Machine Learning in HEP: An Overview

Machine learning in top physics, including measurements, phenomenology and detector performance

Michael Fenton
on behalf of the ATLAS and CMS collaborations

University of California, Irvine
m.fenton@cern.ch

September 28, 2023

ML in HEP

ORIGINAL
16th TOP Physics Workshop
Traverse City, MI, USA – 24th to 29th September 2023

- Machine learning is now ubiquitous in high energy physics, and the world at large
- ATLAS public results with MVA/ML: 46 papers, 17 conf notes, 20 pub notes
  - no equivalent filter for CMS, but number should be similar
- Mostly: ML b/top-tagging, Signal vs Background Separation
- Instead of presenting a bunch of results using ML, I'd like to go over how ML can enter into every stage of analysis from data taking to final measurement

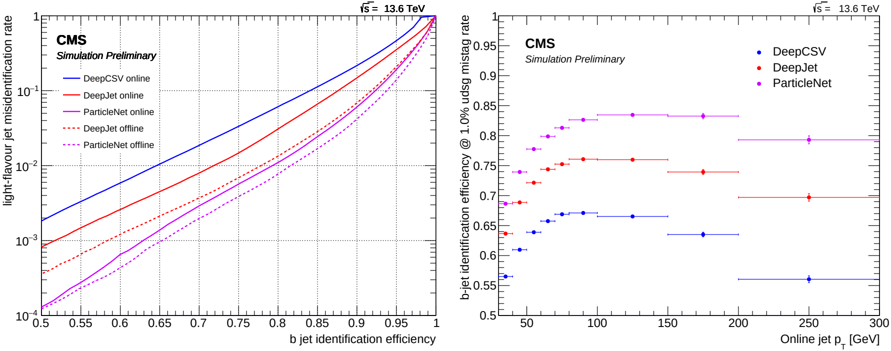
Outline

1. Data Taking / MC Generation
2. Object Selection
3. Calibration
4. Background Estimation
5. Event Reconstruction
6. Systematics
7. Unfolding
8. Paper / Talk Writing
9. Conclusions

Outline

1. Data Taking / MC Generation
2. Object Selection
3. Calibration
4. Background Estimation
5. Event Reconstruction
6. Systematics
7. Unfolding
8. Paper / Talk Writing
9. Conclusions

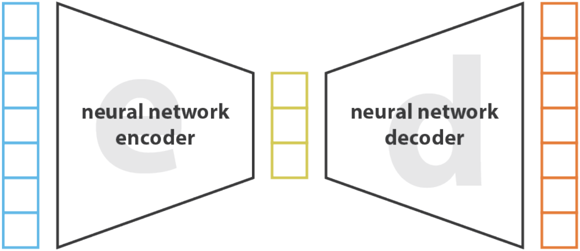
Artificially Intelligent Data Taking



- CMS have a complex new trigger strategy for HH and HHH that includes running ParticleNet in the HLT on both small and large radius jets
- ATLAS also has ML b-tagging in trigger for small jets
- Not sure if either experiment is running a dedicated top tagging trigger, but perhaps we should?

AutoEncoded Data Taking ▸ CMS

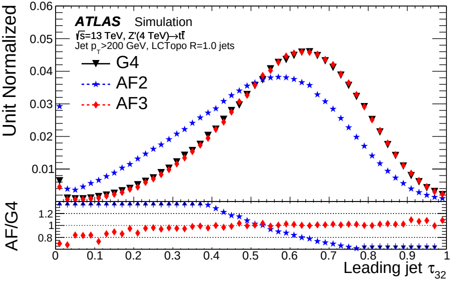
- Variational AutoEncoders (VAE) simple picture: compress into a smaller latent space, then un-compress

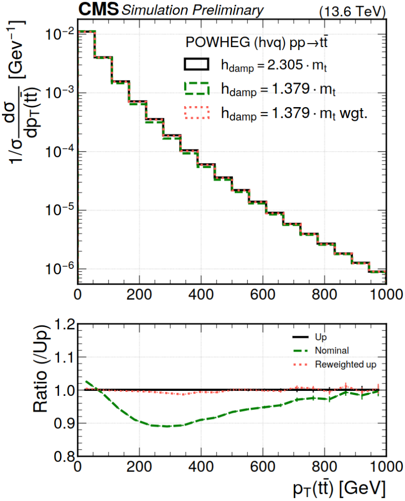


- Compress trigger level data to increase write out rates? ▸ Butter et al
- Also used for "anomaly detection": if the network struggles to undo the compression, event is "anomalous" and flagged for further study
  - ATLAS has some results using VAEs offline
  - Could be used directly in the trigger? ▸ Cerri et al
- CMS use anomaly detection tech to check DQ and flag bad runs

Machine Learned Monte-Carlo Generation

▸ Snowmass Review



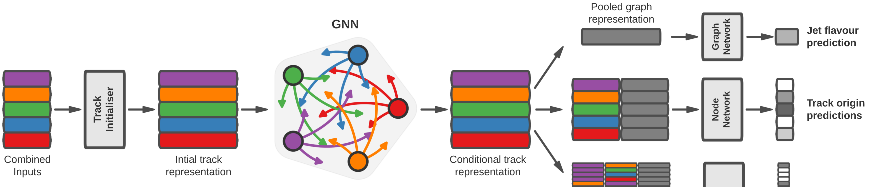


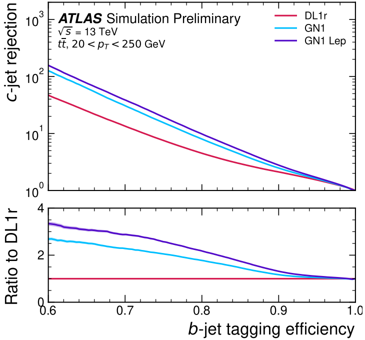
- ATLAS fast detector sim "AF3" now extensively uses GANs and VAEs
- CMS have replaced some generator comparisons with weights using NNs
  - Same procedure could be used for background estimation, calibration...
- ML useful in many places for calculations: phase space sampling, amplitudes, loop integrals, parton showers...
  - NNPDF has been a staple for many years

Outline

1. Data Taking / MC Generation
2. **Object Selection**
3. Calibration
4. Background Estimation
5. Event Reconstruction
6. Systematics
7. Unfolding
8. Paper / Talk Writing
9. Conclusions

Object Selection: *b*-tagging





ATLAS now using GN1 tagger in Run 3

→ No more low level algorithms (Run2), just jets and tracks into a Graph (GNN)!

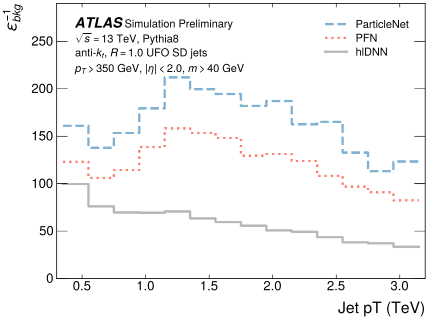
- CMS uses DeepJet which takes PFlow objects and SVs into LSTMs
- Hard to make a direct performance comparison

- Also $X \to bb$ tagging CMS, ATLAS, fully ML based PFlow, ...

Top Tagging

▸ The Landscape

- There are MANY ML-based top taggers out there these days
  - Current SOTA: Particle Transformer
- CMS using ParticleNet (GNN), ATLAS using DNN with HL vars
- Best performance *usually* comes from constituent based taggers, which use GNNs/Attention to input the entire jet (cf S.H. Lim yesterday, ADO, LASSO)
- Absolute performance improvements probably ∼saturated by now



- ATLAS has released a public dataset for benchmarking on realistic detector sim / jet reco / wider p_T range
  - Non-uniform detector → performance can vary!
  - In the near future this will be updated to include estimates of systematic uncertainties for more robust studies

Outline

1 Data Taking / MC Generation

2 Object Selection

3 **Calibration**

4 Background Estimation

5 Event Reconstruction

6 Systematics

7 Unfolding

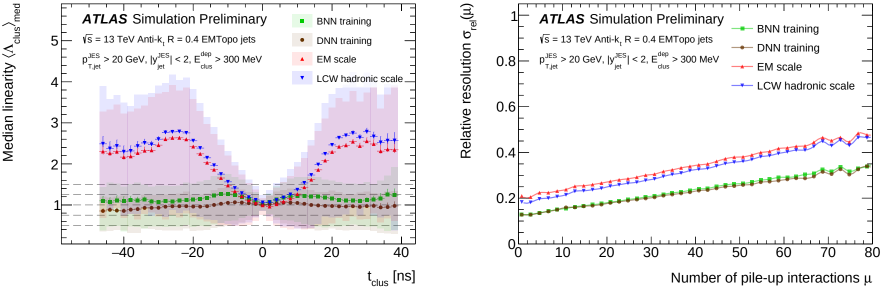
8 Paper / Talk Writing

9 Conclusions

Learning to Calibrate

- New ATLAS ML-based calibration for jets
- Fully connected or Bayesian NN to regress topocluster response
- Inputs:

$$\Big\{\underbrace{E^{\mathrm{EM}}_{\mathrm{clus}}, y^{\mathrm{EM}}_{\mathrm{clus}}}_{\text{kinematics}}, \overbrace{\zeta^{\mathrm{EM}}_{\mathrm{clus}}, t_{\mathrm{clus}}, \mathrm{Var}_{\mathrm{clus}}(t_{\mathrm{cell}})}^{\text{signal strength and timing}}, \underbrace{\lambda_{\mathrm{clus}}, |\vec{c}_{\mathrm{clus}}|, \langle\rho_{\mathrm{cell}}\rangle, \langle \mathfrak{m}^2_{\mathrm{long}}\rangle, \langle \mathfrak{m}^2_{\mathrm{lat}}\rangle, p_{\mathrm{T}}D, f_{\mathrm{emc}}}_{\text{shower location (depth), shapes and compactness}}, \overbrace{f_{\mathrm{iso}}}^{\text{topology (isolation)}}, \underbrace{N_{\mathrm{PV}}, \mu}_{\text{event/pile-up}}\Big\}.$$



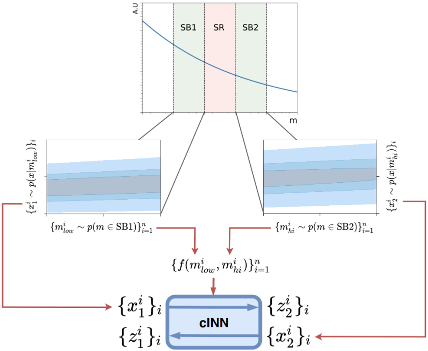
- Leads to improved resolution and decreased sensitivity to out-of-time pileup

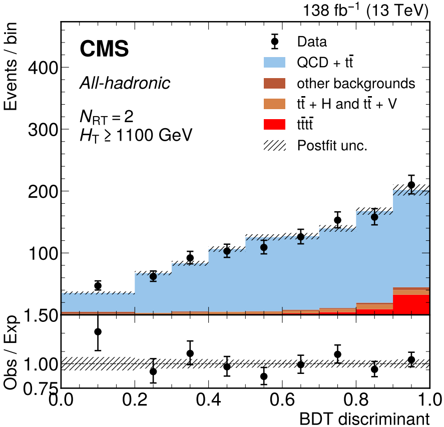
Outline

1. Data Taking / MC Generation
2. Object Selection
3. Calibration
4. **Background Estimation**
5. Event Reconstruction
6. Systematics
7. Unfolding
8. Paper / Talk Writing
9. Conclusions

Background Estimation

- Several methods available for background estimation
  - Extrapolate from sidebands to SRs with Invertible NNs (INN)
  - Replace functional forms with Gaussian Processes


A.U
SB1
SR
SB2
m
$\{x_1^i \sim p(x|m_{low}^i)\}_i$
$\{x_2^i \sim p(x|m_{hi}^i)\}_i$
$\{m_{low}^i \sim p(m \in \mathrm{SB1})\}_{i=1}^n$
$\{m_{hi}^i \sim p(m \in \mathrm{SB2})\}_{i=1}^n$
$\{f(m_{low}^i, m_{hi}^i)\}_{i=1}^n$
$\{x_1^i\}_i$
$\{z_1^i\}_i$
cINN
$\{z_2^i\}_i$
$\{x_2^i\}_i$



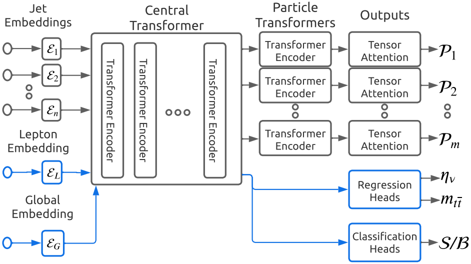
- Enhanced ABCD With Normalising Flows used in CMS all-hadronic 4-top search presented at this conference last year!

Outline

1. Data Taking / MC Generation
2. Object Selection
3. Calibration
4. Background Estimation
5. **Event Reconstruction**
6. Systematics
7. Unfolding
8. Paper / Talk Writing
9. Conclusions

Event Reconstruction: SPA-NET ▸ Fenton et al ▸ Shmakov et al

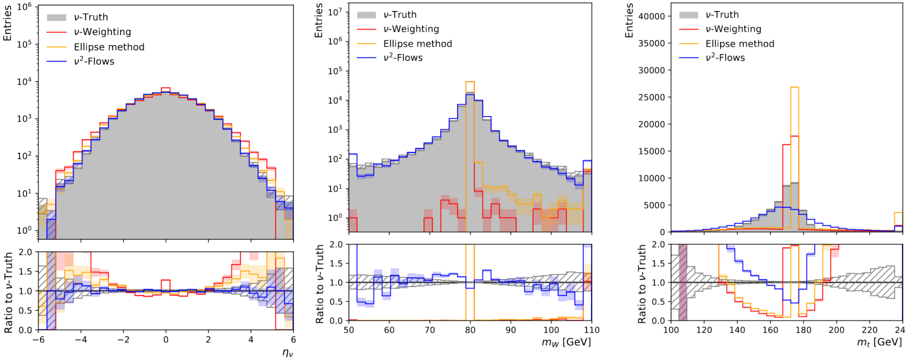
- Complete package for event reconstruction and aux tasks
  - Avoids combinatoric explosion in baseline methods like KLFitter or PDNN
- Applies to arbitrarily complex final states (jets, leptons, anything else) with easy configuration and training



- Aux outputs to remove backgrounds, bad reconstructions, partial events
- Directly regress kinematics (η^ν?)
- Extremely robust training
- Less mass sculpting, faster inference than baselines

- Overall efficiency: 75% SPA-NET, vs 41% KLFitter
- Significant gains in final sensitivity on top mass, $t\bar{t}H, H \to b\bar{b}$, $Z' \to t\bar{t}$
  - Quantum entanglement? $t\bar{t}$+HF?
- **If you reconstruct your events, you should be using this!**

ν^2-Flows ▸ Raine et al



- Use normalising flows to regress neutrino kinematics in dilepton events
  - In: 4vectors of all objects, charge/btag, N(b)jets, MET
  - Out: p_x, p_y, p_z for each neutrino
- Impressive agreement with many kinematics including η_ν, m_W, $m_{t\bar{t}}$...
  - m_t not perfect
- Possible to combine with SPA-NET, WIP!

Outline

1 Data Taking / MC Generation

2 Object Selection

3 Calibration

4 Background Estimation

5 Event Reconstruction

6 **Systematics**

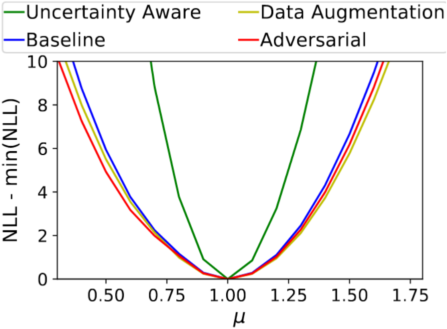
7 Unfolding

8 Paper / Talk Writing

9 Conclusions

Uncertainty Aware Training ▸ Ghosh et al

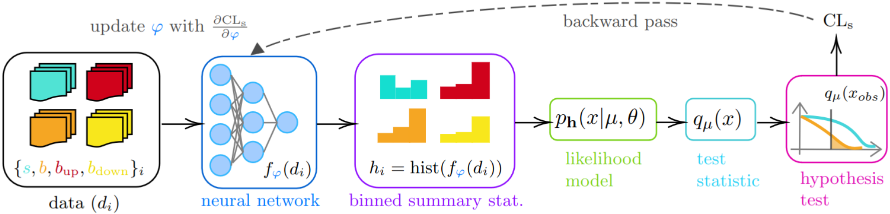
- Idea: parameterise NN vs NPs to reduce impact of uncertainties on final measurement



- Data augmentation: include the syst shifted events in training
- Adversarial: train network to be insensitive to the NP
- Uncertainty aware: give the NP to the network during training, then profile
- Tested on HiggsML dataset with τ energy scale systematic

- Significant improvement in sensitivity possible!
- BUT: unclear how well this scales to multiple NPs
- Best strategy in general depends on analysis design / nature of leading systs
- See also; parameterised NNs ▸ Baldi et al, adversarial decorrelation ▸ Englert et al

Optimal Analyses / Summary Statistics



update φ with $\frac{\partial \mathrm{CL_s}}{\partial \varphi}$

backward pass

$\mathrm{CL_s}$

$\{s, b, b_{\mathrm{up}}, b_{\mathrm{down}}\}_i$

data (d_i)

$f_\varphi(d_i)$

neural network

$h_i = \mathrm{hist}(f_\varphi(d_i))$

binned summary stat.

$p_{\mathbf{h}}(x|\mu,\theta)$

likelihood model

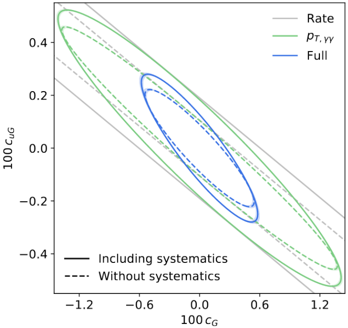
$q_\mu(x)$

test statistic

$q_\mu(x_{obs})$

hypothesis test

- NEOS: end-to-end differentiable analysis using gradient descent to find optimal sensitivity ▸ Simpson, Heinrich
    - Requires entire pipeline to be differentiable, incl histograms (bKDE), likelihoods (pyhf)
    - see also INFERNO ▸ de Castro, Dorigo
- MadMiner: NN estimation of LR parameterised by eg EFT operator ▸ Brehmer et al



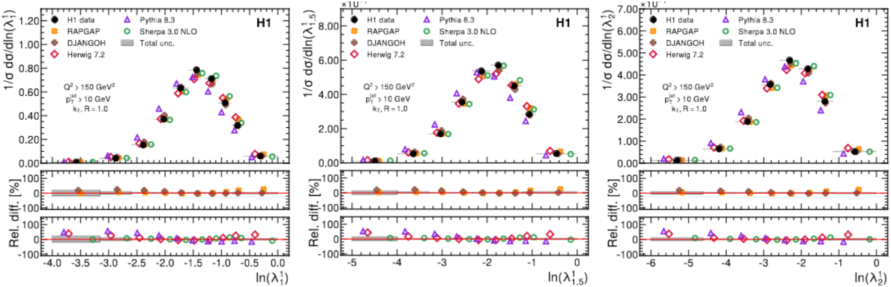
Outline

1. Data Taking / MC Generation
2. Object Selection
3. Calibration
4. Background Estimation
5. Event Reconstruction
6. Systematics
7. **Unfolding**
8. Paper / Talk Writing
9. Conclusions

Unfolding with ML

- A variety of methods on the market these days
  - Omnifold, GANs, INNs, Diffusion, mixed Omnifold+Diffusion

- Omnifold reweights events to match data pre-detector sim; GANs, INNs, & Diffusion take the detector level event as input and output a truth level event
- Only one yet used on real data: ▸ HERA has used Omnifold to measure JSS

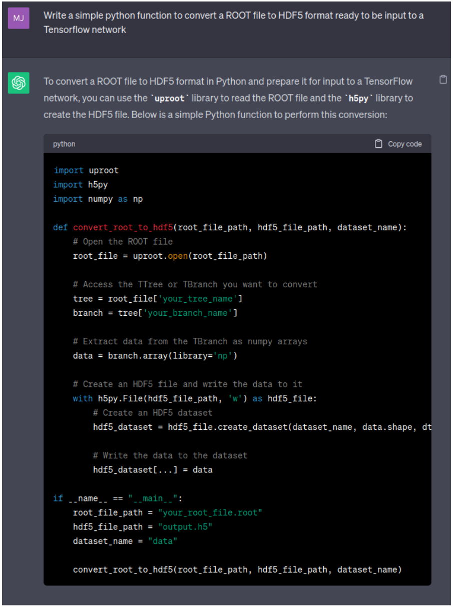


- All methods unfold very high dimensionality, **unbinned** distributions
  - No more need to pre-arrange binning for NN(N...)LO calculations!
  - But unclear how to publish these?

Outline

1. Data Taking / MC Generation
2. Object Selection
3. Calibration
4. Background Estimation
5. Event Reconstruction
6. Systematics
7. Unfolding
8. **Paper / Talk Writing**
9. Conclusions

ChatGPT to write code

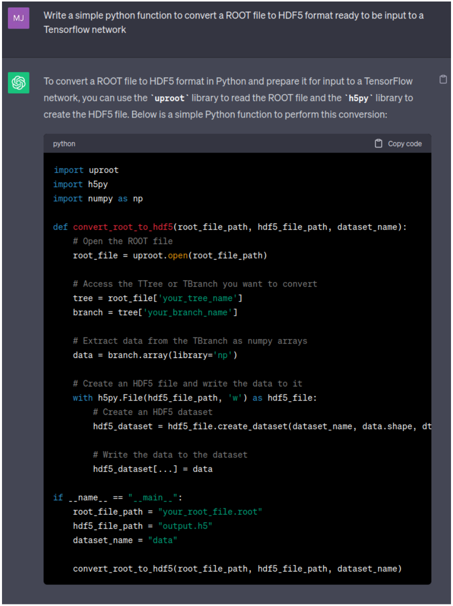


Write a simple python function to convert a ROOT file to HDF5 format ready to be input to a Tensorflow network

To convert a ROOT file to HDF5 format in Python and prepare it for input to a TensorFlow network, you can use the `uproot` library to read the ROOT file and the `h5py` library to create the HDF5 file. Below is a simple Python function to perform this conversion:

```python
import uproot
import h5py
import numpy as np

def convert_root_to_hdf5(root_file_path, hdf5_file_path, dataset_name):
    # Open the ROOT file
    root_file = uproot.open(root_file_path)

    # Access the TTree or TBranch you want to convert
    tree = root_file['your_tree_name']
    branch = tree['your_branch_name']

    # Extract data from the TBranch as numpy arrays
    data = branch.array(library='np')

    # Create an HDF5 file and write the data to it
    with h5py.File(hdf5_file_path, 'w') as hdf5_file:
        # Create an HDF5 dataset
        hdf5_dataset = hdf5_file.create_dataset(dataset_name, data.shape, dt

        # Write the data to the dataset
        hdf5_dataset[...] = data

if __name__ == "__main__":
    root_file_path = "your_root_file.root"
    hdf5_file_path = "output.h5"
    dataset_name = "data"

    convert_root_to_hdf5(root_file_path, hdf5_file_path, dataset_name)
```

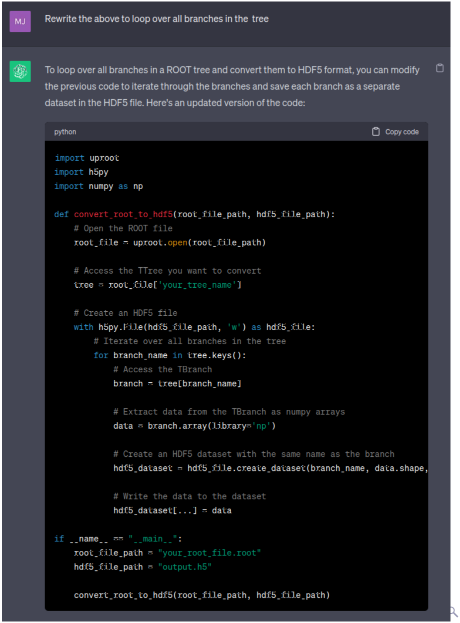
ChatGPT to write code



Write a simple python function to convert a ROOT file to HDF5 format ready to be input to a Tensorflow network

To convert a ROOT file to HDF5 format in Python and prepare it for input to a TensorFlow network, you can use the `uproot` library to read the ROOT file and the `h5py` library to create the HDF5 file. Below is a simple Python function to perform this conversion:

```python
import uproot
import h5py
import numpy as np

def convert_root_to_hdf5(root_file_path, hdf5_file_path, dataset_name):
    # Open the ROOT file
    root_file = uproot.open(root_file_path)

    # Access the TTree or TBranch you want to convert
    tree = root_file['your_tree_name']
    branch = tree['your_branch_name']

    # Extract data from the TBranch as numpy arrays
    data = branch.array(library='np')

    # Create an HDF5 file and write the data to it
    with h5py.File(hdf5_file_path, 'w') as hdf5_file:
        # Create an HDF5 dataset
        hdf5_dataset = hdf5_file.create_dataset(dataset_name, data.shape, dt

        # Write the data to the dataset
        hdf5_dataset[...] = data

if __name__ == "__main__":
    root_file_path = "your_root_file.root"
    hdf5_file_path = "output.h5"
    dataset_name = "data"

    convert_root_to_hdf5(root_file_path, hdf5_file_path, dataset_name)
```



Rewrite the above to loop over all branches in the tree

To loop over all branches in a ROOT tree and convert them to HDF5 format, you can modify the previous code to iterate through the branches and save each branch as a separate dataset in the HDF5 file. Here's an updated version of the code:

```python
import uproot
import h5py
import numpy as np

def convert_root_to_hdf5(root_file_path, hdf5_file_path):
    # Open the ROOT file
    root_file = uproot.open(root_file_path)

    # Access the TTree you want to convert
    tree = root_file['your_tree_name']

    # Create an HDF5 file
    with h5py.File(hdf5_file_path, 'w') as hdf5_file:
        # Iterate over all branches in the tree
        for branch_name in tree.keys():
            # Access the TBranch
            branch = tree[branch_name]

            # Extract data from the TBranch as numpy arrays
            data = branch.array(library='np')

            # Create an HDF5 dataset with the same name as the branch
            hdf5_dataset = hdf5_file.create_dataset(branch_name, data.shape,

            # Write the data to the dataset
            hdf5_dataset[...] = data

if __name__ == "__main__":
    root_file_path = "your_root_file.root"
    hdf5_file_path = "output.h5"

    convert_root_to_hdf5(root_file_path, hdf5_file_path)
```

Leveraging ChatGPT in High Energy Physics Research

- High-energy physics (HEP) is a complex field exploring the fundamental universe.
- ChatGPT and similar tools offer valuable applications in HEP research.

Applications in HEP:

1. Data Analysis: Interpretation and annotation of complex data.
2. Literature Review: Automated summaries and citation assistance.
3. Collaboration: Collaborative writing and multilingual communication.
4. Education and Outreach: Teaching assistance and public engagement.

Challenges and Considerations:

- Ensure AI doesn't introduce bias or compromise accuracy.
- Protect sensitive data and uphold ethical AI practices.

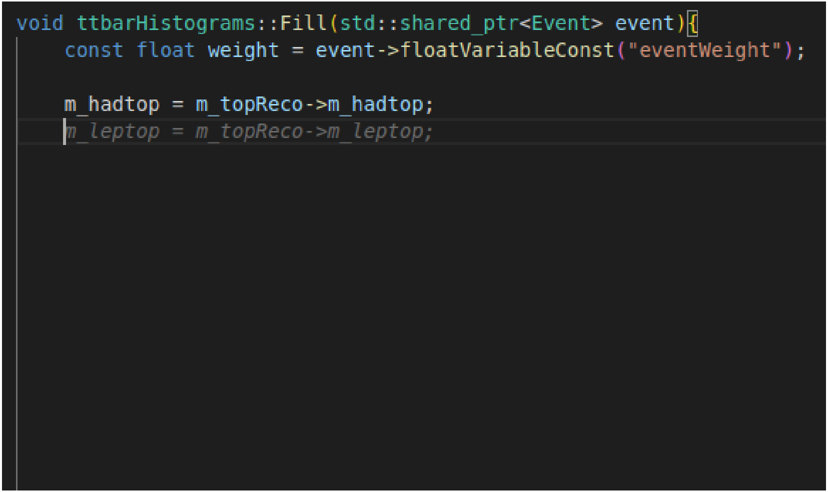
Conclusion:

- AI tools like ChatGPT enhance HEP research in various ways.
- Responsible use and ongoing developments are key for success.

This slide, apart from this footnote, was written entirely by ChatGPT

Github Co-Pilot

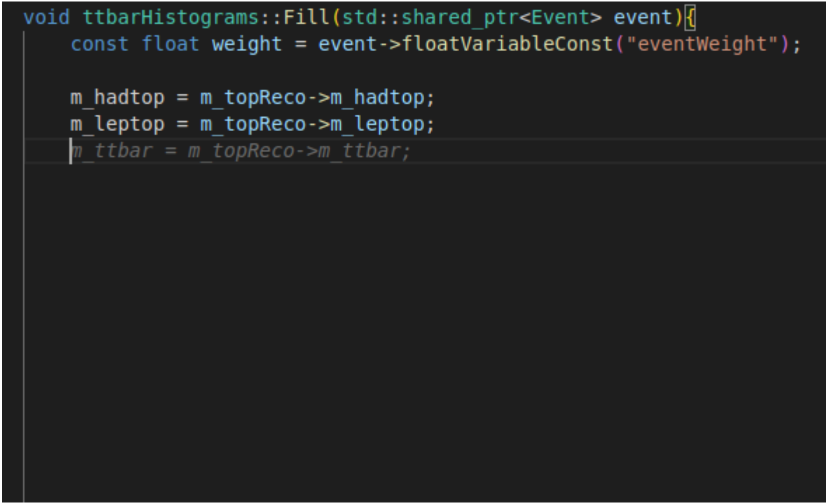
- Plug-in for VSCode, allows tab completion of code based on LLM prediction
- Available free to students and teachers!



```
void ttbarHistograms::Fill(std::shared_ptr<Event> event){
    const float weight = event->floatVariableConst("eventWeight");

    m_hadtop = m_topReco->m_hadtop;
    m_leptop = m_topReco->m_leptop;
```

Github Co-Pilot

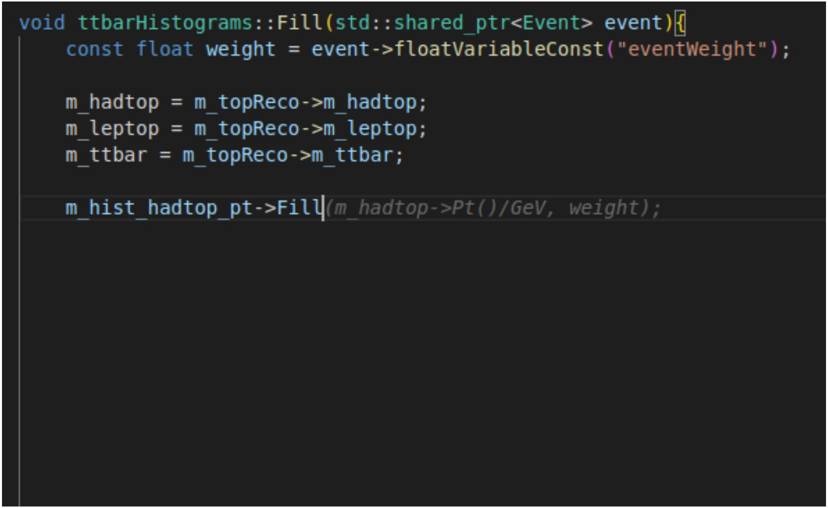
- Plug-in for VSCode, allows tab completion of code based on LLM prediction
- Available free to students and teachers!



```
void ttbarHistograms::Fill(std::shared_ptr<Event> event){
    const float weight = event->floatVariableConst("eventWeight");

    m_hadtop = m_topReco->m_hadtop;
    m_leptop = m_topReco->m_leptop;
    m_ttbar = m_topReco->m_ttbar;
```

Github Co-Pilot

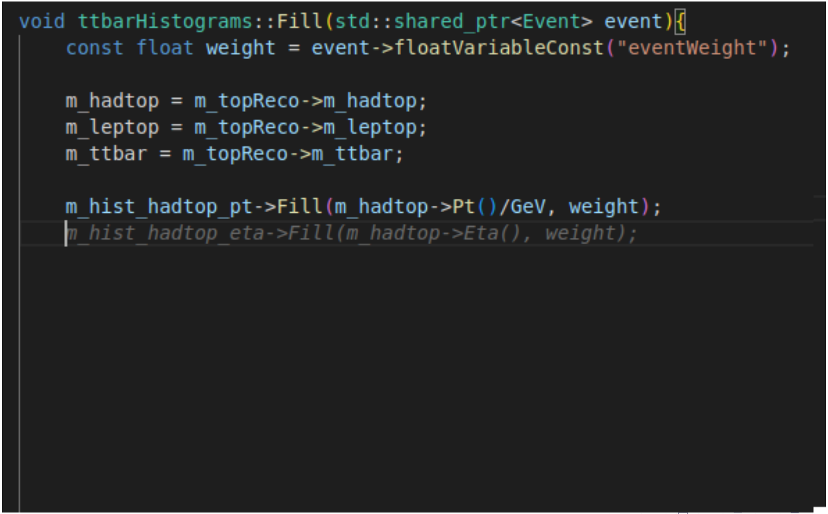
- Plug-in for VSCode, allows tab completion of code based on LLM prediction
- Available free to students and teachers!



```
void ttbarHistograms::Fill(std::shared_ptr<Event> event){
    const float weight = event->floatVariableConst("eventWeight");

    m_hadtop = m_topReco->m_hadtop;
    m_leptop = m_topReco->m_leptop;
    m_ttbar = m_topReco->m_ttbar;

    m_hist_hadtop_pt->Fill(m_hadtop->Pt()/GeV, weight);
```

Github Co-Pilot

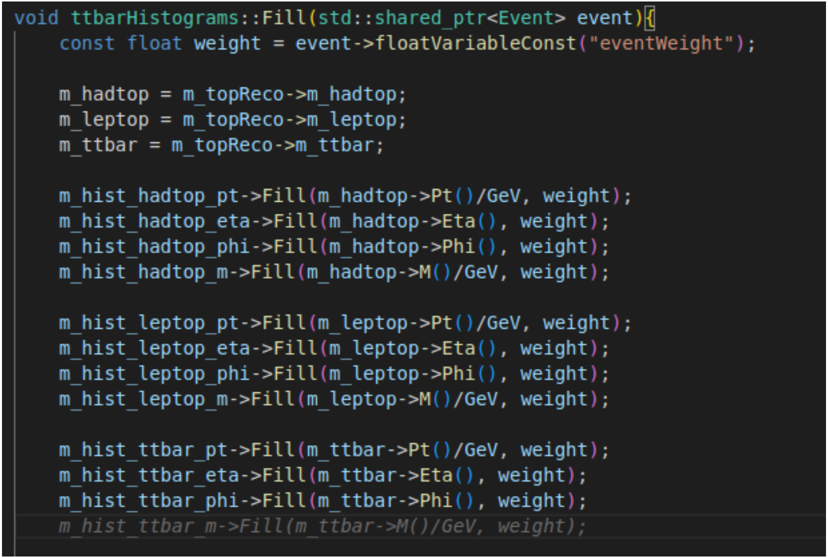
- Plug-in for VSCode, allows tab completion of code based on LLM prediction
- Available free to students and teachers!



```
void ttbarHistograms::Fill(std::shared_ptr<Event> event){
    const float weight = event->floatVariableConst("eventWeight");

    m_hadtop = m_topReco->m_hadtop;
    m_leptop = m_topReco->m_leptop;
    m_ttbar = m_topReco->m_ttbar;

    m_hist_hadtop_pt->Fill(m_hadtop->Pt()/GeV, weight);
    m_hist_hadtop_eta->Fill(m_hadtop->Eta(), weight);
```

Github Co-Pilot

- Plug-in for VSCode, allows tab completion of code based on LLM prediction
- Available free to students and teachers!



```
void ttbarHistograms::Fill(std::shared_ptr<Event> event){
    const float weight = event->floatVariableConst("eventWeight");

    m_hadtop = m_topReco->m_hadtop;
    m_leptop = m_topReco->m_leptop;
    m_ttbar = m_topReco->m_ttbar;

    m_hist_hadtop_pt->Fill(m_hadtop->Pt()/GeV, weight);
    m_hist_hadtop_eta->Fill(m_hadtop->Eta(), weight);
    m_hist_hadtop_phi->Fill(m_hadtop->Phi(), weight);
    m_hist_hadtop_m->Fill(m_hadtop->M()/GeV, weight);

    m_hist_leptop_pt->Fill(m_leptop->Pt()/GeV, weight);
    m_hist_leptop_eta->Fill(m_leptop->Eta(), weight);
    m_hist_leptop_phi->Fill(m_leptop->Phi(), weight);
    m_hist_leptop_m->Fill(m_leptop->M()/GeV, weight);

    m_hist_ttbar_pt->Fill(m_ttbar->Pt()/GeV, weight);
    m_hist_ttbar_eta->Fill(m_ttbar->Eta(), weight);
    m_hist_ttbar_phi->Fill(m_ttbar->Phi(), weight);
    m_hist_ttbar_m->Fill(m_ttbar->M()/GeV, weight);
```

Outline

1 Data Taking / MC Generation

2 Object Selection

3 Calibration

4 Background Estimation

5 Event Reconstruction

6 Systematics

7 Unfolding

8 Paper / Talk Writing

9 **Conclusions**

Summary

ORIGINAL
16th TOP Physics Workshop
Traverse City, MI, USA – 24th to 29th September 2023

- Machine learning is an increasingly ubiquitous part of our work (and personal!) lives: can help solve problems at all stages of analysis, from data collection to final interpretation
- Modern ML tools like ChatGPT, Github Copilot, etc can make our lives easier
    - Can we create CERN/ATLAS/CMS-GPT?
- Happy to discuss any and all ML ideas or questions ~~over coffee/lunch~~ by email: mjfenton@uci.edu
- Useful resource: ▸ HEPML Living Review

Backup