

Leveraging Language Models to Navigate Conference Abstracts

G. Watts (UW/Seattle)



Using AI/ML In Particle Physics



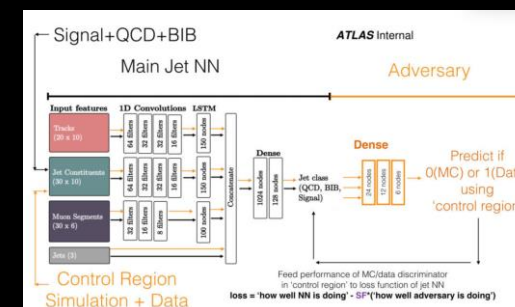
Directly in the Science Process

- Parameter Estimation (mass, calibration, etc.)
- Signal/Background Separation
- Reconstruction (Jet finding, etc.)
- Fast Simulation
- Image processing (astro)
- Anomaly Detection

Indirect Participation

- Coding Assistants
- Drawing graphics for talks
- Retrieval Augmented Generation Help Chatbots (ChATLAS, [AccGPT](#), [Snowmass](#), [HEP-Help](#))
- ...

Custom Built Network Architecture



The New Ingredient: Large Language Models

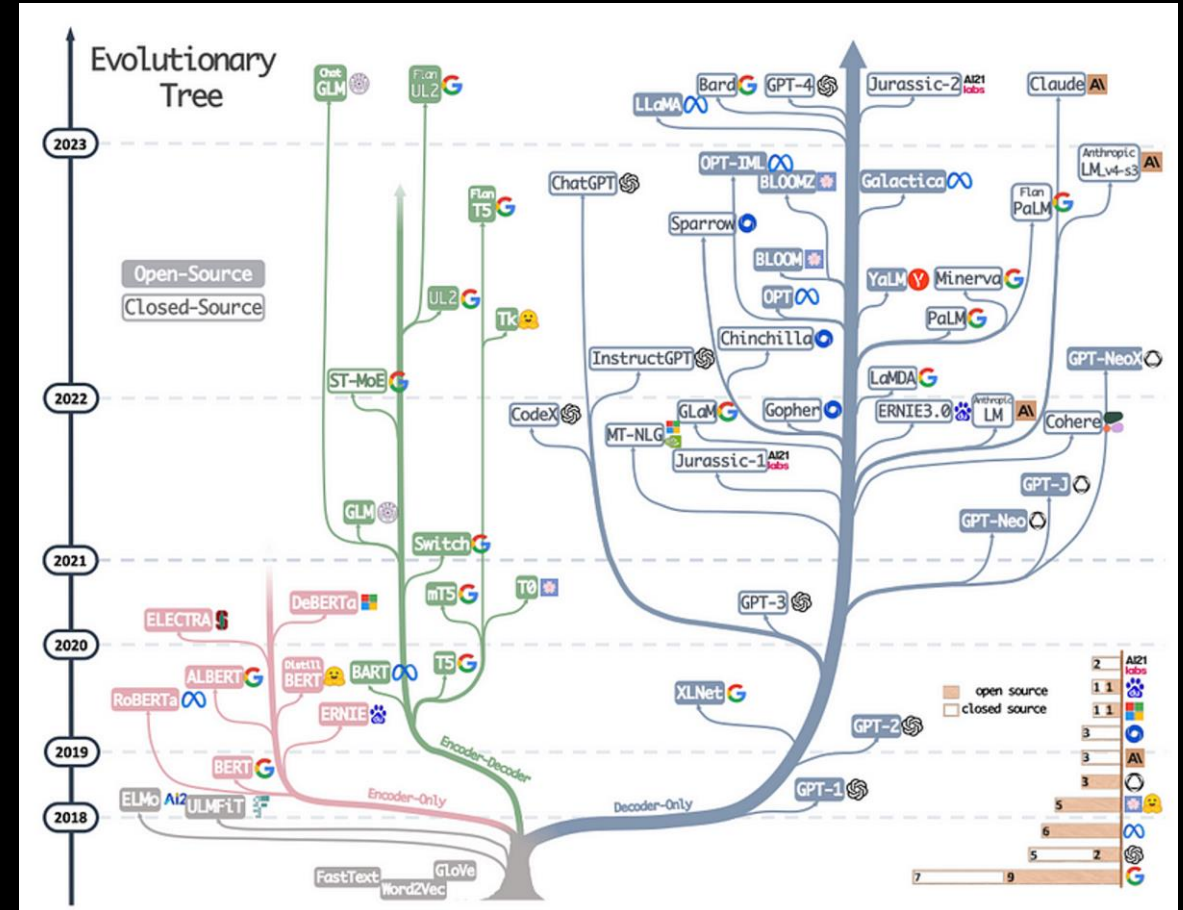
Indirect Participation

- Coding Assistants
- Drawing graphics for talks
- Retrieval Augmented Generation Help Chatbots (ChATLAS, [AccGPT](#), [Snowmass](#), [HEP-Help](#))
- ...

Not totally new, but we are in a new era!

There are LLM's of all types:

- New tech: rapid evolution
- Currently: how small and efficient can they be and still get the job done?
- 3B parameter models (phone, laptop)
- 8B parameter models (GPU)
- O(100)B – the big ones like GPT-4



Characteristics of LLM's

Good at:

- **Generating human-like text:** Crafting articles, emails, and social media posts.
- **Language translation:** Converting text from one language to another with high accuracy.
- **Summarizing information:** Creating concise summaries of long documents.
- **Answering questions:** Providing information and explanations on a wide range of topics.
- **Pattern recognition in data:** Detecting trends and anomalies in large datasets.
- **Natural language understanding:** Interpreting and responding to human queries and commands.
- **Text classification:** Categorizing text into predefined labels for sentiment analysis, topic categorization, etc.

Bad at:

- **Understanding context in depth:** Struggle with nuanced context or deeply technical explanations.
- **Generating original ideas:** Limited in creating truly novel concepts beyond training data.
- **Emotional comprehension:** Can't genuinely understand or process human emotions.
- **Handling ambiguous queries:** May misinterpret or provide inaccurate responses to unclear questions.
- **Ethical decision-making:** Lack moral reasoning and can't weigh ethical considerations.
- **Real-time interaction:** Can't adapt on-the-fly in live, dynamic conversations or events.
- **Memory retention:** Struggle with retaining context over long conversations or recalling previous interactions.

Characteristics of LLM's

Good at:

- **Generating human-like text:** Crafting articles, emails, and social media posts.
- **Language translation:** Converting text from one language to another with high accuracy.
- **Summarizing information:** Creating concise summaries of long documents.
- **Answering questions:** Providing information and explanations on a wide range of topics.
- **Pattern recognition in data:** Detecting trends and anomalies in large datasets.
- **Natural language understanding:** Interpreting and responding to human queries and commands.
- **Text classification:** Categorizing text into predefined labels for sentiment analysis, topic categorization, etc.

Bad at:

- **Understanding context in depth:** Struggle with nuanced context or deeply technical explanations.
- **Generating original ideas:** Limited in creating truly novel concepts beyond training data.
- **Emotional comprehension:** Can't genuinely understand or process human emotions.
- **Handling ambiguous queries:** May misinterpret or provide inaccurate responses to unclear questions.
- **Ethical decision-making:** Lack moral reasoning and can't weigh ethical considerations.
- **Real-time interaction:** Can't adapt on-the-fly in live, dynamic conversations or events.
- **Memory retention:** Struggle with retaining context over long conversations or recalling previous interactions.
- **Hallucinations** – desire to follow instructions even if they can't be

Taking advantage of LLM's

Play to LLM's strengths...

Much of the “bad” can be avoided by tailoring the problems we are trying to address

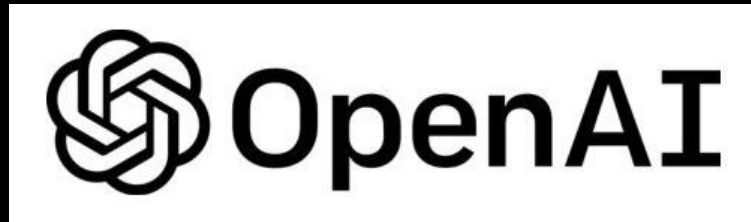
- Summarization and classification – not interpretation
- Tune queries to make unambiguous
- Comparisons rather than decisions
- Avoid (sadly) memory retention: queries must have all context embedded
- Reduce hallucinations by providing specific escape clauses

Bad at:

- **Understanding context in depth:** Struggle with nuanced context or deeply technical explanations.
- **Generating original ideas:** Limited in creating truly novel concepts beyond training data.
- **Emotional comprehension:** Can't genuinely understand or process human emotions.
- **Handling ambiguous queries:** May misinterpret or provide inaccurate responses to unclear questions.
- **Ethical decision-making:** Lack moral reasoning and can't weigh ethical considerations.
- **Real-time interaction:** Can't adapt on-the-fly in live, dynamic conversations or events.
- **Memory retention:** Struggle with retaining context over long conversations or recalling previous interactions.
- **Hallucinations** – desire to follow instructions even if they can't be

Access

Commercial



Plenty of companies out there

- Libraries
- Pay-as-you-go for API access
- Cost depends on model complexity
- Cheap for prototyping projects (\$10's of dollars)

Open-Source



Open Source Serving Platform

- Uses git-like interface
- Tracks many models
- Takes some work to uses any but the most popular models!

OpenAI Calls

Text instructions to model

Many ways to call:

- You program the model in English
- Ask it for json/yaml output
- Tell it what to do, give it the data
- Parse the result
- Calling small models and large ones is only a difference of how much you pay (see later)

For commercial models this is often a single call

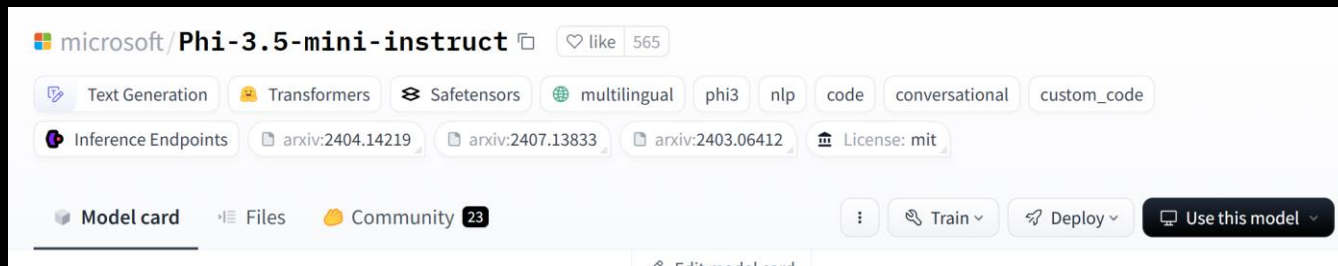
Model Parameters

```
# Generate the completion using OpenAI
openai_client = openai.OpenAI(api_key=get_key())
response = openai_client.chat.completions.create(
    model=model,
    messages=[
        {
            "role": "system",
            "content": "You are a helpful assistant and expert in the field of experimental "
            "particle physics. All responses must be in the JSON format specified.",
        },
        {"role": "user", "content": prompt},
        {
            "role": "user",
            "content": "Topics I'm very interested in\n - "
            + "\n - ".join(context["interested_topics"]),
        },
        {
            "role": "user",
            "content": "Topics I'm not at all interested in\n - "
            + "\n - ".join(context["not_interested_topics"]),
        },
        {
            "role": "user",
            "content": f'Conference Talk Title: "{context["title"]}"',
        },
        {
            "role": "user",
            "content": f'Conference Talk Abstract: "{context["abstract"]}"',
        },
        {
            "role": "user",
            "content": "Your answer should be correct JSON using in the following JSON schema."
            " Everything should be short and succinct with no emoji, and properly escape "
            "latex directives. This is a JSON schema, so "
            "replace the title and type dict with the actual data: \n"
            f"{schema}",
        },
    ],
    max_tokens=1000,
    temperature=0.7,
    n=1,
    stop=None,
)
```

[Code](#)

Open-Source

First, find the [model card](#) of what you need to run



Then...

- Environment is more complex: need TF or pytorch (etc.) that can see your GPU
- You need a beefy GPU
 - 3080 vs A100 is big, even for small models
- Configuration is a little more complex
- Instructions are similar, as is calling.
- Different models use same API, but require different instructions!

```
from transformers import AutoModelForCausalLM, AutoTokenizer, pipeline

model = AutoModelForCausalLM.from_pretrained(
    model_name,
    device_map="cuda",
    torch_dtype="auto",
    trust_remote_code=True,
)

tokenizer = AutoTokenizer.from_pretrained(model_name, trust_remote_code=True)

_hf_models[model_name] = pipeline(
    "text-generation", model=model, tokenizer=tokenizer, trust_remote_code=True
)
```

[Code](#)

Configuration

```
messages = [
    {
        "role": "system",
        "content": "You are my expert AI assistant will help me pick talks and posters I'm interested in.",
    },
    {
        "role": "user",
        "content": query,
    },
    {
        "role": "user",
        "content": "Topics I'm very interested in:\n - "
        + "\n - ".join(context["interested_topics"]),
    },
    {
        "role": "user",
        "content": "Topics I'm not at all interested in:\n - "
        + "\n - ".join(context["not_interested_topics"]),
    },
    {
        "role": "user",
        "content": f'Conference Talk Title: "{context["title"]}"',
    },
    {
        "role": "user",
        "content": f'Conference Talk Abstract: "{context["abstract"]}"',
    },
    {
        "role": "user",
        "content": "Your answer should be correct JSON using in the following schema. And "
        "everything should be short and succinct with no emoji. Here is the answer schema as "
        "a template:\n"
        f'{{AbstractLLMResponse.model_json_schema()["properties"]}}',
    },
]

logger = logging.getLogger(__name__)
logger.debug(f>Loading in model {model_name}")
pipe = create_pipeline(model_name)
parser = JsonSchemaParser(AbstractLLMResponse.model_json_schema())
```

Run

Where do we get the data?

We have a HUGE advantage in HEP

- Years of work to get this – and no one planned for LLM's
- Following the generally good practice of composability.



Many of our **main information websites have API's**

- Any python program can fetch from them easily!
- Popular ones have python libraries
- Easy to build API library yourself if you need it!

We love to hate indico... but...

Conference Ranking



Conference Ranking



Think about this as how you work

- You aren't writing an code algorithm with if statements and loops
- Really have to think about what *context* helps you judge a talk title and abstract!
- How little information do we need? Usually a title is **all** you need!

My Interests

```
# Raw prompt for the LLM
abstract_ranking_prompt = """Help me judge the following conference presentation as interesting or
not by summarizing the abstract and ranking it according to topics I'm interested in or not.
"""

interested_topics = [
    "Hidden Sector Physics",
    "Long Lived Particles (Exotics or RPV SUSY)",
    "Analysis techniques and methods and frameworks, columnar analysis, particularly those based around python or "
    "ROOT's DataFrame (RDF)",
    "Machine Learning and AI for particle physics",
    "The ServiceX tool",
    "Distributed computing for analysis (e.g. Dask, Spark, etc)",
    "Data Preservation and FAIR principles",
    "Differentiable Programming",
]

not_interested_topics = [
    "Quantum Computing",
    "Lattice Gauge Theory",
    "Neutrino Physics",
]
```

[Code](#)

This took a fair amount of tuning...
Still not right!

- Doesn't look at authors...

What do I ask for back from the LLM?

```
class AbstractLLMResponse(BaseModel):
    "Result back from LLM grading of an abstract"

    # Summary of the abstract
    summary: str = Field(
        ...,
        title="A short summary of the abstract that does not repeat the title, no more than 200 "
        "characters. If there is no abstract provided, just repeat the title.",
    )

    # The most likely experiment this is associated with
    experiment: str = Field(
        ...,
        title="The Experiment associated with this work if known (ATLAS, CMS, LHCb, "
        "MATHUSLA, etc.). Blank if unknown. No explanation.",
    )

    # List of keywords
    keywords: List[str] = Field(
        ..., title="Short JSON list of string-keywords associated with the abstract"
    )

    # What is the interest level here?
    interest: str = Field(
        ...,
        title="The string 'high', 'medium', or 'low' indicating how interesting I'll find "
        "the abstract.",
    )
```

```
# A short explanation of why the interest level is what it is
explanation: str = Field(
    ...,
    title="Very short explanation of the interest level in the abstract. No more than a "
    "single sentence, 100 words maximum.",
)

# How confident is the AI of its interest assignment?
confidence: float = Field(
    ...,
    title="A float from 0 to 1 representing the confidence in the interest level.",
)

# Any terms in the abstract that the LLM does not know, but would probably make
# the confidence level higher.
unknown_terms: List[str] = Field(
    ...,
    title="Short JSON list of terms (strings) in the abstract whose definition would "
    "improve your confidence.",
)
```

- Use the JSON schema to prompt LLM for formatting
- Does not work for smaller LLM's, but works just right for larger LLM's!

Example: ACAT 2024 – GPT4o

Title (from indico)	Abstract Summary (LLM)	Experiment (LLM)	Keywords (LLM)
Ahead-of-time (AOT) compilation of Tensorflow models for deployment	The talk presents a novel inference approach using Ahead-of-time (AOT) compilation for TensorFlow models to reduce memory footprint and improve computational performance in CMS workflows.	CMS	['Machine Learning', 'TensorFlow', 'Ahead-of-time Compilation', 'High-Energy Physics', 'Data Analysis']
"Accelerating Particle Physics Simulations with Machine Learning using Normalizing Flows and Flow Matching"	Using Normalizing Flows and Flow Matching to accelerate simulations of high-energy physics collision events, achieving faster and accurate predictions compared to traditional methods.		['Machine Learning', 'Simulations', 'Normalizing Flows', 'Flow Matching', 'High-Energy Physics']
Modern Machine Learning Tools for Unfolding	Modern machine learning tools are used to perform high-dimensional, unbinned unfolding of LHC data with methods such as event reweighting and direct mapping, accounting for correlations.	LHC	['machine learning', 'unfolding', 'LHC data', 'event reweighting', 'high-dimensional methods']
Using Legacy ATLAS C++ Calibration Tools in Modern Columnar Analysis Environments	Efforts to adapt legacy ATLAS C++ calibration tools for use in modern columnar analysis environments like Python/Awkward and RDataFrame.	ATLAS	['ATLAS', 'C++', 'columnar analysis', 'Python', 'RDataFrame', 'calibration tools']
A Function-As-Task Workflow Management Approach with PanDA and iDDS	Overview of the PanDA and iDDS platform for automating multi-step data processing workflows, focusing on python-based user interfaces and distributed task scheduling.	ATLAS	['Distributed Computing', 'Workflow Management', 'Python', 'Machine Learning']
Fair Universe: HiggsML Uncertainty Challenge	The Fair Universe project is creating an AI ecosystem for HEP to minimize systematic uncertainties and predict confidence intervals using large datasets and advanced analysis techniques.		['AI', 'systematic uncertainties', 'confidence intervals', 'Higgs decay', 'tau leptons', 'Codabench', 'NERSC', 'Perlmutter']
AI-driven HPC Workflows Execution with Adaptivity and Asynchronicity in Mind	Investigation into adaptive and asynchronous execution of heterogeneous tasks in scientific workflows using AI-driven middleware for improved resource utilization and reduced costs.	G. Watts (UW/Seattle)	['AI', 'HPC', 'adaptive execution', 'asynchronous execution', 'middleware']

- These are the “most interesting” talks and Posters from the ACAT 2024 workshop.
- Generated with GPT4o
- Cost \$0.43 USD

Example: ACAT 2024 – GPT4o-mini

"Accelerating Particle Physics Simulations with Machine Learning using Normalizing Flows and Flow Matching"	The talk discusses using Normalizing Flows and Flow Matching in machine learning to enhance particle physics simulations, achieving significant speed-ups and accuracy.		['Machine Learning', 'Normalizing Flows', 'Particle Physics Simulation', 'Data Analysis']
Study of columnar data analysis methods to complete an ATLAS analysis	The presentation tests columnar analysis tools, focusing on ServiceX and Coffea, to enhance ATLAS analysis efficiency and publishable results.	ATLAS	['columnar analysis', 'ServiceX', 'Coffea', 'Run-2 analysis', 'supercomputing']
columnflow: Fully automated analysis through flow of columns over arbitrary, distributed resources	The presentation introduces *columnflow*, a Python toolkit for automating large-scale HEP analyses using distributed resources and various data formats.		['automated analysis', 'distributed computing', 'Python', 'machine learning', 'data workflow']
The Good, The Bad, and the Ugly: A Tale of Physics, Software, and ML	The talk discusses using an RNN to improve the identification of long lived particles at the LHC, focusing on software modernization and systematic error control.	ATLAS	['long lived particles', 'machine learning', 'neural networks', 'software modernization', 'signal discrimination']
ServiceX, the novel data delivery system, for physics analysis	ServiceX addresses data extraction challenges for physics analysis, presenting a python-based system with transformer containers and REST API integration.		['ServiceX', 'data delivery', 'python', 'Kubernetes', 'physics analysis']
Quasi interactive analysis of High Energy Physics big data with high throughput	The presentation discusses a new interactive high-throughput data analysis approach for processing large datasets in High Energy Physics, utilizing tools like Dask and ROOT RDataFrame.		['High Energy Physics', 'big data', 'data analysis', 'Dask', 'ROOT RDataFrame', 'high-throughput', 'cloud infrastructure']
Optimizing ANN-Based Triggering for BSM events with Knowledge Distillation	This work explores optimizing ANN-based triggering for BSM events using Knowledge Distillation to enhance data processing efficiency at the LHC.	LHC	['Machine Learning', 'Artificial Neural Networks', 'Knowledge Distillation', 'BSM events', 'trigger optimization']

- These are the “most” interesting talks and Posters from ACAT 2024
- Generated with GPT4o-mini
- Cost less than \$0.02 USD

But...

GPT4o vs GPT4o-mini

GPT-4o

Ahead-of-time (AOT) compilation of Tensorflow models for deployment	The talk presents a novel inference approach using Ahead-of-time (AOT) compilation for TensorFlow models to reduce memory footprint and improve computational performance in CMS workflows.	CMS	['Machine Learning', 'TensorFlow', 'Ahead-of-time Compilation', 'High-Energy Physics', 'Data Analysis']	3	Poster
---	---	-----	---	---	--------

GPT-4o-mini

Ahead-of-time (AOT) compilation of Tensorflow models for deployment	The presentation discusses a novel inference approach for TensorFlow models in high-energy physics, focusing on AOT compilation to reduce memory usage and improve performance.	CMS	['Machine Learning', 'TensorFlow', 'AOT Compilation', 'High-Energy Physics', 'Data Analysis']	2	Poster
---	---	-----	---	---	--------

Example: ACAT 2024 – Phi3-mini

p-adic reconstruction of rational functions in multi-loop calculations	Discusses a new p-adic technique for compact rational function interpolation in multi-loop calculations, requiring fewer samples than traditional methods and resulting in more compact calculations in particle physics. Not quantum computing, lattice theory, or neutrino-specific. 'high' The talk directly addresses techniques using p-adic numbers to improve the efficiency and compactness of multi-loop scattering amplitude calculations, which is within the scope of my interests, especially concerning computational methods applicable to theoretical physics.	Blank	['p-adic reconstruction', 'multi-loop calculations', 'computer algebra', 'interpolation', 'rational functions', 'particle physics']
Optimizing the ATLAS Geant4 detector simulation	The ATLAS experiment enhanced Geant4 simulation for better FL-LHC throughput with recent Woodcock tracking integration and various optimizations like software enhancements, simplifications in geometry/magnetic field descriptions, and code-Geant4 interface tweaks leading to over 100% improvement in performance.	ATLAS	['Geant4', 'ATLAS', 'Monte Carlo Simulation', 'Performance Optimization', 'Electromagnetic Calorimeter', 'Run 3', 'HL-LHC']
Fast and Precise Amplitude Surrogates with Bayesian and Symmetry Preserving Networks	Deep learning with symmetries for accurate matrix elements scaling despite multiple particles in interactions		['Deep learning', 'Symmetry preservation', 'Bayesian networks', 'Machine learning']

- Phi-3 model is much more chatty!
- Has VERY different answers
- No real synthesis for summary
- ~3B parameter model
- Took about 15 minutes to run on a A100, 45 on a RTX 3080

GPT4o vs GPT4o-mini vs Phi3.5-Mini

GPT-4o

Ahead-of-time (AOT) compilation of Tensorflow models for deployment	The talk presents a novel inference approach using Ahead-of-time (AOT) compilation for TensorFlow models to reduce memory footprint and improve computational performance in CMS workflows.	CMS	['Machine Learning', 'TensorFlow', 'Ahead-of-time Compilation', 'High-Energy Physics', 'Data Analysis']	3	Poster
---	---	-----	---	---	--------

GPT-4o-mini

Ahead-of-time (AOT) compilation of Tensorflow models for deployment	The presentation discusses a novel inference approach for TensorFlow models in high-energy physics, focusing on AOT compilation to reduce memory usage and improve performance.	CMS	['Machine Learning', 'TensorFlow', 'AOT Compilation', 'High-Energy Physics', 'Data Analysis']	2	Poster
---	---	-----	---	---	--------

Phi3.5-Mini

No real synthesis

Ahead-of-time (AOT) compilation of Tensorflow models for deployment	The TensorFlow AOT model compilation talk discusses deploying machine learning models in particle physics with low memory and higher computational efficiency for better real-world production scenarios. Integration practices and strategies are also presented for practical implementation in physics experiments. A focus on performance improvement for central IT infrastructure utilization in high-energy physics experiments.	CMS	['Machine Learning', 'Distributed Computing', 'Data Preservation', 'AOT Compilation', 'TensorFlow models', 'Deployment', 'High Energy Physics', 'Resource Optimization']	2	Poster
---	---	-----	--	---	--------

G. Watts (UW/Seattle)

Example: ACAT 2024 – Phi3-small

Optimizing the ATLAS Geant4 detector simulation	Enhancements to Geant4 simulation for ATLAS experiment, improving efficiency by 100%+ since Run 2	ATLAS	['Geant4', 'detector simulation', 'Electromagnetic Calorimeter', 'Woodcock tracking', 'Optimization']
"Accelerating Particle Physics Simulations with Machine Learning using Normalizing Flows and Flow Matching"	Utilizes machine learning, specifically Normalizing Flows, to speed up and improve accuracy of high-energy physics simulations. Discusses oversampling for uncertainty reduction. Short list of keywords associated with the abstract: ['Machine Learning', 'Normalizing Flows', 'Simulation', 'HEP']. 'high': Given my interest in Machine Learning and AI for particle physics, this talk is highly interesting. It pertains directly to the development of analysis frameworks using ML, which aligns well with my interest in improving current methodologies for HEP simulation analysis and meets several of the other topics I am keen on such as Data Analysis techniques and Distributed computing for analysis, though the talk doesn't directly address distributed computing approaches or FAIR data principles. Nonetheless, the proposed approach for direct generation of final data could potentially make use of these topics.	Not specified	['Machine Learning', 'Normalizing Flows', 'Simulation', 'HEP']
Modern Machine Learning Tools for Unfolding	Modern ML tools applied to LHC data unfolding, comparing known/unveiled methods for performance control in many dimensions.	LHC	['unfolding', 'machine learning', 'LHC', 'data analysis', 'performance control']
Using Legacy ATLAS C++ Calibration Tools in Modern Columnar Analysis Environments	Adapting legacy C++ calibration tools for ATLAS experiments to fit columnar analysis in modern environments with minimal code changes for on-the-fly calculations in Python Jupyter notebooks <small>G. Watts (UW/Seattle)</small>	ATLAS	['ATLAS', 'Legacy C++ Tools', 'Columnar Analysis', 'Compatibility Challenges', 'Minimal Code Modifications', 'On-the-fly Calibration', 'Python Jupyter Notebook']

- Better, but not by a lot
- ~8B parameter model
- Took about 45 minutes to run on a A100. Could not fit on a 3080.

GPT4o vs GPT4o-mini vs Phi3.5-Mini

GPT-4o

Ahead-of-time (AOT) compilation of Tensorflow models for deployment	The talk presents a novel inference approach using Ahead-of-time (AOT) compilation for TensorFlow models to reduce memory footprint and improve computational performance in CMS workflows.	CMS	['Machine Learning', 'TensorFlow', 'Ahead-of-time Compilation', 'High-Energy Physics', 'Data Analysis']	3	Poster
---	---	-----	---	---	--------

GPT-4o-mini

Ahead-of-time (AOT) compilation of Tensorflow models for deployment	The presentation discusses a novel inference approach for TensorFlow models in high-energy physics, focusing on AOT compilation to reduce memory usage and improve performance.	CMS	['Machine Learning', 'TensorFlow', 'AOT Compilation', 'High-Energy Physics', 'Data Analysis']	2	Poster
---	---	-----	---	---	--------

Phi3.5-Mini

Ahead-of-time (AOT) compilation of Tensorflow models for deployment	Talk on using AOT compilation to improve memory usage and performance of TensorFlow models for deployment in particle physics applications	CMS	['TensorFlow', 'AOT compilation', 'Particle physics', 'Machine learning', 'CMS experiment']	2	Poster
---	--	-----	---	---	--------

How confident is the LLM?

- Ask it to rate its own confidence on a 0-1.0 scale
- Ask for a list of unknown keywords

(for this one talk I've been discussing...)

	GPT 4o	GPT4o-mini
Rating	3	2
Confidence	0.9	0.7
Unknown Keywords	[]	[]

- Both claimed they knew everything (ha!)
- 4o was more confident of its results

How about unknown terms for all of ACAT?

GPT-4o

RNTuple, JetNet datasets, AtlFast3, DCU, p2r, JLAB Integrating Research Infrastructure Across Facilities (JIRIAF), multi-channel weights, Frontier, Perlmutter, p-adic numbers, FastCaloSimV2, ERSAP, HEPscore23, compute job slot performance, itwinai, LDRD, Foundation Models, DDPM, StashCache, ePIE Conjugate Gradient, VQ-VAE, BNL/SDCC Facility, HEPS, Virtual-Kubelet, Services for Optimized Network Inference on Coprocessors (SONIC), SimCLR, Allen framework, ParticleTransformer, EJFAT, RTDP, Catch2, QuantOM, Digital Twin, CaloDiT, Lamarr, LO and NLO amplitude distributions, Gaussino, ACTS-FATRAS, NAQSS, LArTPC, PQC, Intel Quantum SDK, FPGA Accelerated Transport, Site Sonar, normalizing flow, leading-color approximation, patatrack, VICReg, NERSC, Alpaka library, Woodcock tracking, AdaptivePerf, NPLM, OSDF, HEP-CCE, object condensation approach, OLCF, VEGAS initialization, node filtering, particle-resolved direct numerical simulation, quark-gluon plasma, CaloChallenge, ptycho-W1Net, ODE/SDE-based samplers, SYCLOPS EU project, amplitude surrogates, std::par, flash-simulation paradigm, two-loop five-point amplitudes, GenVectorX, parton-level event file format, HUAWEI NPU 910, Fourier Neural Operators

GPT-4o-mini

CaloChallenge, diffusion networks, T/P separation, NAQSS, Digital Twin Engine, Geant4, bootstrapping, parton shower model uncertainties, Codabench, Variational Autoencoder, D-Wave Ocean, QCD jets, geometric deep learning, transformer, VegasFlow, MadFlow, Perlmutter, persistify, quantum annealing, RayTune, GNN4ITk, transient processing, R3SL, HDFML cluster, INN, adversary, MLFlow, uRWell-based detector, HiggsML, QUBO, recurrent neural network, itwinai, Alpaka library, self-supervision, PDFFlow, PQC, beam induced background, NERSC, Line Segment Tracking

What I learned...

Open-Source configuration and running is painful

The large models really do make a big difference
for this sort of task

8 Billion state-of-the-art models are the minimum
for this sort of work

Prompt tuning requires quite a lot of work

When working with commercial services, caching is
worth the effort
(easy in Python...)

Conclusions

- **Other Projects**
 - Summarize meeting minutes w.r.t. my interests
 - Pick out archive abstracts w.r.t. my interests (alpha version of this exists)
- **Open-Source Models**
 - Easy to fetch, fairly easy to run, but slow!!
 - But low cost of using commercial services means it is likely not to be worth it
 - OpenAI is the only one I tried
 - Others look to be at least as easy to use
 - General rule: The larger the LLM, the better the results and better it is taking direction.
- **The more we standardize our software...**
 - Indico, CDS, the Archive, etc.
 - The more this is done, the easier it is to do projects like this!
 - If you are building a new HEP information website, build in an API!
- **All the libraries are there to enable non-experts (like me) to play**
 - This is mostly just writing a script
 - What little things can you do to make your HEP life easier?
 - Try it out!
- **Go forth and make a (controversial) mess!**
 - It didn't think I was interested in ServiceX...
- [Code](#)
- **CHEP 2024:**
 - 4o-mini: 18 minutes, \$0.08 USD
 - 4o: 20 minutes, \$1.40 USD