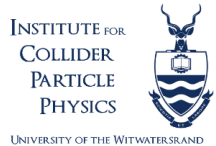
From petabytes to new physics: data compression at the Large Hadron Collider

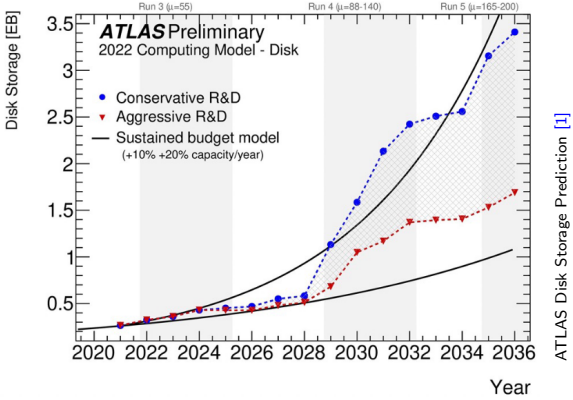
Bralyne Matoukam[1,2], Caterina Doglioni[3], Rachid Mazini[1]

[1]University of the Witwatersrand [2]AIMS Research and Innovation Centre [3]University of Manchester



Motivation: the HL-LHC data challenge

The ATLAS experiment at the Large Hadron Collider (LHC) is entering the High Luminosity era, in which storage will pose a significant challenge.



- The experiment will record data at ten times the current rate
- Increased luminosity $\Longrightarrow$ high event pile-up $\Longrightarrow$ intense storage demands
- Data explosion where storage requirements will continue to increase over the lifetime of the experiment

- There is a need for efficient lossless data compression algorithms $\Longrightarrow$ motivates the work in this presentation

HEP datasets under analysis

Primary Datasets:

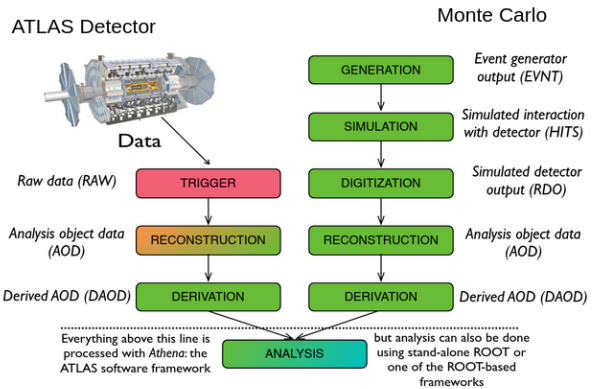
- **ROOT Files:** The standard format for LHC data storage and analysis.
  - `TTree`: Legacy data container used for the past 25 years.
  - `RNTuple`: Next-generation storage layer (See [2], [3])
- **HEPMC Files:** The universal event-record format used to store the pure particle simulation event graph

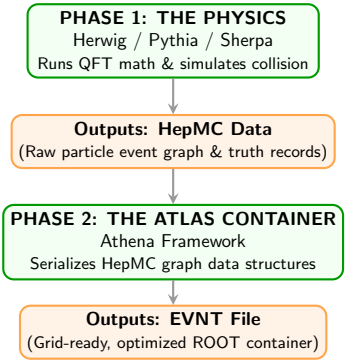
The role of HEPMC

HEPMC files provide the unbiased generator-level "truth" baseline.

It acts as the universal link between theoretical event generators (e.g., `Herwig`, `Pythia`, `Sherpa`) and detector simulation (`Geant4`), making it essential for validating physics models

ATLAS and event generation data formats





Compression impacts every stage of the data-processing pipeline.

Traditional lossless compression

We first establish a performance baseline by evaluating traditional lossless compression algorithms integrated within the `RNTuple` framework inside ATLAS Athena [4].

Core metrics evaluated:

- **File size:** total disk footprint
- **Read throughput:** Measured in events per millisecond, it is the time spent decompressing and deserializing objects

Algorithms evaluated:

- Four primary traditional algorithms (`LZMA`, `ZSTD`, `LZ4`, and `ZLIB`) across tunable compression levels (1-9) [5]

Traditional lossless compression: results

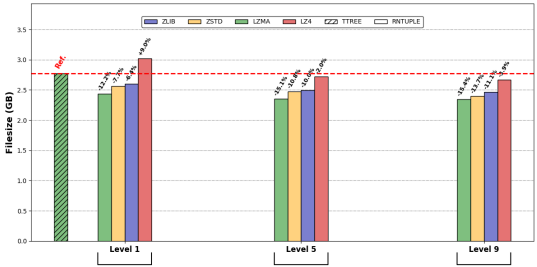


Figure: Total AOD file size scaling across compression algorithms of MC

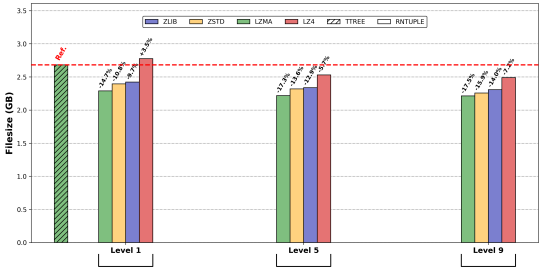


Figure: Total AOD file size scaling across compression algorithms and levels of real data collision

- RNtuple reduces the file size for most algorithms
- LZMA_RNtuple produces the smallest file size at each compression level
- The RNtuple format is more efficient for data. For instance, LZMA RNtuple level 1 provides a 12.2% reduction for MC, compared to a 14.7% reduction for data

Traditional lossless compression: results

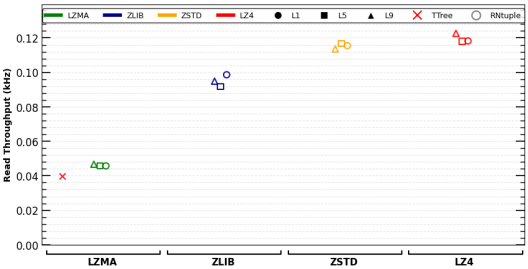


Figure: Read Throughput vs Compression Algorithm for MC

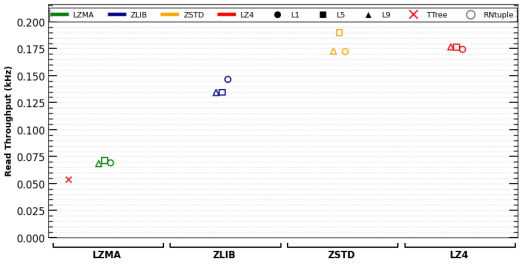


Figure: Read Throughput vs Compression Algorithm for real data collision

- Fastest read throughput: LZ4 with RNtuple for MC and ZSTD for data
- Slowest read throughput: LZMA with TTree
- Read throughput depends on compression algorithms and not on compression levels
- The RNtuple format is more efficient for data. For instance, ZSTD RNtuple level 5 provides up to 0.200 kHz speed for data compared to a 0.12 kHz for MC.

Traditional lossless compression: results

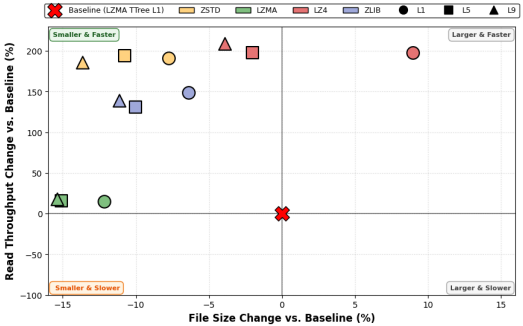


Figure: Read Throughput change vs File size change for MC

Figure: Read Throughput change vs File size change for real data collision

- Significant improvements in read speed are observed for ZSTD, LZ4, and ZLIB compared to LZMA, while achieving a comparable size reduction
- ZSTD provides the best trade-off between storage reduction and read throughput

What have we learned so far?

- RNTuple consistently improves storage efficiency over TTree.
- ZSTD provides the best trade-off between file size and read throughput.
- LZMA with Rntuple produces the smallest file size.
- Traditional compression algorithms are approaching their practical limits.

The next question

Can we exploit the underlying structure of physics data to achieve even better compression?

What is the absolute, ultimate limit to how much we can compress a piece of data without losing information?

– Claude Shannon (1948)

From traditional compression to AI-based compression

Traditional compression

- Relies on repeated byte patterns
- Dictionary matching and entropy coding
- Effective but limited by local redundancy

Machine learning compression

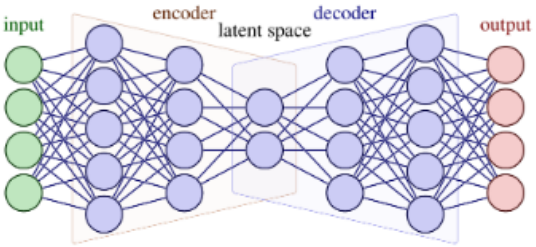
- Learns long-range structure in the data
- Predicts future bytes from previous context
- Converts compression into a sequence modeling problem

Key idea

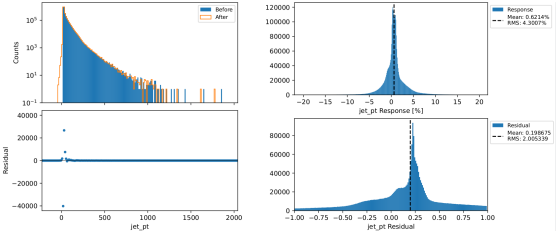
Instead of searching for repeated bytes, can a neural network learn the underlying physics patterns that generated the data?

Lossy ML compression

We can use an autoencoder to learn a compact latent representation of the data [6]



Autoencoder architecture



Results on ATLAS Open Data

Observation

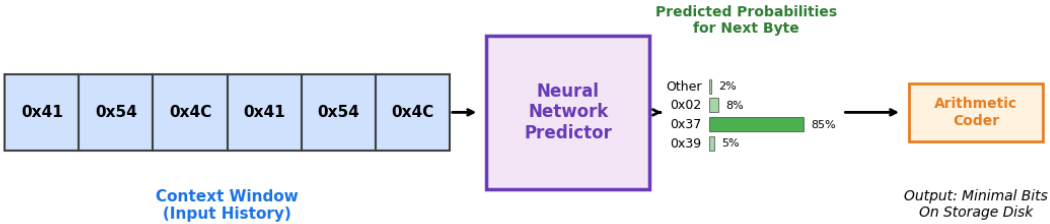
Autoencoders achieve lossy compression by discarding part of the information loss of performance

Limitation of autoencoder compression

Lossless compression is preferable in workflows where precision is important

Language models for lossless compression

- Instead of searching for repeated byte patterns, a language model learns the structure of the data
- Given the previous bytes, the model predicts the most likely next byte in the sequence
- The better the prediction, the fewer bits are required to store the information
- An arithmetic encoder converts these predictions into a compact compressed bitstream.
- Example of such an architecture: DeepMind language model [7]



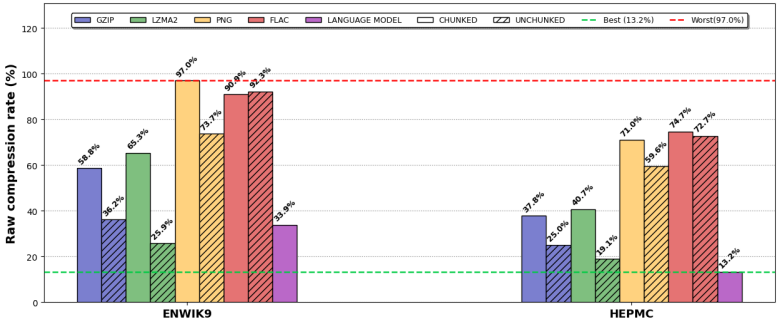
Neural-network predictor combined with arithmetic coding for lossless compression

DeepMind language model results: compression ratio

- On the HepMC dataset, the DeepMind language-model compressor achieves a raw compression rate of $\sim 13.2\%$ of the original file size - better than other methods we use (e.g. GZIP/LZMA).

Take-away

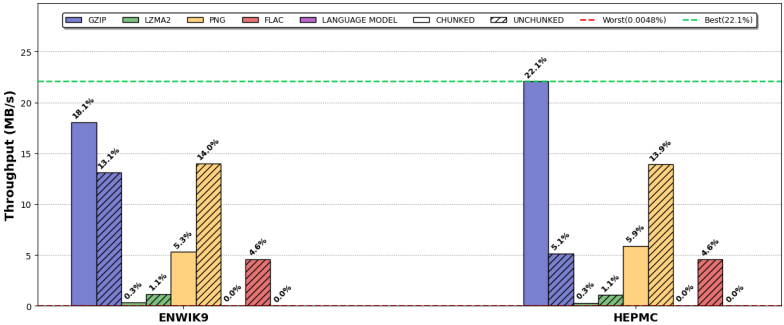
The language model achieves stronger compression than traditional algorithms



- LZMA, the strongest traditional compressor, achieves a raw compression rate of $\sim 19.1\%$
- The results suggest that neural models can exploit long-range correlations that are not fully captured by traditional compression algorithms

DeepMind language model results: throughput

- Despite achieving the best compression ratio, the DeepMind model exhibits the lowest throughput among all evaluated methods.
- The measured throughput is only ~ 0.0048 MB/s.
- Traditional compression algorithms are several orders of magnitude faster.



- Excellent compression is achieved, but the throughput is far too low for large-scale HEP production workflows

Can we preserve the compression gains while improving the throughput?

The throughput challenge

Traditional compression

Fast throughput + Limited compression gains

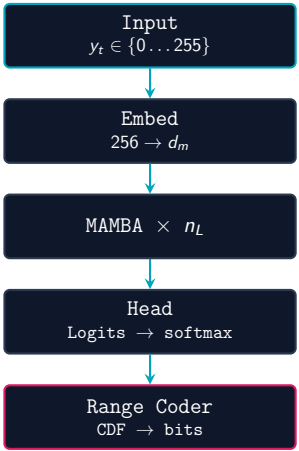
Neural compression (transformer)

Excellent compression + Very slow throughput

Can we achieve both compression and speed?

Mamba network for compression

- Transformer compressors achieve excellent compression ratios
- Their computational cost grows rapidly with sequence length
- HEP datasets contain large, highly structured event records requiring efficient processing



Mamba

A linear-time sequence model designed to capture long-range dependencies while maintaining fast inference [8].

Expected benefit: Smaller model and better throughput, keeping neural compression performance

Mamba network for compression: results on HepMC

- Compression efficiency ratio = Original Size /(Compressed Size + Model Size)

ALGORITHM	EFF. RATIO	Throughput (MB/s)
LZMA(9)	5.33x	0.95
ZSTD (19)	5.07x	1.42
ZLIB (9)	4.00x	**12.76**
BOA (256x1)*	**9.13x**	9.32

- BOA achieves the highest compression efficiency (9.13x), compared to 5.33x for LZMA [8].
- Throughput remains competitive at 9.32 MB/s faster than LZMA (0.95 MB/s) and ZSTD (1.42 MB/s).

From compression to discovery

- Neural compression models estimate how predictable each event is under the learned data distribution
- Highly compressible events $\Rightarrow$ well-represented in training data
- Poorly compressible events $\Rightarrow$ low probability under the model
- This connects compression directly to **likelihood estimation**

Key insight

Compression cost can be interpreted as an anomaly score: Events that are difficult to compress are statistically unusual.

This allows compression models to be repurposed for anomaly detection.

Hypothesis: compression-based anomaly detection

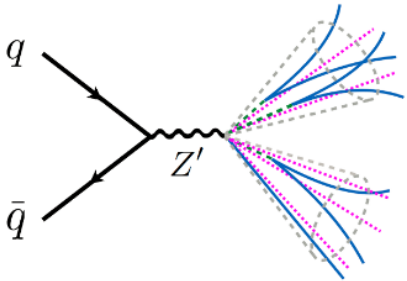
Next research question: can compression models detect physics beyond their training distribution?

Test case to be explored

- Simulate QCD background events and new physics signals
- Train the Mamba BOA model exclusively on the QCD background to learn standard model distributions unsupervised
- Establish an anomaly threshold based purely on model outputs
- Isolate events exceeding this threshold as "anomalous", without using any class labels
- Utilize truth labels strictly to check the filter results and evaluate signal selection performance across different physics models.

Dark QCD & Semi-Visible Jet Topologies

- **The Portal (Brief Theory)**: Standard Model particles decay into a hidden sector via a heavy Z' boson, producing Dark Hadrons
- **Semi-Visible Jets (r_{inv})**: A fraction of these dark hadrons escape the detector entirely. The result is a jet interleaved with missing transverse energy (MET).
- **Exploitable Structural Differences:**
  - **QCD Background** High correlation between jet directions and total balanced event energy
  - **Dark QCD (Anomalous):** Skewed jet energy ratios, unusual substructure variables (τ_{21}, τ_{32}), and misaligned MET vectors.
- **The Compression Metric**: These differences cause the background-trained model to mispredict the signal, allowing to distinguish it from the background.



Conclusions

Technical milestone: efficient neural compression

- `Mamba BOA` significantly accelerates neural throughput, reaching 9.32 MB/s while maintaining a high 9.13× compression ratio.
- It substantially outperforms traditional HEP baselines like `LZMA(9)` and `ZSTD(19)`, while processing data faster than the standard transformer architecture (DeepMind).

Future work: unsupervised anomaly detection

- **Hypothesis:** Physics events that are poorly compressible under the learned Standard Model QCD distribution point directly to statistical anomalies.
- **Target application:** investigating the model's capacity to isolate **Dark QCD** signatures and semi-visible jets via a heavy Z' portal framework.

Thank You

From petabytes to new physics

Backup

Computing resources

To support reproducibility, the source code for this analysis is available:

- DeepMind repository
- Traditional Compression Algorithms repository

Computing resources

Resource	Specification
CERN Hostname	aiatlasbm003 for MC and aiatlasbm003 for data
CPU Model	2 $\times$ AMD EPYC 7302
Clock Speed	3.0 GHz (Base) / 3.3 GHz (Max Boost)
Number of Cores	32 Physical Cores (16 per socket)
Architecture	64 bit
Threads per core	2
Cache	L1: 1024 KB, L2: 16.0 MB, L3: 256 MB
Total RAM	251 GB
Storage Media	Local NVMe SSD (1.8 TB)
Filesystem	XFS
Operating System	AlmaLinux 9.7(GCC)