Competing with AI Scientists: Agent-Driven Approach to Cosmological Parameter Inference

Team Cmbagent

Erwan Allys, Boris Bolliet, Tom Borrett, Celia Lecat, Andy Nilipour, Sebastien Pierre, **Licong Xu**

About our team

Who are we?

Célia Lecat
MSc student

Andy Nilipour
MPhil student

Tom Borrett
1st year PhD student

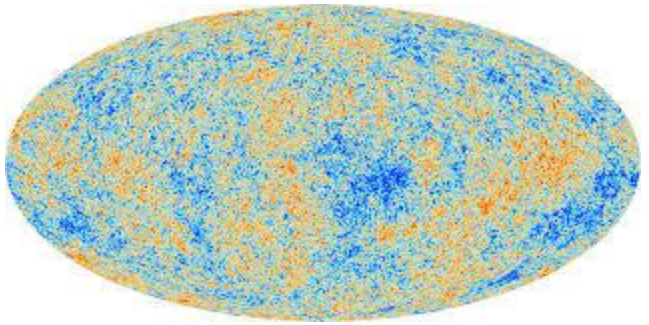
Sébastien Pierre
2nd year PhD student

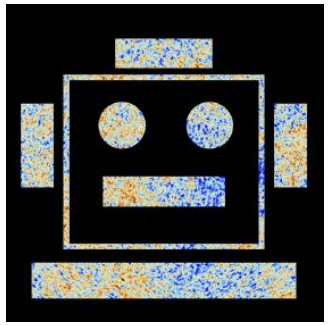
Licong Xu
2nd year PhD student

Advisors: Boris Bolliet (Cambridge), Erwan Allys (ENS)

What are we working on?

- Cosmic Microwave Background (CMB)
- Wavelet Scattering Transform (WST)
- Agentic AI for scientific discovery

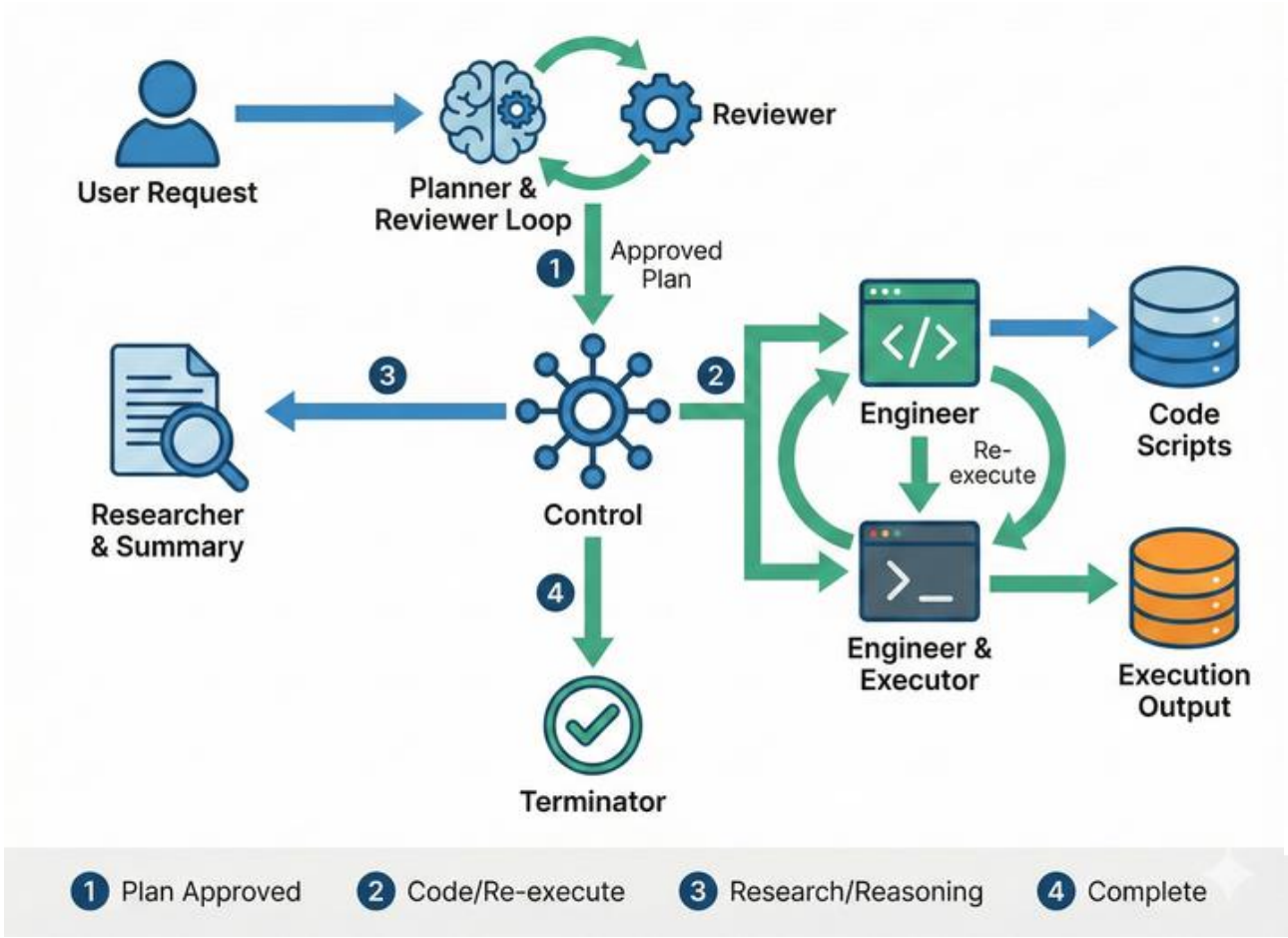




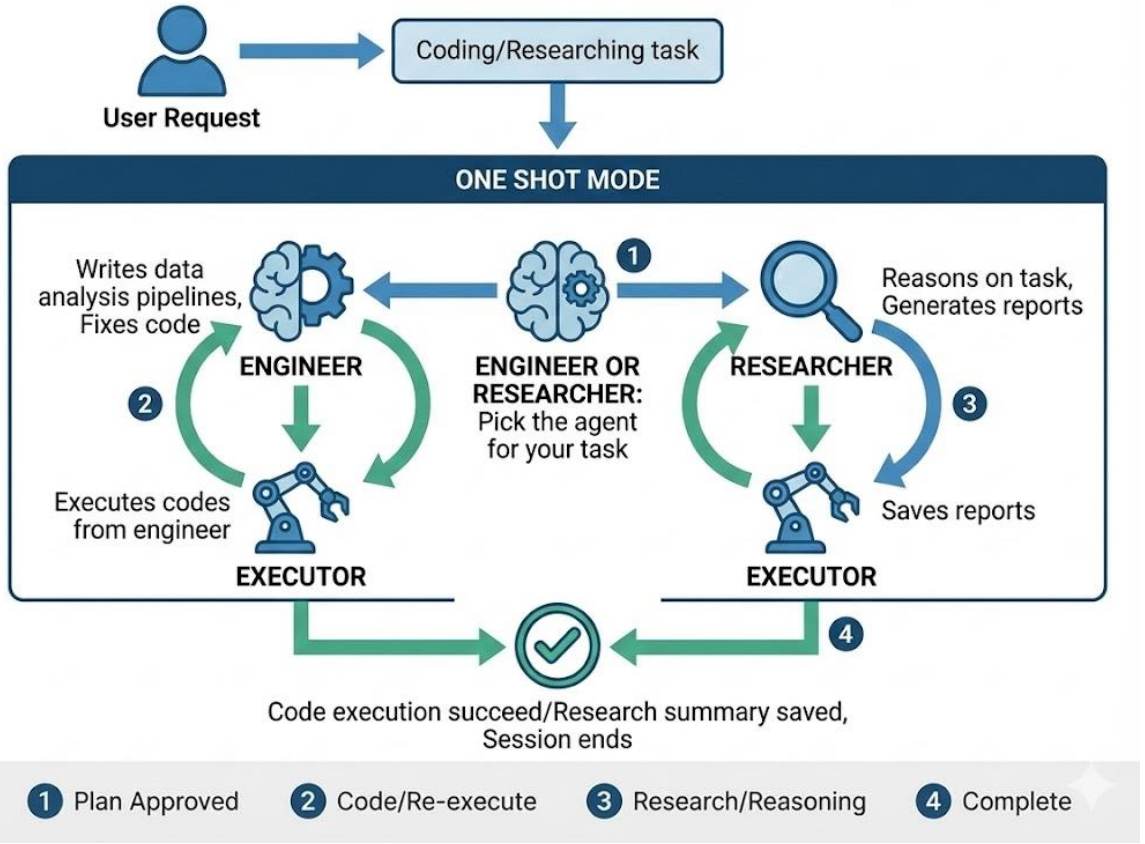
CMBAgent: Agent Orchestration

SCAN ME

Get the open-source code here



Planning & Control
Deep Research



One-Shot

(Xu et al., 2025, Laverick et al., 2024)

First iteration: from starting kit

Find and train a neural network that maximises the score. Best model will achieve above 11.

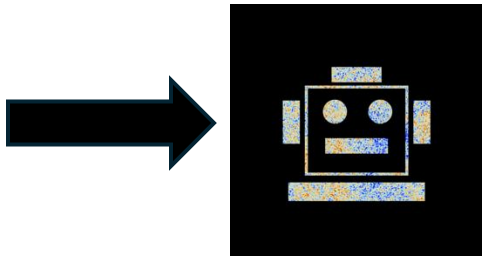
Previous run insights:
Baseline Simple_CNN Score: ~8.2-8.5
Single ResNet18 Score: 8.91
...

Key findings might be important to improve the score:
Data Augmentation: …
Ensembling: …

However, fundamental shift in modelling approach is required to improve further, e.g.:
Architecture modification: …
Loss Function: …

Here is the example code to load the training images:
(example codes)

Hardware constraints:
We are running on an NVIDIA RTX PRO 6000 Blackwell Workstation Edition with 96GB RAM



CMBAgent
Planning & Control

Score: 10.5

Method:
2 × ResNet34
2 × ResNet50
MCMC

Factors affecting parameter inference

- Architecture
  - ResNet/Deep neural network does not work well and expensive to train
- Data preprocessing
  - Non-linear transformation of the raw maps before training
  - Smooth with Gaussian kernels to avoid learning small scale noise & baryons
- Training strategy and Data augmentation
  - Noise resampling
  - Rotation and Flips both in training
- Inference method
  - The default MCMC method might not be the best

Finding the best combination

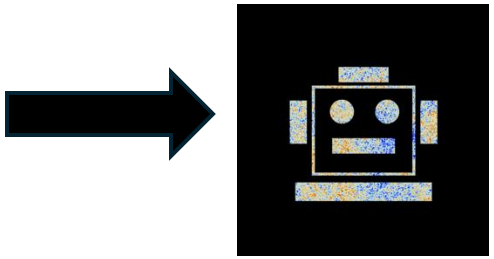
Find and train a neural network that maximises the score. Best model will achieve above 11.5

Previous run insights … +

Deeper is not better. Just keep ResNet18 or models below.

Please pay more attention to **data processing**, like non-linear transformations or smoothing with Gaussian kernel.

Please also consider **data augmentation, and training strategies**, or tweaking the **model architecture**. Since we found architecture is the most important factor from previous experiments



CMBAgent One-shot

Architecture:
Light architecture
Vanilla CNN works well

Preprocessing:
Identity (no preprocessing)

Augmentation:
Flips/Rotations

Training strategy:
LR scheduler
Noise resampling
…

Round 2: Human-in-the-loop

Architecture:
Vanilla CNN + AdaptiveAvgPool

Preprocessing:
Identity (no preprocessing)

Augmentation:
No

Training strategy:
LR scheduler
Noise resampling

→

Score: 11.02

Method:
5 × Vanilla CNN
MCMC inference

Changing the inference strategy:

Previous script +

The current MCMC inference pipeline might not be optimal, please suggest alternative method

→ GPT-5 →

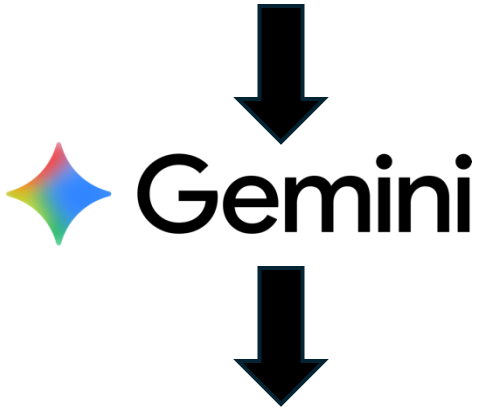
Score: 11.42

Method:
5 × Vanilla CNN
”Grid-based” inference

Round 3: Human-in-the-loop

Previous run insight …
After exploring data preprocessing, we found:

1. **Blurring the map with Gaussian kernel is worse**
2. **Identity preprocessing mode works best**
3. **Residual connections are not helpful**
4. **We need to change the architecture to improve more**
5. **Data augmentation is important**



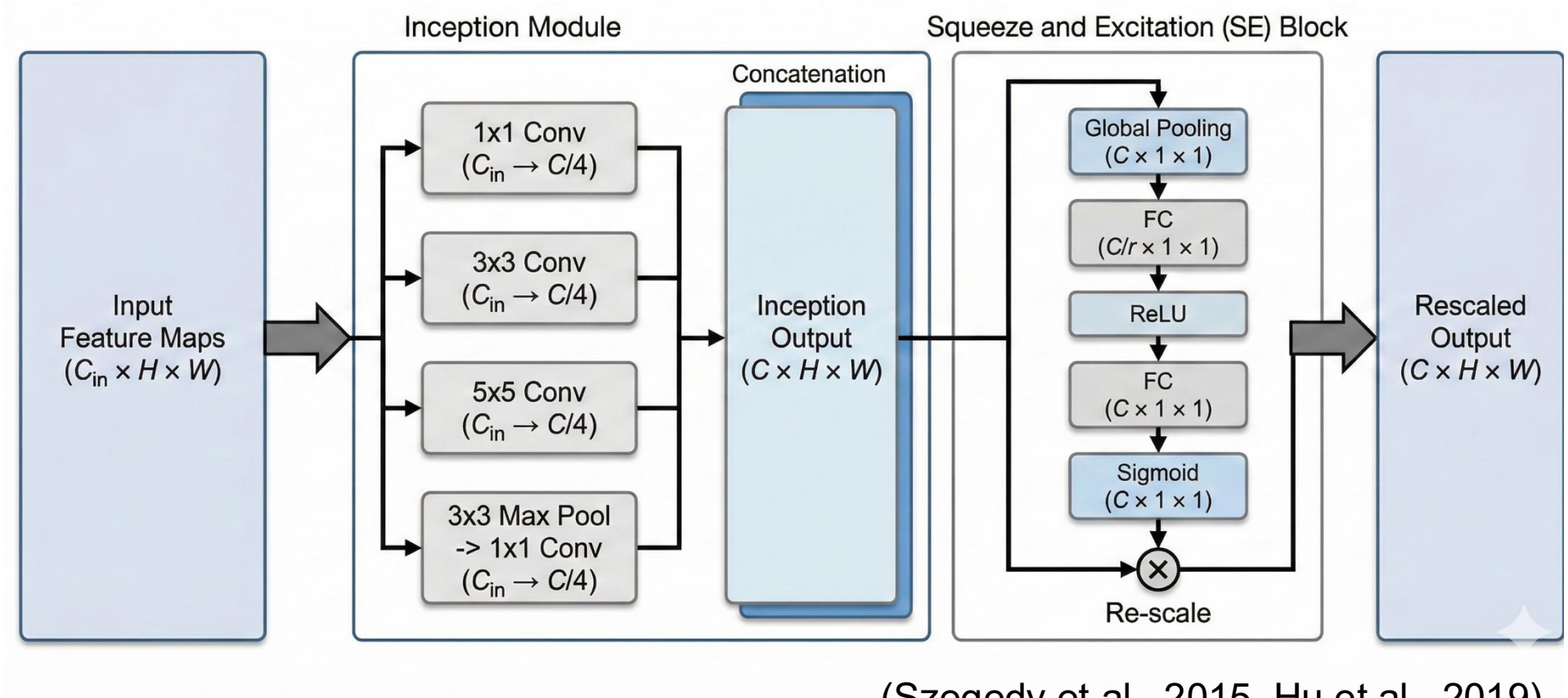
Score: 11.72

Architecture:
5 × InceptionNet
5 × InceptionSENet

Augmentation:
All symmetries of D_4 group (flips, rotations, transpose)
Noise resampling

Inference:
Grid-based inference method
Test-time augmentation using D_4 group

Inception(SE) architecture



(Szegedy et al., 2015, Hu et al., 2019)

Training strategy:

- Pretrain 15 epochs on noiseless maps + training 50 epochs on noisy maps
- Step Learning Rate (StepLR) decay
- MSE loss function

Inference

For each ensemble, use **different train-validation split** indices

$$\boldsymbol{\mu}_g = \frac{1}{N_g} \sum_{i \in G_g} \hat{\boldsymbol{\theta}}_i,$$

$$\mathbf{C}_g = \frac{1}{N_g - 1} \sum_{i \in G_g} \left(\hat{\boldsymbol{\theta}}_i - \boldsymbol{\mu}_g \right) \left(\hat{\boldsymbol{\theta}}_i - \boldsymbol{\mu}_g \right)^\top$$

Smooth with Gaussian kernel

$$\bar{\boldsymbol{\mu}}_g = \sum_{g'} W_{gg'} \, \boldsymbol{\mu}_{g'}$$

$$\bar{\boldsymbol{\Sigma}}_g = \sum_{g'} W_{gg'} \left[\boldsymbol{\Sigma}_{g'} + (\boldsymbol{\mu}_{g'} - \bar{\boldsymbol{\mu}}_g)(\boldsymbol{\mu}_{g'} - \bar{\boldsymbol{\mu}}_g)^\top \right],$$

Shrinkage

$$\widetilde{\boldsymbol{\Sigma}}_g = \tau^2 \left[(1 - \lambda_{\mathrm{LW}}) \, \bar{\boldsymbol{\Sigma}}_g + \lambda_{\mathrm{LW}} \, \mathrm{diag}(\bar{\boldsymbol{\Sigma}}_g) \right]$$

Concat. all ensemble predictions

$$(\bar{\boldsymbol{\mu}}_g, \, \widetilde{\boldsymbol{\Sigma}}_g)$$

Weighted ensemble prediction

$$\hat{\boldsymbol{\theta}}_{\mathrm{ens}} = \sum_m w_m^{(\mathrm{ens})} \, \hat{\boldsymbol{\theta}}^{(m)}$$

where weights given by NLL

Test-time augmentation: take the average predictions across all 8 symmetries of D_4 transformations

Wavelet scattering transform

$$S_1^{\lambda_1} = \langle |I \star \psi^{\lambda_1}| \rangle,$$

$$S_2^{\lambda_1} = \langle |I \star \psi^{\lambda_1}|^2 \rangle,$$

$$S_3^{\lambda_1,\lambda_2} = Cov\left[I \star \psi^{\lambda_1}, |I \star \psi^{\lambda_2}| \star \psi^{\lambda_1}\right],$$

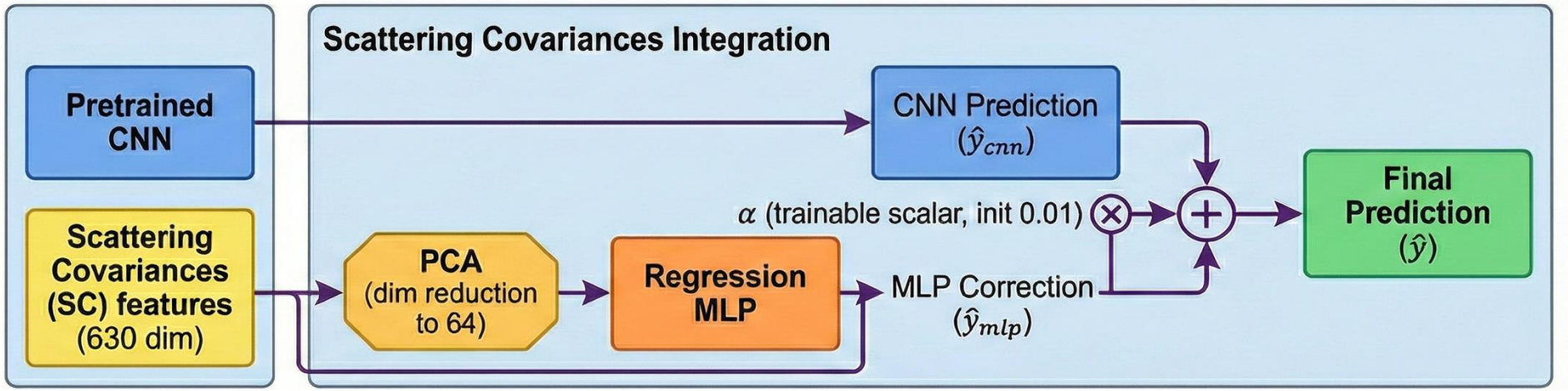
$$S_4^{\lambda_1,\lambda_2,\lambda_3} = Cov\left[|I \star \psi^{\lambda_3}| \star \psi^{\lambda_1}, |I \star \psi^{\lambda_2}| \star \psi^{\lambda_1}\right]$$

$$\mathcal{L} = \mathcal{L}_{\text{mse}} + \lambda_{\text{reg}}\mathcal{L}_{\text{reg}} + \lambda_{\text{cls}}\mathcal{L}_{\text{cls}}$$

MSE loss

L2 penalty between CNN and WST predictions

Auxillary classifier compared to CNN



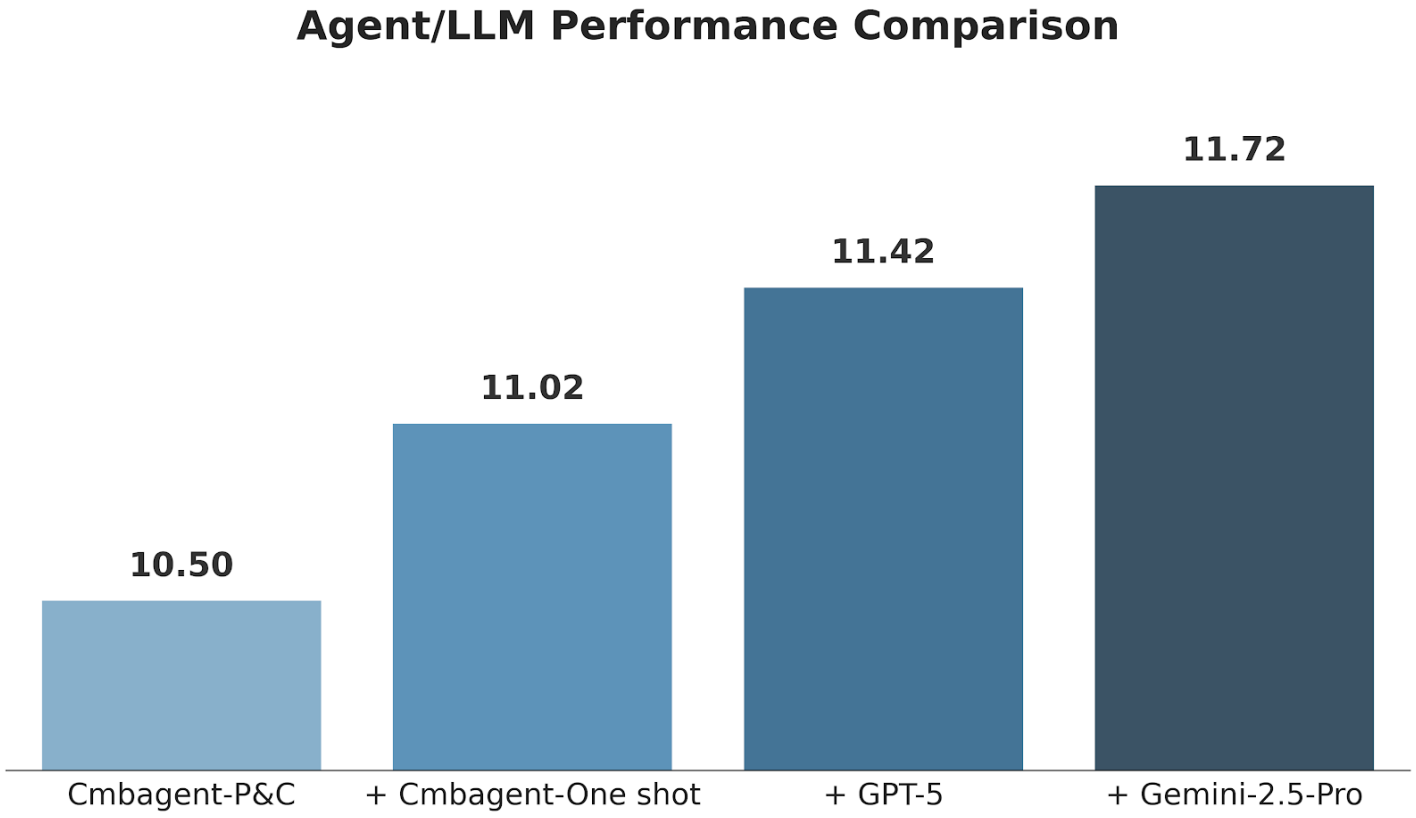
What we learned?

- What Physics we learned?
  - **Vanilla CNN** is already competitive
  - **Data augmentation** matters more than architecture (both train and test-time)
    - Missing a small subset of data augmentation reduces performance significantly
  - **Pooling function**: MaxPool significantly outperforms AvgPool
    - Since MaxPool can extract peak information which is crucial, AvgPool will smooth them out
  - Combining CNN with Wavelet Scattering Coefficient is to be studied

- Can LLM agents solve cosmology problems end-to-end?
  - **Yes**, with an expert in the field
  - **Prompting strategy** is important:
    - loosely specified prompts let LLM explore a wide range of architectures, most of them incompatible
    - Empirically informed prompts are useful to guide LLMs in a correct direction
  - Leveraging a **diverse** set of LLM agents, LLM models and prompts is important

Summary

- LLM agents perform well on this competition, potential for autonomous scientific discovery
- Agents can both **accelerate** and **innovate** new ideas for new scientific discoveries
- Humans can interpret the physical results from LLM agents



Qwen3-Max

11.38
(Wavelet attention)

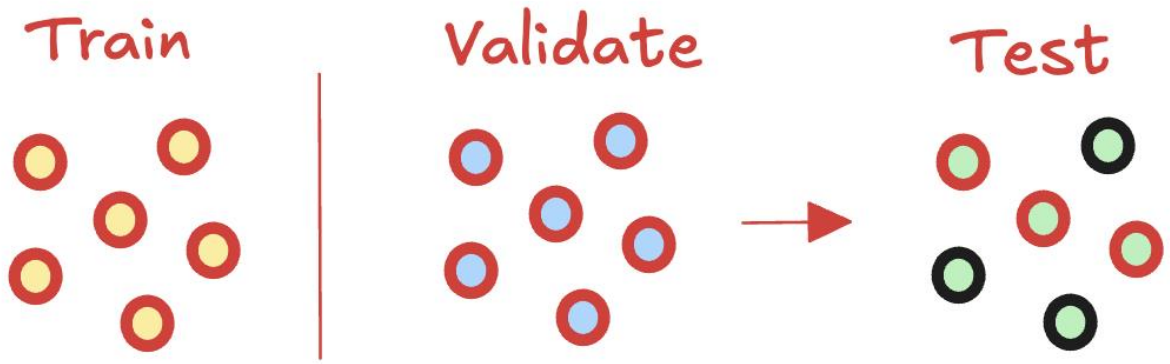
KIMI

- ChannelSpatialAttention
- EncoderDecoderRegression
- …

SCAN ME

Get the open-source code here

What didn't go as planned?
final evaluation on (i) and (ii) was:



Our models did poorly on A.

type-B cosmologies (seen during training)
type-A cosmologies (not seen during training)

Cosmology_Challenge / Phase_1_Startingkit_WL_CNN_MCMC.ipynb

⚠ NOTE:

If you want to split your own training/validation sets to evaluate your model, we recommend splitting the original training data along `axis = 1` (the 256 realizations of nuisance parameters). This will ensure that there are no intrinsic correlations between the training and validation sets.

In [45]:

```
# Split the data into training and validation sets

NP_idx = np.arange(Nsys)  # The indices of Nsys nuisance parameter realizations
split_fraction = 0.2      # Set the fraction of data you want to split (between 0 and 1)
seed = 5566               # Define your random seed for reproducible results

train_NP_idx, val_NP_idx = train_test_split(NP_idx, test_size=split_fraction,
                                            random_state=seed)

noisy_kappa_train = noisy_kappa[:, train_NP_idx]      # shape = (Ncosmo, len(train_NP_idx), 1424, 176)
label_train = data_obj.label[:, train_NP_idx]         # shape = (Ncosmo, len(train_NP_idx), 5)
noisy_kappa_val = noisy_kappa[:, val_NP_idx]          # shape = (Ncosmo, len(val_NP_idx), 1424, 176)
label_val = data_obj.label[:, val_NP_idx]             # shape = (Ncosmo, len(val_NP_idx), 5)

Ntrain = label_train.shape[0]*label_train.shape[1]
Nval = label_val.shape[0]*label_val.shape[1]
```

Splitting along realisations prevent from holding out whole cosmologies.

Recommendation goes against generalisation to unseen cosmologies.

So did public leaderboard (no unseen cosmologies).

We naively followed that recommendation and signals from leaderboard.