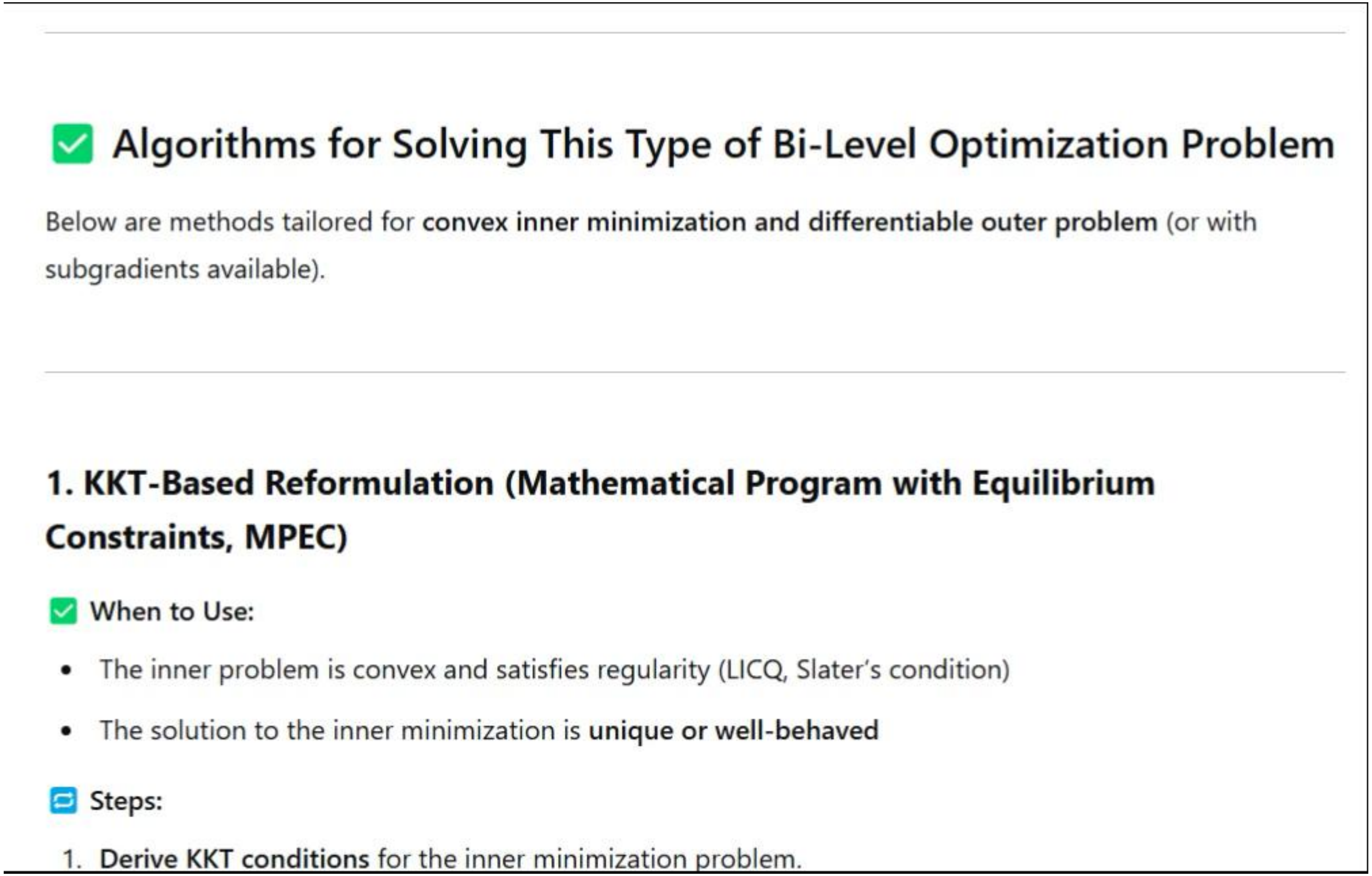
Prompt Instance

\mathop {\sup }\limits_{{\mathbf{L}},\lambda ,{\mathbf{s}}} - \lambda \varepsilon + \sum\limits_{n = 1}^N {{s_n}} \\ {\rm{s}}{\rm{.t}}{\rm{. }} \quad \mathop {\inf }\limits_\xi \left(\sum\limits_{i = 1}^I {{\alpha _i}\left[{\ln ({\gamma _2}\xi + {\gamma _3}{L_i}) - \frac{{{\gamma _1}{L_1}}}{{{\theta _1}}} - {\gamma _1}\sum\limits_{j = 2}^I {\frac{{{L_j} - {L_{j - 1}}}}{{{\theta _j}}}} } \right]} + \lambda \xi - {{\hat \xi }_n} \right) \ge {s_n}, \\ \xi \in [\underline{\xi}, \overline{\xi}], \\ 0 < {L_1} \le \cdots \le {L_i} \le \cdots \le {L_I},\\ \lambda \ge 0. Here, \mathbf{L} \in \mathbb{R}^I and \mathbf{s} \in \mathbb{R}^N. Could you give me some **intuition on** how we can **solve this bi-level optimization problem**?

Prompt Instance

No, I mean whether **some algorithms** are good at **solving this type of bi-level optimization problem**? If so, please list the name and core steps of the algorithm.

LLM for Problem Solving — via ChatGPT



✅ Algorithms for Solving This Type of Bi-Level Optimization Problem

Below are methods tailored for **convex inner minimization and differentiable outer problem** (or with subgradients available).

---

1. KKT-Based Reformulation (Mathematical Program with Equilibrium Constraints, MPEC)

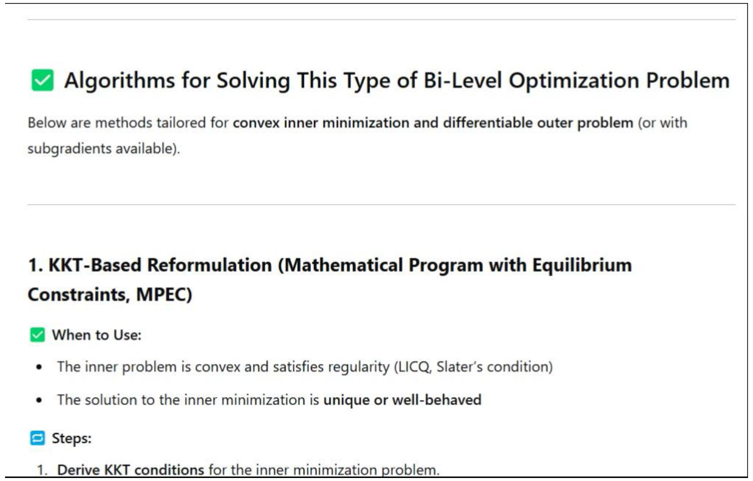
✅ **When to Use:**

- The inner problem is convex and satisfies regularity (LICQ, Slater's condition)
- The solution to the inner minimization is **unique or well-behaved**

🔁 **Steps:**

1. **Derive KKT conditions** for the inner minimization problem.

LLM for Problem Solving — via ChatGPT



☑ Algorithms for Solving This Type of Bi-Level Optimization Problem

Below are methods tailored for **convex inner minimization and differentiable outer problem** (or with subgradients available).

---

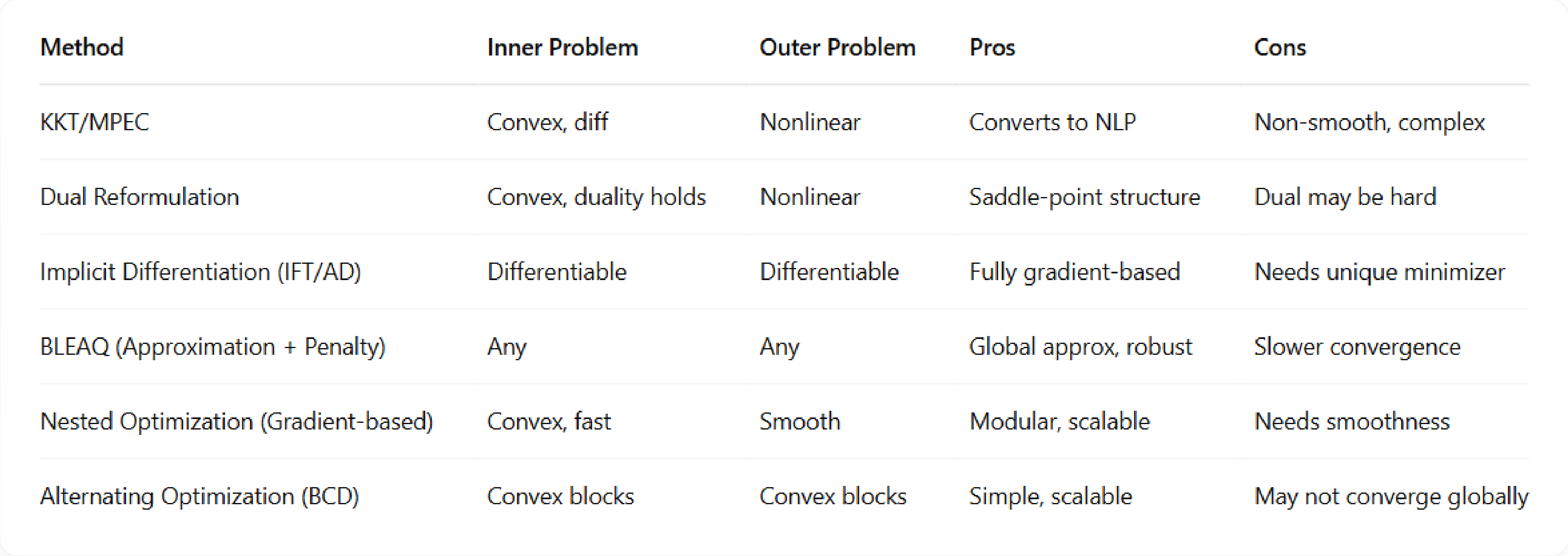
1. KKT-Based Reformulation (Mathematical Program with Equilibrium Constraints, MPEC)

☑ **When to Use:**

- The inner problem is convex and satisfies regularity (LICQ, Slater's condition)
- The solution to the inner minimization is **unique or well-behaved**

Steps:

1. **Derive KKT conditions** for the inner minimization problem.



Method	Inner Problem	Outer Problem	Pros	Cons
KKT/MPEC	Convex, diff	Nonlinear	Converts to NLP	Non-smooth, complex
Dual Reformulation	Convex, duality holds	Nonlinear	Saddle-point structure	Dual may be hard
Implicit Differentiation (IFT/AD)	Differentiable	Differentiable	Fully gradient-based	Needs unique minimizer
BLEAQ (Approximation + Penalty)	Any	Any	Global approx, robust	Slower convergence
Nested Optimization (Gradient-based)	Convex, fast	Smooth	Modular, scalable	Needs smoothness
Alternating Optimization (BCD)	Convex blocks	Convex blocks	Simple, scalable	May not converge globally

LLM for Problem Solving — via ChatGPT

$$\sup_{\mathbf{L},\lambda,\mathbf{s}} -\lambda\varepsilon + \sum_{n=1}^{N} s_n$$

$$\text{s.t.} \quad \inf_{\xi} \left(\sum_{i=1}^{I} \alpha_i \left[\ln(\gamma_2 \xi + \gamma_3 L_i) - \frac{\gamma_1 L_1}{\theta_1} - \gamma_1 \sum_{j=2}^{I} \frac{L_j - L_{j-1}}{\theta_j} \right] + \lambda\xi - \hat{\xi}_n \right) \ge s_n,$$

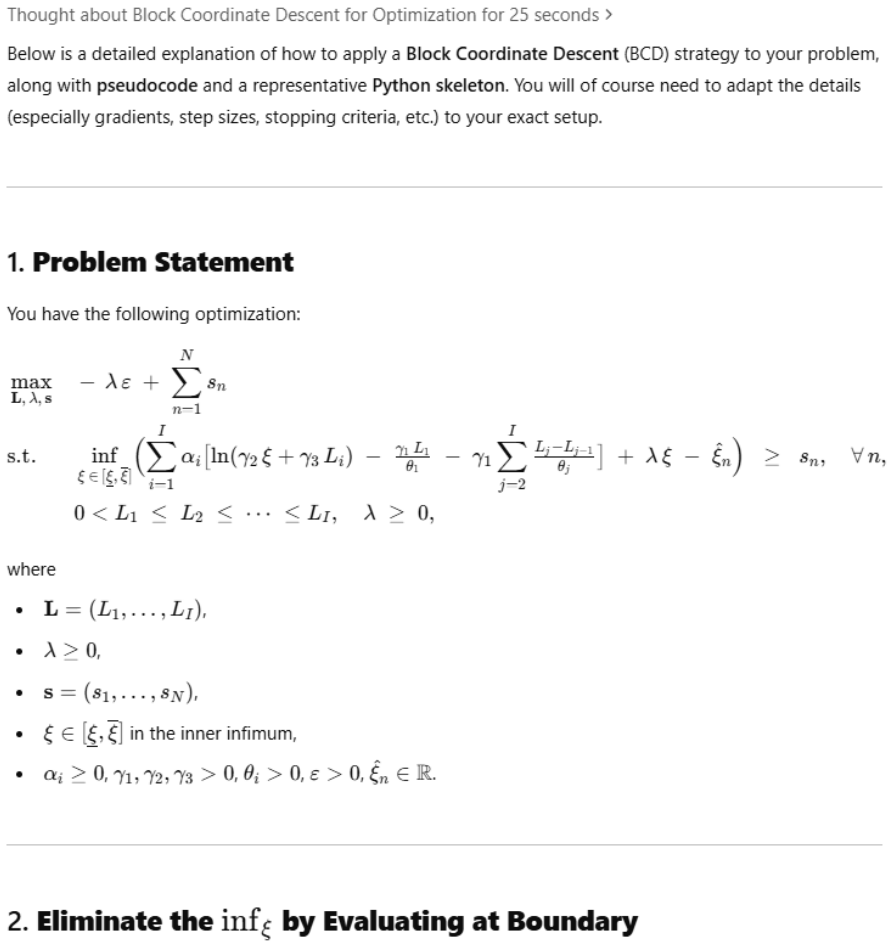
$$\xi \in [\underline{\xi}, \overline{\xi}],$$

$$0 < L_1 \le \cdots \le L_i \le \cdots \le L_I,$$

$$\lambda \ge 0.$$

Prompt Instance

\mathop {\sup }\limits_{{\mathbf{L}},\lambda ,{\mathbf{s}}} - \lambda \varepsilon + \sum\limits_{n = 1}^N {{s_n}} \\ {\rm{s}}{\rm{.t}}{\rm{. }} \quad \mathop {\inf }\limits_\xi \left(\sum\limits_{i = 1}^I {{\alpha _i}\left[{\ln ({\gamma _2}\xi + {\gamma _3}{L_i}) - \frac{{{\gamma _1}{L_1}}}{{{\theta _1}}} - {\gamma _1}\sum\limits_{j = 2}^I {\frac{{{L_j} - {L_{j - 1}}}}{{{\theta _j}}}} } \right]} + \lambda \xi - {{\hat \xi }_n} \right) \ge {s_n}, \\ \xi \in [\underline{\xi}, \overline{\xi}], \\ 0 < {L_1} \le \cdots \le {L_i} \le \cdots \le {L_I},\\ \lambda \ge 0. Here, \mathbf{L} \in \mathbb{R}^I and \mathbf{s} \in \mathbb{R}^N. The optimization problem should be this one. Could you instruct me on how can we use the **BCD algorithm** to resolve it and **provide me the Python code** and **pseudocode**?



Thought about Block Coordinate Descent for Optimization for 25 seconds ›

Below is a detailed explanation of how to apply a **Block Coordinate Descent** (BCD) strategy to your problem, along with **pseudocode** and a representative **Python skeleton**. You will of course need to adapt the details (especially gradients, step sizes, stopping criteria, etc.) to your exact setup.

---

1. **Problem Statement**

You have the following optimization:

$$\max_{\mathbf{L},\lambda,\mathbf{s}} \; -\lambda\varepsilon + \sum_{n=1}^{N} s_n$$

$$\text{s.t.} \quad \inf_{\xi \in [\underline{\xi},\overline{\xi}]} \left(\sum_{i=1}^{I} \alpha_i \left[\ln(\gamma_2 \xi + \gamma_3 L_i) - \tfrac{\gamma_1 L_1}{\theta_1} - \gamma_1 \sum_{j=2}^{I} \tfrac{L_j - L_{j-1}}{\theta_j} \right] + \lambda\xi - \hat{\xi}_n \right) \ge s_n, \quad \forall n,$$

$$0 < L_1 \le L_2 \le \cdots \le L_I, \quad \lambda \ge 0,$$

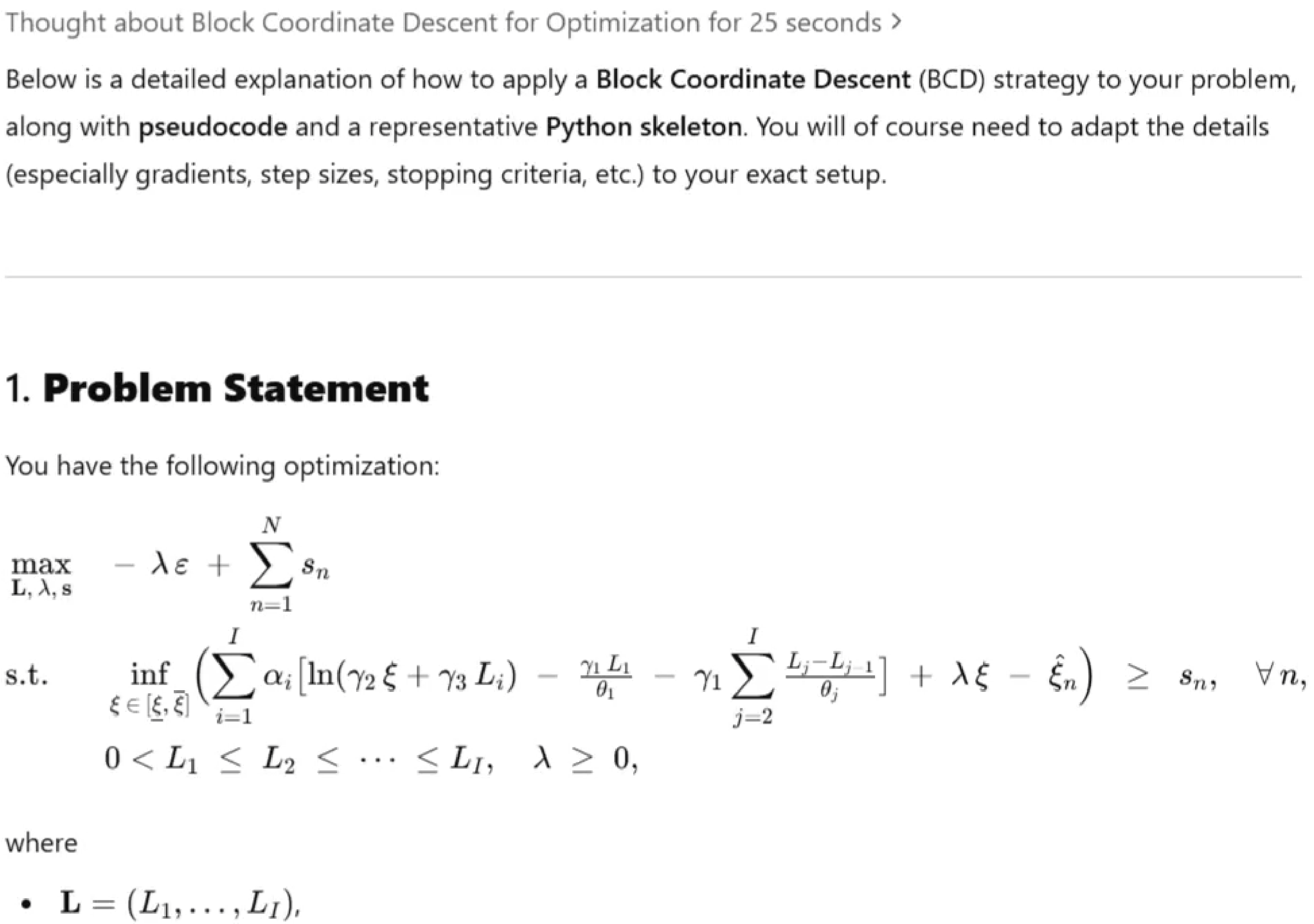
where

- $\mathbf{L} = (L_1, \ldots, L_I)$,
- $\lambda \ge 0$,
- $\mathbf{s} = (s_1, \ldots, s_N)$,
- $\xi \in [\underline{\xi}, \overline{\xi}]$ in the inner infimum,
- $\alpha_i \ge 0$, $\gamma_1, \gamma_2, \gamma_3 > 0$, $\theta_i > 0$, $\varepsilon > 0$, $\hat{\xi}_n \in \mathbb{R}$.

---

2. **Eliminate the $\inf_{\xi}$ by Evaluating at Boundary**

LLM for Problem Solving — via ChatGPT



Thought about Block Coordinate Descent for Optimization for 25 seconds >

Below is a detailed explanation of how to apply a **Block Coordinate Descent** (BCD) strategy to your problem, along with **pseudocode** and a representative **Python skeleton**. You will of course need to adapt the details (especially gradients, step sizes, stopping criteria, etc.) to your exact setup.

---

1. **Problem Statement**

You have the following optimization:

$$\max_{\mathbf{L},\,\lambda,\,\mathbf{s}} \quad -\lambda\varepsilon + \sum_{n=1}^{N} s_n$$

$$\text{s.t.} \quad \inf_{\xi\in[\underline{\xi},\,\overline{\xi}]}\left(\sum_{i=1}^{I}\alpha_i\left[\ln(\gamma_2\,\xi+\gamma_3\,L_i) - \tfrac{\gamma_1\,L_1}{\theta_1} - \gamma_1\sum_{j=2}^{I}\tfrac{L_j-L_{j-1}}{\theta_j}\right] + \lambda\,\xi - \hat{\xi}_n\right) \geq s_n, \quad \forall\, n,$$

$$0 < L_1 \leq L_2 \leq \cdots \leq L_I, \quad \lambda \geq 0,$$

where

- $\mathbf{L} = (L_1,\ldots,L_I)$,

Outline

- Introduce the current mainstream LLMs
- LLM for research
- LLM for problem solving
- LLM for coding

LLM for Coding

External plugin that can be integrated directly into the **VSCode** editor for immediate use

Copilot	https://azure.microsoft.com/en-us/products/github/copilot
Cursor	https://www.cursor.com/en

Use your UH email to register, free to use

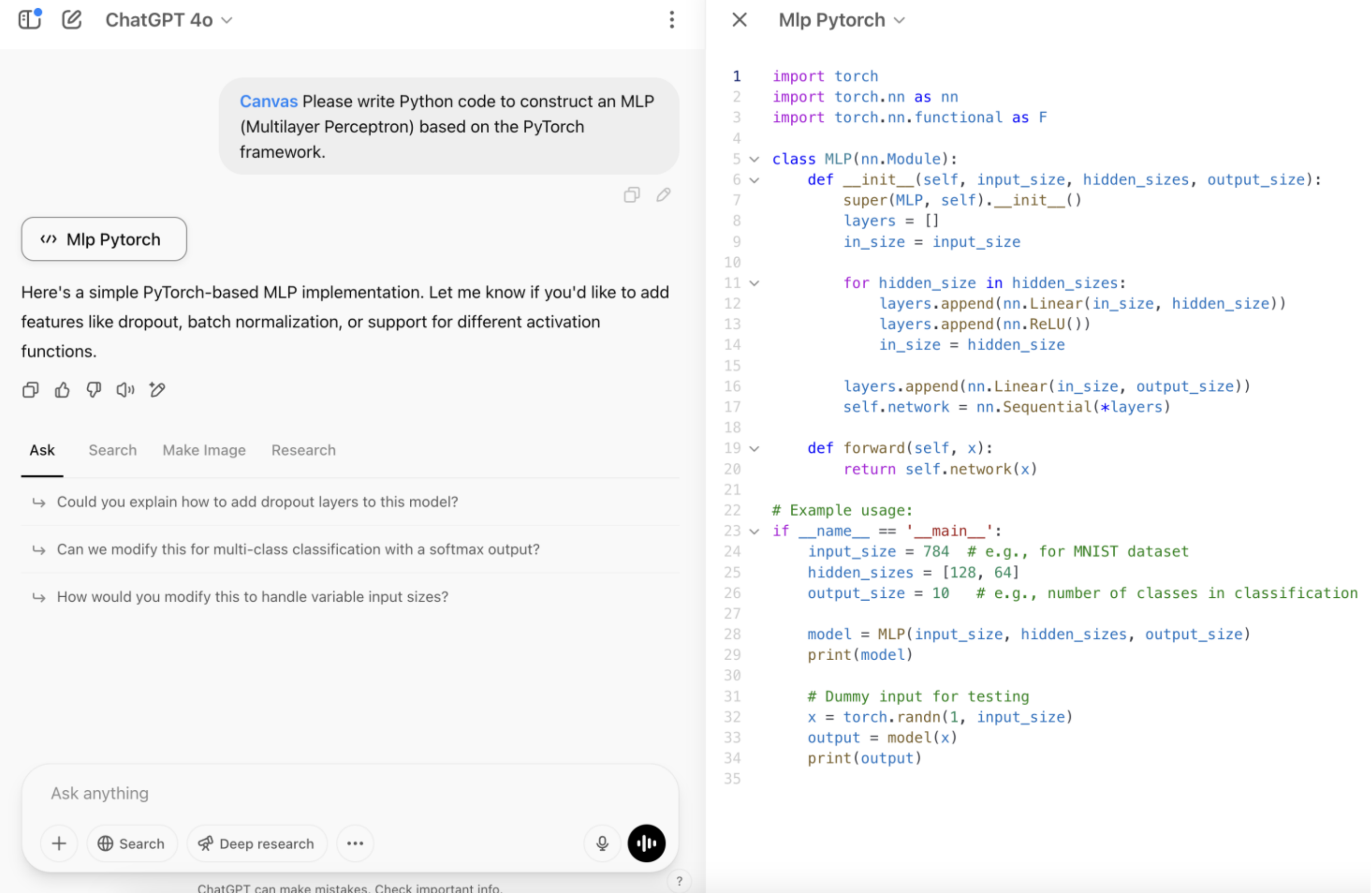
If you expect the AI to implement the entire code for you, that's not realistic

- For **simple code**, use GPT-4o. Don’t waste your quota of advanced models.
- For **complex code**, you need to **break it down into modules** and then communicate the intended functionality of each module to the AI.
- When an error occurs in the code, you should not only copy the error message to the AI, but also **locate the specific part of the code** where the error happened and **provide the surrounding context** to the AI.
- Learn to test code **using examples**; you can ask the AI to generate inputs and expected outputs for the code, and verify them through manual calculation.

LLM for Coding — via ChatGPT

Prompt Instance

Please write **Python** code to construct an **MLP (Multilayer Perceptron)** based on the **PyTorch framework.**



Canvas Please write Python code to construct an MLP (Multilayer Perceptron) based on the PyTorch framework.

Here's a simple PyTorch-based MLP implementation. Let me know if you'd like to add features like dropout, batch normalization, or support for different activation functions.

↳ Could you explain how to add dropout layers to this model?

↳ Can we modify this for multi-class classification with a softmax output?

↳ How would you modify this to handle variable input sizes?

ChatGPT can make mistakes. Check important info.

Mlp Pytorch

```
import torch
import torch.nn as nn
import torch.nn.functional as F

class MLP(nn.Module):
    def __init__(self, input_size, hidden_sizes, output_size):
        super(MLP, self).__init__()
        layers = []
        in_size = input_size

        for hidden_size in hidden_sizes:
            layers.append(nn.Linear(in_size, hidden_size))
            layers.append(nn.ReLU())
            in_size = hidden_size

        layers.append(nn.Linear(in_size, output_size))
        self.network = nn.Sequential(*layers)

    def forward(self, x):
        return self.network(x)

# Example usage:
if __name__ == '__main__':
    input_size = 784  # e.g., for MNIST dataset
    hidden_sizes = [128, 64]
    output_size = 10   # e.g., number of classes in classification

    model = MLP(input_size, hidden_sizes, output_size)
    print(model)

    # Dummy input for testing
    x = torch.randn(1, input_size)
    output = model(x)
    print(output)
```

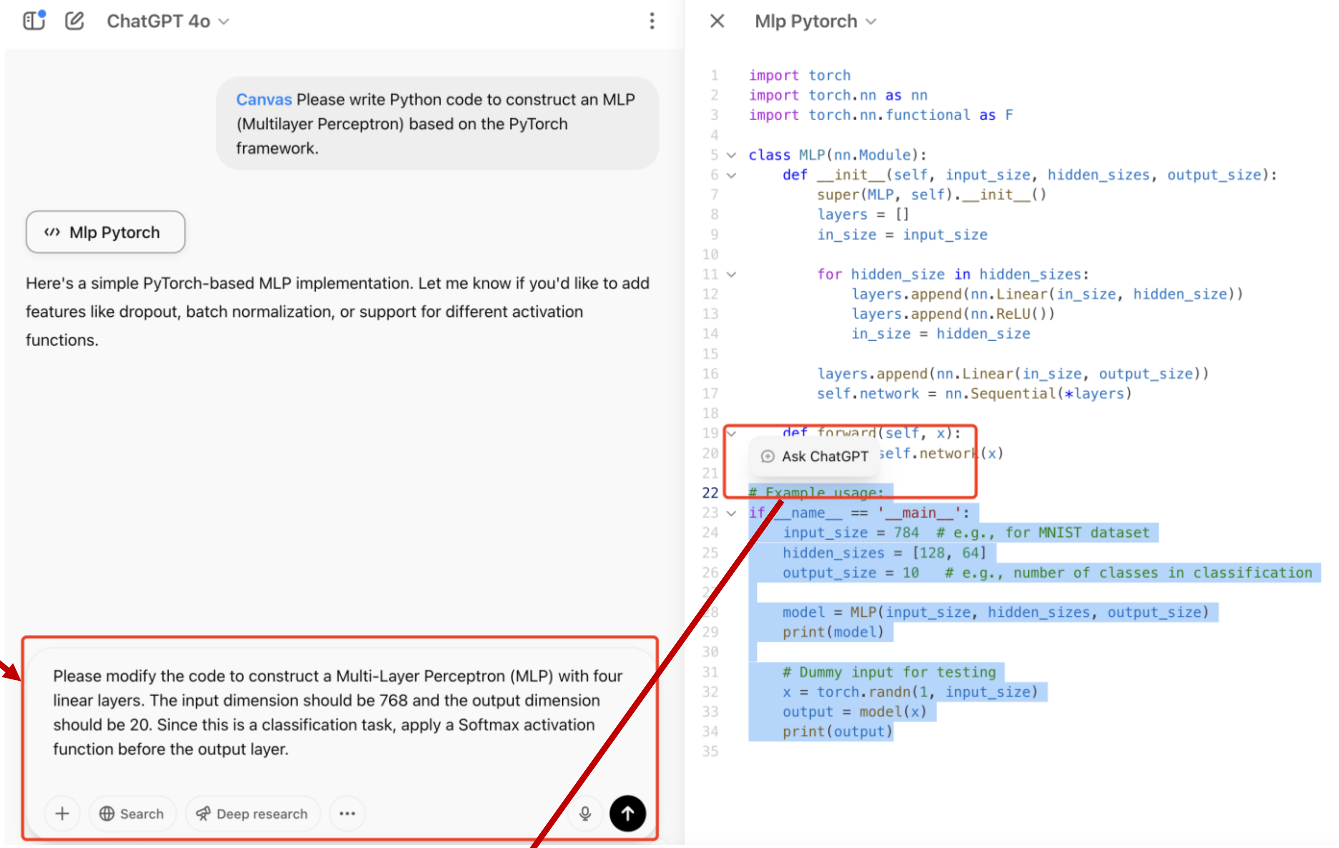
Although GPT can generate the corresponding code, is this a good prompt? **No!**

LLM for Coding — via ChatGPT

Prompt Instance

Please write **Python** code to construct an **MLP (Multilayer Perceptron)** based on the **PyTorch framework.**

Please modify the code to construct a Multi-Layer Perceptron (MLP) with **four linear layers**. The **input dimension** should be 768 and the **output dimension** should be 20. Since this is a **classification task, apply a Softmax activation function** before the output layer.



Canvas Please write Python code to construct an MLP (Multilayer Perceptron) based on the PyTorch framework.

Mlp Pytorch

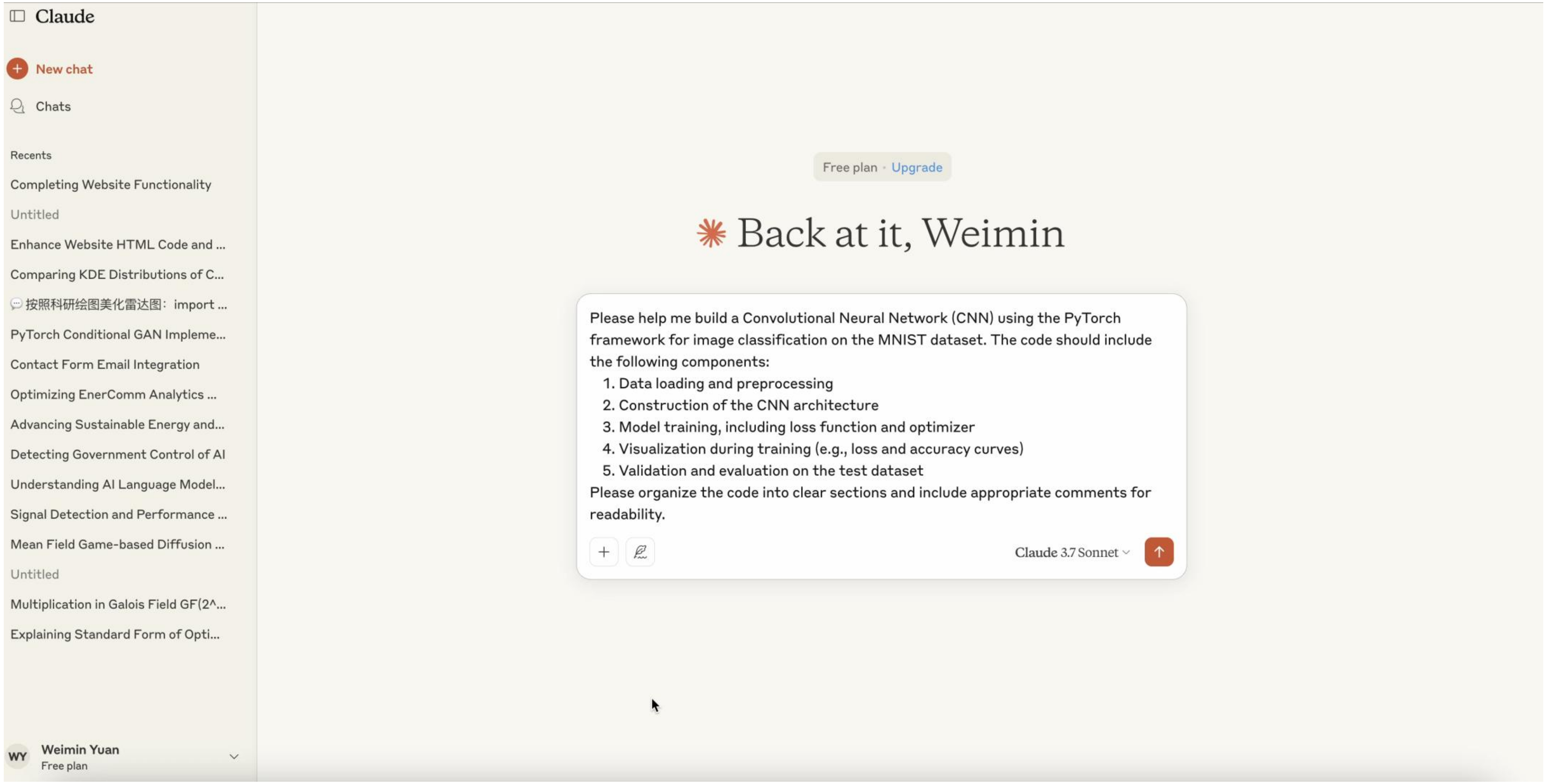
Here's a simple PyTorch-based MLP implementation. Let me know if you'd like to add features like dropout, batch normalization, or support for different activation functions.

Please modify the code to construct a Multi-Layer Perceptron (MLP) with four linear layers. The input dimension should be 768 and the output dimension should be 20. Since this is a classification task, apply a Softmax activation function before the output layer.

```python
import torch
import torch.nn as nn
import torch.nn.functional as F

class MLP(nn.Module):
    def __init__(self, input_size, hidden_sizes, output_size):
        super(MLP, self).__init__()
        layers = []
        in_size = input_size

        for hidden_size in hidden_sizes:
            layers.append(nn.Linear(in_size, hidden_size))
            layers.append(nn.ReLU())
            in_size = hidden_size

        layers.append(nn.Linear(in_size, output_size))
        self.network = nn.Sequential(*layers)

    def forward(self, x):
               self.network(x)

# Example usage:
if __name__ == '__main__':
    input_size = 784  # e.g., for MNIST dataset
    hidden_sizes = [128, 64]
    output_size = 10   # e.g., number of classes in classification

    model = MLP(input_size, hidden_sizes, output_size)
    print(model)

    # Dummy input for testing
    x = torch.randn(1, input_size)
    output = model(x)
    print(output)
```

You can also select a specific module and modify only that part

LLM for Coding — via Claude


Claude
New chat
Chats
Recents
Completing Website Functionality
Untitled
Enhance Website HTML Code and ...
Comparing KDE Distributions of C...
按照科研绘图美化雷达图：import ...
PyTorch Conditional GAN Impleme...
Contact Form Email Integration
Optimizing EnerComm Analytics ...
Advancing Sustainable Energy and...
Detecting Government Control of AI
Understanding AI Language Model...
Signal Detection and Performance ...
Mean Field Game-based Diffusion ...
Untitled
Multiplication in Galois Field GF(2^...
Explaining Standard Form of Opti...
WY Weimin Yuan
Free plan
Free plan · Upgrade
Back at it, Weimin
Claude 3.7 Sonnet

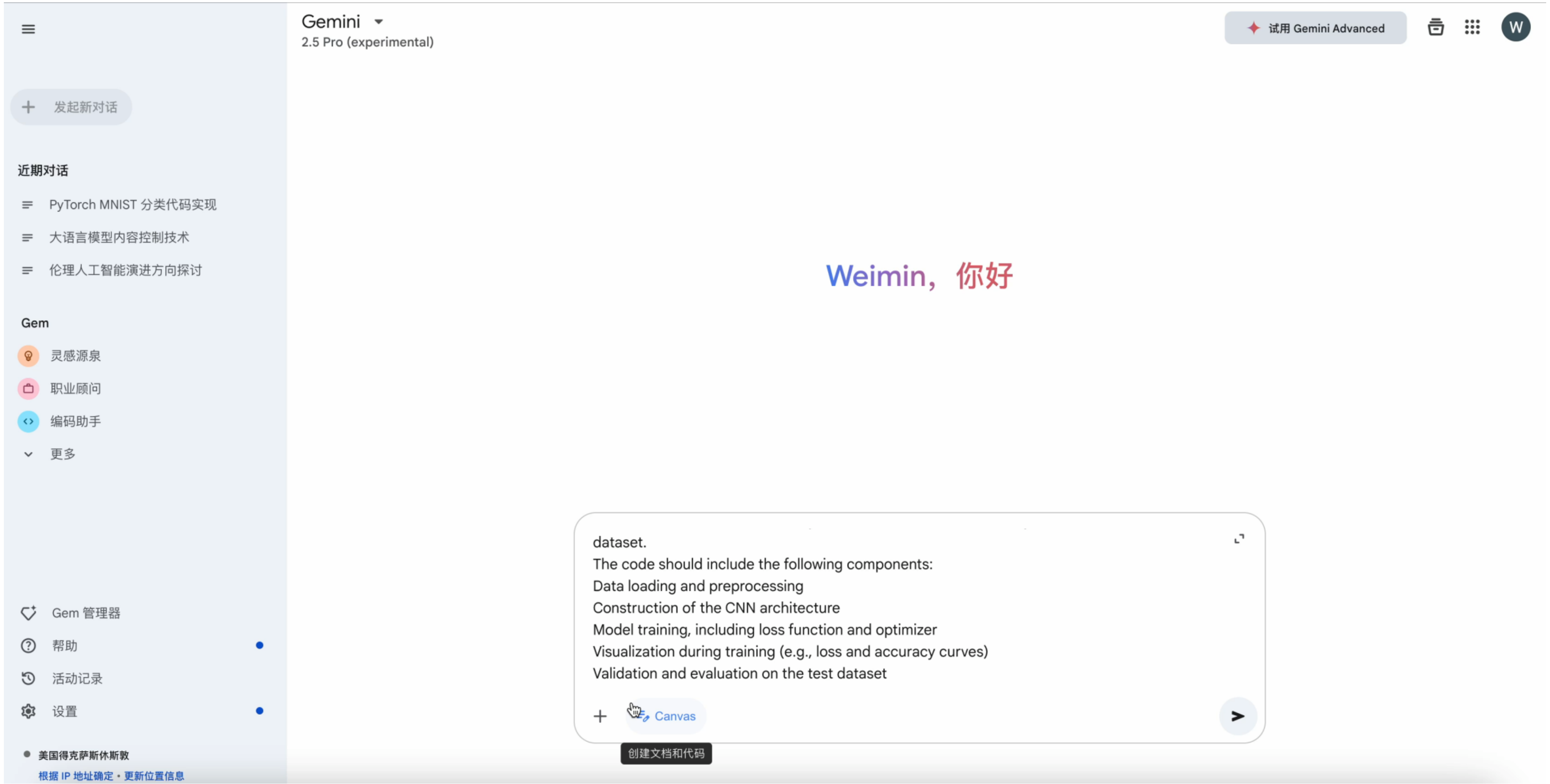
Please help me build a Convolutional Neural Network (CNN) using the PyTorch framework for image classification on the MNIST dataset. The code should include the following components:

1. Data loading and preprocessing
2. Construction of the CNN architecture
3. Model training, including loss function and optimizer
4. Visualization during training (e.g., loss and accuracy curves)
5. Validation and evaluation on the test dataset

Please organize the code into clear sections and include appropriate comments for readability.

LLM for Coding — via Gemini

You need to choose **Canvas mode**



dataset.
The code should include the following components:
Data loading and preprocessing
Construction of the CNN architecture
Model training, including loss function and optimizer
Visualization during training (e.g., loss and accuracy curves)
Validation and evaluation on the test dataset

THANK YOU

Department of Electrical
and Computer Engineering

Cullen College of Engineering