Artificial Intelligence and the Uncertainty challenge in Fundamental Physics

CONTROLLING SYSTEMATIC ERRORS WITH AN ADVERSARY

G. Watts
For the ATLAS Collaboration and the CalRatio Analysis Team
2023-11-30

SYSTEMATIC ERRORS

Reduce the precision and accuracy of your experimental measurement

Known Unknowns

e.g. Known measurement errors in your experiment

- Usually accounted for with some sort of calibration or characterization
- Should affect only the precision
- If done right, can be differentiated through and included in optimization

Unknown Unknowns

e.g. An unknown bias in measurement technique

- Discovered via closure tests failing
- Difference is often added to a overall systematic error
- Often discovered as a measurement nears completion.

SYSTEMATIC ERRORS

- **Discovered via closure tests failing**
- Difference is often added to a overall systematic error
- Often discovered as a measurement nears completion.

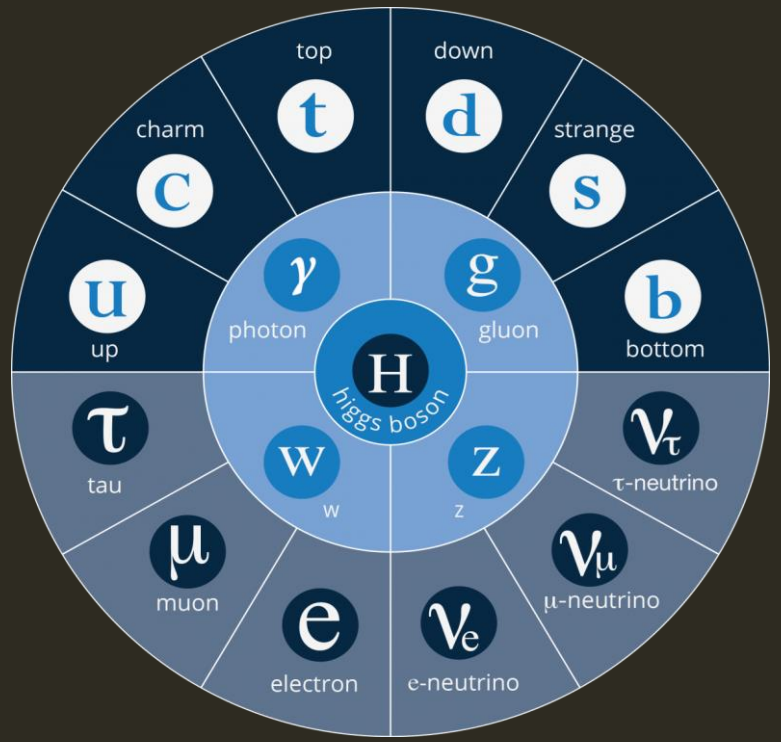
Can this closure test help us train the analysis to steer away from the discrepancy?

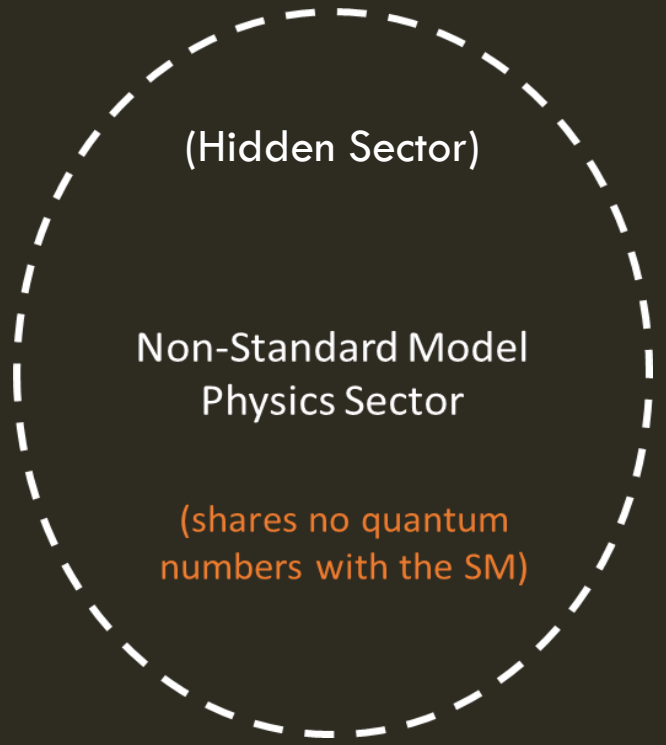
Existing examples from ATLAS

SUSY-2018-22	Search for squarks and gluinos: jets+MET BDT weights in XML format on HEPData + simpleAnalysis implementation	Nov 2020
SUSY-2019-04	RPV SUSY search, leptons + many jets ONNX files for 5 NNs (4-8 jets SRs) on HEPData + simpleAnalysis implementation	Sep 2021
SUSY-2018-30	SUSY search with MET and many b-jets simpleAnalysis implementation with ONNX-serialised NN model	Nov 2022
EXOT-2019-23	Search for neutral LLPs with displaced hadronic jets ("CalRatio LLP search") preserved NNs as ONNX, BDTs as executables with petrify-bdt; low level inputs; also 6d efficiency maps parametrising the BDT+NN selection + example code	June 2022
HDBS-2019-23	Anomaly detection search for new resonances Y → X+H in hadronic final states VRNN python code + post-training weights (PyTorch .pth file)	June 2023

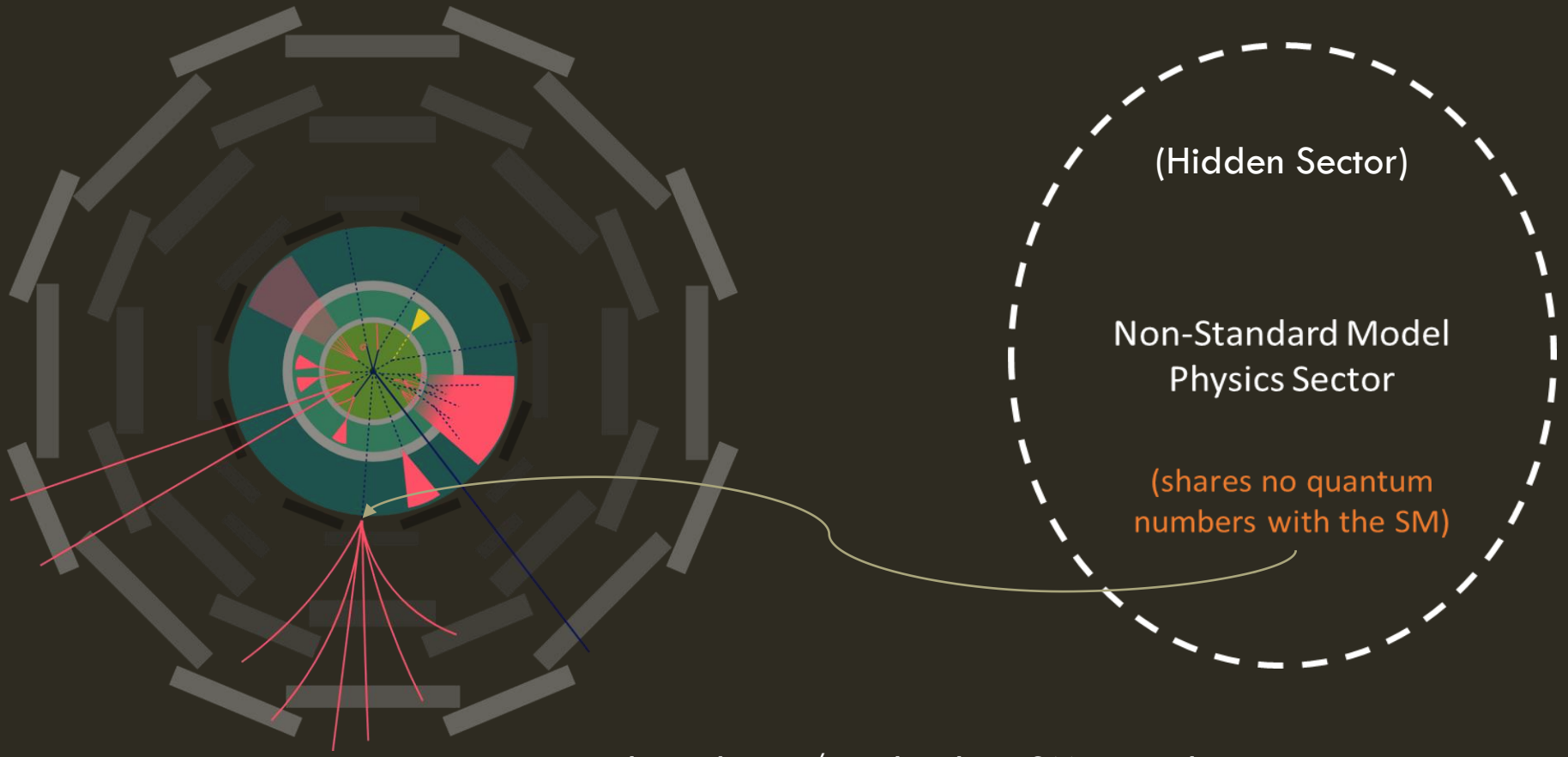
HEPDATA

SEARCHING FOR HIDDEN SECTORS





SEARCHING FOR HIDDEN SECTORS



But it does decay/mix back to SM particles!

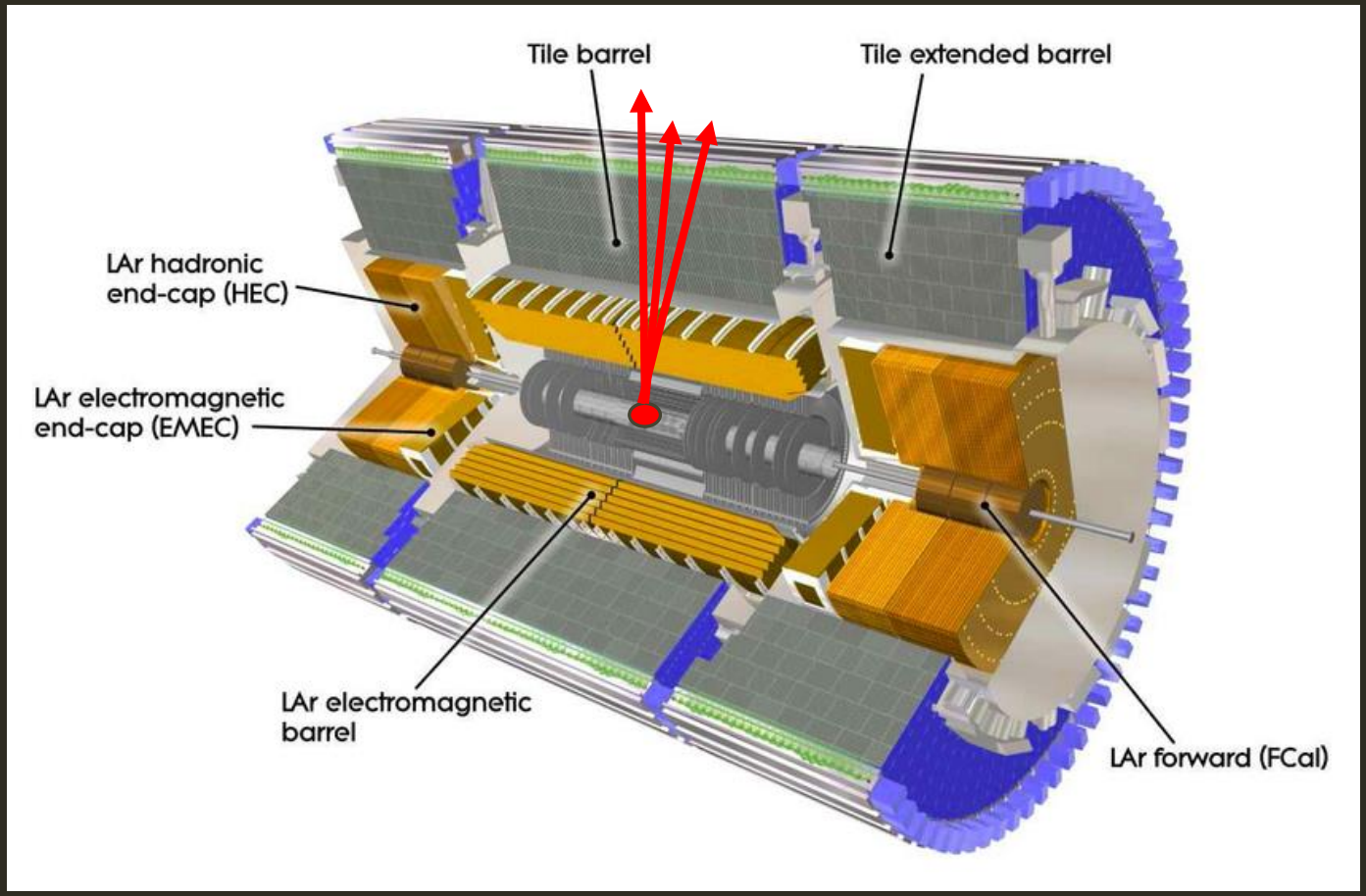
SIGNAL

Standard Model

The ATLAS Calorimeter

Measures energy deposited from particle jets

- Originates from Interaction Point
- Leaves tracks, calorimeter energy, etc.



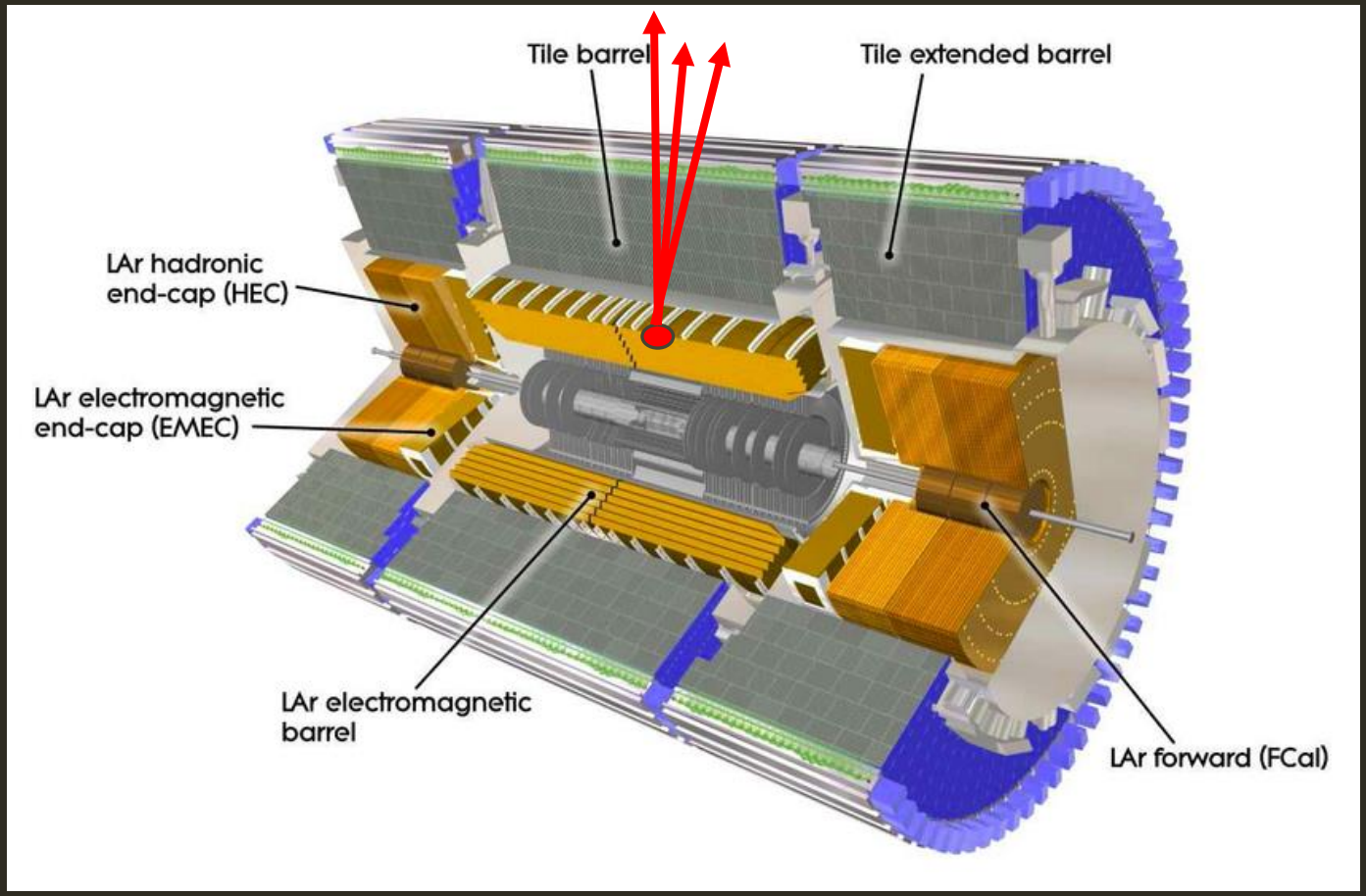
SIGNAL

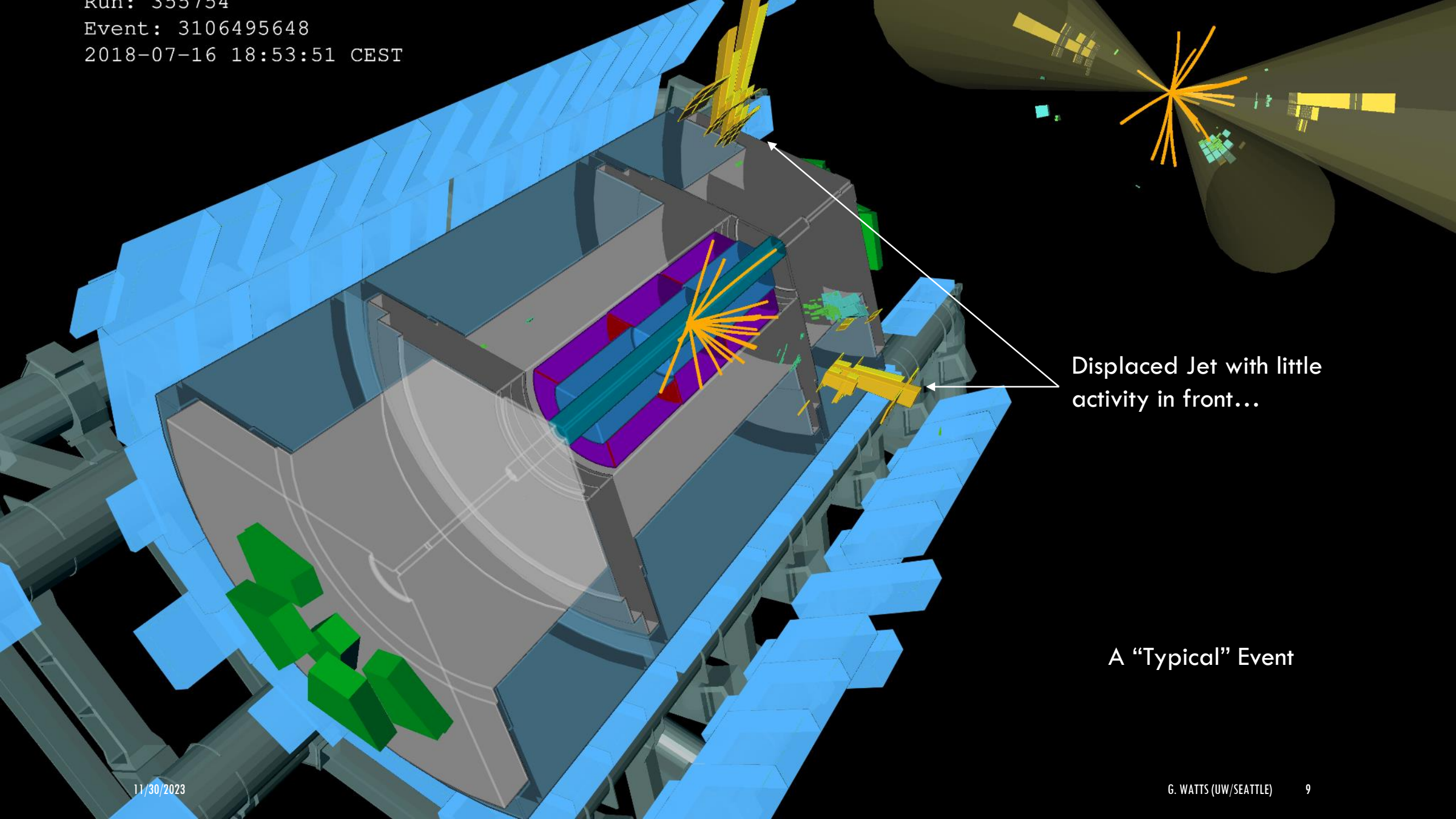
The ATLAS Calorimeter

Measures energy deposited from particle jets

- Displaced from Interaction Point
- Little activity between IP and the displaced jet

Long Lived Particle





A “Typical” Event

IRREDUCIBLE BACKGROUNDS

This caused all the trouble!

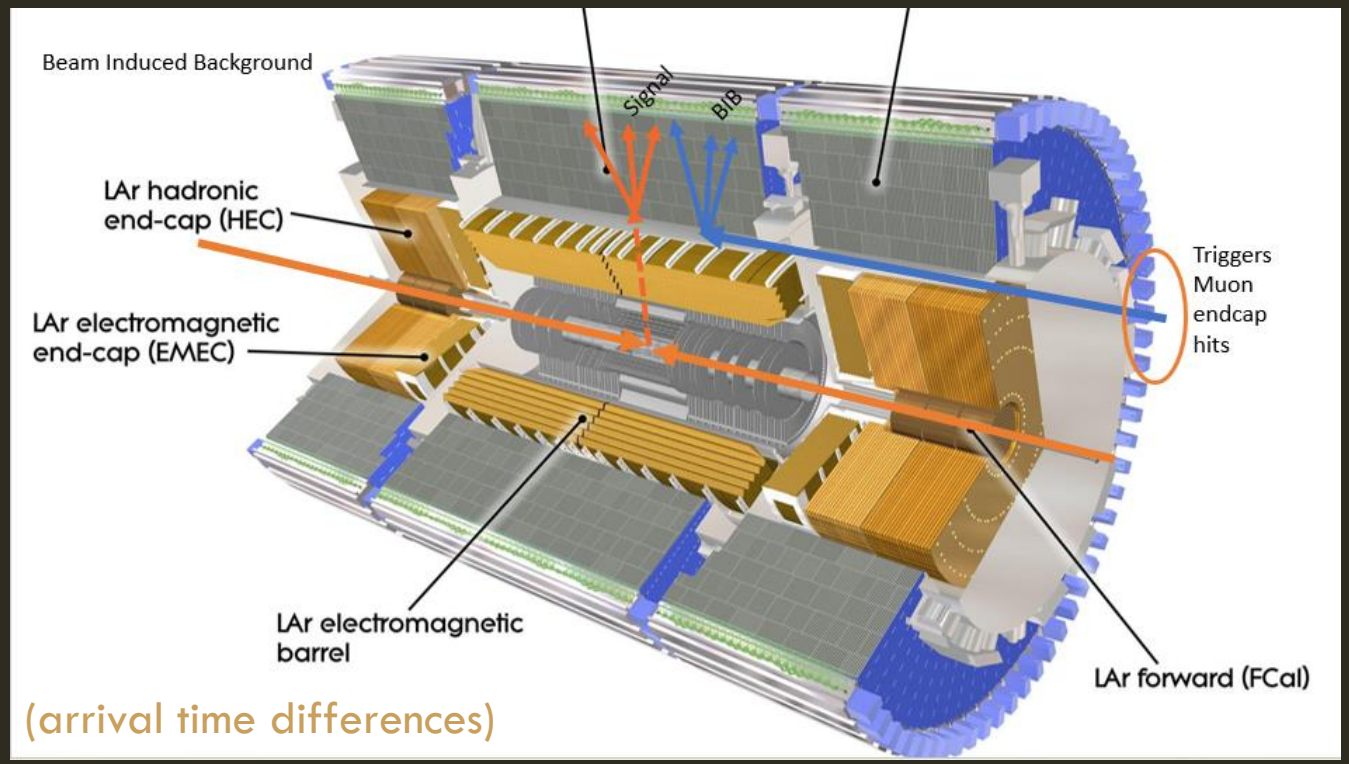
Standard Model MultiJet Background (Multijet)

Cross section of SM Jets is about 10^{10}

Our signal is $\sim 10^{-2}$

Regular fluctuations in jet evolution will make a signal-like jet every now and then!

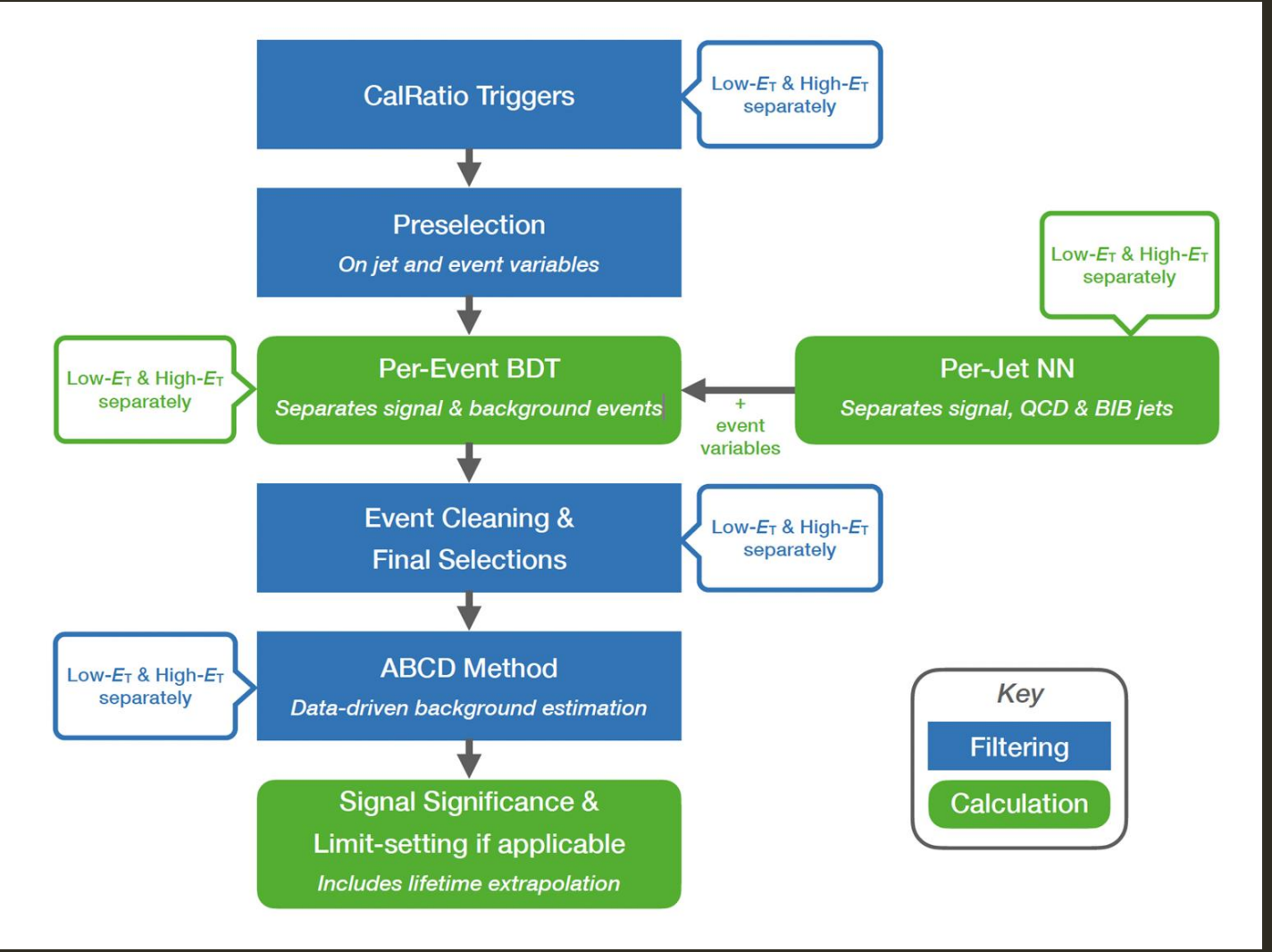
Beam Induced Background (BIB)



(arrival time differences)

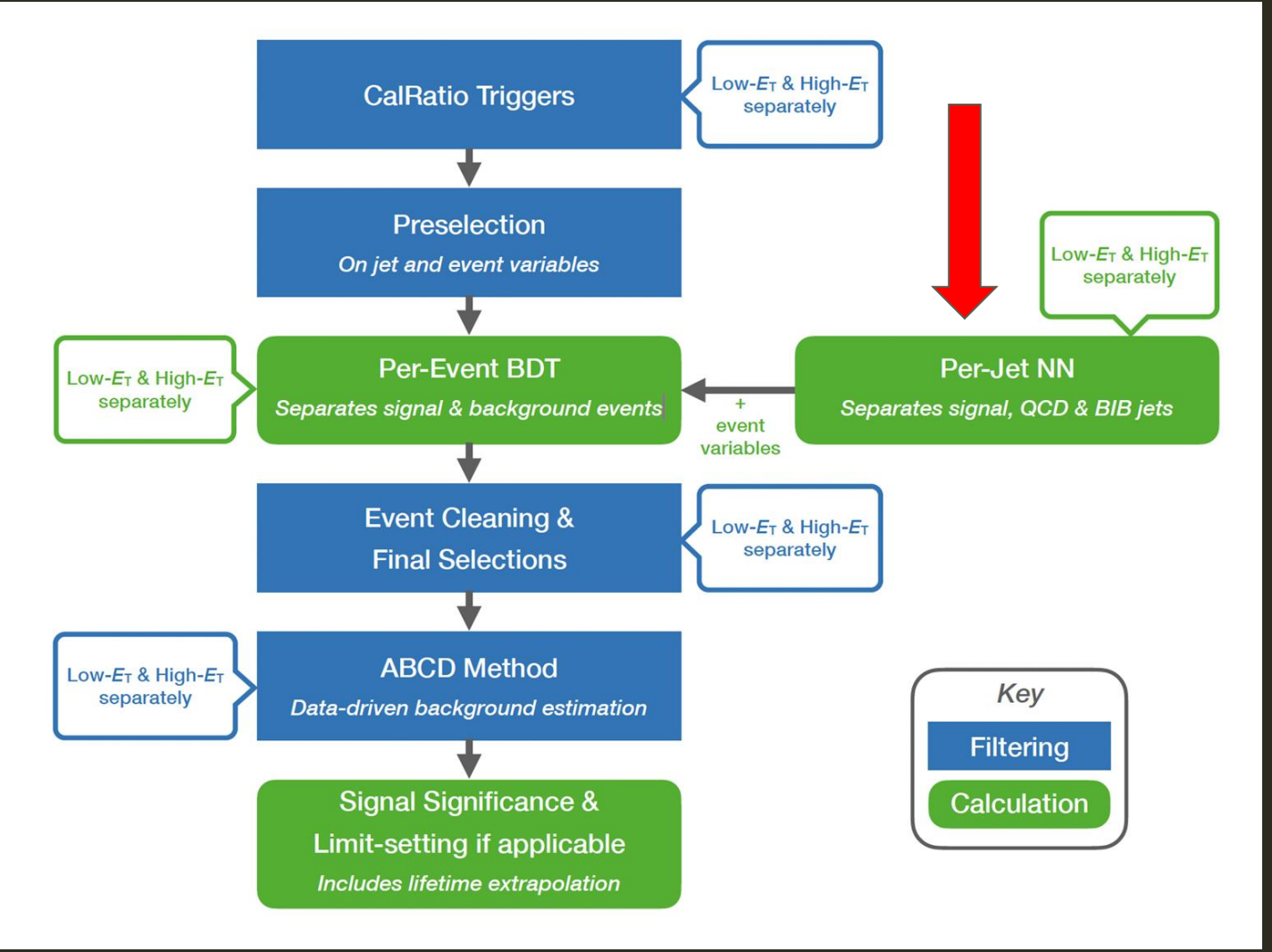
ANALYSIS WORKFLOW

- Analysis Follows Standard HEP pattern
- After triggers, Per-Jet NN classifies each jet
  - Beam Induced Background, Multi-jet Background, or Signal
- The scoring from the jets is combined by a Per-Event BDT
- And final selections are fed to standard limit setting algorithms

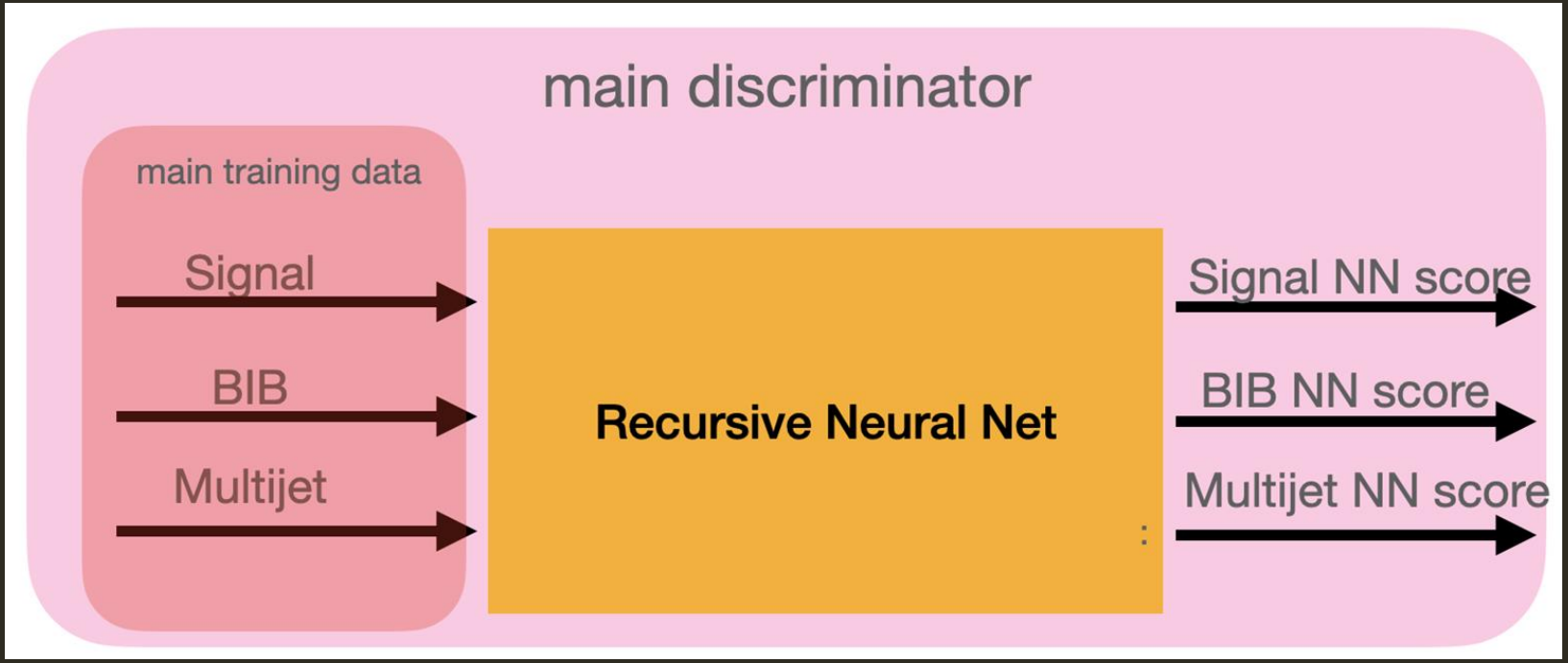


THE PER-JET NN – WHERE THE PROBLEM IS

- Per-Jet NN powers most of the signal/background discrimination.
- Using a **recurrent neural net** as one part of the process
  - Uses recurrent connections to capture and utilize information from previous steps in the sequence. Each step is a physics object!
- Data in the network has already gone through various processing steps
  - Eliminating and selecting most relevant data
  - Various pre-selection and processing

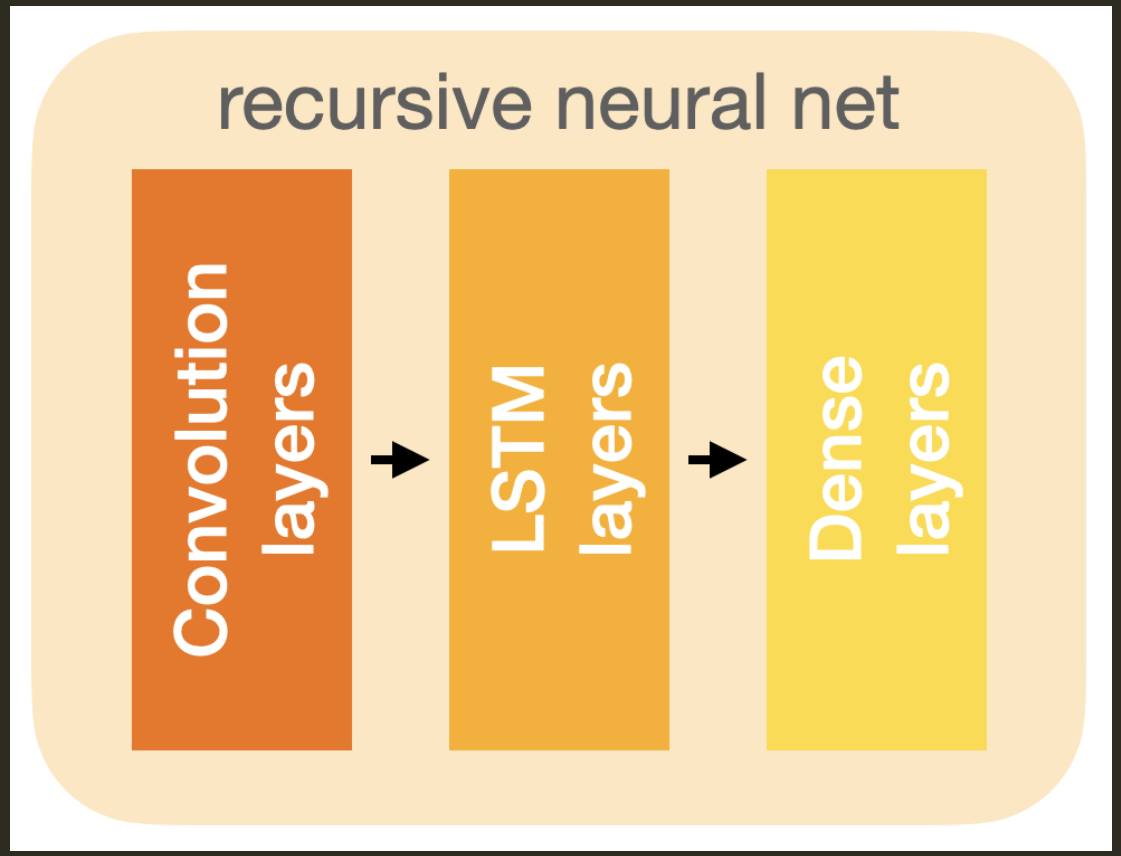


THE PER-JET NN



- Uses topo-clusters, muon segments, and track information in training, along with jet and LLP data
- Neural net architecture is a set of convolutional layers feeding into an LSTM layer
- Optimal hyperparameters achieved through grid search

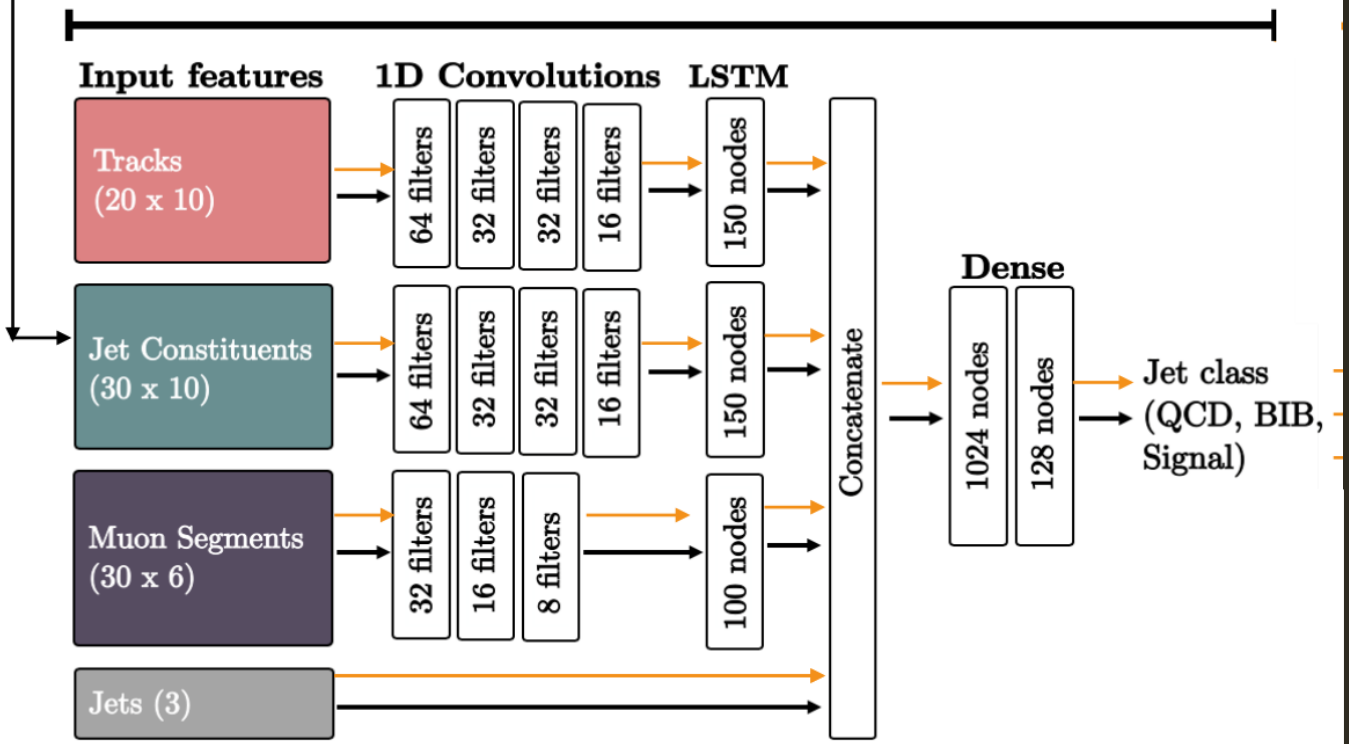
RNN DESIGN



- Convolutional neural network
  - Feature extractor
  - Exploits correlations between input variables
- LSTM network
  - Memory remembers information between subsequent inputs
  - Exploits correlations between cluster/tracks/muon segments
- Dense network
  - Information concatenated with jet inputs
  - Outputs predictions on classification

← Signal+QCD+BIB

Main Jet NN



OPS!

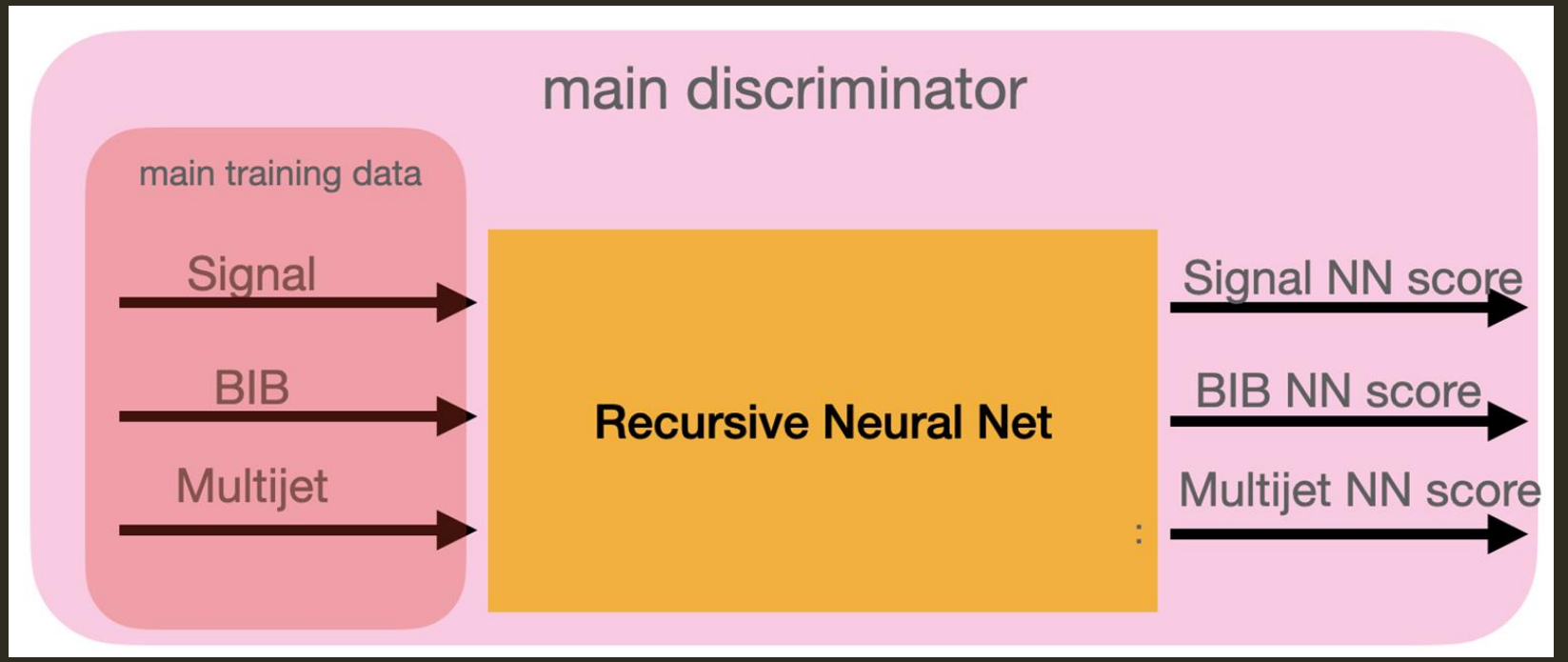
Our (internal) closure plots were not working!

Eventually we traced it to a MC/Data difference.

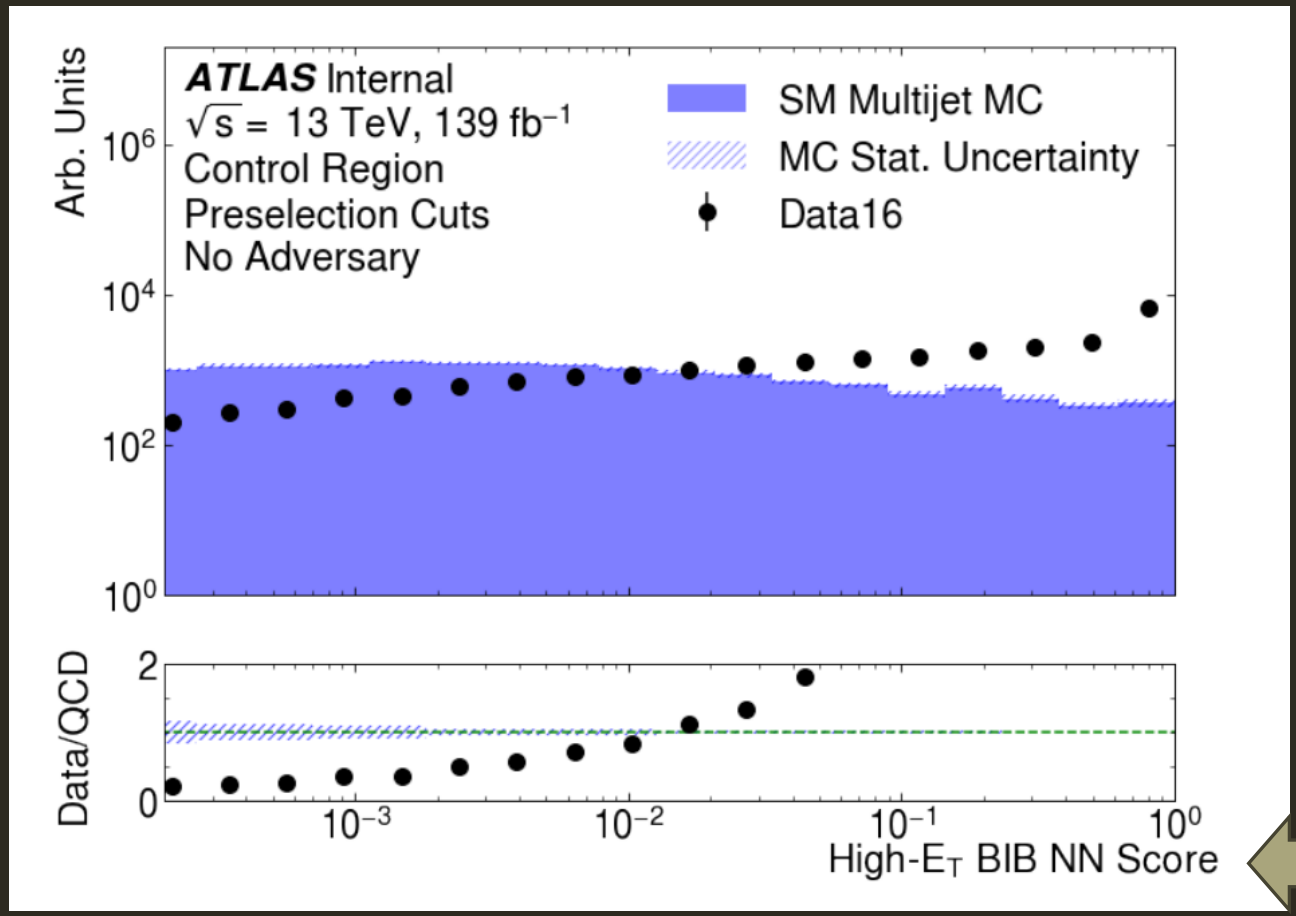
MC

Data

MC



OPS!



Canonical HEP Data/MC Comparison

NN Output

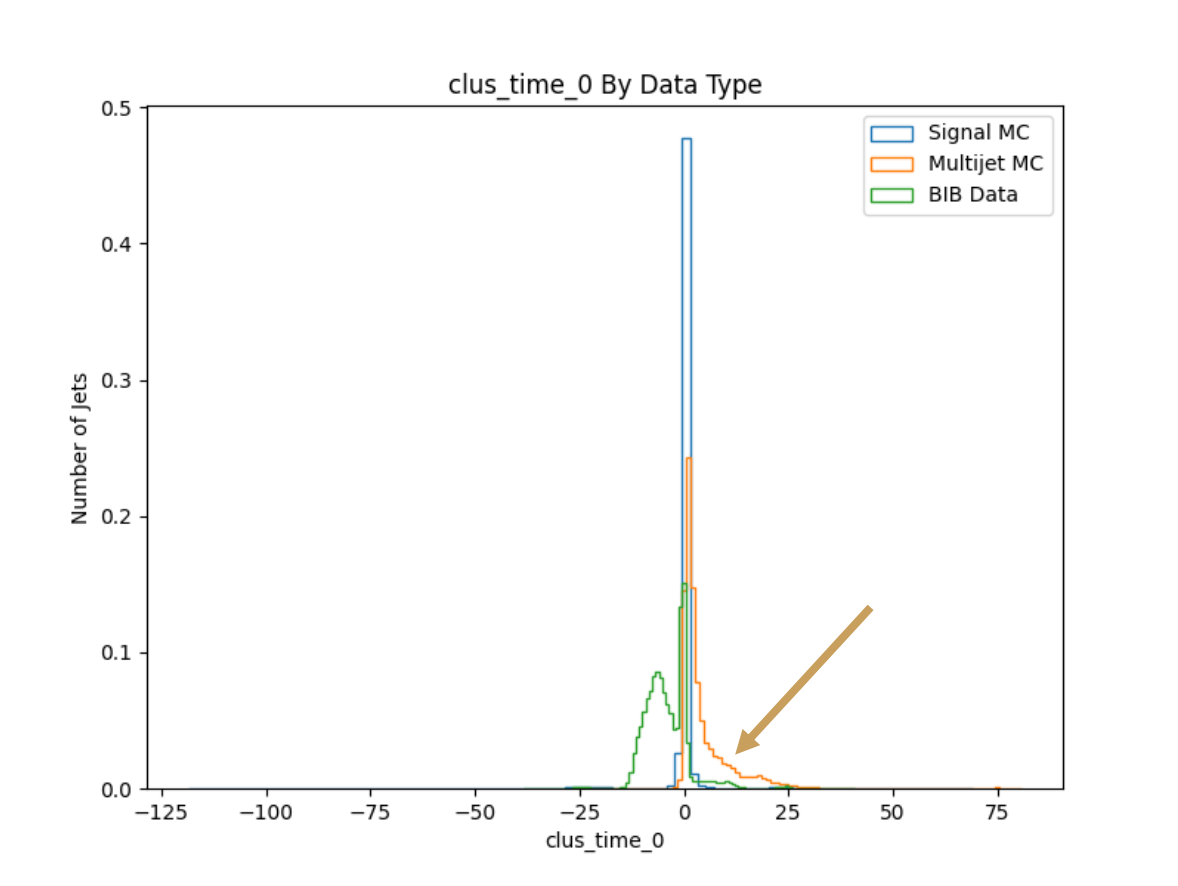
OPS!

The problem was in the simulation of the cluster timing for the calorimeter.

1) Remove this entirely from the training inputs
2) Correct it block-wise
3) Find a cleverer approach

Problem: Timing is a vital way to discriminate BIB from signal and background.

Problem: Even trained only on MC (if we could), it would make a mistake on the actual data!*



* Since the background calculate is data-driven, the error would have only been on the sensitivity calculation

A WAY FORWARD

➡ Prevent the RNN from learning Data/MC features

Build an adversary that tries to tell the difference between Data and MC solely from the 3 outputs of the RNN.

But we need to do this on a sample that should be otherwise identical!

Can't use the signal and background!
Need a new Control Sample!

A WAY FORWARD

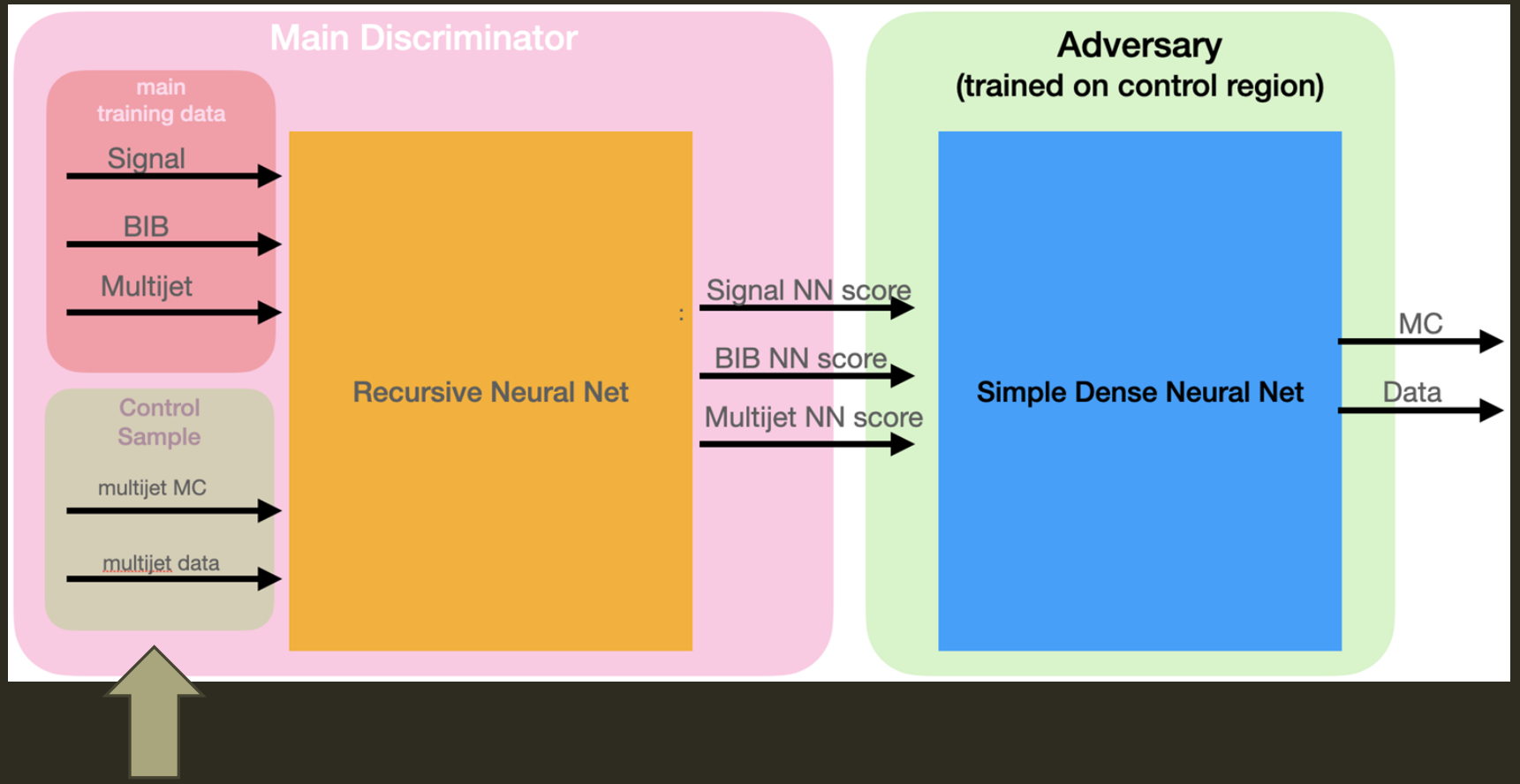
➡ Prevent the RNN from learning Data/MC features

But we need to do this on a sample that should be otherwise identical!

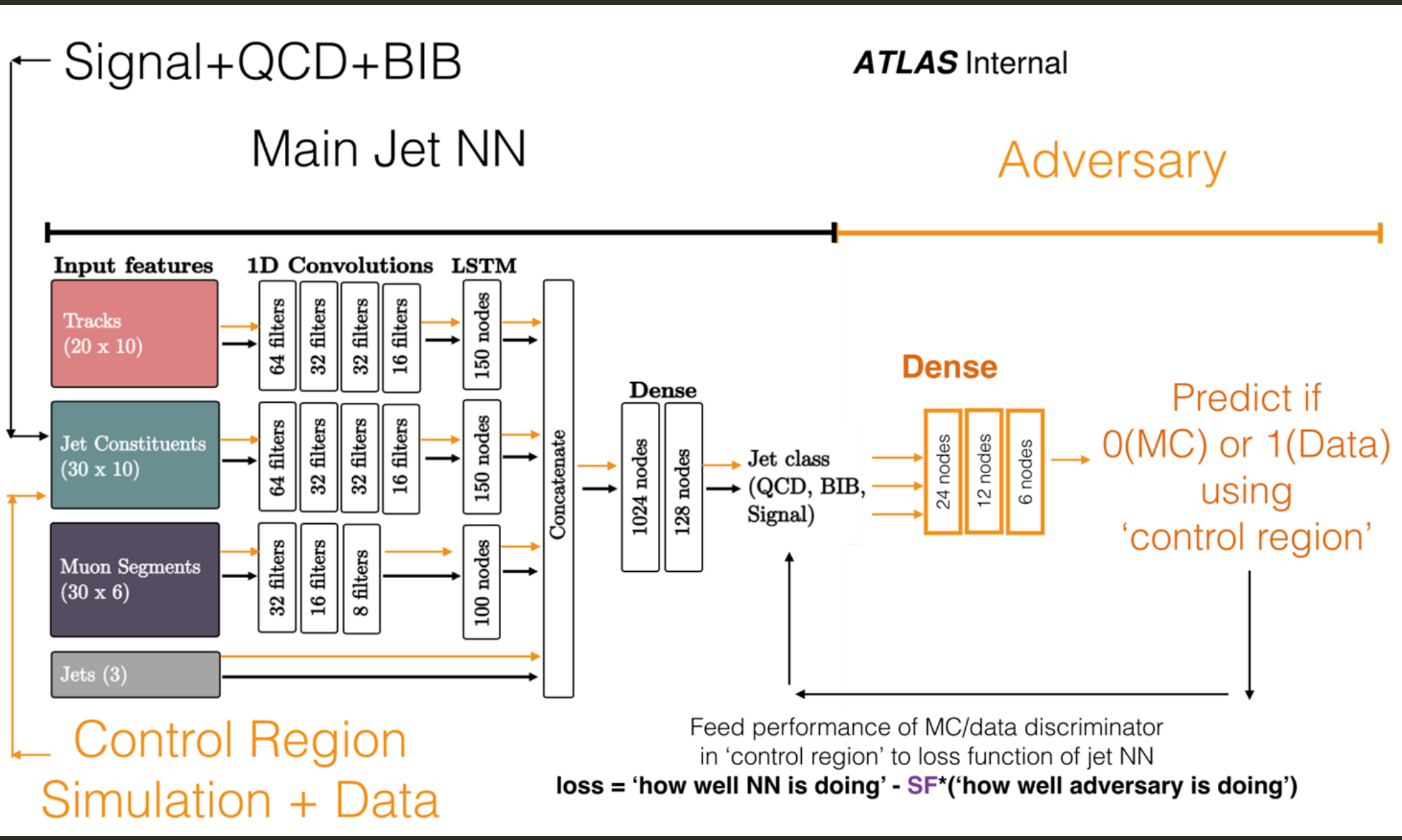
Can't use the signal and background!
Need a new Control Sample!

1) Control sample should have no signal
2) Controls sample should have the same NN score!
3) Control sample must be well modeled otherwise in both MC and Data

THE ADVERSARY



Anything other than 50-50 is fed back into the loss function for the main network training.

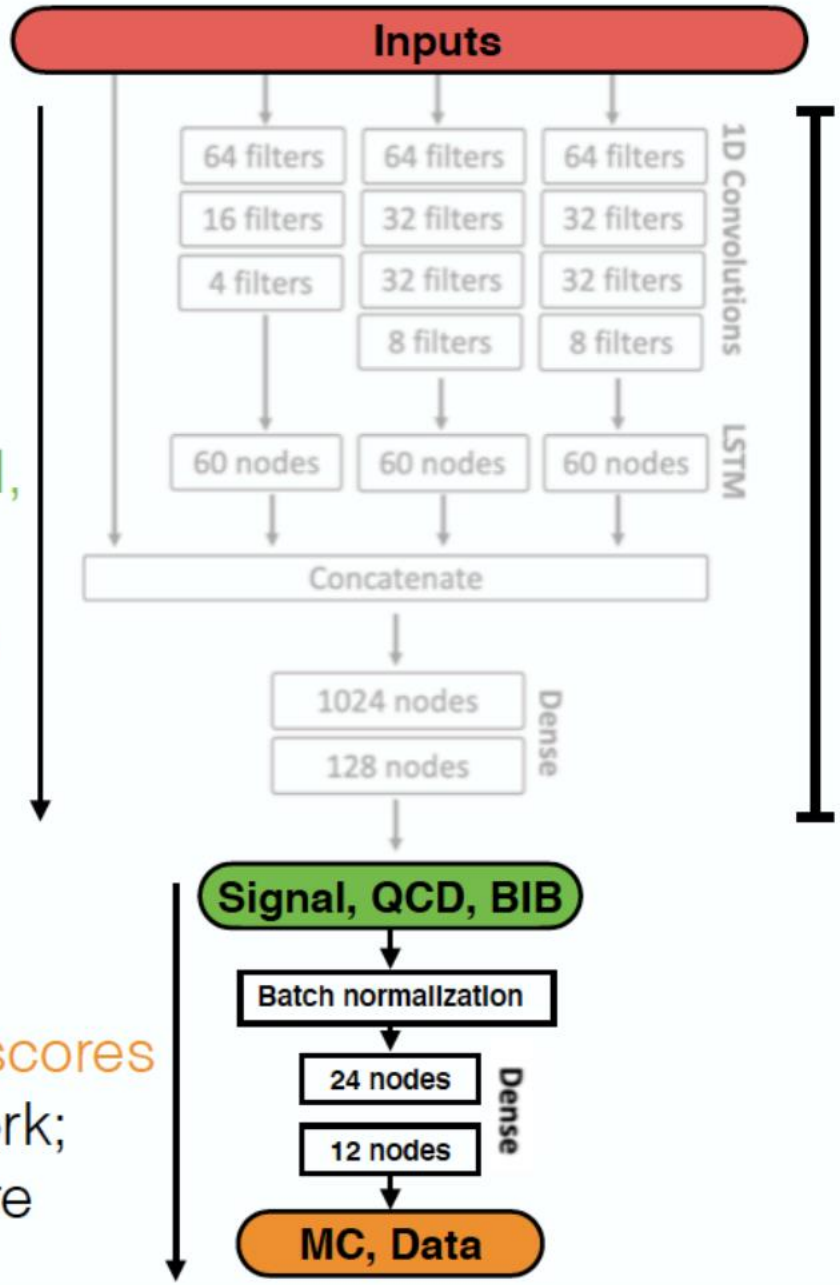

Signal+QCD+BIB
ATLAS Internal
Main Jet NN
Adversary
Input features
1D Convolutions
LSTM
Tracks (20 x 10)
Jet Constituents (30 x 10)
Muon Segments (30 x 6)
Jets (3)
64 filters
32 filters
32 filters
16 filters
150 nodes
64 filters
32 filters
32 filters
16 filters
150 nodes
32 filters
16 filters
8 filters
100 nodes
Concatenate
Dense
1024 nodes
128 nodes
Jet class (QCD, BIB, Signal)
Dense
24 nodes
12 nodes
6 nodes
Predict if 0(MC) or 1(Data) using 'control region'
Control Region
Simulation + Data
Feed performance of MC/data discriminator in 'control region' to loss function of jet NN
loss = 'how well NN is doing' - SF*('how well adversary is doing')

HOW DOES THE TRAINING WORK?

3 steps!

1. Adversary step

QCD, Data from control region



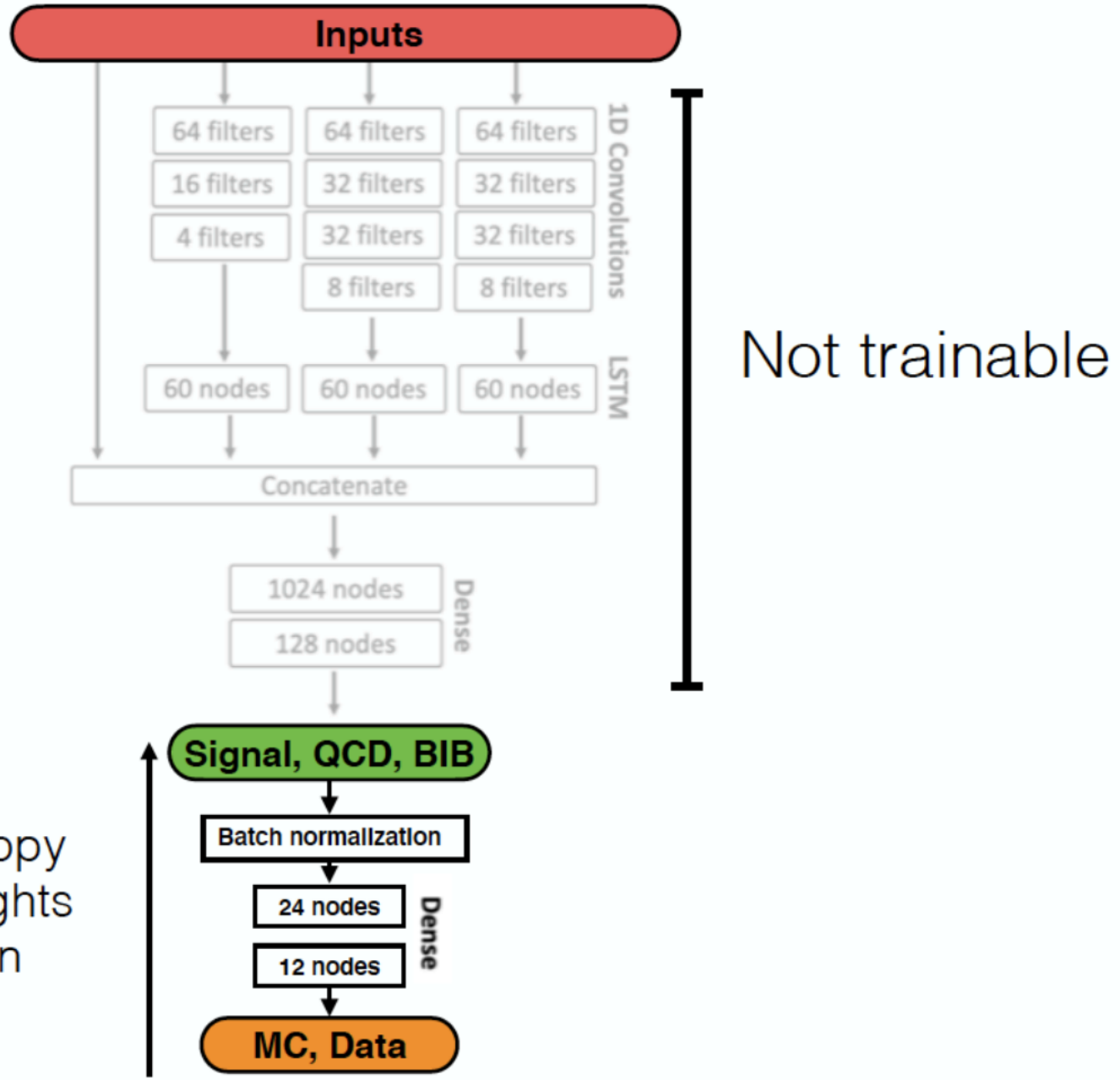
Calculates Signal, QCD, BIB scores for QCD, Data in control region

Not trainable

Calculates MC, Data scores in adversary network; trains weights here

1. Adversary step

QCD, Data
from control region



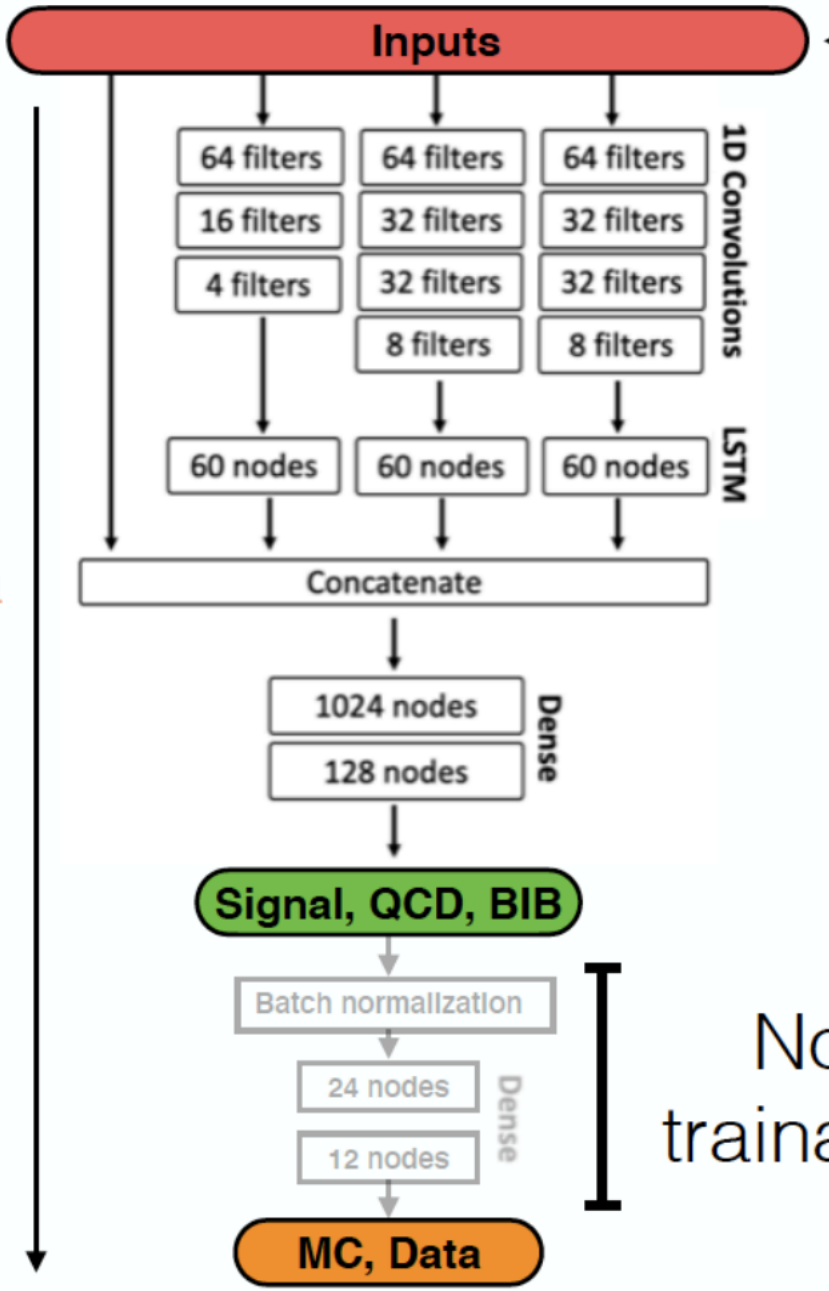
Not trainable

Backpropagate
positive loss
of MC, Data cross entropy
such that trainable weights
differentiate between
QCD, Data

2. Main step

QCD, Data from control region

Signal, QCD, BIB



Calculates MC, Data scores for QCD, Data in control region

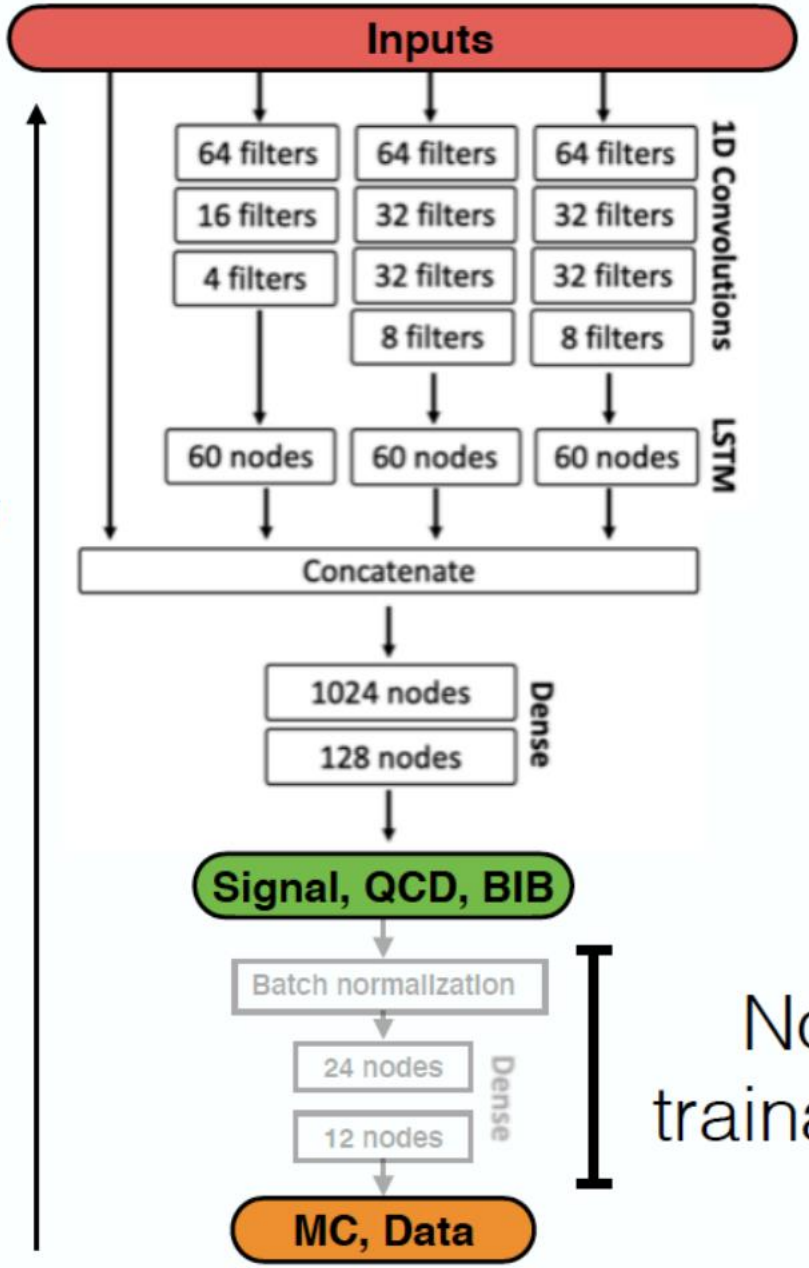
Calculates Signal, QCD, BIB scores for Signal, QCD, BIB inputs

Not trainable

2. Main step

QCD, Data
from control region

Signal, QCD, BIB

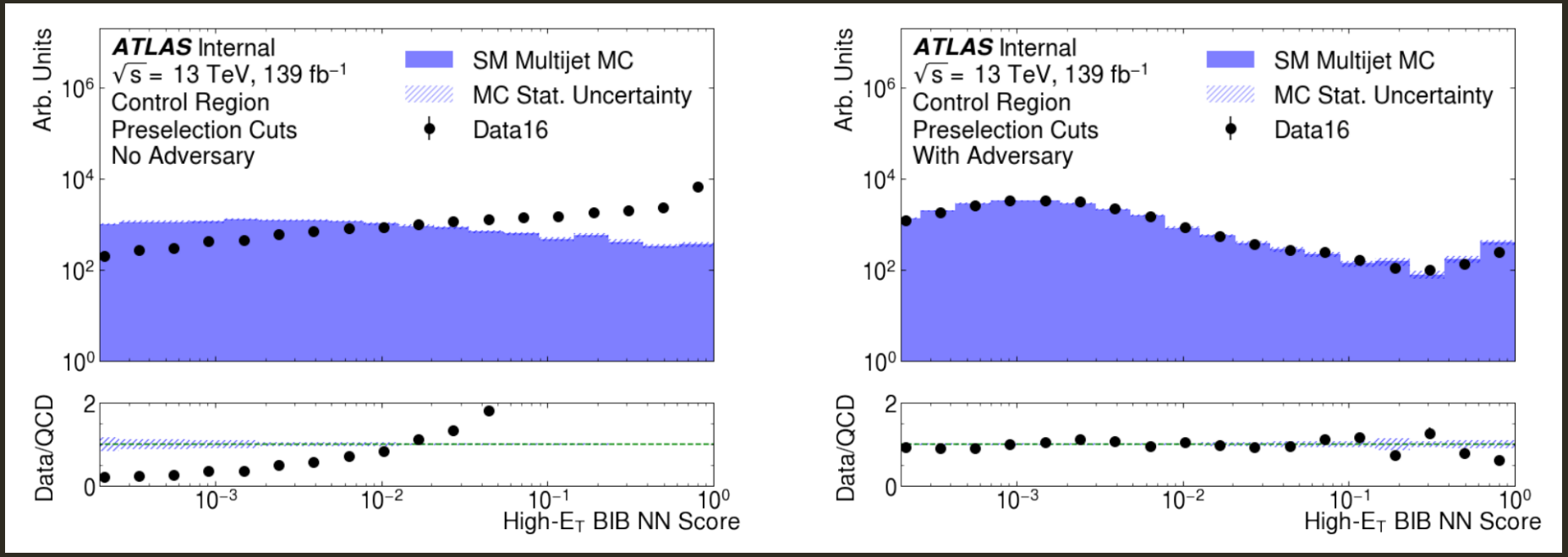


Backpropagate
negative loss
of MC, Data cross entropy
such that trainable
weights
do not differentiate
between
QCD, Data

Backpropagate
positive loss of
Signal, QCD, BIB
cross entropy
such that trainable
weights
differentiate
between Signal,
QCD, BIB

Not
trainable

NN OUTPUT IS MUCH BETTER REPRESENTED



CONCLUSIONS

There is evidence this reduced the systematics in other areas of modeling as well

- Cluster distributions seem to be a little better
- We started thinking about this like in-situ calibrations, or simultaneous fits in a profile-likelihood.

Overall, the analysis was some 35% more sensitive than the previous version

- Previous version used a BDT for the per-jet NN
- The timing was still an issue – but the BDT was not sensitive enough to really bring out the error!

The mechanics were complex

- Training frameworks aren't really built for this.

As a side plea

- Moving this code from a graduate student's experiment to production level took almost 6 months!