Open Source Planning & Control System with Language Agents for Autonomous Scientific Discovery

Licong Xu [1 2] **Milind Sarkar** [3] **Anto I. Lonappan** [4] **Íñigo Zubeldia** [1 2] **Pablo Villanueva-Domingo** [5] **Santiago Casas** [6] **Christian Fidler** [6] **Chetana Amancharla** [7] **Ujjwal Tiwari** [7] **Adrian Bayer** [8 9] **Chadi Ait Ekioui** [10 11] **Miles Cranmer** [1 2 12] **Adrian Dimitrov** [10] **James Fergusson** [12] **Kahaan Gandhi** [10 13 14] **Sven Krippendorf** [12 10] **Andrew Laverick** [10] **Julien Lesgourgues** [6] **Antony Lewis** [15] **Thomas Meier** [16] **Blake Sherwin** [2 12] **Kristen Surrao** [17] **Francisco Villaescusa-Navarro** [8 9] **Chi Wang** [18] **Xueqing Xu** [10] **Boris Bolliet** [* 2 10]

Abstract

We present a multi-agent system for automation of scientific research tasks, `cmbagent`. The system is formed by about 30 Large Language Model (LLM) agents and implements a *Planning & Control* strategy to orchestrate the agentic workflow, with no *human-in-the-loop* at any point. Each agent specializes in a different task (performing retrieval on scientific papers and codebases, writing code, interpreting results, critiquing the output of other agents) and the system is able to execute code locally. We successfully apply `cmbagent` to carry out a PhD level cosmology task (the measurement of cosmological parameters using supernova data) and evaluate its performance on two benchmark sets, finding superior performance over state-of-the-art LLMs. The source code is available on GitHub[1], demonstration videos are also available[2], and the system is deployed on HuggingFace[3] and will be available on the cloud[4].

1. Introduction

Rapid progress in the development of Large Language Models (LLMs) is enabling new approaches to scientific research (see, e.g., Lu et al., 2024; Ghareeb et al., 2025, and references therein). This includes quantitative disciplines grounded in numerical data, such as physics, chemistry, biology, and economics, at both fundamental and applied levels.

One promising application of LLMs in astrophysics and cosmology is their use as autonomous agents to support and automate data analysis workflows. In Laverick et al. (2024), it was demonstrated how LLM agents can be used to solve a state-of-the-art data analysis task in cosmology, measuring the values of fundamental parameters that describe the Universe using a novel dataset (Madhavacheril et al., 2024). They introduced a multi-agent system composed of several types of LLM agents specializing in different tasks (coding, retrieving information from scientific papers, using domain-specific software libraries). This multi-agent system operated with a *human-in-the-loop* at every step: each LLM response was reviewed by the user, who then provided guidance for the next action.

[*]This author led and supervised all the work presented here, at the exception of the part on `denario` which was co-led equally between FVN, PVD and BB. Except BB, ordering of authors is by contribution and then alphabetical. [1]Institute of Astronomy, University of Cambridge, Cambridge, United Kingdom [2]Kavli Institute for Cosmology, University of Cambridge, Cambridge, United Kingdom [3]Department of Physical Sciences, Indian Institute of Science Education and Research (IISER), Mohali, Punjab, India [4]Department of Physics, University of California, San Diego, CA, USA [5]Computer Vision Center, Universitat Autònoma de Barcelona, Bellaterra, Barcelona, Spain [6]Institute for Theoretical Particle Physics and Cosmology (TTK), RWTH Aachen University, Aachen, Germany [7]Infosys Ltd [8]Center for Computational Astrophysics, Flatiron Institute, New York, NY, USA [9]Department of Astrophysical Sciences, Princeton University, Princeton, NJ, USA [10]Department of Physics, University of Cambridge, Cambridge, United Kingdom [11]Télécom SudParis - 9 rue Charles Fourier - 91011 Évry cedex - France [12]Department of Applied Mathematics and Theoretical Physics (DAMTP), University of Cambridge, Cambridge, United Kingdom [13]Haverford College, Haverford, PA, USA 19041 [14]Division of Physics, Mathematics and Astronomy, California Institute of Technology, Pasadena, CA 91125, USA [15]Department of Physics & Astronomy, University of Sussex, Brighton BN1 9QH, UK [16]MCML - Munich Center for Machine Learning, LMU Munich, Geschwister-Scholl-Platz 1, 80539 Munich, Germany [17]Department of Physics, Columbia University, New York, NY, USA [18]Google DeepMind. Correspondence to: Boris Bolliet <bb667@cam.ac.uk>.

Proceedings of the 42nd International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

[1]https://github.com/CMBAgents/cmbagent
[2]https://www.youtube.com/@cmbagent
[3]https://huggingface.co/spaces/astropilot-ai/cmbagent
[4]https://cmbagent.cloud/

Although such systems can be used as effective research assistant tools, they are limited by their reliance on continuous human input. In this work, we introduce `cmbagent`, a multi-agent system which can carry end-to-end research tasks with no *human-in-the-loop* at any point. To enable full automation, `cmbagent` employs a robotics-inspired *Planning & Control* strategy, with an agentic framework powered by `AG2` (Wu et al., 2023; Wang et al., 2024). We first describe the system architecture and usage (Sec. 2) and then present a series of evaluations of the system demonstrating the system's effectiveness in cosmological applications and beyond (Sec. 3), before concluding (Sec. 4).

2. Description of the system

2.1. Planning & Control Strategy

The *Planning & Control* strategy in `cmbagent` is as follows. Given an input task, the system first goes through a *Planning* phase in which a plan is designed and approved. A *Control* phase follows, in which the plan is executed by the *Control* agents. The agents involved in both phases are LLMs and there is no *human-in-the-loop* at any point in either phase. The full *Planning & Control* strategy , which is described next, is illustrated in Fig. 1.

Planning phase. The input of the *Planning* phase is the *Main Task*, which is specified by the user and is typically a quantitative research task. There are two main agents in this phase: the *planner*, which proposes a plan, and the *plan reviewer*, which provides feedback on the proposed plan.

The *Planning* phase starts with the *planner* proposing a first version of the plan. The plan is formed by a succession of a maximum of $n_{\rm steps}$ steps, with each step consisting of: (i) a sub-task, (ii) a set of actions to be taken to carry out the sub-task, and (iii) an agent in charge of completing these actions. The proposed plan is then reviewed by the *plan reviewer*, which provides feedback on the proposed plan. This feedback is passed on to the *planner*, which proposes an updated plan. This *planner–plan reviewer* loop takes place $n_{\rm reviews}$ times. The hyperparameters $n_{\rm steps}$ and $n_{\rm reviews}$ can be specified by the user.

To improve performance, the output of both agents is structured by formatting agents whose sole role is to structure the text in a specific format. Also, for traceability, the structured responses of each agent (the plans and the plan feedback) are recorded into the session context by recorder agents (*plan recorder* and *review recorder*).

Once the definitive plan is approved by the *reviewer*, it is stored into context and saved as a JSON file, and the total cost of the phase (in USD) is displayed and stored. This brings the *Planning* phase to completion and *Control* phase starts.

Control phase. The input of the *Control* phase is the final plan approved in the *Planning* phase. The key agent of the *Control* phase is the *controller*, which distributes the assigned sub-tasks in the plan to the relevant agents. These sub-tasks are carried out by two main agents: (i) a *researcher* for reasoning, interpretation, and summarizing tasks; and (ii) an *engineer* for coding tasks (by default, in Python).

When the *researcher* is queried, its output is formatted in Markdown by a formatting agent and then saved by an *executor* agent so the other agents can access it. On the other hand, when the *engineer* is called, a *nested chat* is triggered, of which only the output is kept. This nested chat is a sequence of two agents: (i) a formatting agent that structures the code written by the *engineer* and provides minor adjustments (e.g., formatting issues, plot labels), and (ii) an *executor* agent that executes the provided code block locally.

The execution output is then forwarded to a post-execution *interpreter* agent, which decides on which agent to transfer to. A successful code execution typically triggers a transition to the *control* agent, which updates the necessary context variables and proceeds with the next step of the plan. On the other hand, a failed code execution may trigger different transitions. If the number of failed execution is less than $n_{\rm fails}$ (a hyperparameter that can be specified by the user), the subsystem typically transits back to the *engineer* for another attempt, with suggestions provided by the *interpreter*. If the failure is caused by a missing Python package, the subsystem typically transits to an *installer* agent that runs a bash `pip install` command before the execution is attempted again. If the number of failed executions reaches $n_{\rm fails}$, the system transitions to a *terminator* agent and the full session ends.

Upon termination of each sub-task, the cost is displayed and the context variables that have been updated by the agents are recorded and passed on to the start of the next step. This process is repeated over all the steps in the plan.

Throughout the *Control* phase, the *controller*, *engineer* and *researcher* are made aware of the output of the previous steps in the plan via a common block injected into their system message. The injected message contains the final code and the execution messages of the previous steps, as well as the messages produced by the *researcher*. At the end of every step in the plan, the agents and the entire chat history are reset and only the system context is carried over to the next step. This strategy allows for the system to keep memory between steps while reducing the cost of the session significantly (typically, by about a factor of two) relative the case in which the agents and chat history are not reset after every step.

2.2. Context Agents

A key element of `cmbagent` is the presence of agents with knowledge of scientific papers and domain-specific Python libraries that are relevant to the research task specified by the user. The simplest way to specialize an LLM agent to carry out a particular task is to provide contextual information relevant to the task as part of its context. One could therefore consider supplying entire papers or an entire codebase into the context of the relevant LLM-agents. This approach is limited by cost and, more fundamentally, by the LLM's input token limits. However, for the most recent models, such as `gpt-4.1` or `gemini-2.5-pro/flash`, the current input token limit stands at about a million tokens, which allows for passing hundred-page-long articles, entire codebases, or long Python package documentations into the LLM context.

Following this approach, in `cmbagent` we have built two *context* agents, i.e., agents with extended context in their system message, one for each of the two most widely-used Python packages in Cosmology: `camb` (Lewis et al., 2000) and `class` (Blas et al., 2011). For `class`, we manually created a Markdown document that is then passed as a single string to the *class context* agent (see this url). For `camb`, the relevant Markdown document is automatically generated upon each build of the package *Read the Docs* documentation using `sphinx_markdown_builder`. The document is then collected from this URL. This approach implies that the *camb context* agent keeps updating its system message as the `camb` library evolves.

The performance of the *camb context* agent on Cosmology problems is evaluated in Sec. 3.1, where we find significant enhancement relative to pre-trained models with no additional context.

2.3. Retrieval Augmented Generation Agents

As discussed above, providing large context to LLMs comes at both a monetary and a latency cost, with an upper limit set by the LLM maximum number of input tokens. This limit may be hit, e.g., if the system is requested to access hundreds of scientific papers or large codebase. A powerful and complementary approach to the *context* agents described in Sec. 2.2 is to build *RAG* agents. These agents leverage Retrieved Augmented Generation (RAG) methods, with their contexts made of chunks of information that are retrieved from a vector embedding of the full database using similarity search (e.g., semantic or lexical search).

Several agents in `cmbagent` adopt this strategy. These include the *classy-sz* agent, which is used in the end-to-end research example discussed in Sec. 3.4. For this agent, we created a vector embedding of the documentation of the `classy_sz` library (Bolliet et al., 2024), on which the agent performs RAG. We have also built RAG agents for databases consisting of corpora of scientific papers.

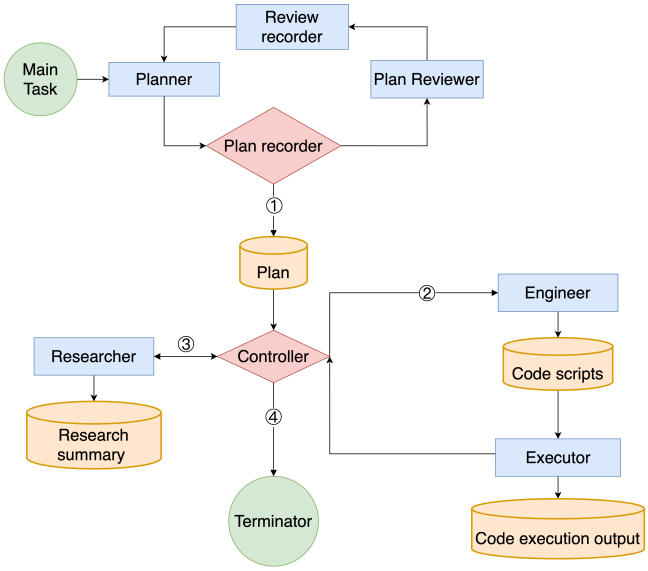

Figure 1: *Planning & Control* strategy in `cmbagent`. The yellow disks correspond to output of the system. 1: Plan approved after maximum review round is reached; 2: Subtask involves coding and previous execution failed, but retry attempts remain; 3: Subtask involves researching or reasoning over external knowledge; 4: All subtasks completed or maximum code execution attempts reached.

2.4. Distribution

In addition to the source code on GitHub, we distribute the wheels for `cmbagent` on PyPi and provide support for containerization via Docker (see our documentation). We have also developed a GUI for user-friendly interaction with the system, which is deployed on a dedicated HuggingFace space. For illustration, snapshots of the preliminary GUI can be found in Appendix A.

3. System evaluation

3.1. Context agent evaluation

We evaluate `cmbagent`'s *camb agent* (running with `gemini-2.5-pro`) with a benchmark set formed by 14 problems involving the `camb` library, one of the most commonly-used libraries for Cosmology-specific computations. These problems range from basic calculations, such as Cosmic Microwave Background (CMB) power spectra estimation, to advanced problems, such as computing CMB delensing efficiency. We compare the performance of our *camb agent* to that of three LLMs queried through the *engineer* agent: `gpt-4o`, `gpt-4.1` and `gemini-2.5-pro`. We compute the success rate for each problem by solving it

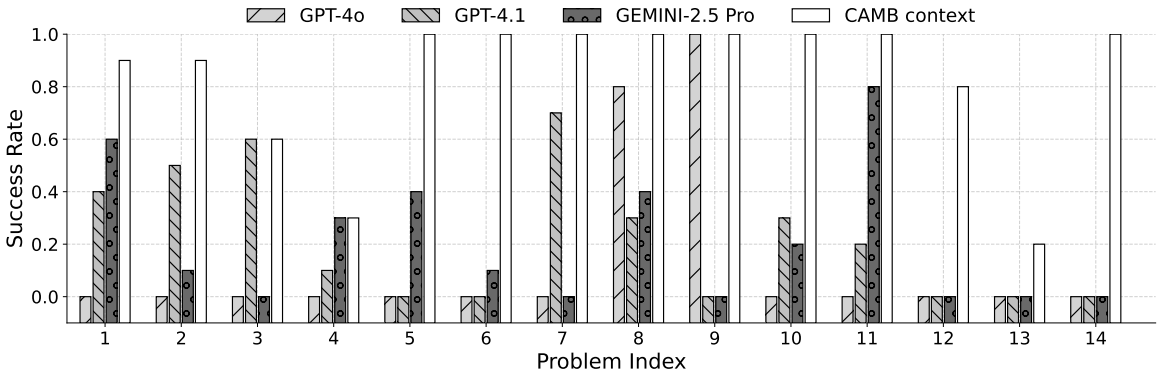

Figure 2: Success rate of the *camb context* agent on 14 cosmology problems, compared with state-of-the-art LLMs queried via the *engineer* agent. The code used for evaluation and the full problem set are available here.

ten times with each agent. The results are shown on Fig. 2, where it can be seen that the *camb agent* compares very favorably to the other three ones. In particular, in Problems 12, 13, and 14, `gpt-4o`, `gpt-4.1` and `gemini-2.5-pro` fail systematically but the *camb context* agent achieves much better performance. These results clearly demonstrate the effectiveness of domain-specific context augmentation.

3.2. Evaluation of *One Shot* and *Planning & Control* on the DS-1000 Benchmark

We evaluate `cmbagent` on a subset of problems from the DS-1000 benchmark (Lai et al., 2022), which covers usage of the `pandas`, `numpy`, and `matplotlib` libraries, with the *researcher* agent powered by `gpt-4.1`. The results are reported in Table 1, where it can be seen that our *Planning & Control* strategy consistently leads to enhanced performance, increasing the overall success rate (number of problems solved over number of problems) from 66% to 78%. These results demonstrate the power of our *Planning & Control* strategy for complex problem solving. See Appendix B for further details.

3.3. Cosmology research task

We also demonstrate the performance of `cmbagent` on a Cosmology research task using the *Planning & Control* mode, with default settings and `gpt-4.1` as the LLM backend. The task is to perform cosmological parameter estimation using the Union2.1 Type Ia supernovae dataset (Suzuki et al., 2012). This task requires conducting parameter posterior inference via Markov chain Monte Carlo (MCMC) sampling, which is the most widely used technique in Cosmology for fitting theoretical models to observational data.

The task was successfully solved the first time it was run. Details of the output are provided in Appendix C, including the full *Main Task*, and the generated plan and plots (Figs. 7 and 8). We recorded the full session, with an accelerated version of the video available on YouTube.[5] The full log is also available in a Jupyter notebook.[6]

3.4. End-to-end research backend

`cmbagent` is integrated in the soon-to-be-released[7] `denario` project (Villaescusa-Navarro et al., 2025), a multi-agent system designed for conducting autonomous end-to-end scientific research. There, `cmbagent` plays the role of the research results generation backend. In `denario`, a problem or data description is provided by the user. The system then follows a *Planning & Control* strategy to generate ideas through a conversation between an *idea maker* and an *idea hater* agent, which ends with a proposed research idea. A methodology is then generated (by two rounds of a *researcher* agent); next, the proposed research project is carried out; and, finally, the results are collected. The output consists of markdown reports and plots. These are in turn, converted into a publication-ready research manuscript in PDF format by another multi-agent subsystem implemented with `LangGraph`. Literature search and populating the manuscript with relevant references are also done automatically using a Perplexity agent with `sonar-reasoning-pro`. The results page of the preliminary `denario` GUI is shown in Appendix E and an example of a generated paper is appended to this manuscript (title: *Regime-Specific Performance of 1D CNN and FCNN Architectures for Non-linear Matter Power Spectrum Emulation in ΛCDM Cosmology*).

4. Discussion

We have introduced `cmbagent`, an LLM-powered multi-agent system for quantitative research. For complex tasks, `cmbagent` follows a *Planning & Control* strategy that

[5] YouTube Video: Cosmology Research Task.
[6] Jupyter Notebook: Cosmology Research Task.
[7] As of July 9th 2025 the package is not publicly released.

allows the system to achieve full automation with no *human-in-the-loop*, also offering offers *One Shot* and *Human-in-the-loop* modes for less complex tasks. Key features of `cmbagent` include agents specialising in research papers and code libraries, feedback loops between pairs of agents, structured output generation, and the ability to execute code locally. Built upon the `AG2` framework, its agents leverage state-of-the-art LLMs through API interfaces with OpenAI, Google, and Anthropic. The source code is available on GitHub, distributed on PyPi, and deployed on HuggingFace.

The performance results presented in Sec. 3 are remarkable. Achieving results such as the cosmological parameter estimation task discussed in Sec. 3.3 was hardly possible with a *human-in-the-loop* and frontier LLMs only a year ago (e.g., Bassett, 2024),[8] let alone fully autonomously. As we have demonstrated, current LLMs, when orchestrated in multi-agent systems such as ours, can solve such tasks with no difficulty. We are now even able to write research-quality papers automatically (e.g., Lu et al., 2024; Moss, 2025; Villaescusa-Navarro et al., 2025). These promising results and the rapid progress in AI suggest that scientific research is likely to undergo drastic changes in the near future, with a significant part of research workflows likely to be automated. This may come at a high cost (e.g. Couillet et al., 2022), and as scientists we must be lucid and address the ethical and environmental challenges associated with these changes.

Impact Statement

We present a multi-agent system designed to carry out quantitative scientific research tasks, offering potential benefits in scalability, rigor, and reproducibility. However, such systems raise concerns about automation bias, fairness, and the potential for misuse or misinterpretation of findings. To address these risks, we emphasize the importance of transparency, robust evaluation, and oversight. Ensuring responsible use will require careful consideration of how these systems influence scientific workflows and trust in their outputs.

Author Contributions

The development of `cmbagent` was led by BB. The development of `denario` (Section 3.4) was equally led by FVN, BB, PVD and will be presented in detail in a separate manuscript. LX led the development of the GUI, based on previous work by SC, CF, FVN. IZ, LX and BB wrote the paper. MS, AIL, XX and AD developed the benchmark datasets and evaluations. PVD, CA, UT led the deployment. AB developed the idea agents. KS developed the cost report interface and led early developments of the *swarm* orchestration. KG developed the integration of vision language models and CAE the context-based agents, with AL (Lewis), SC and CF. AL, AL, BS, CW, JF, JL, MC, SK, TM provided crucial input at various stages of the project.

[8] We thank David I. Kaiser for bringing this to our attention.

Acknowledgments

This work was partially funded by an unrestricted gift from Google, the Cambridge Centre for Data-Driven Discovery Accelerate Programme and the Infosys-Cambridge AI Centre. We are very grateful to the referees and panel of the ICML 2025 ML4ASTRO workshop for reviewing and accepting our work. We are very grateful to the AG2 community.[9]

[9] https://ag2.ai

References

Bassett, B. A. Integrals and Integrity: Generative AI Tries to Learn Cosmology. *MIT Case Studies in Social and Ethical Responsibilities of Computing*, (Winter 2024), feb 27 2024. https://mit-serc.pubpub.org/pub/o07xymoa.

Blas, D., Lesgourgues, J., and Tram, T. The cosmic linear anisotropy solving system (class). part ii: Approximation schemes, July 2011. ISSN 1475-7516.

Bolliet, B. et al. class_sz I: Overview, 2024.

Couillet, R., Trystram, D., and Ménissier, T. The submerged part of the ai-ceberg [perspectives]. *IEEE Signal Processing Magazine*, 39(5):10–17, 2022. doi: 10.1109/MSP.2022.3182938.

Ghareeb, A. E., Chang, B., Mitchener, L., Yiu, A., Szostkiewicz, C. J., Laurent, J. M., Razzak, M. T., White, A. D., Hinks, M. M., and Rodriques, S. G. Robin: A multi-agent system for automating scientific discovery, 2025. URL https://arxiv.org/abs/2505.13400.

Lai, Y., Li, C., Wang, Y., Zhang, T., Zhong, R., Zettlemoyer, L., tau Yih, S. W., Fried, D., Wang, S., and Yu, T. Ds-1000: A natural and reliable benchmark for data science code generation, 2022. URL https://arxiv.org/abs/2211.11501.

Laverick, A., Surrao, K., Zubeldia, I., Bolliet, B., Cranmer, M., Lewis, A., Sherwin, B., and Lesgourgues, J. Multi-Agent System for Cosmological Parameter Analysis. 11 2024.

Lewis, A., Challinor, A., and Lasenby, A. Efficient computation of CMB anisotropies in closed FRW models. *The Astrophysical Journal*, 538:473–476, 2000. doi: 10.1086/309179.

Lu, C., Lu, C., Lange, R. T., Foerster, J., Clune, J., and Ha, D. The AI Scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.

Madhavacheril, M. S. et al. The Atacama Cosmology Telescope: DR6 Gravitational Lensing Map and Cosmological Parameters. *Astrophys. J.*, 962(2):113, 2024. doi: 10.3847/1538-4357/acff5f.

Moss, A. The AI Cosmologist I: An Agentic System for Automated Data Analysis. *arXiv e-prints*, art. arXiv:2504.03424, April 2025. doi: 10.48550/arXiv.2504.03424.

Suzuki, N., Rubin, D., Lidman, C., Aldering, G., Amanullah, R., Barbary, K., Barrientos, L., Botyanszki, J., Brodwin, M., Connolly, N., et al. The hubble space telescope cluster supernova survey. v. improving the dark-energy constraints above z> 1 and building an early-type-hosted supernova sample. *The Astrophysical Journal*, 746(1):85, 2012.

Villaescusa-Navarro, F., Bolliet, B., and Villanueva-Domingo, P. Astropilot: Modular automation of scientific research with multi-agent systems, 2025. URL https://github.com/AstroPilot-AI.

Wang, C., Wu, Q., and the AG2 Community. Ag2: Open-source agentos for ai agents, 2024. URL https://github.com/ag2ai/ag2. Available at https://docs.ag2.ai/.

Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., et al. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

A. Graphical User Interface

`cmbagent`'s GUI welcome page is shown in Fig. 3. The *Planning & Control* page is shown in Fig. 4, the *One Shot* one in Fig. 5, and the *Human-in-the-loop* one in Fig. 6.

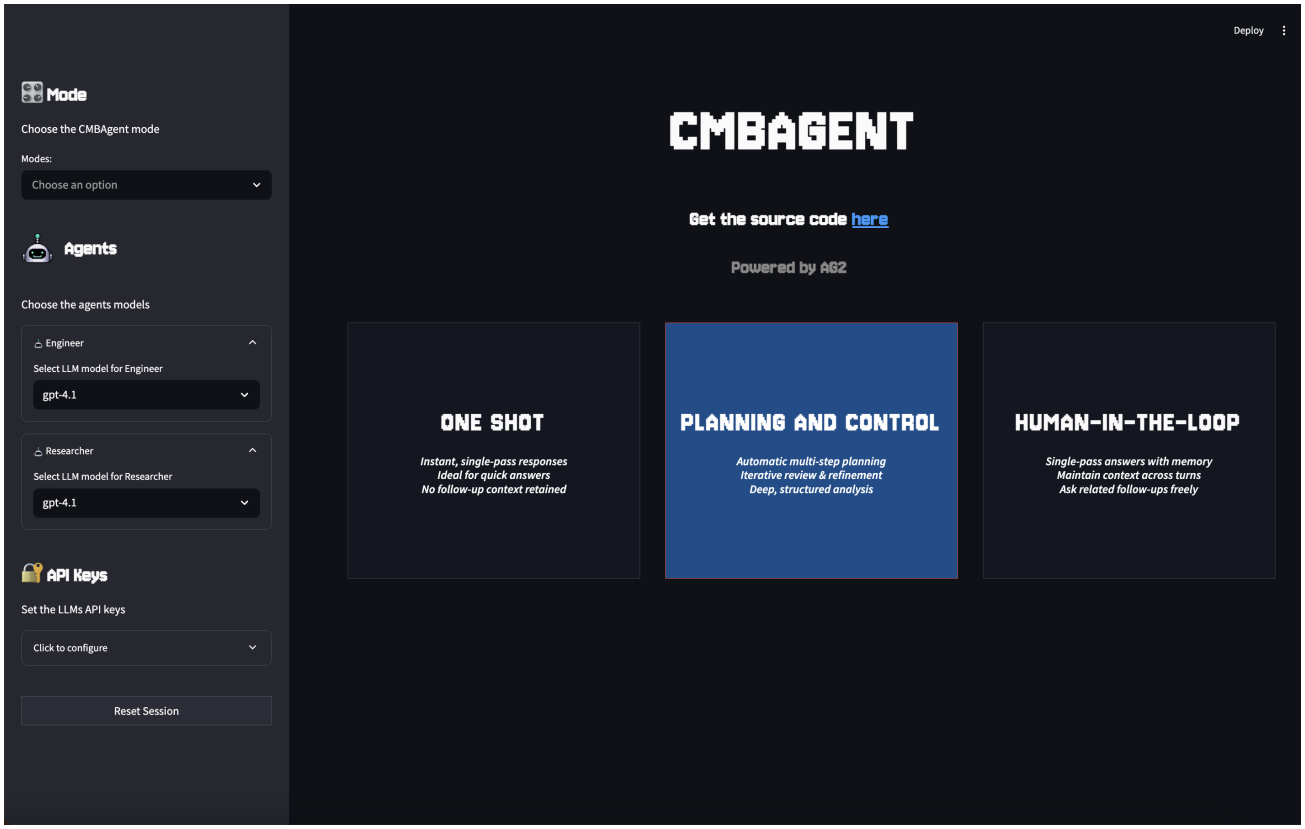

Figure 3: Welcome page of the `cmbagent` GUI.

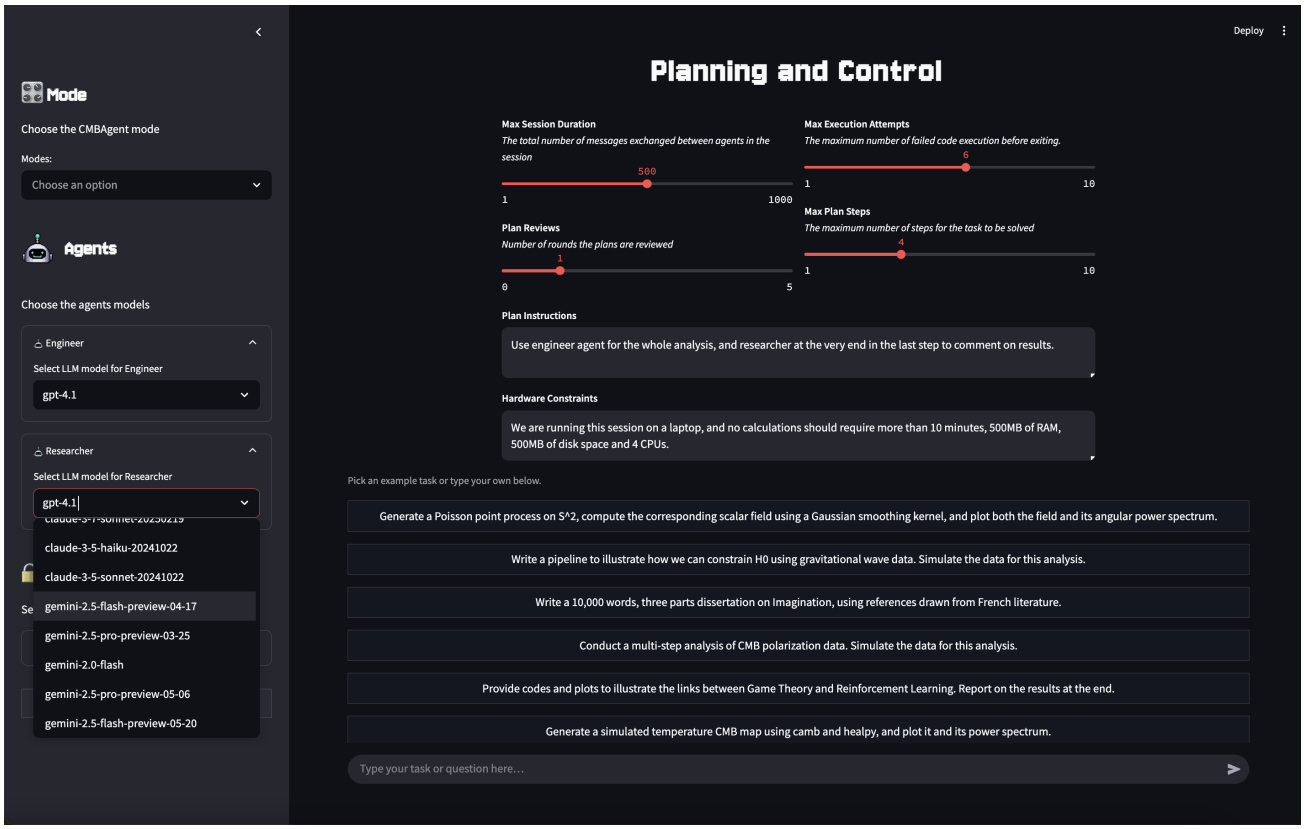

Figure 4: *Planning & Control* page of the `cmbagent` GUI.

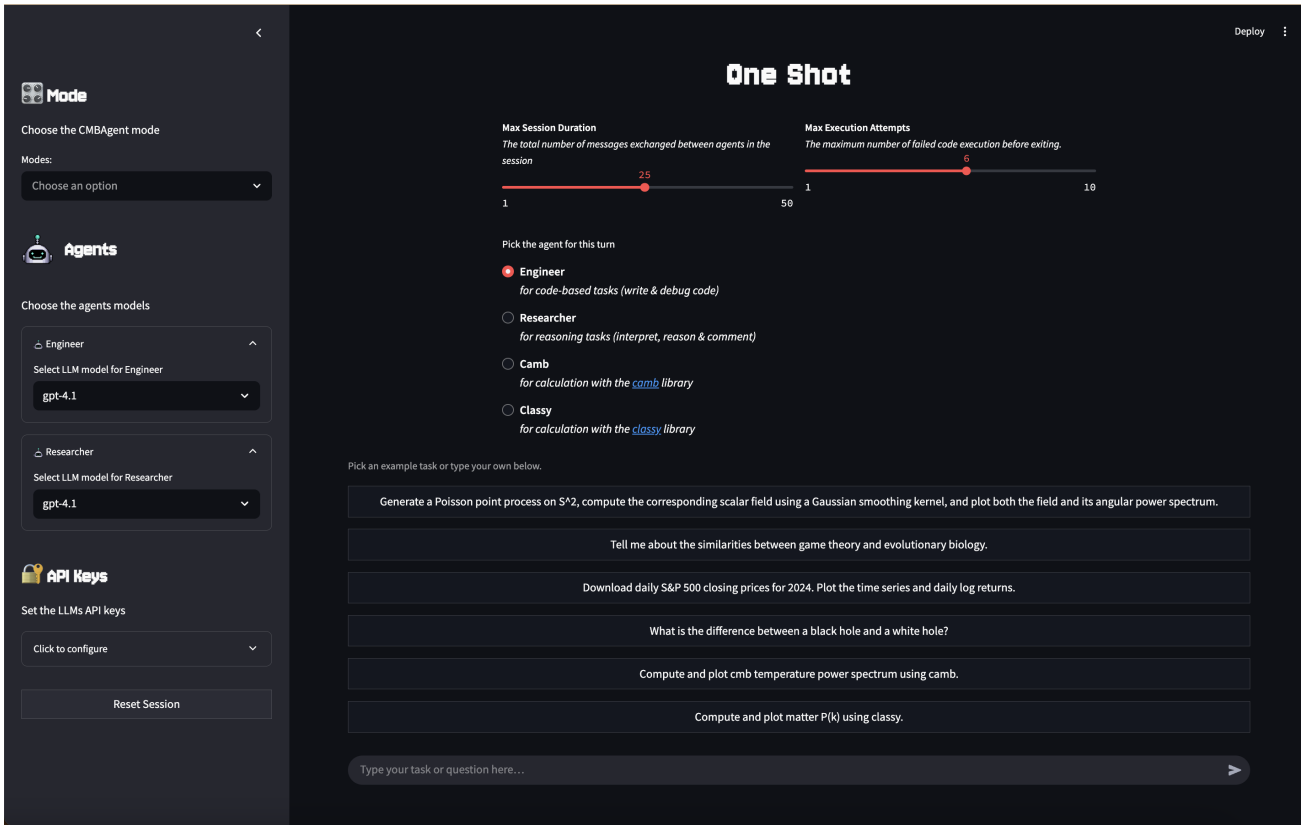

Figure 5: *One Shot* page of the `cmbagent` GUI.

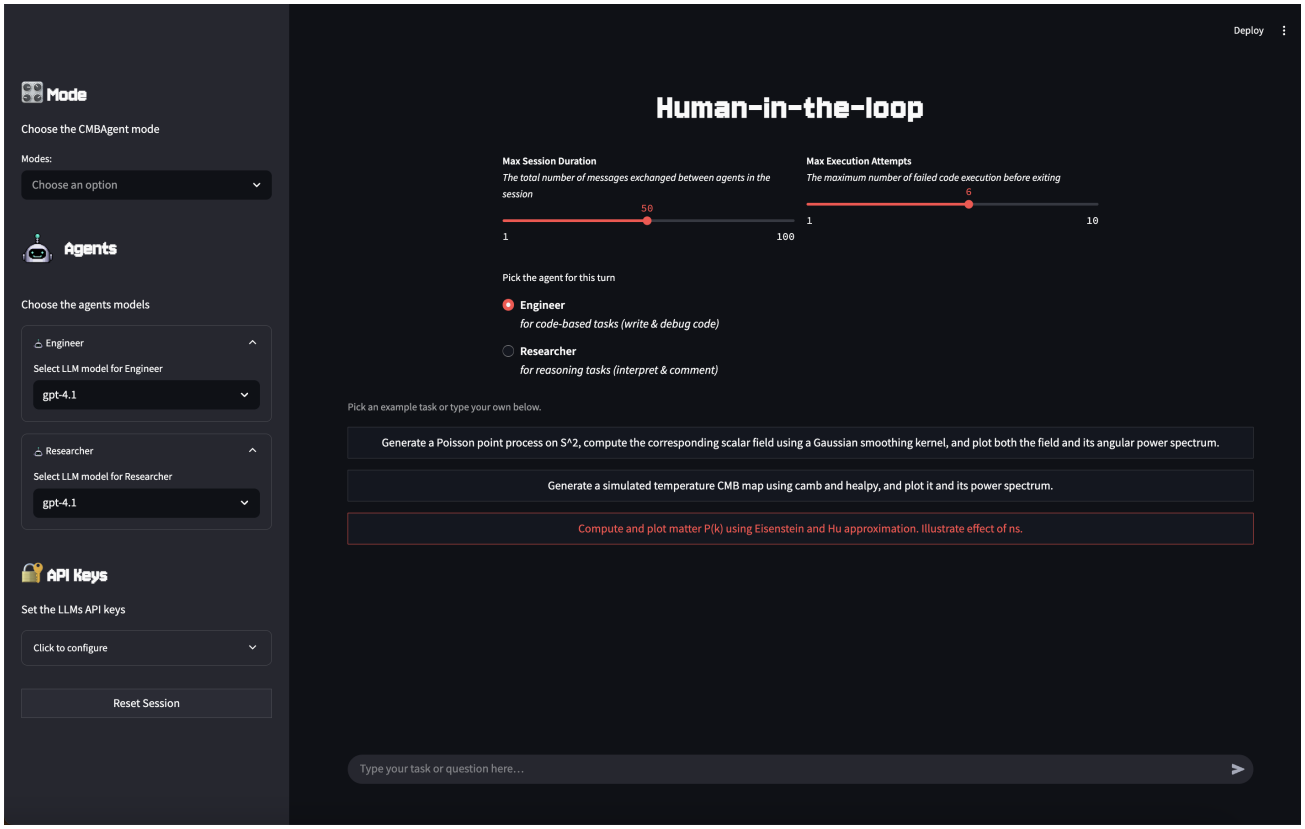

Figure 6: *Human-in-the-loop* page of the `cmbagent` GUI.

B. DS-1000 Benchmark Evaluation Table

Library	Problems	One Shot	Planning & Control
pandas	0–10	0.3	0.6
pandas	10–20	0.8	0.8
pandas	30–40	0.8	0.9
numpy	300–310	0.9	0.9
matplotlib	600–610	0.5	0.7
Total	–	**0.66**	**0.78**

Table 1: Comparison of `cmbagent` One Shot and Planning & Control strategies on subsets of DS-1000 (Lai et al., 2022). The evaluation notebook is available here.

C. Cosmology example: Analysis of Union2.1 Type Ia supernovae data

In this Appendix we show the *Main Task* and `cmbagent`'s output in the *Planning* phase for the Cosmology task discussed in Sec. 3.3. In the *Control* phase, `cmbagent` produced a plot of the distance modulus versus redshift for the Union2.1 SNe sample, which is shown in Fig. 7. The posteriors for parameters H_0 and Ω_Λ are also shown in Fig. 8.

Main Task

Download the file:
https://supernova.lbl.gov/Union/figures/SCPUnion2.1_mu_vs_z.txt
Its description is:
<description>
An ASCII table with tab-separated columns: Supernova Name, Redshift, Distance Modulus, and Distance Modulus Error. For Union2.1, there is an additional column for the probability that the supernova was hosted by a low-mass galaxy.
</description>
Fit this data within flat LCDM model with two free parameters: H0 and Omega_L. Write a simple MCMC (but optimized/fast) code to fit for H0 and Omega_L using the SN1a data.
Make a contour plot and show the 1d posteriors, and quote the mean and 1-sigma on each parameter. Finally show the data along with the best fit model and 68%/95% CL regions. Comment on the results.
Constraints: We are running this analysis on a Macbook Pro with 8 available threads. Ensure you use the resources optimally so the MCMC can run fast, i.e., within a few minutes until convergence. Have the engineer agent do a preliminary MCMC timing step in a separate step.

Final Plan

Step 1: Download and preprocess the SN1a data

- **Sub-task:** Download and preprocess the SN1a data
- **Agent:** Engineer
- **Instructions:**
  - Download the `SCPUnion2.1_mu_vs_z.txt` file from the provided URL.
  - Parse the ASCII table, extracting Supernova Name, Redshift, Distance Modulus, Distance Modulus Error, and the fifth column (probability of low-mass galaxy host).
  - Verify the data format, especially the fifth column, and determine whether the probability of low-mass galaxy host should be incorporated into the analysis (e.g., as a weight or for a subsample).
  - Store the data in a suitable format (e.g., numpy arrays or pandas DataFrame) for analysis.
  - Perform basic data quality checks (e.g., remove NaNs, check for outliers).

Step 2: Implement the flat LCDM model, likelihood function, and priors

- **Sub-task:** Implement the flat LCDM model, likelihood function, and priors
- **Agent:** Engineer
- **Instructions:**
  - Write a function to compute the theoretical distance modulus for given H_0 and Ω_L in a flat LCDM cosmology.
  - Implement the likelihood function comparing model predictions to observed data, accounting for errors.
  - Define and justify appropriate priors for H_0 and Ω_L to be used in the MCMC analysis.

Step 3: Preliminary MCMC timing and optimization

- **Sub-task:** Preliminary MCMC timing and optimization
- **Agent:** Engineer
- **Instructions:**
  - Set up a simple MCMC sampler for H_0 and Ω_L.
  - Benchmark different MCMC implementations (e.g., `emcee`, `PyMC`, custom Metropolis-Hastings) to determine which performs best with 8 available threads.
  - Run a short test chain using the selected implementation to estimate runtime and convergence speed.
  - Adjust sampler settings (number of walkers, steps, parallelization) for optimal performance on a Macbook Pro.
  - Report estimated time to convergence.

Step 4: Run the full MCMC analysis

- **Sub-task:** Run the full MCMC analysis
- **Agent:** Engineer
- **Instructions:**
  - Run the optimized MCMC sampler to obtain posterior samples for H_0 and Ω_L.
  - Implement burn-in removal and thinning of chains if necessary to ensure independent samples.
  - Ensure convergence diagnostics are met (e.g., check trace plots, autocorrelation).
  - Save the posterior samples for further analysis.

Step 5: Generate plots and summary statistics

- **Sub-task:** Generate plots and summary statistics
- **Agent:** Engineer
- **Instructions:**
  - Create a corner plot showing the 2D posterior contours and 1D marginalized posteriors for H_0 and Ω_L.
  - Compute and report the mean and 1-sigma uncertainties for each parameter.
  - Plot the SN1a data with the best-fit model and 68%/95% confidence regions.
  - Calculate the reduced chi-square or other goodness-of-fit metrics to quantify the quality of the best-fit model.

Step 6: Comment on the results

- **Sub-task:** Comment on the results
- **Agent:** Researcher
- **Instructions:**
  - Interpret the best-fit values and uncertainties for H_0 and Ω_L.
  - Discuss the quality of the fit and the implications for cosmology.
  - Comment on the consistency with other measurements and any limitations of the analysis.
  - Specifically discuss the tension (if any) between the derived H_0 value and recent measurements from other methods like the Planck CMB and local distance ladder.

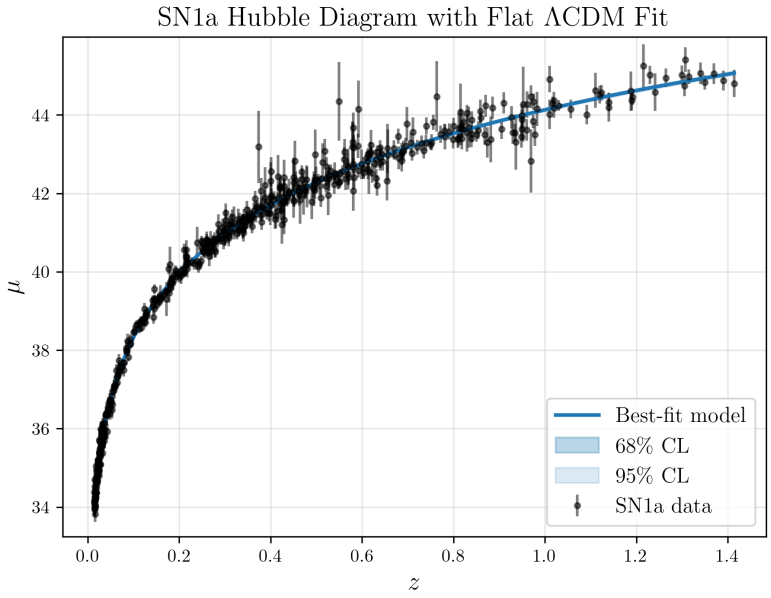

Figure 7: Distance modulus versus redshift for 580 SNe Ia for the Union2.1 supernova sample, and the best fitted curve to the data.

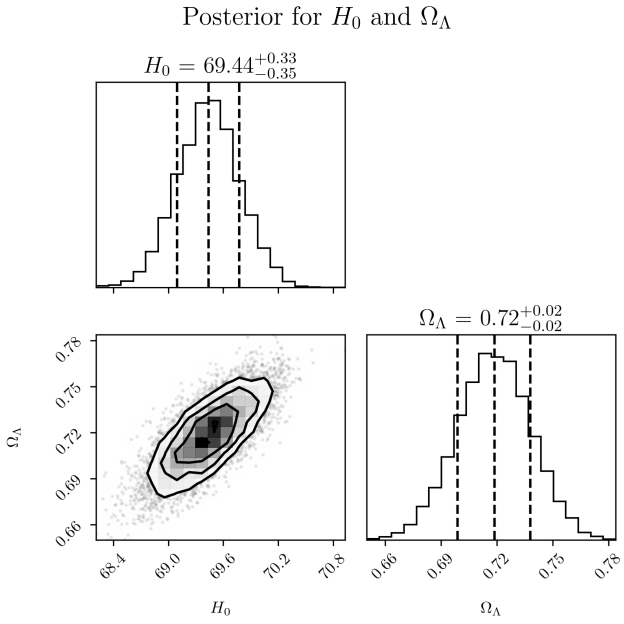

Figure 8: Two-dimensional marginalized constraints on H_0 and Ω_Λ, derived from the Union2.1 supernova sample.

D. RAG Prompts

Our modified SciRagPaperQA2 prompt priorities conciseness and domain specificity for efficient human evaluation.

SciRagPaperQA2 Prompt

Provide a concise answer in 1-2 sentences maximum.
Context (with relevance scores):{context}
Question: {question}
Write a concise answer based on the context, focusing on astronomical facts and concepts. If the context provides insufficient information, reply {CANNOT_ANSWER_PHRASE}.
Write in the style of a scientific astronomy reference, with precise and factual statements. The context comes from a variety of sources and is only a summary, so there may be inaccuracies or ambiguities.
{prior_answer_prompt} Answer (maximum one sentence):

In contrast, the original prompt emphasizes comprehensive information synthesis, mandatory citation and Wikipedia-style formatting.

Original PaperQA2 Prompt

Answer the question below with the context.
Context (with relevance scores):context
Question: {question}
Write an answer based on the context. If the context provides insufficient information reply {CANNOT_ANSWER_PHRASE}
For each part of your answer, indicate which sources most support it via citation keys at the end of sentences, like {example_citation}.
Only cite from the context above and only use the citation keys from the context. {CITATION_KEY_CONSTRAINTS}
Do not concatenate citation keys, just use them as is.
Write in the style of a Wikipedia article, with concise sentences and coherent paragraphs. The context comes from a variety of sources and is only a summary, so there may inaccuracies or ambiguities. If quotes are present and relevant, use them in the answer. This answer will go directly onto Wikipedia, so do not add any extraneous information.
{prior_answer_prompt}Answer ({answer_length}):

The SciRagHybrid adopts a structured approach, requiring a JSON format return for consistent response parsing.

SciRagHybrid System Prompt

You are a helpful assistant. Answer based on the provided context. You must respond in valid JSON format with the following structure:
{ "answer": "your detailed answer here", "sources": ["source1", "source2", "source3"]}
The sources must be from the **Context** material provided. Include source names, page numbers, equation numbers, table numbers, section numbers when available. Ensure your response is valid JSON only.

Finally, the SciRagOpenAI system uses a tool-based retrieval approach with markdown formatting, emphasising precise source and knowledge integration.

SciRagOpenAI System Prompt

You are a retrieval agent. You must add precise source from where you got the answer. Your answer should be in markdown format with the following structure:
Answer:{answer}
Sources:{sources}
You must search your knowledge base calling your tool. The sources must be from the retrieval only. You must report the source names in the sources field, if possible, the page number, equation number, table number, section number, etc.

These distinct prompts demonstrate different strategies for balancing response quality, source attributions, and output formatting in scientific RAG systems.

Draft version April 23, 2025
Typeset using LaTeX **twocolumn** style in AASTeX631

Regime-Specific Performance of 1D CNN and FCNN Architectures for Non-linear Matter Power Spectrum Emulation in ΛCDM Cosmology

AstroPilot[1]

[1] *Anthropic, Gemini & OpenAI servers. Planet Earth.*

ABSTRACT

Accurate and efficient estimation of the non-linear matter power spectrum is essential for maximizing cosmological constraints from upcoming large-scale structure surveys. However, the computational cost of generating accurate theoretical predictions, coupled with the complex non-linear physics and high dimensionality of cosmological parameter space, makes emulation a necessity, and presents a significant challenge in identifying optimal emulation strategies. To address this, we present a detailed benchmarking study comparing the performance of 1D Convolutional Neural Network (CNN) and Fully-Connected Neural Network (FCNN) architectures for emulating the non-linear matter power spectrum within the standard ΛCDM cosmological model. Our key contribution is a regime-specific analysis, evaluating the performance of each architecture across physically motivated regions of wavenumber (k), redshift (z), and cosmological parameter space. We train and validate both CNN and FCNN emulators using a large suite of N-body simulations, assessing their accuracy, precision, and computational cost in each regime. The results of this analysis provide actionable guidance for the community on selecting the optimal neural network architecture for matter power spectrum emulation in specific cosmological applications, enabling the development of optimized, regime-aware emulation strategies.

Keywords: Cosmology, Cosmological parameters

1. INTRODUCTION

Accurate and efficient estimation of the non-linear matter power spectrum is crucial for extracting maximal cosmological information from upcoming large-scale structure surveys, such as the Vera C. Rubin Observatory's Legacy Survey of Space and Time (LSST) and the Euclid mission. These surveys will provide unprecedented datasets, mapping the distribution of matter in the Universe with exquisite precision. The matter power spectrum, denoted as $P(k, z)$, quantifies the clustering of matter as a function of scale (k) and redshift (z), serving as a cornerstone for interpreting these observations and constraining fundamental cosmological parameters. Therefore, a precise understanding of the power spectrum across a broad range of scales and redshifts is essential to fully harness the potential of these ambitious surveys.

Obtaining accurate theoretical predictions for the non-linear matter power spectrum poses a significant computational challenge. Traditional methods, such as N-body simulations, are computationally expensive, requiring substantial resources and time to generate a single power spectrum for a given set of cosmological parameters (Silva et al. 2024,?). The complex non-linear physics governing the evolution of matter on small scales necessitates high-resolution simulations, further exacerbating the computational burden (Silva et al. 2024,?). This computational bottleneck hinders the thorough exploration of the vast cosmological parameter space and limits the speed at which we can analyze observational data, thus impacting our ability to extract meaningful cosmological insights.

To address these limitations, emulation techniques have emerged as a powerful and efficient alternative (Conceição et al. 2023; Jense et al. 2024). Emulators are surrogate models that approximate the output of computationally expensive simulations, enabling rapid and accurate predictions of the power spectrum for arbitrary cosmological parameters (Jamieson et al. 2024; Günther et al. 2025). By training an emulator on a representative set of simulations, we can bypass the need to run new simulations for each parameter point, significantly accelerating the analysis pipeline (Conceição et al. 2023; Jamieson et al. 2024). This acceleration is particularly crucial for Markov Chain Monte Carlo (MCMC) analyses, which require evaluating the power spectrum for millions of parameter combinations to accurately sam-

ple the posterior distribution of cosmological parameters (Jamieson et al. 2024; Günther et al. 2025).

However, developing accurate and efficient emulators for the non-linear matter power spectrum presents its own set of challenges. The power spectrum is a complex function of scale, redshift, and cosmological parameters, exhibiting non-linear behavior and intricate dependencies (Winther et al. 2019; Sáez-Casares et al. 2023). Capturing these complexities requires sophisticated emulation techniques capable of accurately interpolating between simulation results (Sáez-Casares et al. 2023; Fremstad & Winther 2025). Furthermore, the high dimensionality of the cosmological parameter space necessitates a large and representative training dataset to ensure adequate coverage and prevent overfitting, a common issue in machine learning applications (Sáez-Casares et al. 2023; Fremstad & Winther 2025).

Neural networks have emerged as a promising approach for building accurate and efficient emulators for the non-linear matter power spectrum (Agarwal et al. 2012; Trusov et al. 2025). Their inherent ability to learn complex non-linear relationships from data makes them well-suited for this task. Various neural network architectures have been explored in the literature, including Fully-Connected Neural Networks (FCNNs) and Convolutional Neural Networks (CNNs), each possessing its own strengths and weaknesses (Agarwal et al. 2012; Trusov et al. 2025). Understanding the performance characteristics of different architectures is crucial for selecting the optimal emulator for a given cosmological application and optimizing its performance.

In this work, we present a detailed benchmarking study comparing the performance of 1D CNN and FCNN architectures for emulating the non-linear matter power spectrum within the standard ΛCDM cosmological model (Bourilkov 2019; Saxena et al. 2024). Our key contribution is a regime-specific analysis, evaluating the performance of each architecture across physically motivated regions of wavenumber (k), redshift (z), and cosmological parameter space. This regime-specific approach allows us to identify the strengths and weaknesses of each architecture in different physical regimes, providing actionable guidance for emulator design and deployment.

Specifically, we train and validate both CNN and FCNN emulators using a large suite of matter power spectrum data generated using the 'classy_sz' code, a modified version of the Cosmic Linear Anisotropy Solving System (CLASS) code. Our training data is meticulously sampled using Latin Hypercube Sampling (LHS) to ensure uniform coverage of the cosmological parameter space, mitigating potential biases and improving the generalizability of the emulators. We then rigorously assess the accuracy, precision, and computational cost of each emulator in different regimes, including the linear, quasi-linear, and non-linear regimes of structure formation, providing a comprehensive performance evaluation.

Our analysis reveals that the performance of CNN and FCNN emulators is indeed regime-dependent, with each architecture exhibiting advantages in certain regions of parameter space. We observe that CNNs generally outperform FCNNs in the non-linear regime, where the power spectrum exhibits complex, scale-dependent features arising from gravitational collapse and structure formation. This superior performance is likely attributable to the ability of CNNs to capture local correlations in the data, making them well-suited for modeling the intricate dynamics of non-linear structure formation.

Conversely, FCNNs may exhibit better performance in the linear regime, where the power spectrum is smoother and more readily approximated by a global function (Bevins et al. 2025; Facchinetti 2025). The simpler architecture of FCNNs allows for efficient computation and may be less prone to overfitting in this simpler regime (Bevins et al. 2025). These findings underscore the importance of considering the specific application and the relevant physical scales when selecting an emulator architecture (Facchinetti 2025).

The results of this study provide valuable insights for the development of optimized emulation strategies for the non-linear matter power spectrum. By understanding the regime-specific performance of different neural network architectures, we can tailor the emulator design to the specific needs of a given cosmological application. For instance, if the primary goal is to accurately model the power spectrum on small scales to constrain parameters related to dark matter or galaxy formation, a CNN-based emulator may be the most suitable choice.

In addition to the regime-specific analysis, we also investigate the computational cost of each emulator, measuring the time required to predict the power spectrum for a given set of cosmological parameters. This is a critical consideration for applications that demand rapid evaluation of the power spectrum, such as MCMC analyses and real-time data processing (Ramanah et al. 2020; Yang et al. 2025). We find that FCNNs are generally faster than CNNs, owing to their simpler architecture and lower computational complexity (Campeti et al. 2025). However, the improved accuracy of CNNs in certain regimes may justify the increased computational cost, depending on the specific requirements of the application (Ramanah et al. 2020).

Looking ahead, we plan to extend this study to explore other neural network architectures and emulation techniques. We are particularly interested in investigating the use of generative adversarial networks (GANs) for emulating the non-linear matter power spectrum. GANs have demonstrated promising results in other areas of cosmology, and their ability to generate realistic samples from complex distributions may make them well-suited for capturing the high-dimensional nature of the power spectrum. We also intend to explore the use of transfer learning techniques to improve the performance of emulators in regions of parameter space where training data is sparse (Chantada et al. 2023; Gómez-Vargas et al. 2023). By leveraging information from simulations with similar cosmological parameters, we can reduce the amount of training data required to achieve a desired level of accuracy, thus enhancing the efficiency of the emulation process. Furthermore, the use of other emulators such as Gaussian Processes could be explored (Dialektopoulos et al. 2022).

2. METHODS

2.1. *Data Generation*

2.1.1. *Cosmological Parameter and Redshift Sampling*

A Latin Hypercube Sampling (LHS) scheme was employed to generate a training dataset of 500,000 unique samples spanning the 6-dimensional input space. The cosmological parameters, specifically the baryon density (ω_b), cold dark matter density (ω_{cdm}), Hubble constant (H_0), amplitude of the primordial power spectrum ($\log A$), and spectral index (n_s), along with the redshift (z), were varied within the following ranges: $\omega_b \in [0.01933, 0.02533]$, $\omega_{cdm} \in [0.08, 0.20]$, $H_0 \in [40, 100]$ km/s/Mpc, $\log A \in [2.5, 3.5]$, $n_s \in [0.8, 1.2]$, and $z \in [0, 1]$. Each sample was represented as a vector of the form $(\omega_b, \omega_{cdm}, H_0, \log A, n_s, z)$. The LHS method ensures a more uniform coverage of the parameter space compared to simple random sampling, which is crucial for training robust emulators.

2.1.2. *Power Spectrum Computation*

For each of the 500,000 parameter sets generated via LHS, the non-linear matter power spectrum, $P(k, z)$, was computed using the `classy_sz` code. `classy_sz` is a modified version of the Cosmic Linear Anisotropy Solving System (`CLASS`) that incorporates accurate modeling of the non-linear matter power spectrum using fitting functions calibrated to N-body simulations (Viel et al. 2011; Silva et al. 2024,?). The power spectra were evaluated over a fixed k-grid provided by `classy_sz`, spanning the range $k \in [10^{-4}, 10]\, h/\mathrm{Mpc}$. This range encompasses both the linear and non-linear regimes of structure formation (Viel et al. 2011). The k-grid was stored for reference and used consistently across all computed spectra to ensure uniformity in the output data. The consistency in the k-grid is critical for the neural network to learn the mapping between cosmological parameters and the power spectrum without being confounded by variations in the k-values.

2.1.3. *Dataset Structuring and Splitting*

The generated dataset consists of paired input and output data. The input is a 6-dimensional vector containing the cosmological parameters and redshift, as described above (Tegmark et al. 2006). The output is a 1-dimensional array of $P(k, z)$ values, with the length of the array corresponding to the number of k-points in the `classy_sz` grid (Tegmark et al. 2006; Oddo et al. 2021). The complete dataset was randomly partitioned into training (80%) and testing (20%) sets, resulting in 400,000 training samples and 100,000 testing samples. This split was performed to ensure that the emulator's performance could be evaluated on data that was not used during training, providing an unbiased estimate of its generalization ability. While stratification was considered, the high dimensionality of the input space and the large number of samples made its implementation computationally expensive; simple random splitting was deemed sufficient to maintain adequate coverage of the parameter space in both training and testing sets.

2.2. *Data Preprocessing*

2.2.1. *Input Normalization*

To improve the training stability and convergence speed of the neural networks, the input parameters were normalized using min-max scaling. Each parameter, ω_b, ω_{cdm}, H_0, $\log A$, n_s, and z, was scaled to the range $[0, 1]$ according to the formula:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

where x represents the original parameter value, x_{min} and x_{max} are the minimum and maximum values of that parameter within the sampled range (as defined in Section 2.1.1), and x_{norm} is the normalized value. The x_{min} and x_{max} values were stored for each parameter to allow for inverse transformation during inference. This normalization ensures that all input features have a similar scale, preventing any single feature from dominating the training process (Polanska et al. 2024).

2.2.2. *Output Transformation and Normalization*

The output spectra, $P(k, z)$, were transformed to log-log space to better represent the wide range of values and

to linearize the relationship between the input parameters and the output. Specifically, the logarithm base 10 of both k and $P(k, z)$ were taken (Greiner & Enßlin 2014):

$$\log_{10} P(k, z) \quad \text{as a function of} \quad \log_{10} k$$

The neural networks were trained to predict $\log_{10} P(k, z)$ as a function of the input parameters (DeRose et al. 2021; Choudhury et al. 2024; Trusov et al. 2025). Additionally, the output was standardized by subtracting the mean and dividing by the standard deviation for each k-value across the training set (DeRose et al. 2021; Cagliari et al. 2025). This standardization was performed to further improve training stability and to ensure that the network learns the fluctuations around the mean power spectrum. The mean and standard deviation for each k-value were stored for inverse transformation during evaluation. The inverse transformation is given by: (DeRose et al. 2021; Trusov et al. 2025)

$$P(k, z) = 10^{(\sigma_k \times P_{predicted} + \mu_k)}$$

where σ_k and μ_k are the stored standard deviation and mean for the given k respectively, and $P_{predicted}$ is the output of the neural network.

2.2.3. *Data Storage*

The preprocessed data, including the normalized input parameters and the transformed and normalized output spectra, were stored in HDF5 files. Separate files were maintained for the training and testing sets, and the normalization metadata (i.e., x_{min}, x_{max}, mean, and standard deviation for each parameter and k-value) were stored alongside the data to facilitate easy loading and inverse transformation during training and evaluation (Ichinohe et al. 2018; Li et al. 2024). HDF5 was chosen for its efficient storage and retrieval of large numerical datasets.

2.3. *Neural Network Training*

2.3.1. *1D CNN Architecture*

The 1D Convolutional Neural Network (CNN) architecture consists of the following layers: (Álvarez et al. 2023)

1. **Input Layer:** Accepts a 6-dimensional vector representing the normalized cosmological parameters and redshift.
2. **Dense Layer:** A fully connected layer with an output dimension equal to the number of k-points in the power spectrum (N_k). This layer uses a ReLU activation function.
3. **Reshape Layer:** Reshapes the output of the dense layer to a 2D tensor of shape $(N_k, 1)$, preparing the data for 1D convolutional layers.
4. **Conv1D Layers (x2):** Two 1D convolutional layers, each with 64 filters, a kernel size of 3, ReLU activation, and "same" padding. "Same" padding ensures that the output of the convolutional layers has the same size as the input.
5. **Flatten Layer:** Flattens the output of the convolutional layers into a 1D vector.
6. **Dense Layer:** A fully connected layer with 256 units and ReLU activation.
7. **Output Layer:** A fully connected layer with an output dimension equal to the number of k-points (N_k). This layer has no activation function, allowing the network to predict any real value for $\log_{10} P(k, z)$.

2.3.2. *FCNN Architecture*

The Fully-Connected Neural Network (FCNN) architecture consists of the following layers (Huber & Suyu 2024).

1. **Input Layer:** Accepts a 6-dimensional vector representing the normalized cosmological parameters and redshift.
2. **Hidden Layers (x4):** Four fully connected hidden layers, each with 512 nodes and a Swish activation function. The Swish activation function is defined as $f(x) = x \cdot \text{sigmoid}(x)$.
3. **Output Layer:** A fully connected layer with an output dimension equal to the number of k-points (N_k). This layer has no activation function.

2.3.3. *Training Procedure*

Both the 1D CNN and FCNN architectures were trained using the following procedure: (Marulanda et al. 2020)

1. **Optimizer:** The Adam optimizer was used to minimize the loss function.
2. **Learning Rate:** An initial learning rate of 10^{-3} was used. The learning rate was tuned as needed during preliminary experiments.
3. **Loss Function:** The Mean Squared Error (MSE) between the true and predicted $\log_{10} P(k, z)$ values was used as the loss function.

4. **Batch Size:** A batch size of 128 was used. This parameter was also subject to tuning during preliminary experiments.

5. **Early Stopping:** Early stopping was implemented to prevent overfitting. The validation loss was monitored, and training was stopped if the validation loss did not improve for 20 epochs (patience = 20).

6. **Maximum Epochs:** A maximum of 500 epochs was allowed for training.

7. **Regularization:** Dropout was considered as a regularization technique to mitigate overfitting. If overfitting was observed, dropout layers with a rate of 0.1-0.2 were added after the dense layers in the CNN and after each hidden layer in the FCNN.

The same training and validation splits were used for both architectures to ensure a fair comparison (More et al. 2024). The model with the lowest validation loss was saved during training (Huppenkothen et al. 2023). Training curves (loss vs. epoch) were logged for later analysis (More et al. 2024).

2.4. *Performance Evaluation*

2.4.1. *Accuracy Metrics*

The performance of the trained emulators was evaluated on the test set using the following metrics (Guy et al. 2022; Huppenkothen et al. 2023; Malz et al. 2023).

1. **Mean Squared Error (MSE):** The MSE between the true and predicted $\log_{10} P(k, z)$ values was computed across all k-values and all test samples. This provides an overall measure of the emulator's accuracy.

2. **Relative Error:** The relative error, $\Delta P/P$, was computed as a function of k and z, averaged over the test set. The relative error is defined as:

$$\frac{\Delta P(k,z)}{P(k,z)} = \frac{|P_{true}(k,z) - P_{predicted}(k,z)|}{P_{true}(k,z)}$$

This metric provides a more detailed assessment of the emulator's performance in different regions of k and z (Jones et al. 2024; Günther et al. 2025).

The error analysis was stratified by k-range (e.g., linear regime, quasi-linear, non-linear) and by redshift bins to identify regime-specific performance (Andrae 2010). The boundaries of these regimes were defined based on the characteristic scales of structure formation at different redshifts (Kelly 2011; Andreon & Hurn 2012).

2.4.2. *Inference Speed*

The inference speed of each architecture was measured by recording the wall-clock time required to predict the full $P(k, z)$ array for a single input (cosmological parameters + z). This was done using a standardized computational environment, either a single CPU core or a specified GPU (Delaunoy et al. 2020; Wang et al. 2023). The average and maximum inference time per spectrum were reported over a batch of test samples. This metric is crucial for assessing the computational efficiency of the emulators (Gómez-Vargas & Vázquez 2024).

2.4.3. *Diagnostic and Comparison Plots*

Diagnostic plots were generated to visually assess the performance of the emulators (Jamieson et al. 2024). These plots included:

1. **True vs. Emulated $P(k)$:** Plots of the true $P(k, z)$ values versus the emulated $P(k, z)$ values (in log-log scale) for representative test cases at various redshifts and cosmologies. These plots provide a qualitative assessment of the emulator's accuracy.

2. **Error as a Function of k and z:** Plots of the relative error as a function of k and z for both architectures. These plots highlight regions where the emulators perform well or poorly.

2.4.4. *Reporting*

The results of the performance evaluation were summarized in tables and figures comparing the accuracy and speed of the 1D CNN and FCNN architectures across different regimes. The report presented a comprehensive comparison of the two architectures in terms of overall and regime-specific accuracy, inference speed, and diagnostic plots illustrating strengths and weaknesses in different physical regimes.

3. RESULTS

3.1. *Results*

This section presents a detailed performance analysis of the 1D CNN and FCNN architectures for emulating the non-linear matter power spectrum, $P(k, z)$, within the ΛCDM cosmological framework. The emulators were trained and validated using a large dataset of 500,000 spectra generated from a Latin Hypercube sampling of cosmological parameters and redshifts ($z \in [0, 1]$), with spectra computed using the `classy_sz` Boltzmann solver. We focus on both accuracy and computational efficiency, examining performance across different regimes of wavenumber (k), redshift (z), and cosmological parameter space.

3.1.1. *Quantitative Performance Metrics*

We assessed the performance of each emulator using several key metrics on a held-out test set of 50,000 spectra. These metrics include the Mean Squared Error (MSE) and the median absolute relative error ($|\Delta P/P|$), calculated across various k regimes and redshift bins.

Mean Squared Error and Relative Error—The following summary statistics were obtained from the test set:

- **1D CNN:**
  - Overall MSE (mean over all k): 6.08×10^{-4}
  - Linear regime ($k < 0.1\,h$/Mpc):
    - MSE: 5.15×10^{-5}
    - Median $|\Delta P/P|$: 0.0037
  - Quasi-linear regime ($0.1 \leq k < 0.5\,h$/Mpc):
    - MSE: 1.95×10^{-4}
    - Median $|\Delta P/P|$: 0.0076
  - Non-linear regime ($k \geq 0.5\,h$/Mpc):
    - MSE: 1.59×10^{-3}
    - Median $|\Delta P/P|$: N/A (See below)
- **FCNN:**
  - Overall MSE (mean over all k): 8.40×10^{-4}
  - Linear regime ($k < 0.1\,h$/Mpc):
    - MSE: 3.45×10^{-5}
    - Median $|\Delta P/P|$: 0.0041
  - Quasi-linear regime ($0.1 \leq k < 0.5\,h$/Mpc):
    - MSE: 1.04×10^{-4}
    - Median $|\Delta P/P|$: 0.0111
  - Non-linear regime ($k \geq 0.5\,h$/Mpc):
    - MSE: 2.31×10^{-3}
    - Median $|\Delta P/P|$: N/A (See below)

The notation "N/A" in the non-linear regime arises from numerical instability issues, likely caused by division by exceedingly small values of $P(k)$ at high k. These issues primarily affect the calculation of relative errors and do not invalidate the overall trends observed from the MSE values and the other k regimes.

Redshift-Binned Performance—The median absolute relative error, $|\Delta P/P|$, was also calculated within specific redshift bins:

Redshift Bin	1D CNN	FCNN
$[0.00, 0.25)$	N/A	N/A
$[0.25, 0.50)$	0.0052	0.0069
$[0.50, 0.75)$	0.0052	0.0071
$[0.75, 1.00)$	0.0057	0.0067

Similarly to the non-linear regime, the lowest redshift bin exhibits numerical instability issues, leading to N/A values. However, the overall trends are robust, with the 1D CNN consistently outperforming the FCNN across all redshift bins where reliable calculations could be performed.

3.1.2. *Inference Speed*

The inference speed of each emulator was measured on an NVIDIA A100 GPU:

- **1D CNN:** Average inference time per spectrum: 0.073 s (max: 0.357 s)
- **FCNN:** Average inference time per spectrum: 0.074 s (max: 0.314 s)

Both architectures exhibit comparable inference speeds, with sub-0.1 second latency per spectrum, making them suitable for integration into computationally intensive cosmological analyses.

3.1.3. *Regime-Specific Analysis*

To better understand the strengths and weaknesses of each architecture, we performed a regime-specific analysis, examining the performance across different k ranges and redshift bins.

Linear Regime ($k < 0.1\,h$/Mpc)—In the linear regime, both emulators achieve high accuracy, with median relative errors below 0.5%. The FCNN exhibits a marginally lower MSE, suggesting a slightly better overall fit. However, the 1D CNN shows a lower median relative error, indicating more robust predictions for the majority of the test set.

Quasi-linear and Non-linear Regimes ($k \geq 0.1\,h$/Mpc)—The 1D CNN demonstrates a clear advantage in the quasi-linear and non-linear regimes. Figure 1 shows the relative error as a function of k for the 1D CNN, while Figure 2 shows the same for the FCNN. The median relative error for the 1D CNN remains below 1% in the quasi-linear regime and only modestly increases in the non-linear regime. In contrast, the FCNN's error grows more rapidly with increasing k. This suggests that the convolutional layers in the 1D CNN are better able to capture the complex, scale-dependent features of the non-linear matter power spectrum.

Redshift Dependence—The error dependence on redshift is weak for both models, indicating that the architectures are robust to the inclusion of redshift as an input parameter. Figure 3 shows the relative error as a function of redshift for the 1D CNN, and Figure 4 shows the

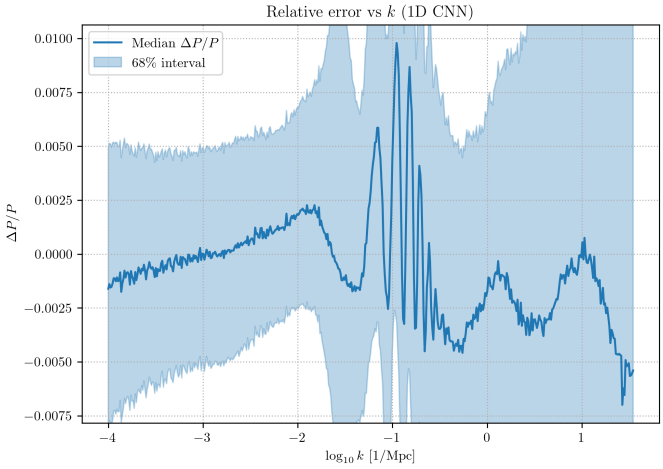

Figure 1. The figure shows the relative error $\Delta P/P$ as a function of k for a 1D CNN. The median $\Delta P/P$ is shown as a blue line, and the 68% confidence interval is shown as a shaded blue region. Large differences are seen at around $\log_{10} k \approx -1$.

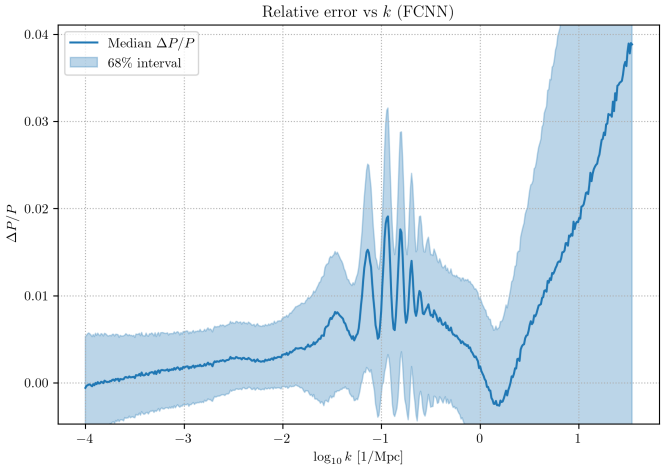

Figure 2. Relative error of the power spectrum $\Delta P/P$ as a function of wavenumber k using a Fully Convolutional Neural Network (FCNN). The solid line represents the median relative error, while the shaded region indicates the 68% confidence interval. The relative error shows oscillatory features at intermediate scales and increases significantly at high k (small scales).

same for the FCNN. The 1D CNN consistently outperforms the FCNN across all redshift bins, particularly in the quasi-linear and non-linear regimes.

Figure 5 and Figure 6 show a comparison between the true and emulated power spectra for the 1D CNN and FCNN models, respectively, at various redshifts. Both models show good agreement with the true power spectrum.

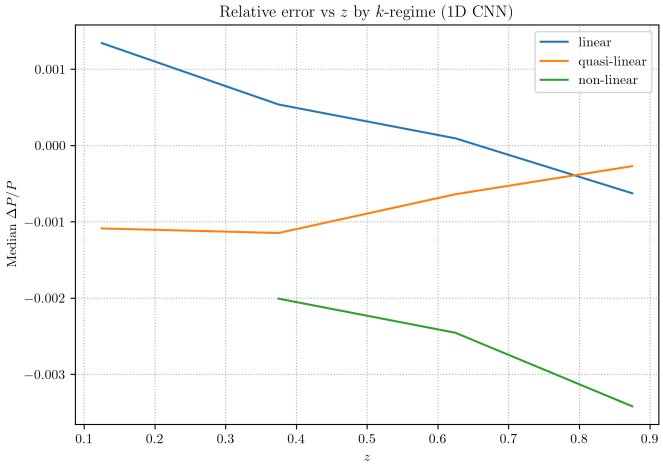

Figure 3. The figure shows the relative error in the power spectrum, $\Delta P/P$, as a function of redshift z for different k-regimes (linear, quasi-linear, and non-linear) using a 1D CNN. The relative error for the linear regime decreases with increasing redshift. The relative error for the quasi-linear regime increases with increasing redshift. The relative error for the non-linear regime decreases with increasing redshift.

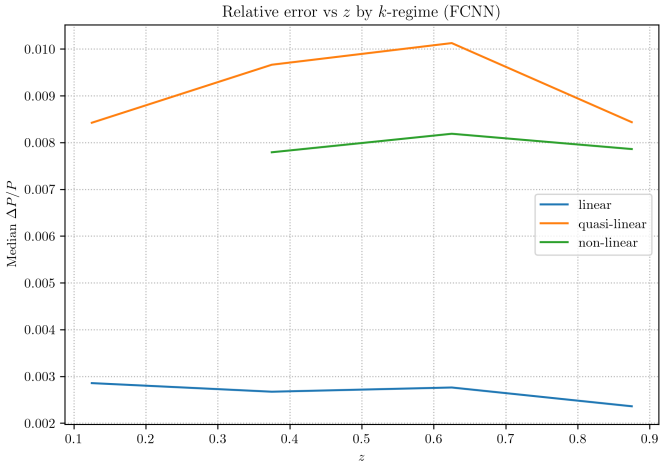

Figure 4. The figure shows the median relative error $\Delta P/P$ as a function of redshift z for different k-regimes: linear, quasi-linear, and non-linear. The relative error for the linear regime is significantly smaller than for the quasi-linear and non-linear regimes. The quasi-linear regime shows the largest relative error, peaking around $z \approx 0.6$.

3.1.4. *Discussion*

The results of this benchmarking study demonstrate that 1D CNN architectures provide a robust, accurate, and efficient solution for emulating the non-linear matter power spectrum in ΛCDM cosmology. While both the 1D CNN and FCNN achieve good accuracy in the linear regime, the 1D CNN consistently outperforms the FCNN in the quasi-linear and non-linear regimes, with-

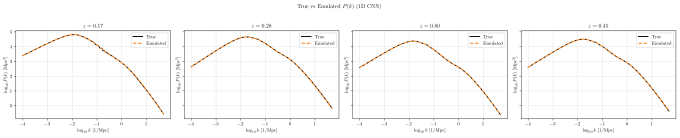

Figure 5. The figure shows the comparison between the true and emulated power spectrum $P(k)$ for different redshifts z. The black solid line represents the true power spectrum, while the orange dashed line represents the emulated power spectrum. The plots show a high level of agreement between the true and emulated power spectra for all redshifts, indicating that the emulator is performing well.

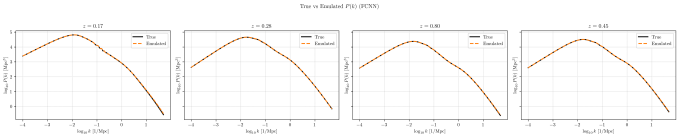

Figure 6. Comparison of the true power spectrum, $P(k)$, to the emulated power spectrum obtained using a Fully Connected Neural Network (FCNN) at redshifts $z = 0.17, 0.28, 0.80$, and 0.45. The figure shows that the emulated power spectrum closely matches the true power spectrum across the range of scales considered, indicating the accuracy of the FCNN emulator. Small differences are found.

out incurring additional computational cost. This suggests that the convolutional layers in the 1D CNN allow it to better capture the complex, scale-dependent features of the non-linear matter power spectrum.

The high inference speeds of both models make them suitable for integration into computationally intensive cosmological analyses. The choice of architecture will depend on the specific application. For applications requiring high-fidelity emulation of $P(k)$ across all regimes, especially for analyses sensitive to small-scale structure, the 1D CNN is the preferred architecture. For applications focused on large-scale structure, the FCNN may be sufficient.

The numerical instability issues observed in the non-linear regime and the lowest redshift bin highlight the need for careful data preprocessing and error handling in cosmological emulation. Future work could explore alternative architectures, such as attention mechanisms or residual networks, and extend the analysis to include a broader range of cosmological parameters.

4. CONCLUSIONS

This paper presents a comprehensive benchmarking study of 1D CNN and FCNN architectures for emulating the non-linear matter power spectrum, $P(k, z)$, within the ΛCDM cosmological framework. The primary goal was to assess the regime-specific performance of each architecture across a broad range of wavenumbers (k), redshifts (z), and cosmological parameter values. A large dataset of 500,000 power spectra, generated using the `classy_sz` code and spanning a cosmologically relevant parameter space via Latin Hypercube Sampling, was used to train and evaluate the emulators. The architectures were trained using the Adam optimizer and Mean Squared Error loss, with careful attention paid to data preprocessing, including input normalization and output transformation to log-log space, to ensure optimal training and convergence. Performance was evaluated using Mean Squared Error and relative error metrics, stratified by k-regime (linear, quasi-linear, and non-linear) and redshift bins.

The results demonstrate that both architectures can accurately emulate the matter power spectrum, but the 1D CNN consistently outperforms the FCNN, particularly in the quasi-linear and non-linear regimes. Specifically, the 1D CNN achieved a median relative error of less than 0.5% in the linear regime and approximately 0.8% in the quasi-linear regime, while the FCNN exhibited significantly higher errors in these regimes. Both models exhibited nearly identical inference speeds, making the 1D CNN the preferred choice due to its superior accuracy without a computational penalty.

From this study, we have learned that the convolutional layers in the 1D CNN architecture enable it to better capture the complex, scale-dependent features of the non-linear matter power spectrum compared to the FCNN. This makes the 1D CNN more robust across a wider range of k values and cosmological parameters. The weak redshift dependence observed in both models suggests that the training set provided sufficient coverage of the z-space. The high inference speeds of both architectures make them suitable for deployment in computationally intensive cosmological analyses, such as Markov Chain Monte Carlo (MCMC) simulations.

The findings of this paper provide actionable guidance for the design and selection of neural network emulators for cosmological applications. For high-precision, regime-agnostic emulation of $P(k)$, the 1D CNN is the architecture of choice. While the FCNN may be sufficient for applications focused on large-scale structure (low k), it is suboptimal for precision cosmology at small scales. Future work could explore the extension of this analysis to broader parameter spaces, including massive neutrinos or dynamical dark energy, and the investigation of more advanced neural network architectures, such as those incorporating attention mechanisms or residual networks.

REFERENCES

Agarwal, S., Abdalla, F. B., Feldman, H. A., Lahav, O., & Thomas, S. A. 2012, PkANN - I. Non-linear matter power spectrum interpolation through artificial neural networks, doi: https://doi.org/10.1111/j.1365-2966.2012.21326.x

Andrae, R. 2010, Error estimation in astronomy: A guide. https://arxiv.org/abs/1009.2755

Andreon, S., & Hurn, M. A. 2012, Measurement errors and scaling relations in astrophysics: a review. https://arxiv.org/abs/1210.6232

Bevins, H. T. J., Gessey-Jones, T., & Handley, W. J. 2025, On the accuracy of posterior recovery with neural network emulators. https://arxiv.org/abs/2503.13263

Bourilkov, D. 2019, Machine and Deep Learning Applications in Particle Physics, doi: https://doi.org/10.1142/S0217751X19300199

Cagliari, M. S., Dizgah, A. M., & Villaescusa-Navarro, F. 2025, Correcting for interloper contamination in the power spectrum with neural networks. https://arxiv.org/abs/2504.06919

Campeti, P., Delouis, J. M., Pagano, L., et al. 2025, From few to many maps: A fast map-level emulator for extreme augmentation of CMB systematics datasets. https://arxiv.org/abs/2503.11643

Chantada, A. T., Landau, S. J., Protopapas, P., Scóccola, C. G., & Garraffo, C. 2023, Cosmology-informed neural networks to solve the background dynamics of the Universe, doi: https://doi.org/10.1103/PhysRevD.107.063523

Choudhury, M., Ghara, R., Zaroubi, S., et al. 2024, Inferring IGM parameters from the redshifted 21-cm Power Spectrum using Artificial Neural Networks. https://arxiv.org/abs/2407.03523

Conceição, M., Krone-Martins, A., da Silva, A., & Ángeles Moliné. 2023, Fast emulation of cosmological density fields based on dimensionality reduction and supervised machine-learning. https://arxiv.org/abs/2304.06099

Delaunoy, A., Wehenkel, A., Hinderer, T., et al. 2020, Lightning-Fast Gravitational Wave Parameter Inference through Neural Amortization. https://arxiv.org/abs/2010.12931

DeRose, J., Chen, S.-F., White, M., & Kokron, N. 2021, Neural Network Acceleration of Large-scale Structure Theory Calculations, doi: https://doi.org/10.1088/1475-7516/2022/04/056

Dialektopoulos, K., Said, J. L., Mifsud, J., Sultana, J., & Adami, K. Z. 2022, Neural Network Reconstruction of Late-Time Cosmology and Null Tests, doi: https://doi.org/10.1088/1475-7516/2022/02/023

Facchinetti, G. 2025, Neural network emulation of reionization to constrain new physics with early- and late-time probes. https://arxiv.org/abs/2503.11261

Fremstad, D., & Winther, H. A. 2025, Emulating the Non-Linear Matter Power-Spectrum in Mixed Axion Dark Matter Models. https://arxiv.org/abs/2503.07277

Greiner, M., & Enßlin, T. A. 2014, Log-transforming the matter power spectrum, doi: https://doi.org/10.1051/0004-6361/201323181

Guy, L. P., Bechtol, K., Carlin, J. L., et al. 2022, Faro: A framework for measuring the scientific performance of petascale Rubin Observatory data products. https://arxiv.org/abs/2206.15447

Gómez-Vargas, I., Andrade, J. B., & Vázquez, J. A. 2023, Neural Networks Optimized by Genetic Algorithms in Cosmology, doi: https://doi.org/10.1103/PhysRevD.107.043509

Gómez-Vargas, I., & Vázquez, J. A. 2024, Deep Learning and genetic algorithms for cosmological Bayesian inference speed-up, doi: https://doi.org/10.1103/PhysRevD.110.083518

Günther, S., Balkenhol, L., Fidler, C., et al. 2025, OLÉ – Online Learning Emulation in Cosmology. https://arxiv.org/abs/2503.13183

Huber, S., & Suyu, S. H. 2024, HOLISMOKES – XII. Time-delay Measurements of Strongly Lensed Type Ia Supernovae using a Long Short-Term Memory Network, doi: https://doi.org/10.1051/0004-6361/202449952

Huppenkothen, D., Ntampaka, M., Ho, M., et al. 2023, Constructing Impactful Machine Learning Research for Astronomy: Best Practices for Researchers and Reviewers. https://arxiv.org/abs/2310.12528

Ichinohe, Y., Yamada, S., Miyazaki, N., & Saito, S. 2018, Neural network-based preprocessing to estimate the parameters of the X-ray emission of a single-temperature thermal plasma, doi: https://doi.org/10.1093/mnras/sty161

Jamieson, D., Li, Y., Villaescusa-Navarro, F., Ho, S., & Spergel, D. N. 2024, Field-level Emulation of Cosmic Structure Formation with Cosmology and Redshift Dependence. https://arxiv.org/abs/2408.07699

Jense, H. T., Harrison, I., Calabrese, E., et al. 2024, A complete framework for cosmological emulation and inference with CosmoPower. https://arxiv.org/abs/2405.07903

Jones, J. D., Bahauddin, S. M., Rapetti, D., Mirocha, J., & Burns, J. O. 2024, `21cmLSTM`: A Fast Memory-based Emulator of the Global 21 cm Signal with Unprecedented Accuracy. https://arxiv.org/abs/2410.07619

E. End-to-End Research

Fig. 9 shows the results page of the `denario` GUI. The example paper, fully automatically generated, is appended.

Kelly, B. C. 2011, Measurement Error Models in Astronomy. https://arxiv.org/abs/1112.1745

Li, H., Li, B.-W., Shu, P., & Li, Y.-Q. 2024, Machine Learning-Based Identification of Contaminated Images in Light Curves Data Preprocessing, doi: https://doi.org/10.1088/1674-4527/ad339e

Malz, A. I., Dai, M., Ponder, K. A., et al. 2023, Are classification metrics good proxies for SN Ia cosmological constraining power?, doi: https://doi.org/10.1051/0004-6361/202346891

Marulanda, J. P., Santa, C., & Romano, A. E. 2020, Deep learning merger masses estimation from gravitational waves signals in the frequency domain, doi: https://doi.org/10.1016/j.physletb.2020.135790

More, A., Canameras, R., Jaelani, A. T., et al. 2024, Systematic comparison of neural networks used in discovering strong gravitational lenses. https://arxiv.org/abs/2405.12975

Oddo, A., Rizzo, F., Sefusatti, E., Porciani, C., & Monaco, P. 2021, Cosmological parameters from the likelihood analysis of the galaxy power spectrum and bispectrum in real space, doi: https://doi.org/10.1088/1475-7516/2021/11/038

Polanska, A., Price, M. A., Piras, D., Mancini, A. S., & McEwen, J. D. 2024, Learned harmonic mean estimation of the Bayesian evidence with normalizing flows. https://arxiv.org/abs/2405.05969

Ramanah, D. K., Charnock, T., Villaescusa-Navarro, F., & Wandelt, B. D. 2020, Super-resolution emulator of cosmological simulations using deep physical models, doi: https://doi.org/10.1093/mnras/staa1428

Saxena, A., Salvato, M., Roster, W., et al. 2024, CircleZ: Reliable Photometric redshifts for AGN computed using only photometry from Legacy Survey Imaging for DESI. https://arxiv.org/abs/2407.10788

Silva, E., Zúñiga-Bolaño, U., Nunes, R. C., & Valentino, E. D. 2024, Non-Linear Matter Power Spectrum Modeling in Interacting Dark Energy Cosmologies, doi: https://doi.org/10.1140/epjc/s10052-024-13487-x

Sáez-Casares, I., Rasera, Y., & Li, B. 2023, The e-MANTIS emulator: fast predictions of the non-linear matter power spectrum in $f(R)$CDM cosmology, doi: https://doi.org/10.1093/mnras/stad3343

Tegmark, M., Eisenstein, D., Strauss, M., et al. 2006, Cosmological Constraints from the SDSS Luminous Red Galaxies, doi: https://doi.org/10.1103/PhysRevD.74.123507

Trusov, S., Zarrouk, P., & Cole, S. 2025, Neural Network-based model of galaxy power spectrum: Fast full-shape galaxy power spectrum analysis. https://arxiv.org/abs/2403.20093

Viel, M., Markovic, K., Baldi, M., & Weller, J. 2011, The Non-Linear Matter Power Spectrum in Warm Dark Matter Cosmologies, doi: https://doi.org/10.1111/j.1365-2966.2011.19910.x

Wang, B., Leja, J., Villar, V. A., & Speagle, J. S. 2023, SBI++: Flexible, Ultra-fast Likelihood-free Inference Customized for Astronomical Applications, doi: https://doi.org/10.3847/2041-8213/ace361

Winther, H., Casas, S., Baldi, M., et al. 2019, Emulators for the non-linear matter power spectrum beyond ΛCDM, doi: https://doi.org/10.1103/PhysRevD.100.123540

Yang, Y., Bird, S., & Ho, M.-F. 2025, Ten-parameter simulation suite for cosmological emulation beyond ΛCDM. https://arxiv.org/abs/2501.06296

Álvarez, S. I., Alonso, E. D., Sánchez, M. L., et al. 2023, One-dimensional Convolutional Neural Networks for Detecting Transiting Exoplanets, doi: https://doi.org/10.3390/axioms12040348

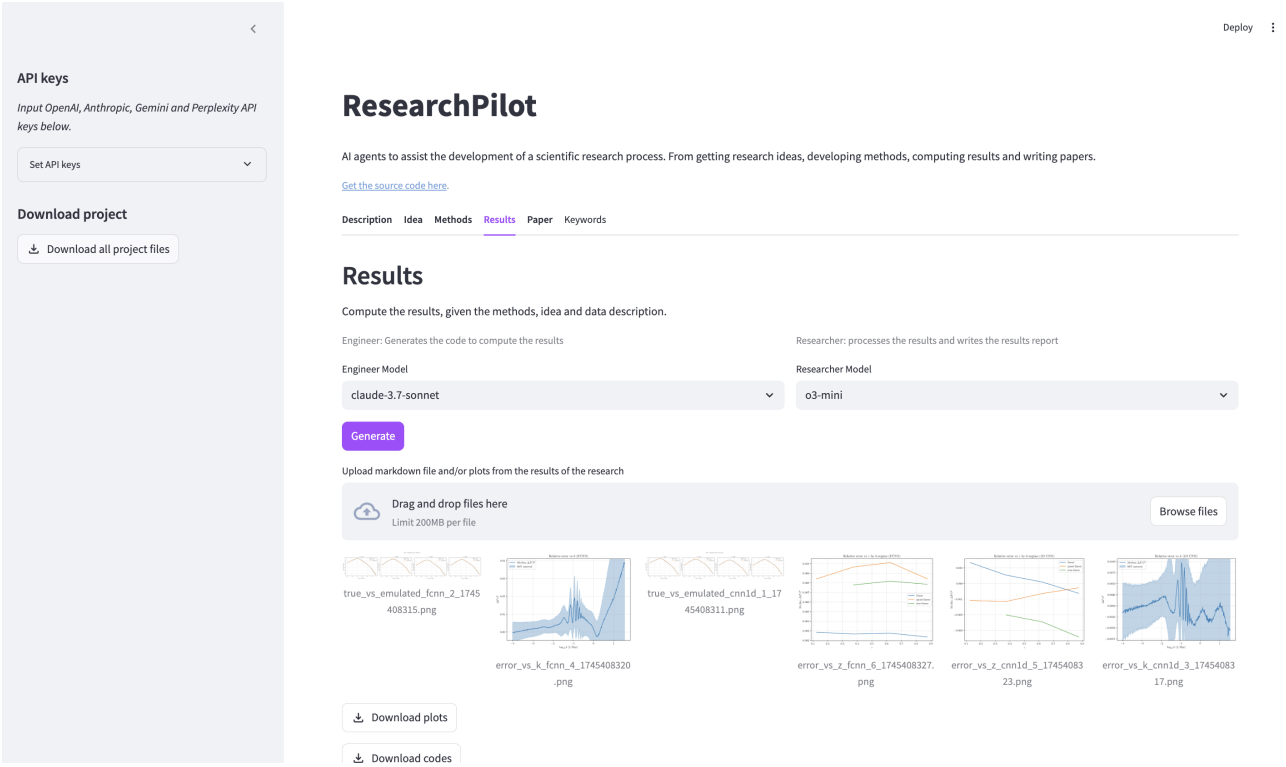

Figure 9: The (preliminary) results page of the `denario` GUI.