東京大学
素粒子物理国際研究センター
International Center for Elementary Particle Physics
The University of Tokyo

Development of an LLM-Based System for Automatic Code Generation from HEP Publications

March 20, 2026

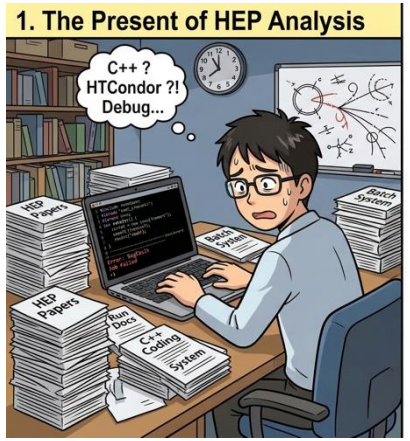
ISGC 2026

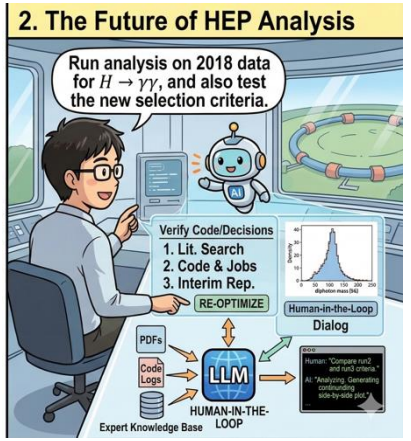
ICEPP/UTokyo[a], KEK[b]

Masahiko Saito[a], Tomoe Kishimoto[b], Junichi Tanaka[a]

Challenges in HEP Analysis and Future Vision

- HEP analyses are becoming **increasingly complex**
  - EB-scale data and large software stacks
  - Physicists need substantial computing expertise
- LLMs may **lower the barrier**
  - Natural-language interaction for analysis tasks
  - Let physicists focus more on physics
- Our vision
  - Integrate coding, batch processing, and documentation access
  - Keep humans in the loop with interpretable intermediate outputs





Generated by Gemini

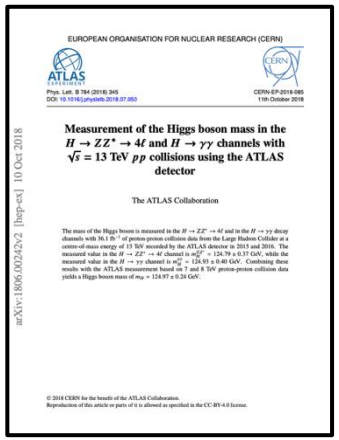
<u>Goal</u>: Use LLMs to make HEP analysis more accessible while keeping it verifiable.

Project Overview: PoC for Reproducibility

Project goal
(1st milestone)

Build a proof-of-concept system that extracts analysis procedures from HEP papers and reproduces published results.

Paper (PDF)

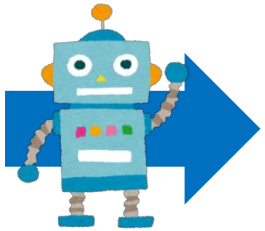


EUROPEAN ORGANISATION FOR NUCLEAR RESEARCH (CERN)

Measurement of the Higgs boson mass in the $H \to ZZ^* \to 4\ell$ and $H \to \gamma\gamma$ channels with $\sqrt{s} = 13$ TeV pp collisions using the ATLAS detector

The ATLAS Collaboration

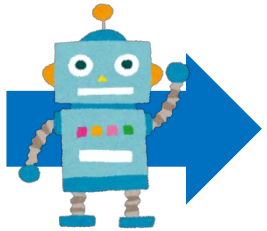
LLM extraction



Extracted cuts

Description
1. Pre-selection
1.1 Good Run List
1.2 Trigger requirements
1.3 (# of PV) > 0
2. Electron
2.1 loose criteria
2.2 eT > 7GeV
2.3 abs(eta) < 2.47
2.4 (ptcone20/eT) < 0.15
2.5 (topoetcone20/eT) < 0.20
2.6 z0 * sin(theta) < 0.5mm
2.7 d0/sig(d0) < 5
3. Muon
3.1 abs(eta) < 2.7

LLM generation

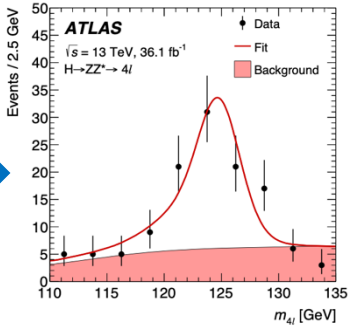


Generated analysis code

```
# Helper function to select good electrons
def select_good_electrons(event):
    """Select good electrons based on quality and kinematic criteria."""
    # Check if electron branches exist
    if 'AnalysisElectronsAuxDyn.pt' not in event.fields:
        return np.array([], dtype=bool)

    # Get electron branches
    electrons_pt = event['AnalysisElectronsAuxDyn.pt'].to_numpy()  # MeV
    electrons_eta = event['AnalysisElectronsAuxDyn.eta'].to_numpy()
    electrons_phi = event['AnalysisElectronsAuxDyn.phi'].to_numpy()
    electrons_charge = event['AnalysisElectronsAuxDyn.charge'].to_numpy()
    electrons_ptcone20 = event['AnalysisElectronsAuxDyn.ptcone20_Nonprompt_All_MaxWeightTTVALooseCone_p
    electrons_topoetcone20 = event['AnalysisElectronsAuxDyn.topoetcone20_CloseByCorr'].to_numpy()  # Me
    electrons_lhloose = event['AnalysisElectronsAuxDyn.DFCommonElectronsLHLooseBL'].to_numpy()

    # Get track particle indices - handle nested structure
    electron_trk_indices = []
    for i in range(len(event['AnalysisElectronsAuxDyn.trackParticleLinks'])):
        if len(event['AnalysisElectronsAuxDyn.trackParticleLinks'][i]) > 0:
            electron_trk_indices.append(event['AnalysisElectronsAuxDyn.trackParticleLinks'][i][0]['m_pe
        else:
            electron_trk_indices.append(-1)
    electron_trk_indices = np.array(electron_trk_indices)

    # Get track particle properties
    gsf_z0 = event['GSFTrackParticlesAuxDyn.z0'].to_numpy()
    gsf_vz = event['GSFTrackParticlesAuxDyn.vz'].to_numpy()
    gsf_d0 = event['GSFTrackParticlesAuxDyn.d0'].to_numpy()
    gsf_theta = event['GSFTrackParticlesAuxDyn.theta'].to_numpy()
    gsf_cov_diag = event['GSFTrackParticlesAuxDyn.definingParametersCovMatrixDiag'].to_numpy()

    # Get primary vertex z0
    pv_z0 = event['PrimaryVerticesAuxDyn.z'].to_numpy()[0] if 'PrimaryVerticesAuxDyn.z' in event.fields

    # Initialize selection mask
    n_electrons = len(electrons_pt)
```

Execution

Reproduced result



- Tests whether published analyses can be reproduced from paper descriptions
- Reveals missing or ambiguous analysis details
- Toward more accessible and verifiable HEP analysis workflows

Benchmark Design and Evaluation Protocol

Target Analysis

- Target: ATLAS $H \rightarrow ZZ^* \rightarrow 4\ell$ analysis (arXiv: 1806.00242)
- Data: ATLAS Open data, pp collisions in 2015-16
- Suitable benchmark because it is:
  - publicly reproducible
  - referenced selection details

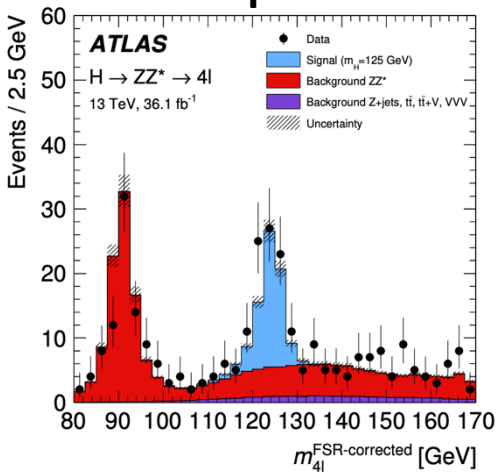
Ground truth

- Manually curated from the main paper and cited references
- Includes:
  - curated selection-cut list
  - hand-written baseline code
- 27 selection cuts used for the extraction benchmark
- Only cuts reproducible with ATLAS Open Data were included for the coding benchmark

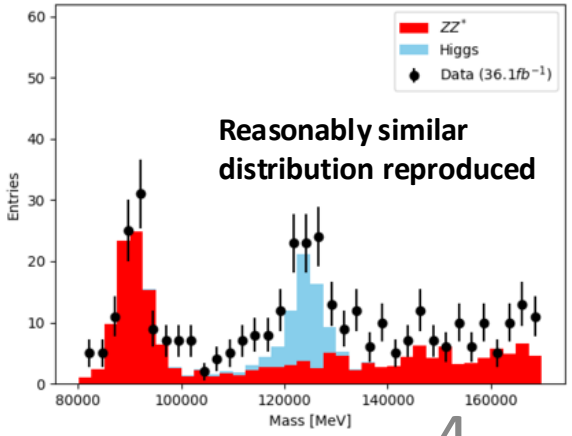
Evaluation

- Step 1: Selection extraction
  - Compare LLM output with curated cut list
  - Metrics: correct cuts / hallucinations
- Step 2: Code generation
  - Compare filtered events of generated code with baseline
  - Labels: exact match / not match / exec failure
- Extraction and code generation were evaluated separately to assess feasibility at each stage.

Paper

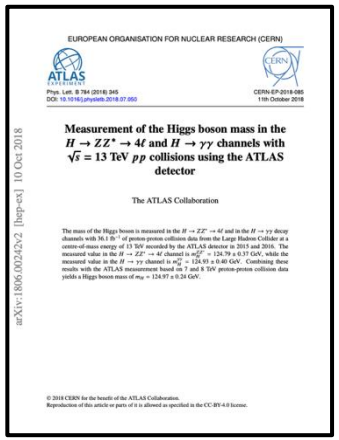


Hand-written code



Step 1: Selection Extraction

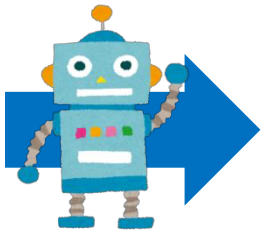
Paper (PDF)


EUROPEAN ORGANISATION FOR NUCLEAR RESEARCH (CERN)

Measurement of the Higgs boson mass in the $H \to ZZ^* \to 4\ell$ and $H \to \gamma\gamma$ channels with $\sqrt{s} = 13$ TeV pp collisions using the ATLAS detector

The ATLAS Collaboration

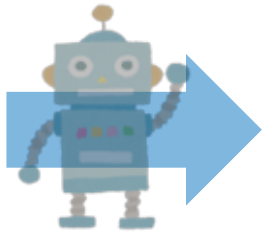
LLM extraction



Extracted cuts

Description
1. Pre-selection
1.1 `Good Run List`
1.2 `Trigger requirements`
1.3 `(# of PV) > 0`
2. Electron
2.1 `loose criteria`
2.2 `eT > 7GeV`
2.3 `abs(eta) < 2.47`
2.4 `(ptcone20/eT) < 0.15`
2.5 `(topoetcone20/eT) < 0.20`
2.6 `z0 * sin(theta) < 0.5mm`
2.7 `d0/sig(d0) < 5`
3. Muon
3.1 `abs(eta) < 2.7`

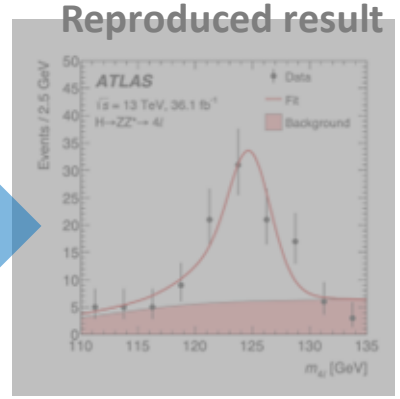
LLM generation



Generated analysis code

Execution

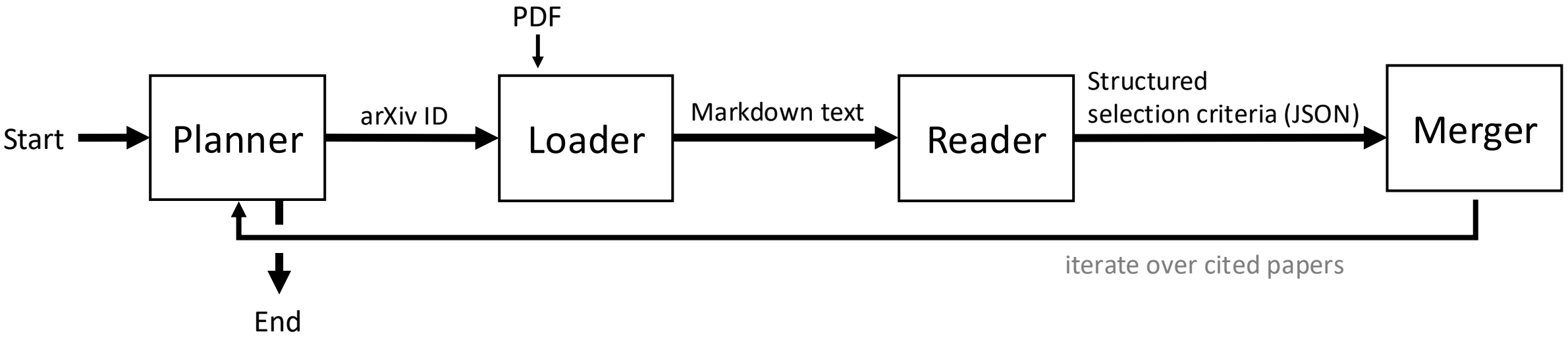
Reproduced result



Step 1 Goal: extract event-selection cuts from the main paper and cite references

Step 1: Selection Extraction - Workflow

1. **Planner**: selects the next paper
2. **Loader**: converts PDF to text and extracts references
3. **Reader**: extracts selection criteria
4. **Merger**: merges new criteria and continues the iteration



Selection criteria are **extracted iteratively** from the paper and its cited references.

Step 1: From Paper Text to Structured Selection Criteria

Paper text (PDF)

7 Mass measurement in the $H \to ZZ^* \to 4\ell$ channel

7.1 Event selection

Events are required to contain at least four isolated leptons ($\ell = e, \mu$) that emerge from a common vertex, form two pairs of oppositely charged same-flavour leptons. Electrons are required to be within the full pseudorapidity range of the inner tracking detector ($|\eta| < 2.47$) and have transverse energy $E_T > 7$ GeV, while muons are required to be within the pseudorapidity range of the muon spectrometer ($|\eta| < 2.7$) and have transverse momentum $p_T > 5$ GeV. The three higher-p_T (E_T) leptons in each quadruplet are required to pass thresholds of 20, 15, and 10 GeV, respectively. A detailed description of the event selection can be found in Ref. [11, 44].

PDF converted into parser-friendly text

Converted Markdown text

```
# 7 Mass measurement in the $H \rightarrow ZZ^* \rightarrow 4\ell$ channel

### 7.1 Event selection

Events are required to contain at least four isolated leptons ( $\ell = e, \mu$ ) that emerge from a common vertex, form two pairs of oppositely charged same-flavour leptons. Electrons are required to be within the full pseudorapidity range of the inner tracking detector ( $|\eta| < 2.47$ ) and have transverse energy $E_T > 7$ GeV, while muons are required to be within the pseudorapidity range of the muon spectrometer ( $|\eta| < 2.7$ ) and have transverse momentum $p_T > 5$ GeV. The three higher- $p_T$ ( $E_T$ ) leptons in each quadruplet are required to pass thresholds of 20, 15, and 10 GeV, respectively. A detailed description of the event selection can be found in Ref. [11, 44].
```

Structured LLM output


```
{
  "selection": {
    "Event": [
      {
        "content": "At least four isolated leptons (ℓ = e, μ) from a common vertex",
        "comments": "Leptons must form two pairs of oppositely charged same-flavour leptons.",
        "references": ["1708.02810", "1712.02304"]
      },
      {
        "content": "Three higher-pT (E_T) leptons in each quadruplet pass thresholds of 20, 15,
        "comments": "Applies to the three highest transverse energy/momentum leptons in the eve
        "references": ["1708.02810", "1712.02304"]
      }
    ],
    "Electron": [
      {
        "content": "|η| < 2.47",
        "comments": "Full pseudorapidity range of the inner tracking detector.",
        "references": ["1708.02810", "1712.02304"]
      },
```


- Structured output makes downstream processing easier
- The output includes cuts, references, and explanatory comments

Step 1: Selection Extraction - Experimental Setup

Models and Runtime

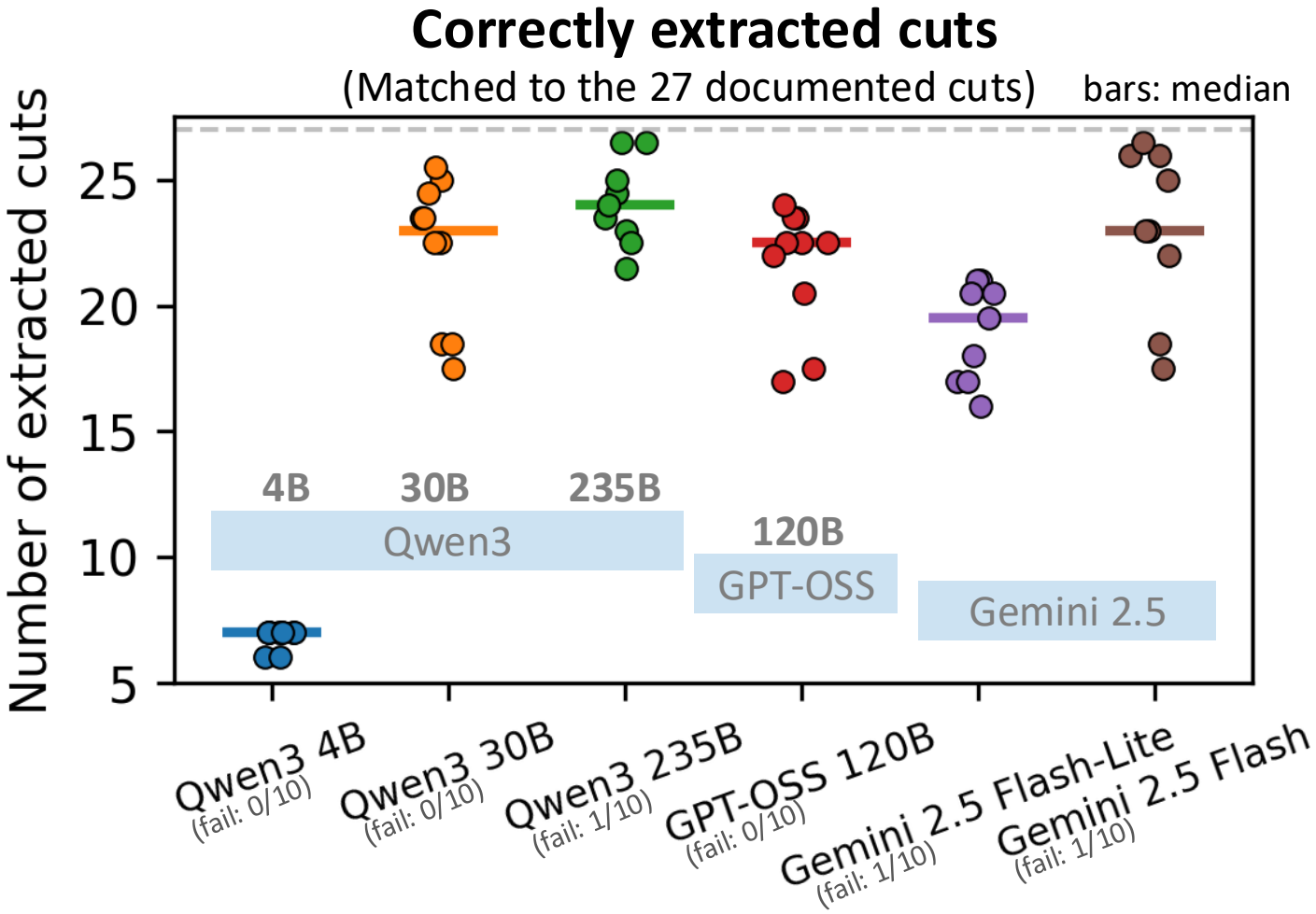
- Framework: LangChain / LangGraph
- Inference backend: vLLM
- Models tested:
  - open-weight LLMs: Qwen3, GPT-OSS
  - commercial model: Gemini 2.5 Flash
- Primary focus: open-weight models for local/private use and lower cost
- Each model was run 10 times with different random seeds

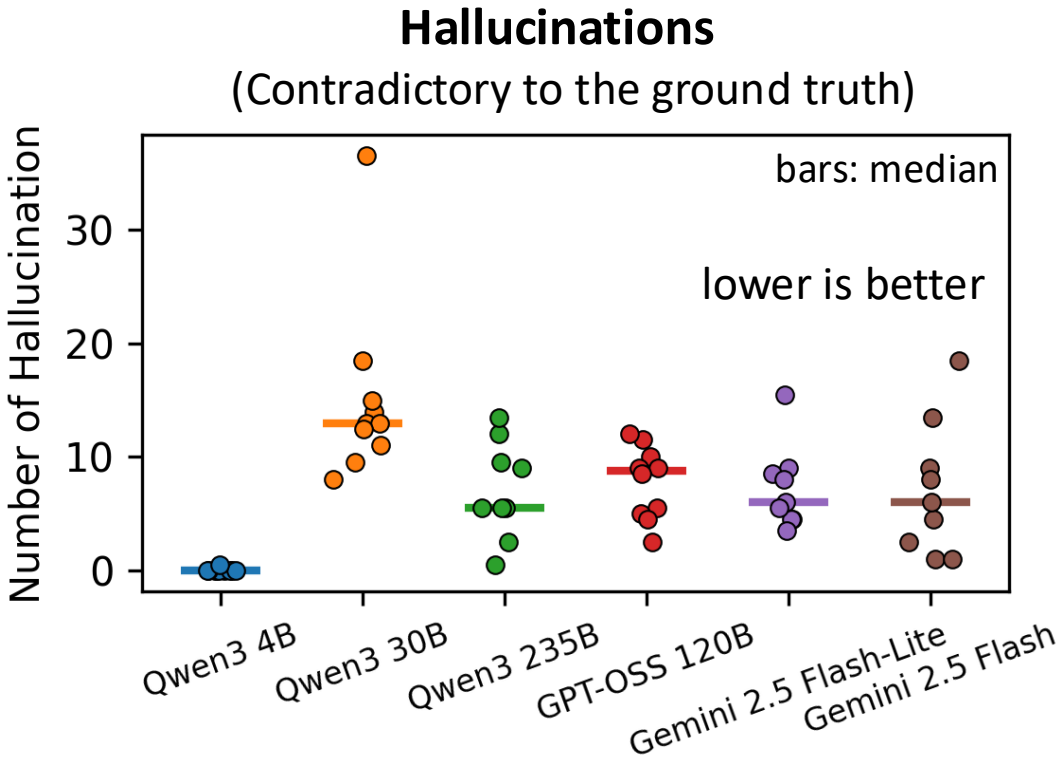
Evaluation protocol

- Generated cut lists were scored against the curated ground-truth list using LLM-based judges
- The final score was defined as the median across repeated judge runs
- Outputs contradictory to the ground truth were counted as hallucinations
- A subset of outputs was manually reviewed to validate the scoring

We benchmarked multiple LLMs under repeated runs and evaluated them with an LLM-based judge.

Step 1: Selection Extraction - Results

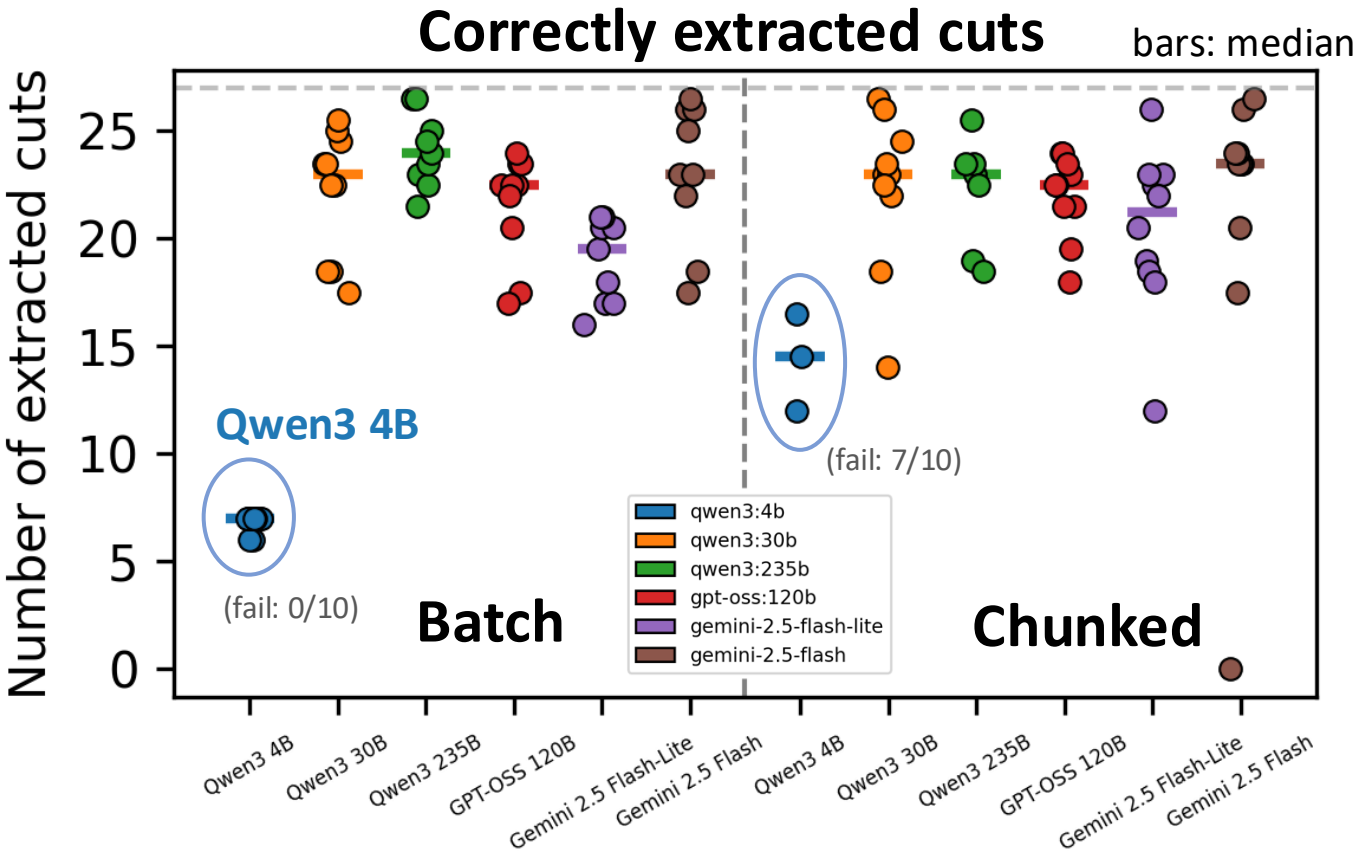


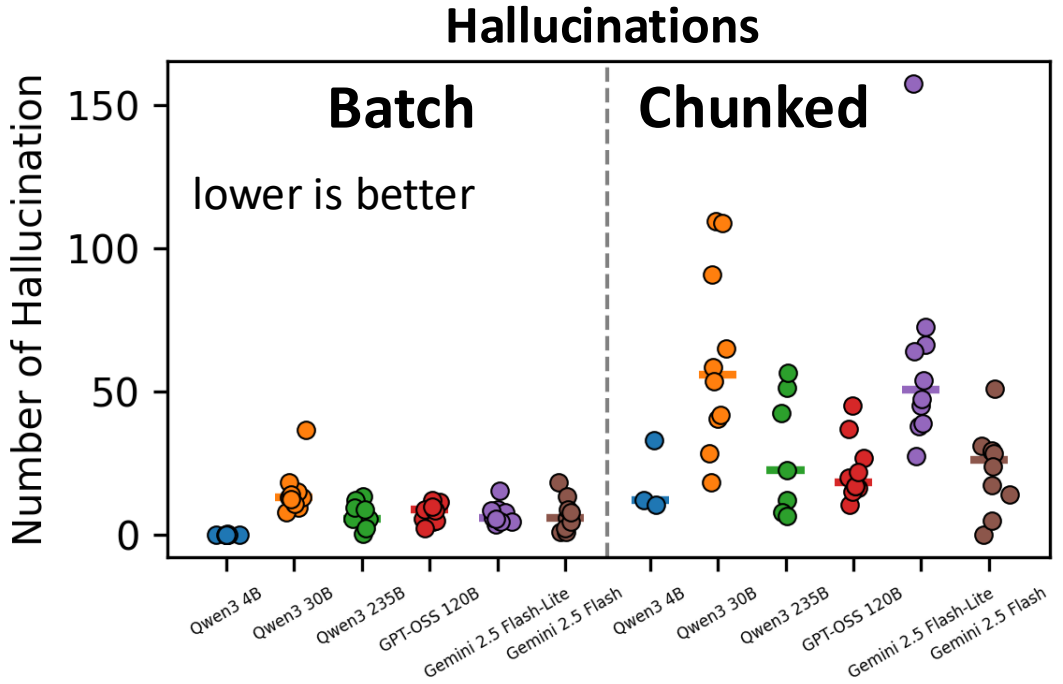


- 30B+ models extract most documented cuts, while 4B remains insufficient.
- Outputs are still stochastic: larger models tend to reduce hallucinations, but not yet fully stable.

Step 1: Selection Extraction - Effect of Chunking

Chunking: instead of processing the full paper at once, the text is split into smaller chunks and processed separately.



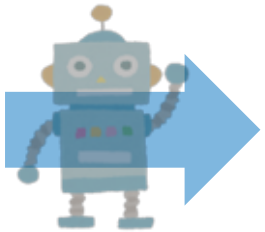


- Chunking improves extraction for small models, especially 4B.
- However, it also increases hallucinations.
- This trade-off remains a key challenge for practical use of small models.

Step 2: Code Generation

Paper (PDF)

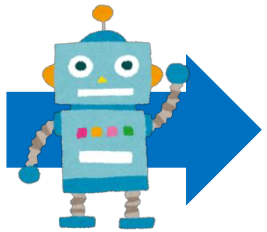
LLM extraction



Extracted cuts

Description
1. Pre-selection
1.1 Good Run List
1.2 Trigger requirements
1.3 (# of PV) > 0
2. Electron
2.1 loose criteria
2.2 eT > 7GeV
2.3 abs(eta) < 2.47
2.4 (ptcone20/eT) < 0.15
2.5 (topoetcone20/eT) < 0.20
2.6 z0 * sin(theta) < 0.5mm
2.7 d0/sig(d0) < 5
3. Muon
3.1 abs(eta) < 2.7

LLM generation

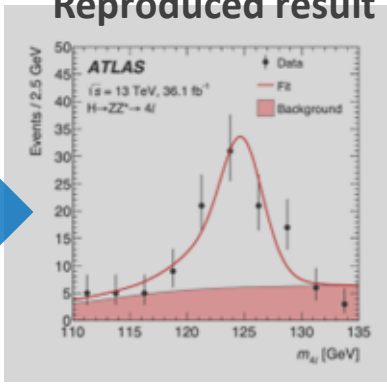


Generated analysis code

```
# Helper function to select good electrons
def select_good_electrons(event):
    """Select good electrons based on quality and kinematic criteria."""
    # Check if electron branches exist
    if 'AnalysisElectronsAuxDyn.pt' not in event.fields:
        return np.array([], dtype=bool)

    # Get electron branches
    electrons_pt = event['AnalysisElectronsAuxDyn.pt'].to_numpy()  # MeV
    electrons_eta = event['AnalysisElectronsAuxDyn.eta'].to_numpy()
    electrons_phi = event['AnalysisElectronsAuxDyn.phi'].to_numpy()
    electrons_charge = event['AnalysisElectronsAuxDyn.charge'].to_numpy()
    electrons_ptcone20 = event['AnalysisElectronsAuxDyn.ptcone20_Nonprompt_All_MaxWeightTTVALooseCone_p
    electrons_topoetcone20 = event['AnalysisElectronsAuxDyn.topoetcone20_CloseByCorr'].to_numpy()  # Me
    electrons_lhloose = event['AnalysisElectronsAuxDyn.DFCommonElectronsLHLooseBL'].to_numpy()

    # Get track particle indices - handle nested structure
    electron_trk_indices = []
    for i in range(len(event['AnalysisElectronsAuxDyn.trackParticleLinks'])):
        if len(event['AnalysisElectronsAuxDyn.trackParticleLinks'][i]) > 0:
            electron_trk_indices.append(event['AnalysisElectronsAuxDyn.trackParticleLinks'][i][0]['m_pe
        else:
            electron_trk_indices.append(-1)
    electron_trk_indices = np.array(electron_trk_indices)

    # Get track particle properties
    gsf_z0 = event['GSFTrackParticlesAuxDyn.z0'].to_numpy()
    gsf_vz = event['GSFTrackParticlesAuxDyn.vz'].to_numpy()
    gsf_d0 = event['GSFTrackParticlesAuxDyn.d0'].to_numpy()
    gsf_theta = event['GSFTrackParticlesAuxDyn.theta'].to_numpy()
    gsf_cov_diag = event['GSFTrackParticlesAuxDyn.definingParametersCovMatrixDiag'].to_numpy()

    # Get primary vertex z0
    pv_z0 = event['PrimaryVerticesAuxDyn.z'].to_numpy()[0] if 'PrimaryVerticesAuxDyn.z' in event.fields

    # Initialize selection mask
    n_electrons = len(electrons_pt)
```

Execution

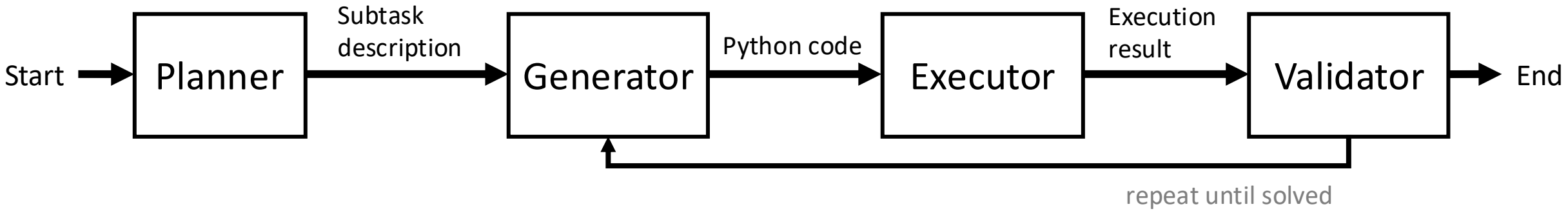
Reproduced result



Step 2 Goal: generate executable analysis code from the extracted cut list

Step 2: Code Generation - Workflow

1. **Planner**: decomposes the task into subtasks
2. **Generator**: generates code for each subtask
3. **Executor**: runs the generated code in a container
4. **Validator**: checks the execution result and decides whether to continue



Code generation is performed iteratively until each subtask is solved.

Step 2: Code Generation - Preliminary Results

Setup

- 10 runs per model for 3 LLMs
- The prompt included the task description, selection criteria, variable information, and runtime environment
- This setup is intended to evaluate the LLMs' code-generation ability under controlled inputs

Evaluation

- Event-level selection results on 1,000 MC $H \rightarrow ZZ^* \rightarrow 4\ell$ events were compared with the reference implementation

Result categories

- *Exactly Matched*: identical event-level selection result
- *Not Matched*: valid output, but a different selection was implemented
- *Execution Failed*: no valid final code produced

Model	Exactly Matched	Not Matched	Execution Failed
Qwen3-Coder-Next-80B	**3**	2	5
Qwen3-Coder-30B	**0**	5	5
GPT-OSS-120B	**2**	1	7

- Open LLMs occasionally reproduced the reference event-level result exactly.
- However, robustness is still limited, with many runs failing to produce valid final code.

Limitations and Future Directions

Current limitations

- PDF parsing remains challenging: conversion is slow and may fail on tables or complex layouts
- LLM outputs remain stochastic and not yet fully reliable
- Hallucinations remain a persistent risk: verification mechanisms are essential

Future directions

Short-term

- Integrate Step 1 and Step 2 into an end-to-end workflow
- Add RAG support for the code-generation stage
- Evaluate generalization on additional benchmarks

Long-term

- Move toward an analysis system where humans can inspect intermediate states and collaborate with the LLM

Summary

- **Prototype**: We developed an LLM-based system that extracts analysis procedures from HEP papers and generates executable analysis code.
- **Step 1**: Open LLMs extracted structured selection criteria across the main paper and cited references; 30B+ models were especially promising, although stochasticity and hallucinations remain.
- **Step 2**: Some runs produced the reference event-level result exactly, but robustness remains limited.
- **Take-home**: LLMs are a viable direction for HEP analysis assistance, with a long-term path toward human-in-the-loop collaborative agents.

Open LLMs are not yet sufficient for fully autonomous analysis,
but already useful as collaborative tools.