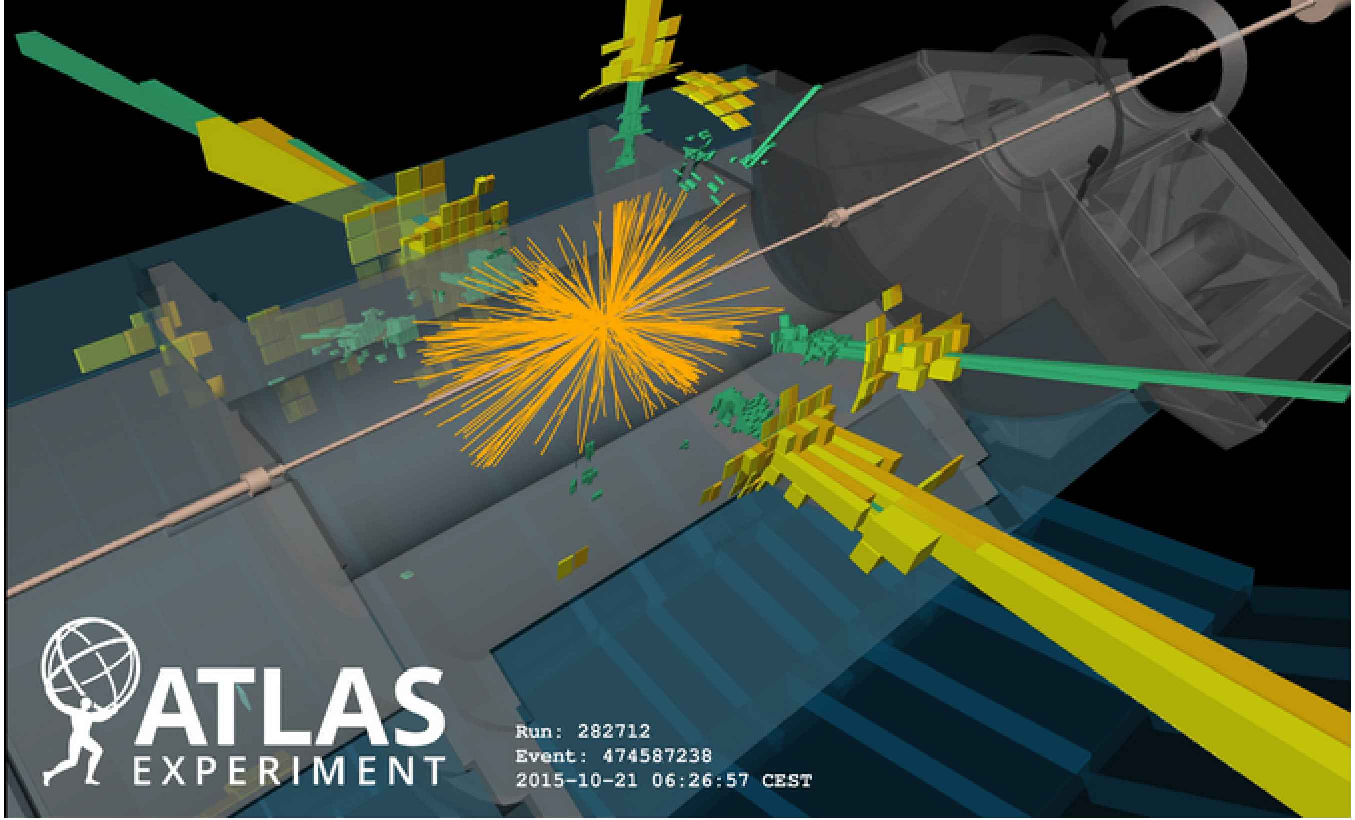
Hadronic jets E and M calibration with DNN

P-A Delsart

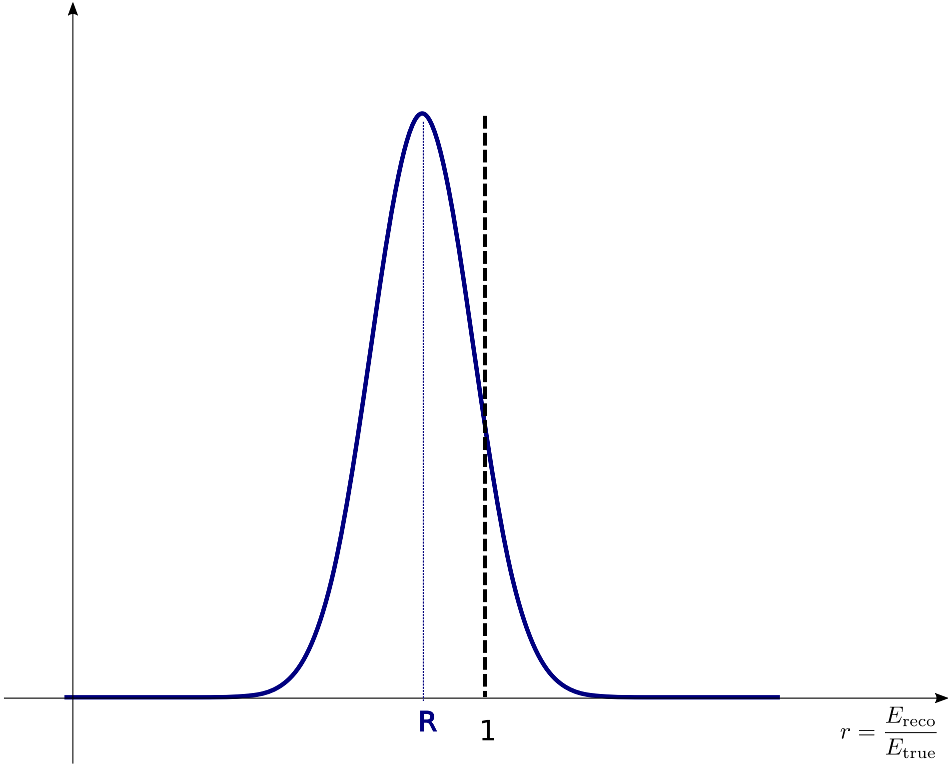
Jet Calibration



- Hadronic jets detected by ATLAS need to be **calibrated**
- Developed a DNN-based method to simultaneously calibrate jet E and mass
- Continuing pioneer work from this PUB Note

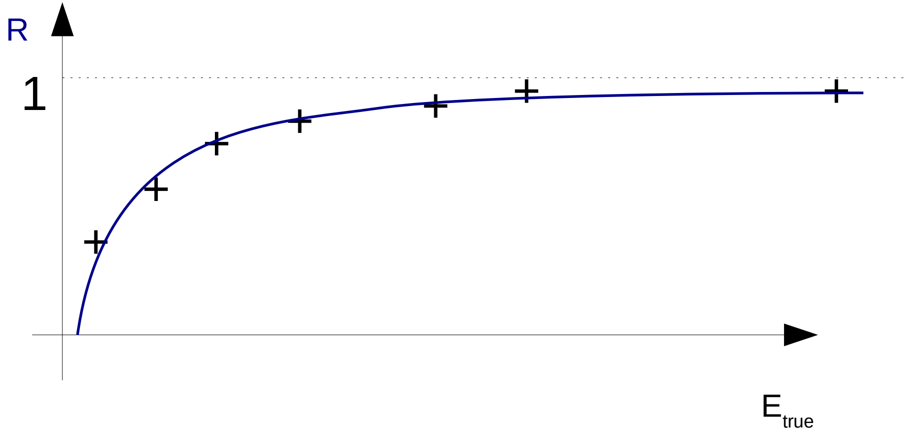
Introduction : jet response

- Particle jets with a given E_{true} are reconstructed with a E_{reco} distribution
- Usually, discuss jet E response
  - $r = E_{reco}/E_{true}$: individual response for 1 jet
  - R = mode of r distrib, the "response" at E_{true} ↔ Jet Energy Scale (JES)



Introduction : jet response

- Particle jets with a given E_{true} are reconstructed with a E_{reco} distribution
- Usually, discuss jet E response
  - $r=E_{reco}/E_{true}$: individual response for 1 jet
  - R = mode of r distrib, the “response” at E_{true}
- R depends on E_{true} … and other parameters : η, m, $EM_{fraction}$,...



Introduction : jet calibration

- Goal : find the correction factor **C** defining E_{calib}=C E_{reco}

  Such as : $\mathbf{R_{calib}}$ = mode(r_{calib}) = **1**

- C must depend on reconstructed quantities E_{reco},η_{reco}, m_{reco}, ….
- We want to calibrate E and mass at the same time. So we need a function

  $$C : \mathbb{R}^N \rightarrow \mathbb{R}^2$$

- Looks like a regression problem...

Calibration with DNN : difficulties

Basic idea : regress R_E and R_{mass} vs (E,mass,η,etc...)

- Need to learn the **mode** of the targets, not the targets
  - Use dedicated loss functions
- R varies strongly vs η because of the detector structure (calorimeter boundaries)
  - Hard to model sharp variations → use “input annotation”
- ...

Learning the mode

How to learn the mode of the response

- Can not use any loss function
  - MSE loss $||r_{pred}-r_{true}||^2$ → NN learns the **mean**
  - Bias when r distrib is asymmetric
- Considering 2 approaches
  - Leaky Gaussian Kernel (LGK) (introduced here)
    - Mode exactly learned by $\delta(y-y_{pred})$ (Dirac function)
    - LGK is a surrogate function : $$\text{LGK loss} = \exp\left(-\frac{(r_{\text{target}} - r_{\text{pred}})^2}{2\alpha}\right) + \beta|r_{\text{target}} - r_{\text{pred}}|$$
    - Fixed parameters α,β ~1e-3
  - Mixture Density Network (MDN)

MDN loss

- Goal is to "predict" the distribution of response given E_{true}
- First, assume distrib is gaussian : $P(r|E_{\text{true}}) = g(r|\mu(E_{\text{true}}), \sigma(E_{\text{true}}))$
- Thus the NN must learn μ and σ
- Optimal μ and σ are obtain when maximizing the likelihood : $\prod_{i \in \text{inputs}} P(r^i|E^i_{\text{true}})$
- In practice :
  - have the NN predicts **μ and σ**
  - choose the **log likelihood as the loss**
- $$\text{loss}((\mu, \sigma), r_{\text{target}}) = \log(\sigma) + \frac{1}{2}(\frac{\mu - r_{\text{target}}}{\sigma})^2$$

MDN loss real case

- But real distributions are not gaussian !
- We can assume the core of distribution are ~gaussian
  - Core is what matters : we want the **mode**
- Proceed as follows :
  - Start training the NN until reasonable μ and σ are predicted
    - Typically, 1 or 2 epochs are enough
  - Replace gaussian in previous formula by truncated gaussian at Nσ (ex: N=3 or N=1)
  - Continue training, possibly reducing N from time to time

How to learn the mode of the response

- LGK loss

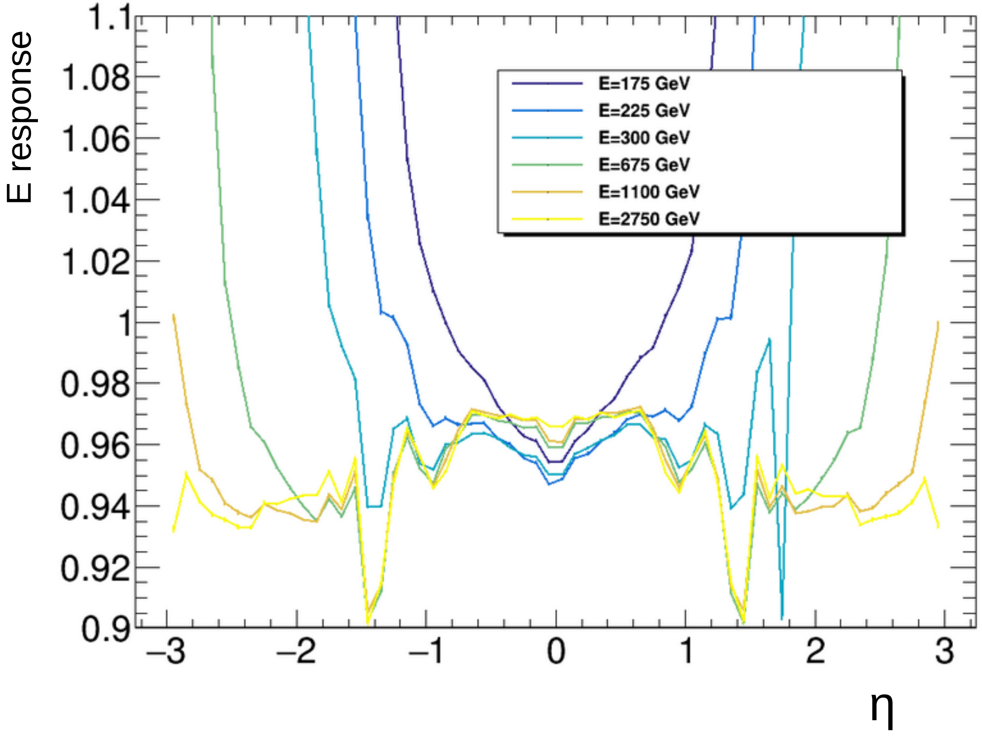
$$\text{LGK loss} = \exp(-\frac{(r_{\text{target}} - r_{\text{pred}})^2}{2\alpha}) + \beta|r_{\text{target}} - r_{\text{pred}}|$$

- MDN loss
  - prediction = (μ,σ)

$$\text{loss}((\mu,\sigma), r_{\text{target}}) = \log(\sigma) + \frac{1}{2}(\frac{\mu - r_{\text{target}}}{\sigma})^2$$

Learning the η structure

What do we expect ?

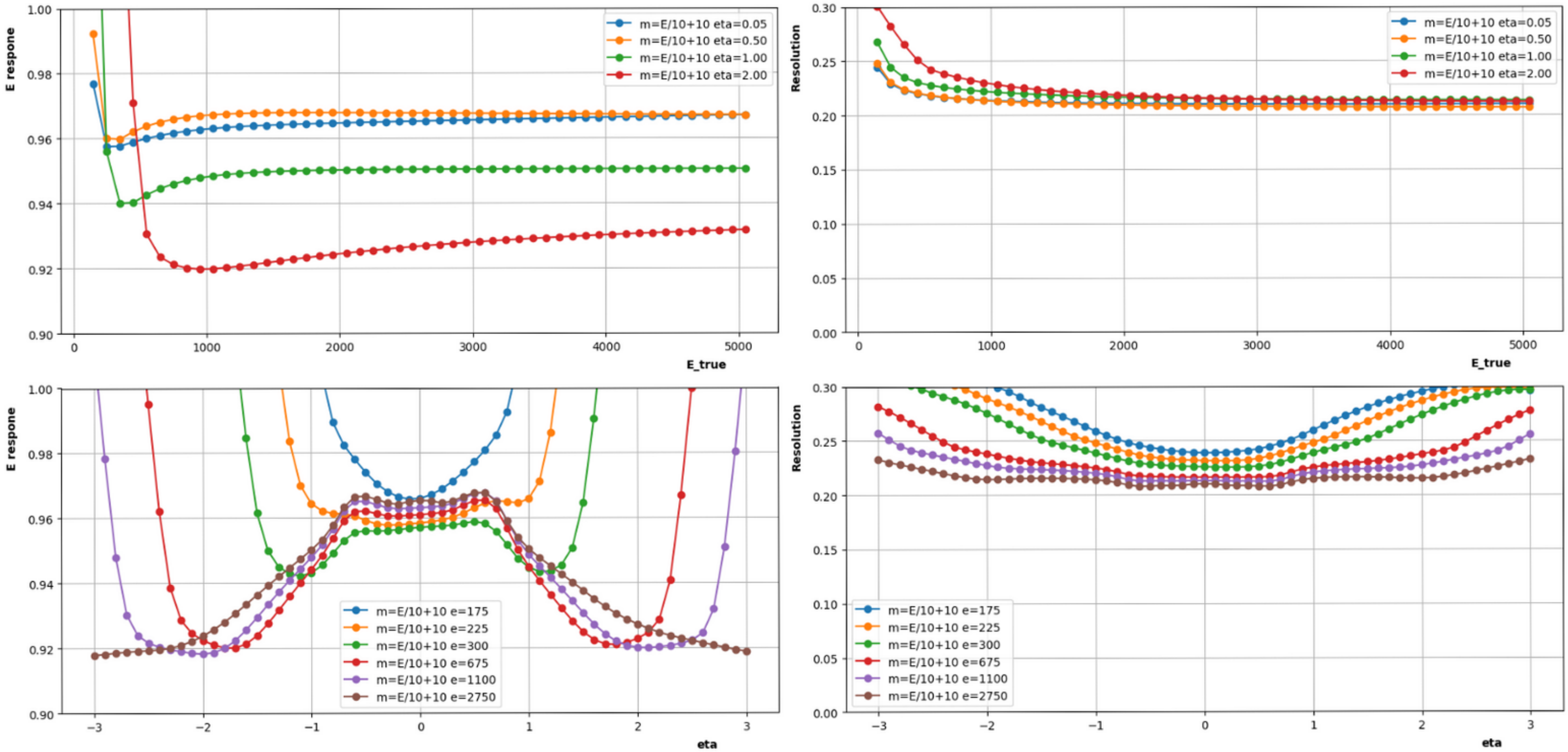


- E response calculated “manually”
  - In bins of (E,eta) as in standard EtaJES
    - Thus ignoring dependencies on mass & NPV
  - Fit distribution in each of these bins → obtain R
  - Plot R vs η for a few E bins

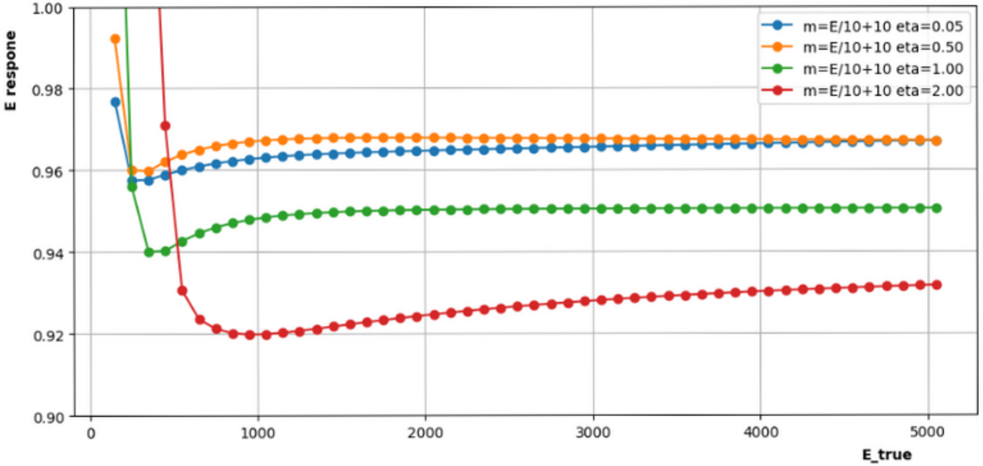
NN prediction for response and resolution

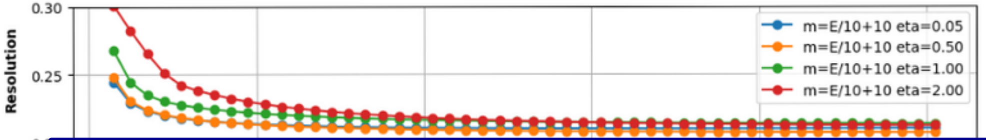
- Learn just E Response with MDN loss
  - Thus also predicting E resolution
  - Details in following slides
- Try to replicate previous plot

NN prediction for response and resolution



NN prediction for response and resolution

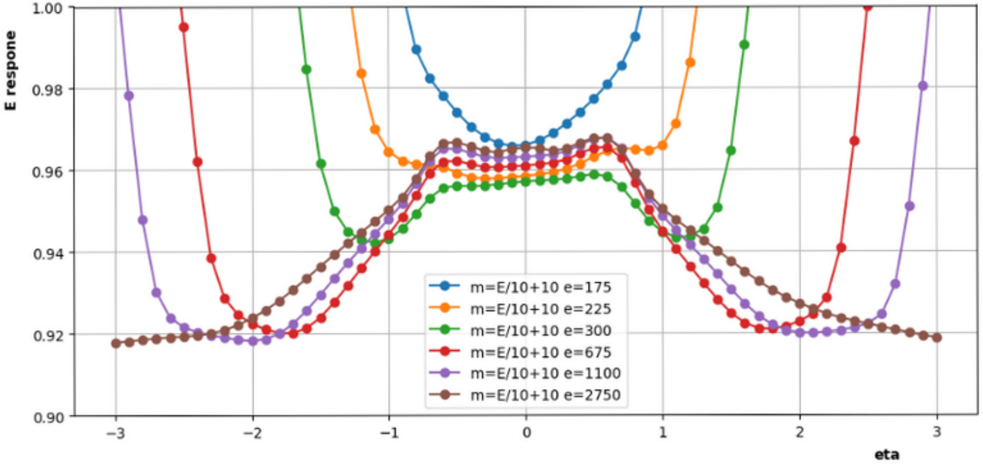


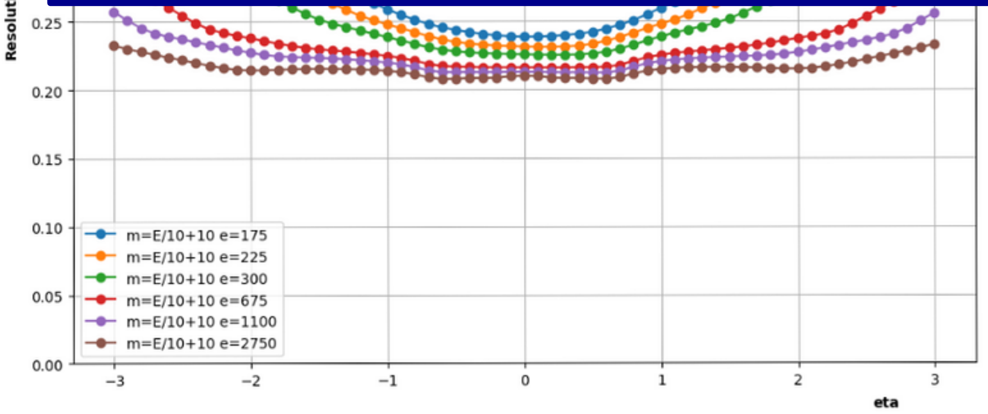


- Response shape are roughly correct

But :

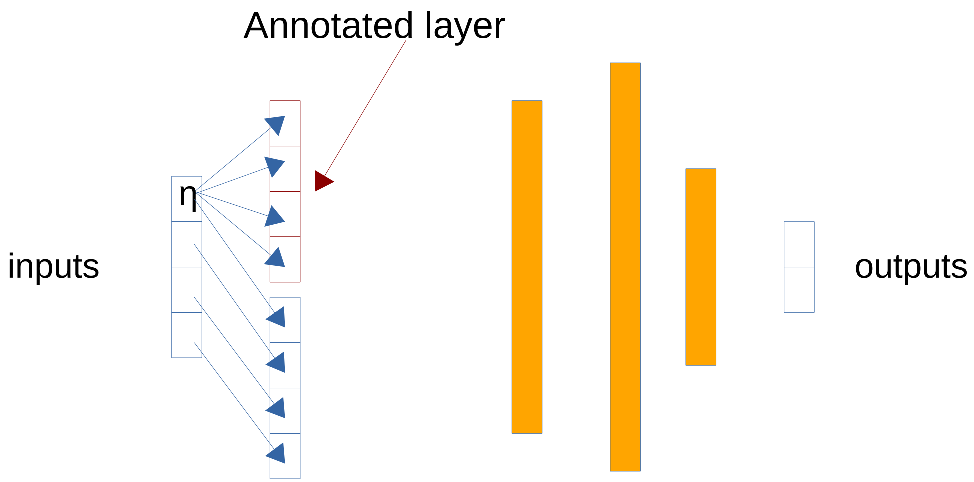
- Missing fine structure in eta
- Resolution much higher than expected





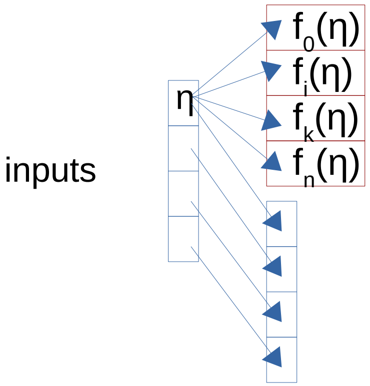
Input Annotation

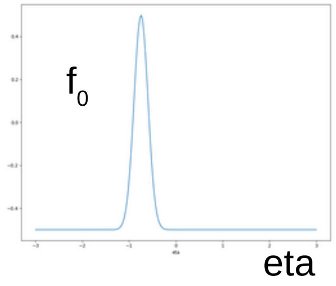
- Increase the input by adding “features”

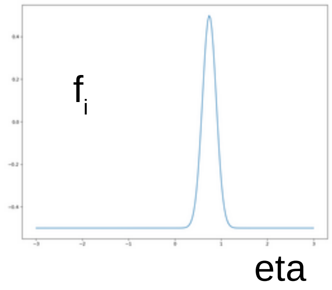


Input Annotation

- Increase the input by adding “features”







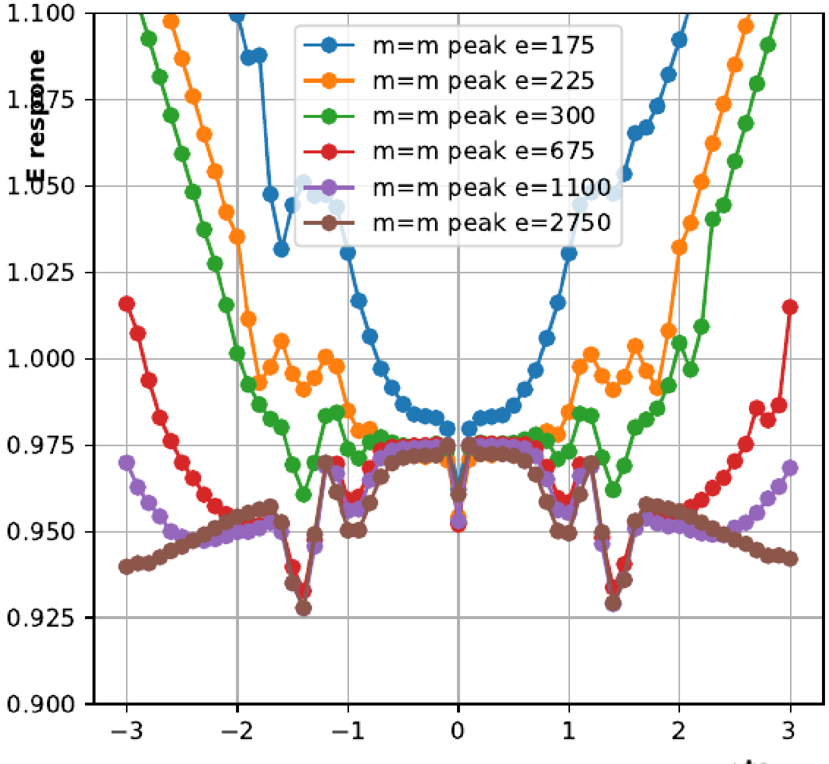
Gaussian Annotation

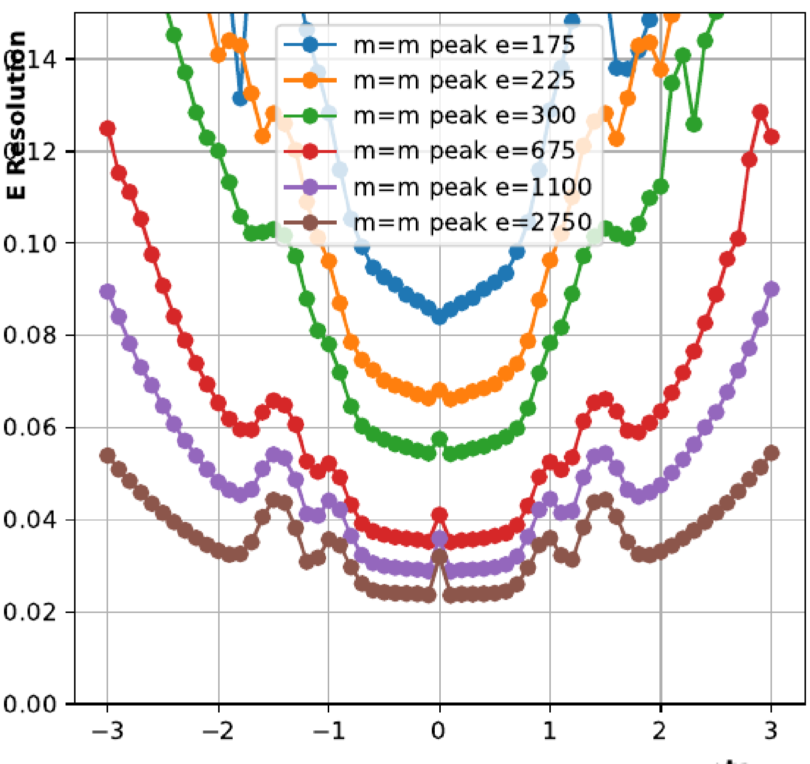
- Gaussian centers set on detector cracks

Intention : add the “distance to the crack” information to the NN

NN predictions with Input Annotation

- NN E response predictions
  - Recover the η structure of the response





NN implementation details

Input handling

- Use 260M simulated large-Radius jets
  - do not fit in memory
  - Custom solution :
    - Randomly place jets data in ~10M entries flat TTrees/TFiles
    - streaming inputs from files with uproot
  - Other suggestion of workflow ? Use TFRecord + protobuff files ?
- Features and targets normalization
  - Linear scaling to ~[-1,1]
  - Although targets are naturally ~1, normalization still important to avoid training unstability
    - Or use a final activation centred around 1 (like 1+tanh(x))

Input handling

- Use 260M simulated large-Radius jets
  - do not fit in memory
  - Custom solution :
    - Randomly place jets data in ~10M entries flat TTrees/TFiles
    - streaming inputs from files with uproot
  - Other suggestion of workflow ? Use TFRecord + protobu
- Features and targets normalization
  - Linear scaling to ~[-1,1]
  - Although targets are naturally ~1, normalization still impo unstability
    - Or use a final activation centered around 1 (like 1+tanh(x))

15 Input features :
E, mass, η

NPV, μ

EMFrac, EM3Frac, Tile0Frac

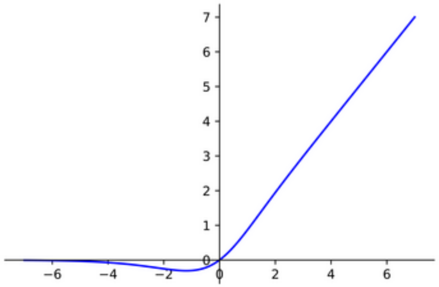
NeutralFrac,
EffNTracks (== $\Sigma(p_{Ttrack})^2/\Sigma(p_{Ttrack}^2)$)
SumPtTrkFrac

D2, Qw
EffNConst (==$\Sigma(p_T)^2/\Sigma(p_T^2)$),
GroomMratio (==$M_{softdrop}/M_{ungroomed}$)

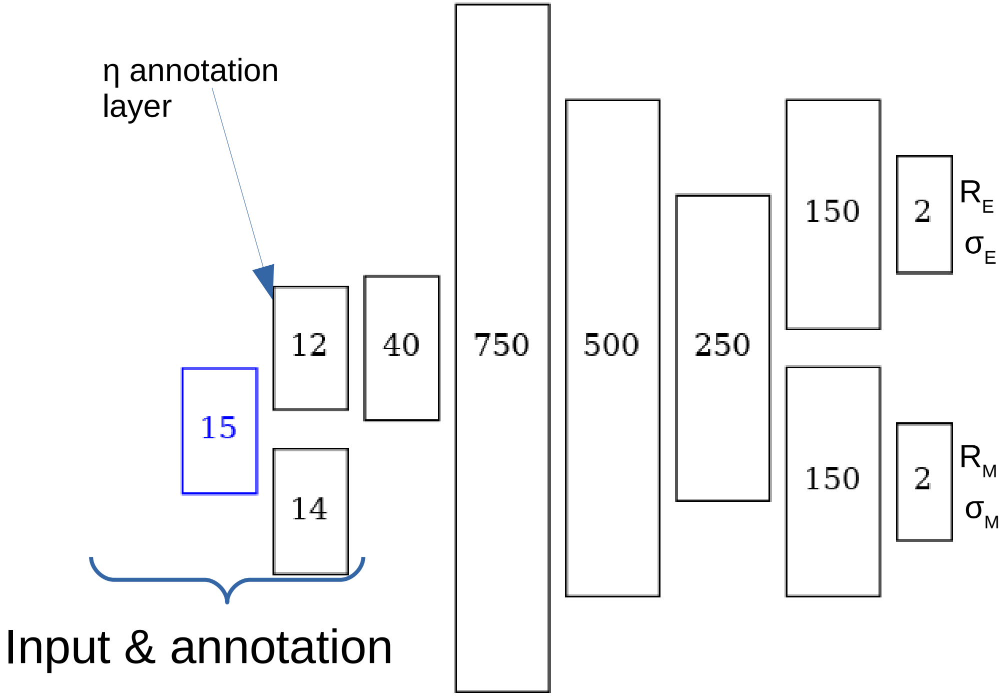
NN architectures

Framework : tensorflow+keras

- Tested various architectures
  - Including various layer sizes
  - MDN and/or LGK losses
- Presenting here 2 architectures with MDN loss, 1 with LGK loss
- Activation functions
  - Internal layers : mish (a smooth ReLU)
  - Last layer : tanh

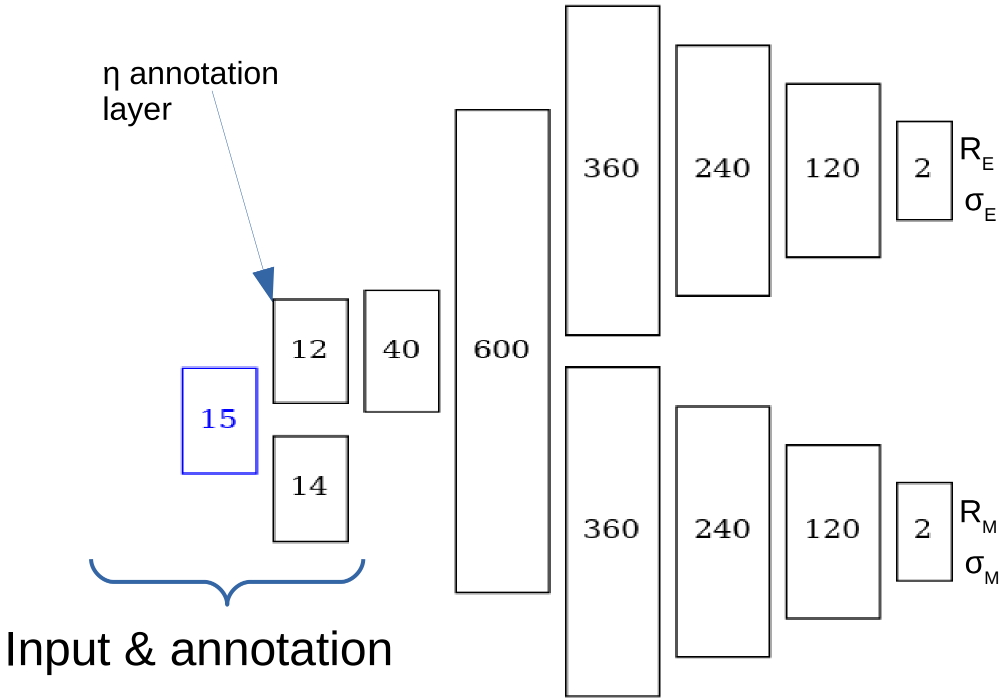


NN architectures



- Triangular shape (labelled T0 in following plots)
- ~620K trainable weights
- Fork at the tip allows
  - 2 Dedicated sets of weights for R_E and for R_M
  - 2 loss functions, tunable independently

NN architectures



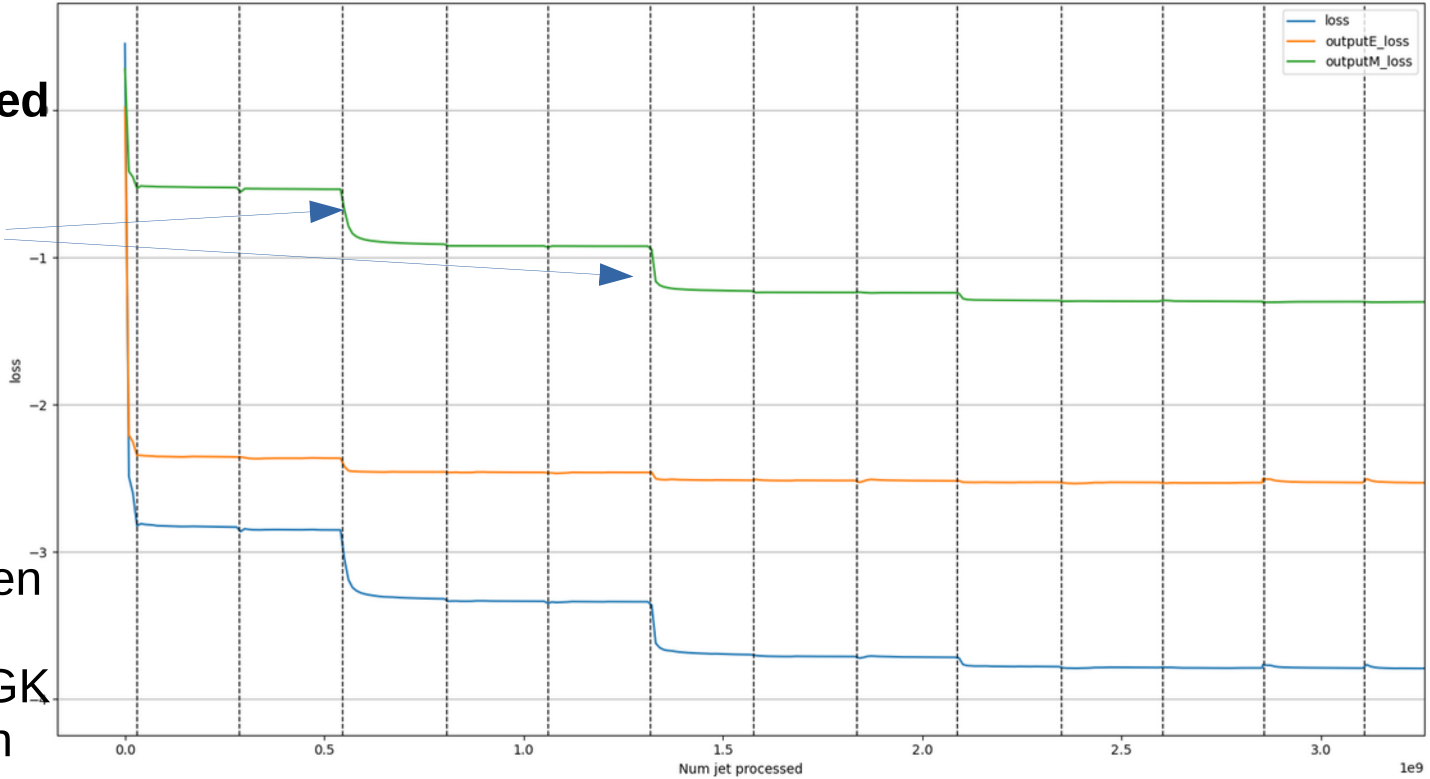
- Deep fork shape (labelled DeepF in following plots)
- ~690K trainable weights
- Much deeper fork :
  - 2 independent deep NN to predict R_E and R_M with a common base
- Architecture also tested for the LGK loss (labelled LGK)

NN training convergence

End of epoch

MDN loss vs njet processed

Steps due to change in the gaussian truncation in the MDN formula.

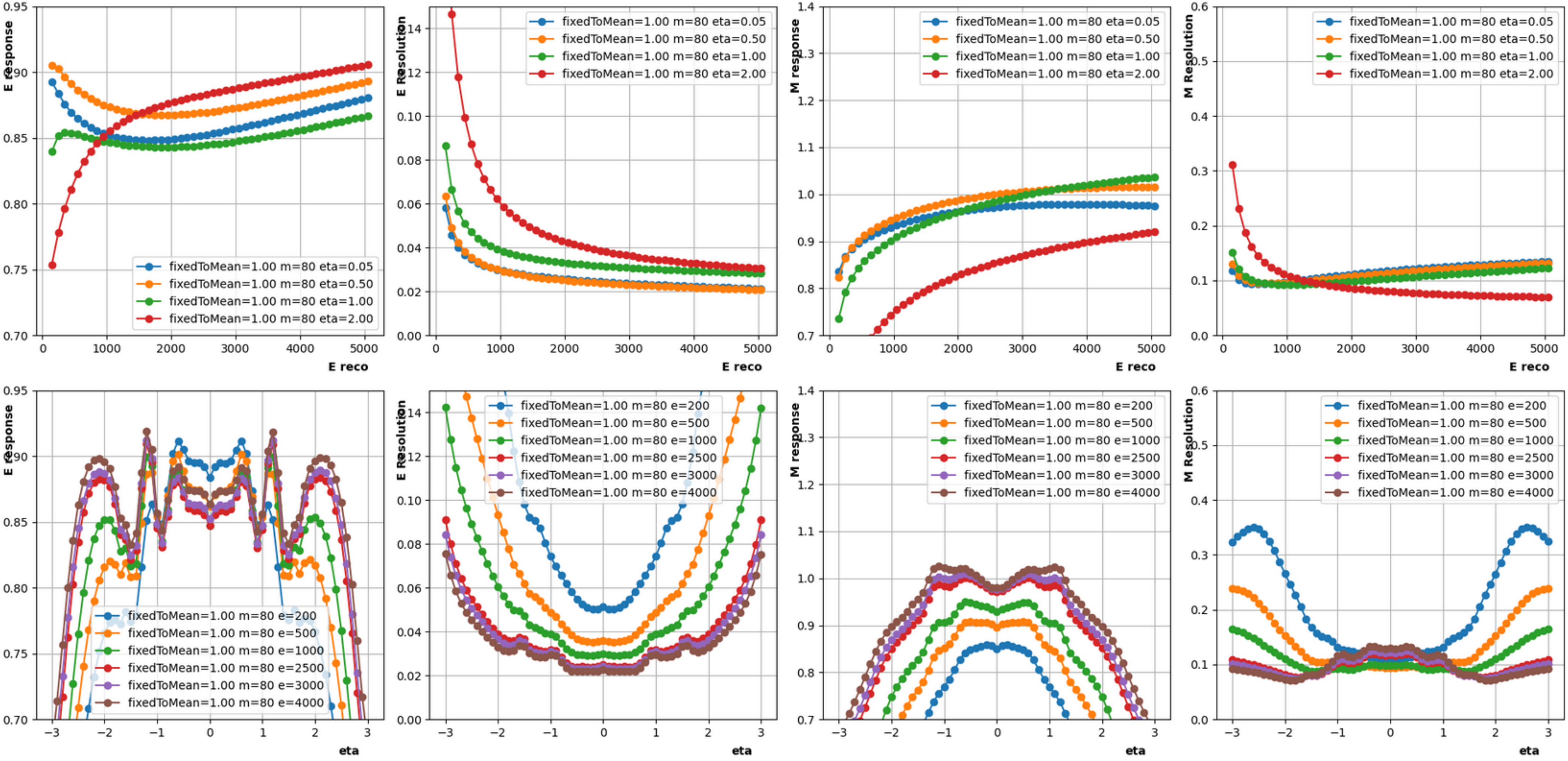


Note :
Can evaluate LGK loss when training with MDN :
usualy get slightly **lower** LGK loss than when training with LGK

→ MDN seems to perform better

Training time :
1 epoch (260M jets) → O(5min) on Nvidia V100

NN predictions


E response
E reco
fixedToMean=1.00 m=80 eta=0.05
fixedToMean=1.00 m=80 eta=0.50
fixedToMean=1.00 m=80 eta=1.00
fixedToMean=1.00 m=80 eta=2.00
E Resolution
E reco
fixedToMean=1.00 m=80 eta=0.05
fixedToMean=1.00 m=80 eta=0.50
fixedToMean=1.00 m=80 eta=1.00
fixedToMean=1.00 m=80 eta=2.00
M response
E reco
fixedToMean=1.00 m=80 eta=0.05
fixedToMean=1.00 m=80 eta=0.50
fixedToMean=1.00 m=80 eta=1.00
fixedToMean=1.00 m=80 eta=2.00
M Resolution
E reco
fixedToMean=1.00 m=80 eta=0.05
fixedToMean=1.00 m=80 eta=0.50
fixedToMean=1.00 m=80 eta=1.00
fixedToMean=1.00 m=80 eta=2.00
E response
eta
fixedToMean=1.00 m=80 e=200
fixedToMean=1.00 m=80 e=500
fixedToMean=1.00 m=80 e=1000
fixedToMean=1.00 m=80 e=2500
fixedToMean=1.00 m=80 e=3000
fixedToMean=1.00 m=80 e=4000
E Resolution
eta
fixedToMean=1.00 m=80 e=200
fixedToMean=1.00 m=80 e=500
fixedToMean=1.00 m=80 e=1000
fixedToMean=1.00 m=80 e=2500
fixedToMean=1.00 m=80 e=3000
fixedToMean=1.00 m=80 e=4000
M response
eta
fixedToMean=1.00 m=80 e=200
fixedToMean=1.00 m=80 e=500
fixedToMean=1.00 m=80 e=1000
fixedToMean=1.00 m=80 e=2500
fixedToMean=1.00 m=80 e=3000
fixedToMean=1.00 m=80 e=4000
M Resolution
eta
fixedToMean=1.00 m=80 e=200
fixedToMean=1.00 m=80 e=500
fixedToMean=1.00 m=80 e=1000
fixedToMean=1.00 m=80 e=2500
fixedToMean=1.00 m=80 e=3000
fixedToMean=1.00 m=80 e=4000

NN predictions

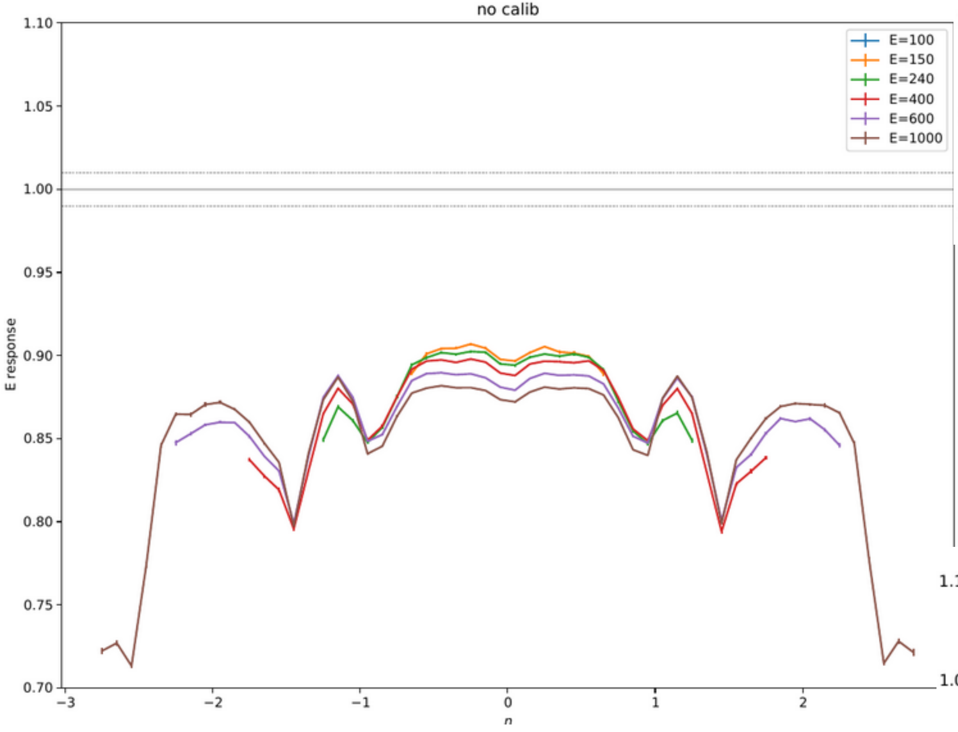
- All 3 tested NN predict similar E&M responses
  - No identical though, predictions vary within ~1% for E and a 2-3% for M

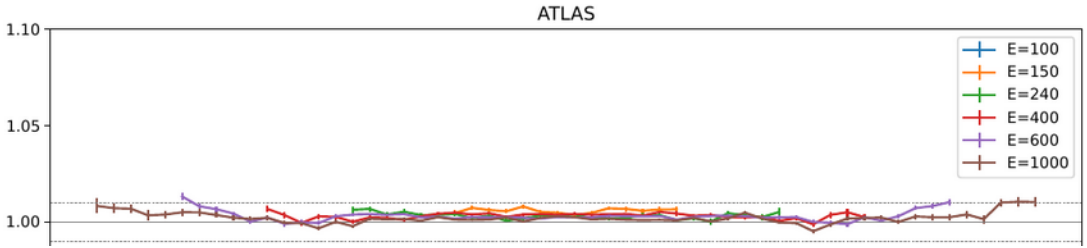
NN performances

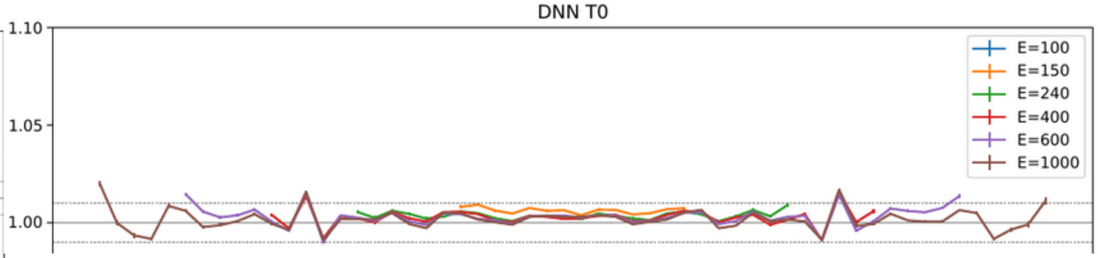
- Comparing NN calibrated responses with ATLAS standard JES and JMS calibration
  - And uncalibrated responses
- Use simple bins in (E,η), (E,η, NPV) or (p_T,m,η) to evaluate mode&resolution of response distributions
- Then plot mode&resolution vs variables

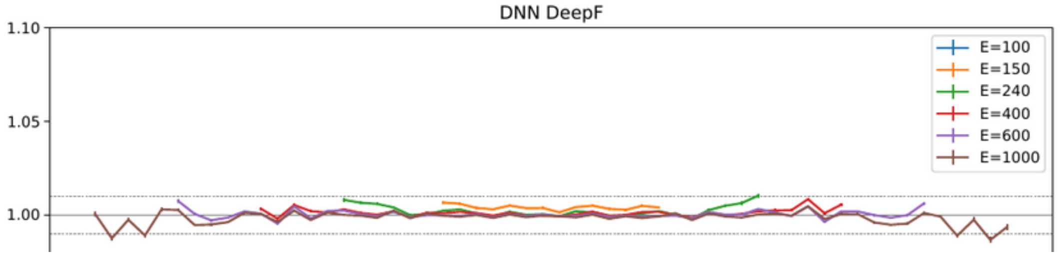
Resolution = IQR/response

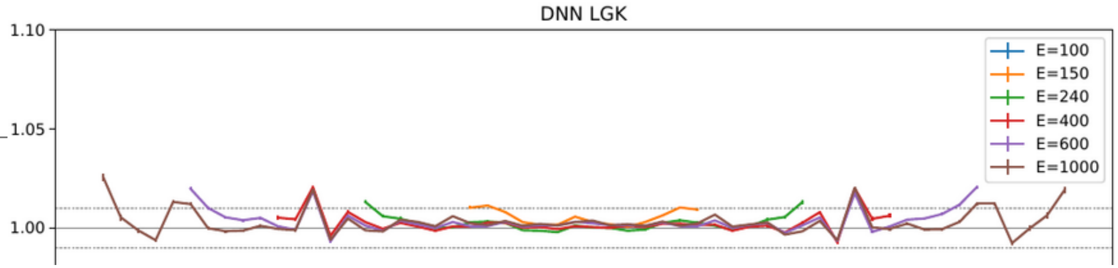
E response vs η



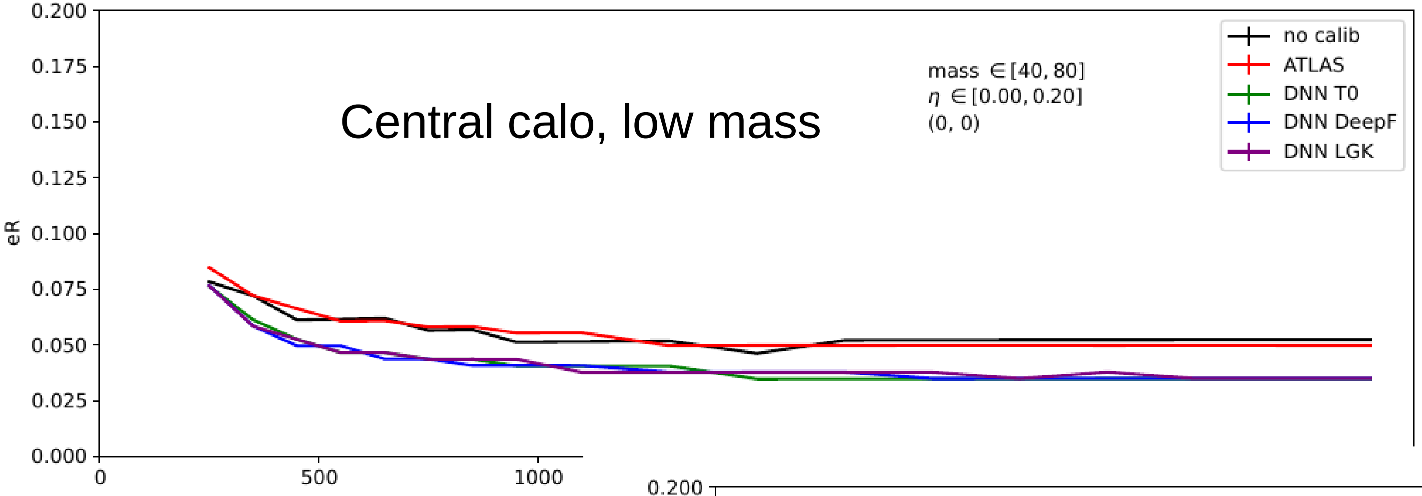


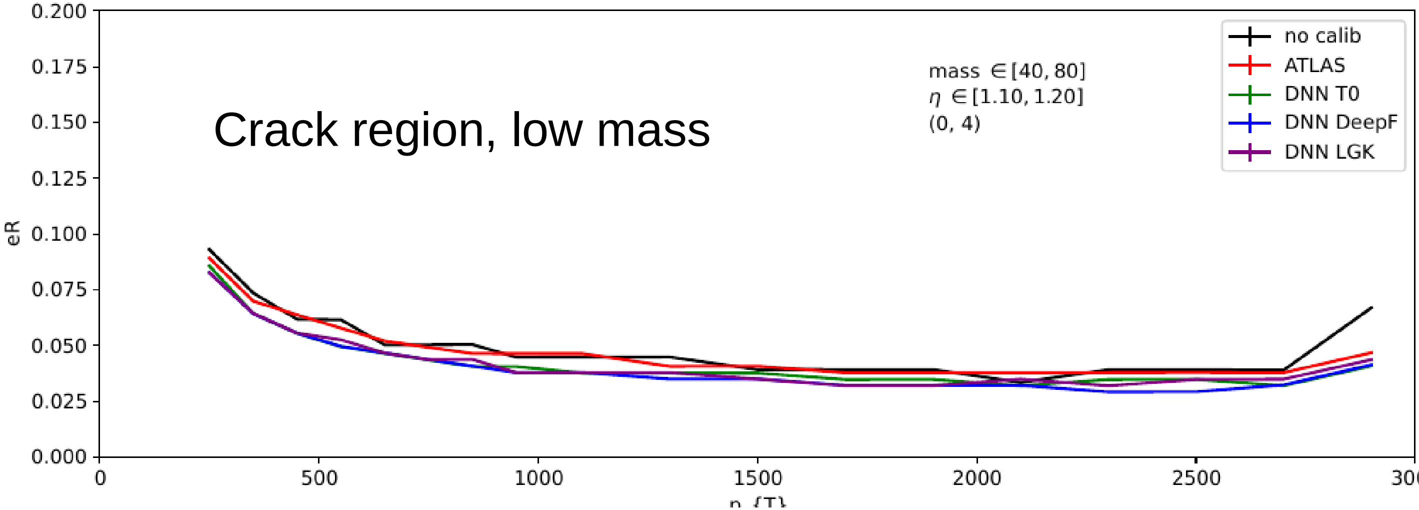






E resolution vs p_T



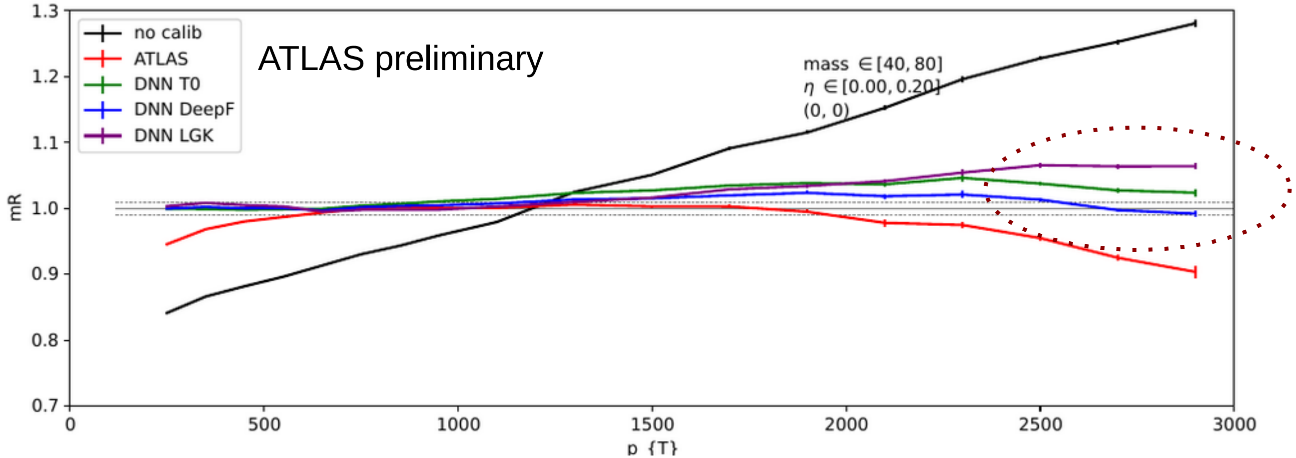


NN performances

- DNN perform very good E and mass calibration
  - Better than standard calib in almost all respects
    - Energy/mass scale and resolution
  - lower pile-up dependence
- All DNN perform similarly
  - “DeepF” variant looking a bit better

NN performances, limitations ?

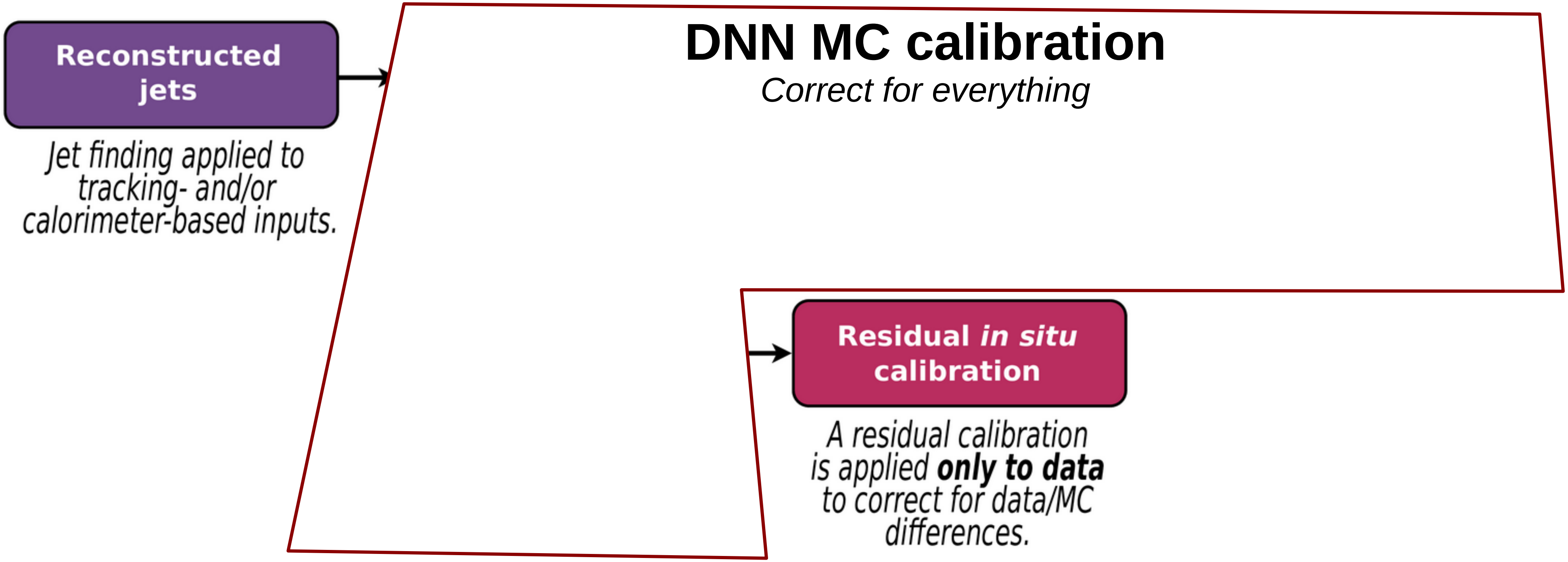
- Some differences in mass closure at high E
- difficult to understand and control
  - Not correlated to final loss : very similar in all cases
  - Giving more weight to highE/low mass events doesn't help
  - Maybe we just lack statistics at high E/low mass … ?



Conclusions

- Developed a DNN-based simultaneous calibration of jet E & mass
  - 2 noteworthy aspects : learning distrib mode & input annotation to deal with sharp η variations
- DNNs perfom globally better than ATLAS standard calib
  - Better closure & resolution
  - **But details are hard to understand & control**
    - What does matter ? NN architecture ? Event weights ? Loss function parametrization ?
    - Makes it difficult to define reliable & robust calib procedure
- To do :
  - Check impact of input distributions and weighting schemes (related to above difficulties ?)
  - Check performances with non QCD-initiated jets : W/Z, H and top jets (on-going : looking great !)

backup


DNN MC calibration
Correct for everything
Reconstructed jets
Jet finding applied to tracking- and/or calorimeter-based inputs.
Residual in situ calibration
A residual calibration is applied only to data to correct for data/MC differences.

Numerical Inversion

A procedure to avoid dependence on input sample distribution

- Learn R as function of E_{true} with a 1st DNN, call it R_{DNN1}
- Define $E_{NI} = R_{DNN1}(E_{true})\, E_{true}$
  - Numerical inversion of E_{true} : “best guess of E_{reco} given E_{true}”
- Learn R as a function of E_{NI} with a 2nd DNN, call it R_{DNN2}
- Define $C(E_{reco}) = 1/R_{DNN2}(E_{reco})$

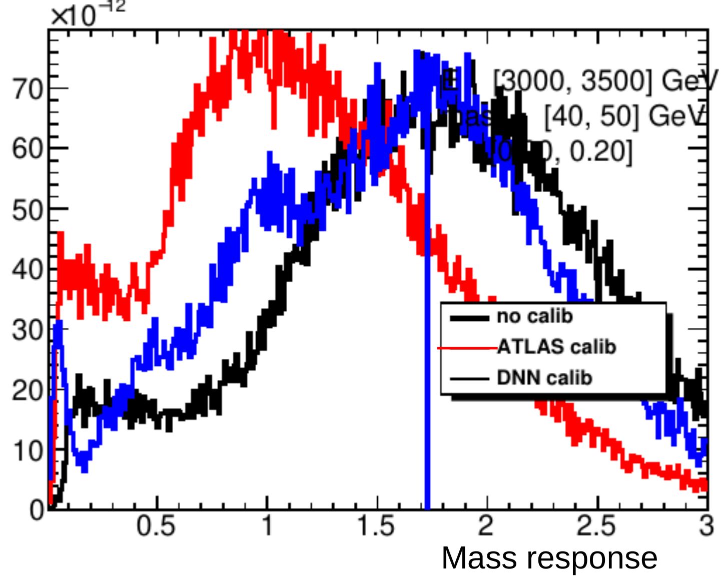
Numerical Inversion

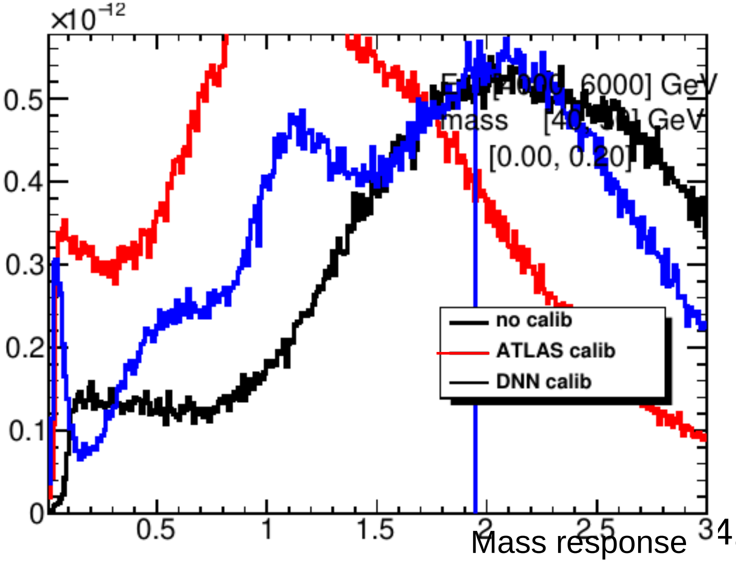
Numerical Inversion

- Procedure used in standard calibration involving learning 2 Response functions
- Using DNN amplifies intrinsic difficulties related to calibrated quantity X as function of X
  - Occurs for mass because response varies strongly and has large width
- Contrary to standard techniques, no numeric mitigation possible with DNN (or very complicated)
- Forget numerical inversion, just regress directly vs reco quantities
  - Will have to carefully evaluate potential bias due to input distrib

Issues with mass calibration

- Calibrated mass distributions also problematic
  - High E, low masses
- Double (triple) peaks appearance
- Known effect due to mathematical features of calibrating X as function of X
- Amplified by NN and very difficult to mitigate
  - mitigation procedure in standard calib unapplicable with DNN





Issues with Numerical Inversion

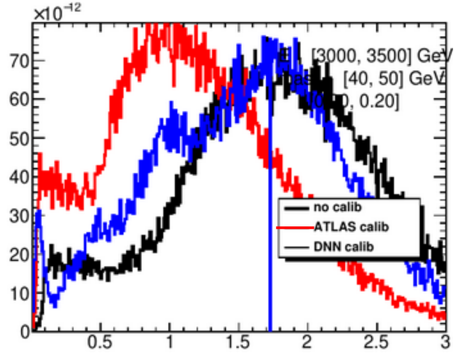
- More complex because needs 2 NN
  - Longer to train
  - Much harder to debug
- Amplifies response distortion issues
  - With no easy way to fix…

Try direct calibration

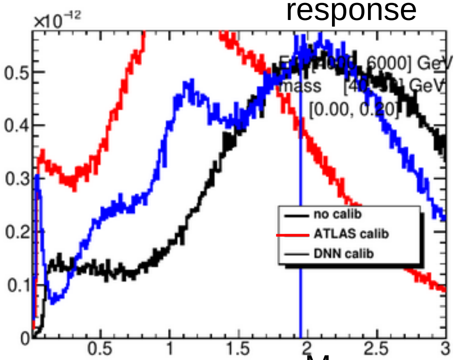
(learn directly $R(E_{reco})$ with 1 NN)

Direct Calibration

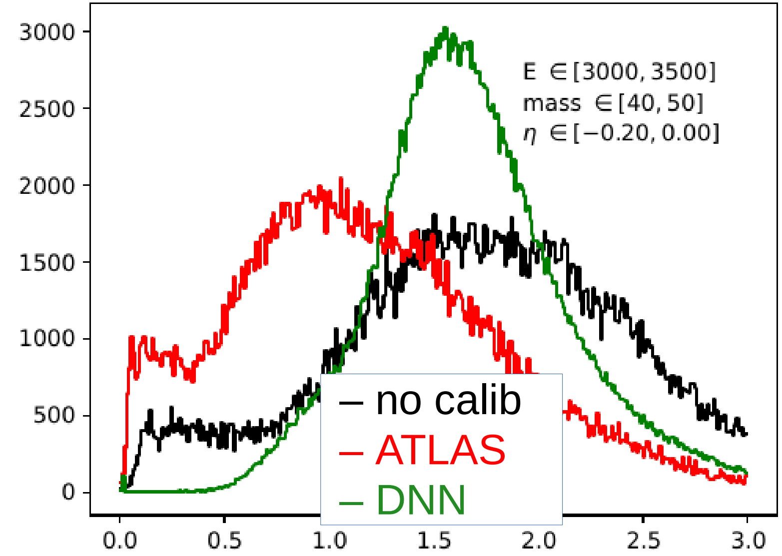
- Much better corrected mass distribution

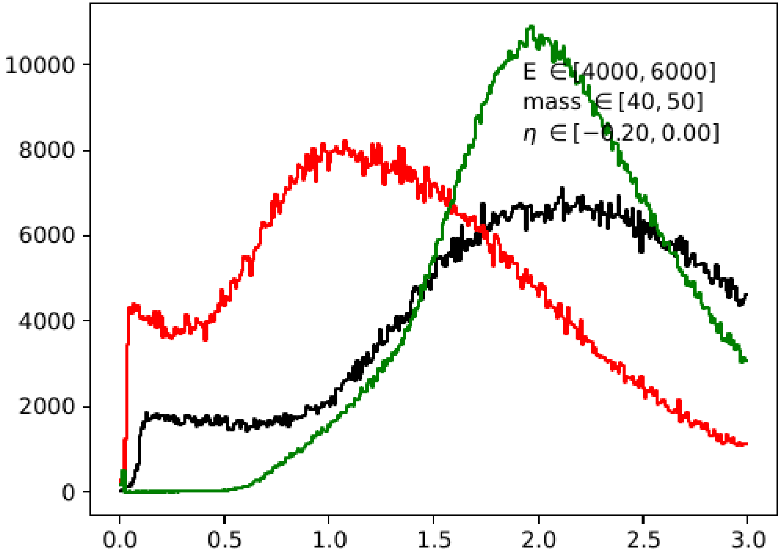


Mass response



Mass response

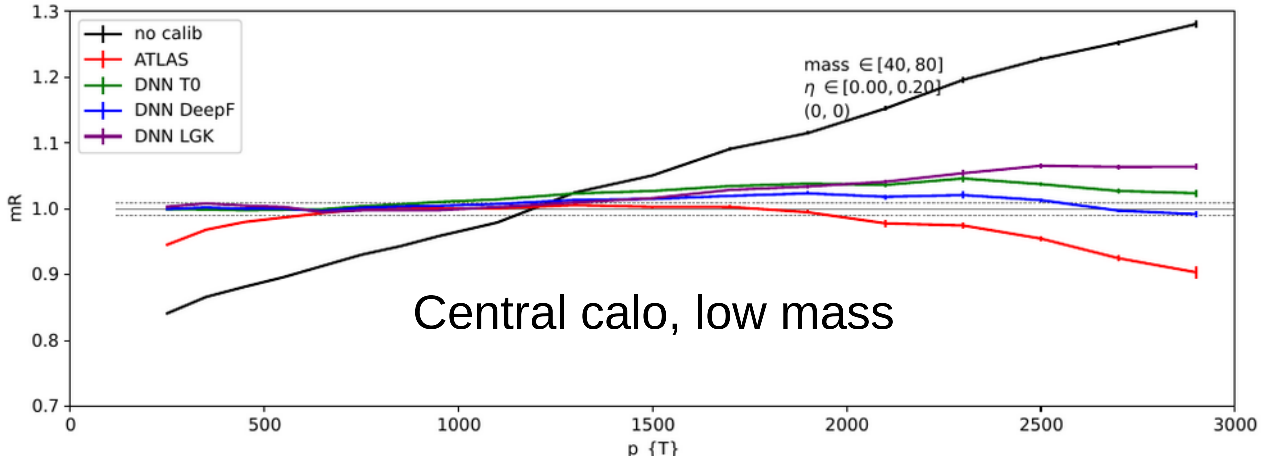


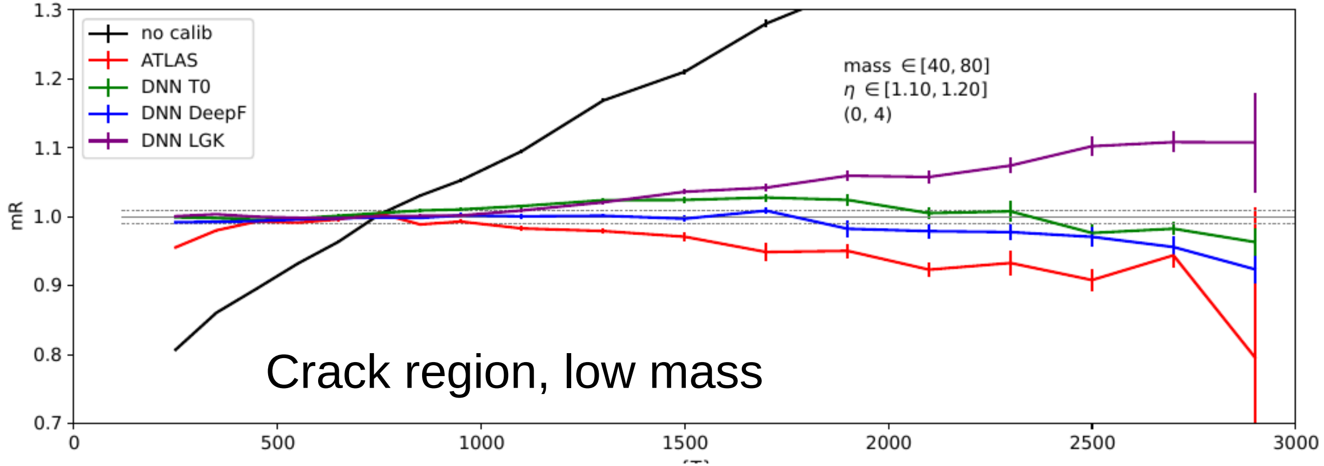


NN training optimizers

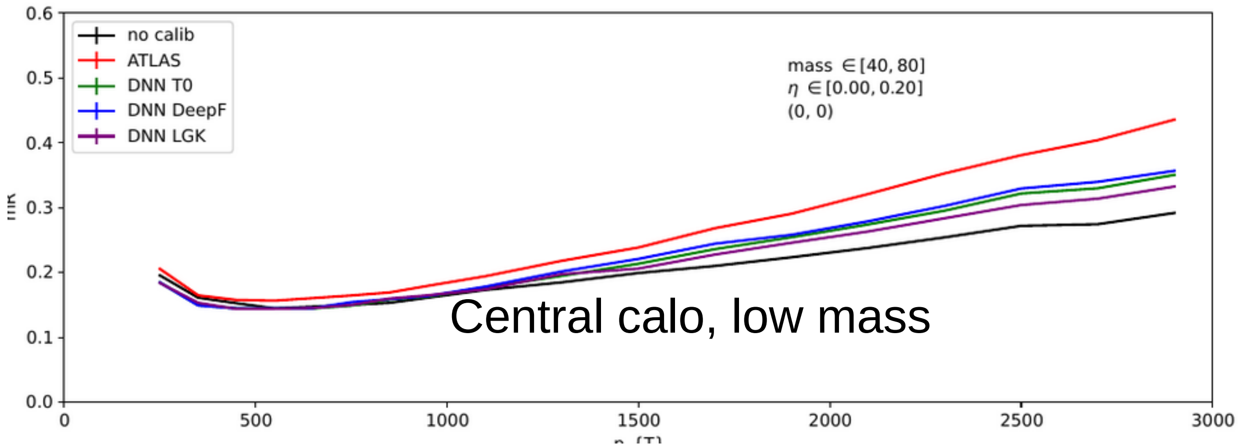
- Tested several optimizers, mostly variants of ADAM
  - Rectified-ADAM
  - ADABelief
  - DiffGrad
  - Applying the “Look-ahead” technique
- Goal was to see if they optimize better the loss
  - Nothing clear, but faster convergence than usual SGD/ADAM

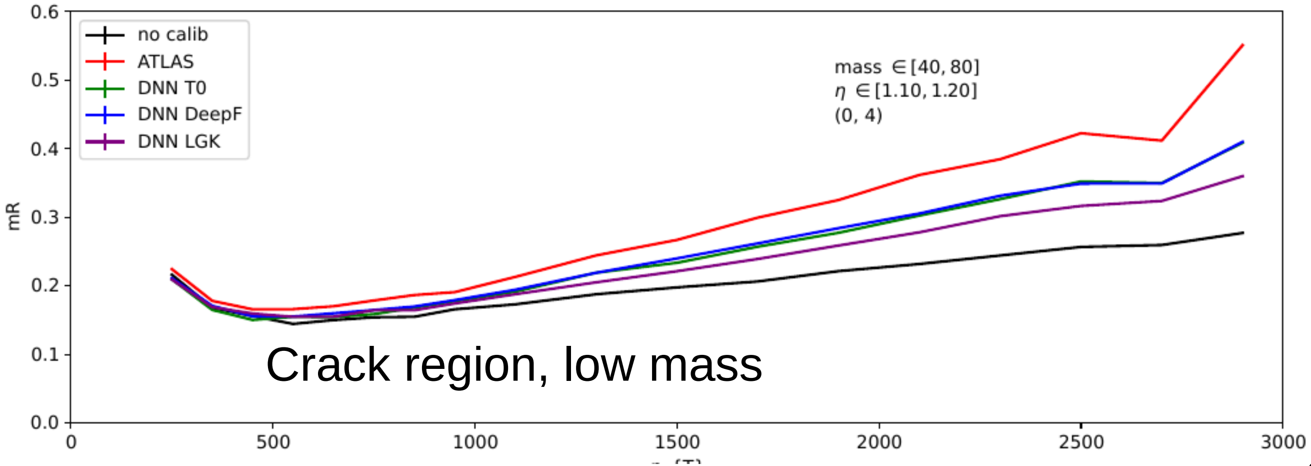
M response vs p_T



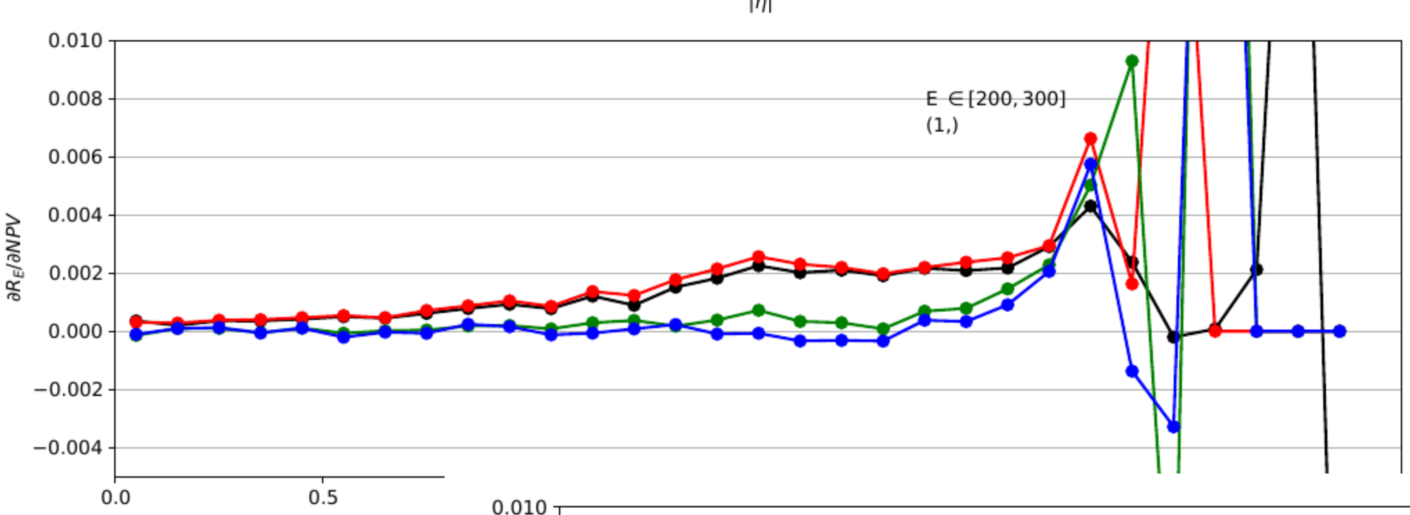


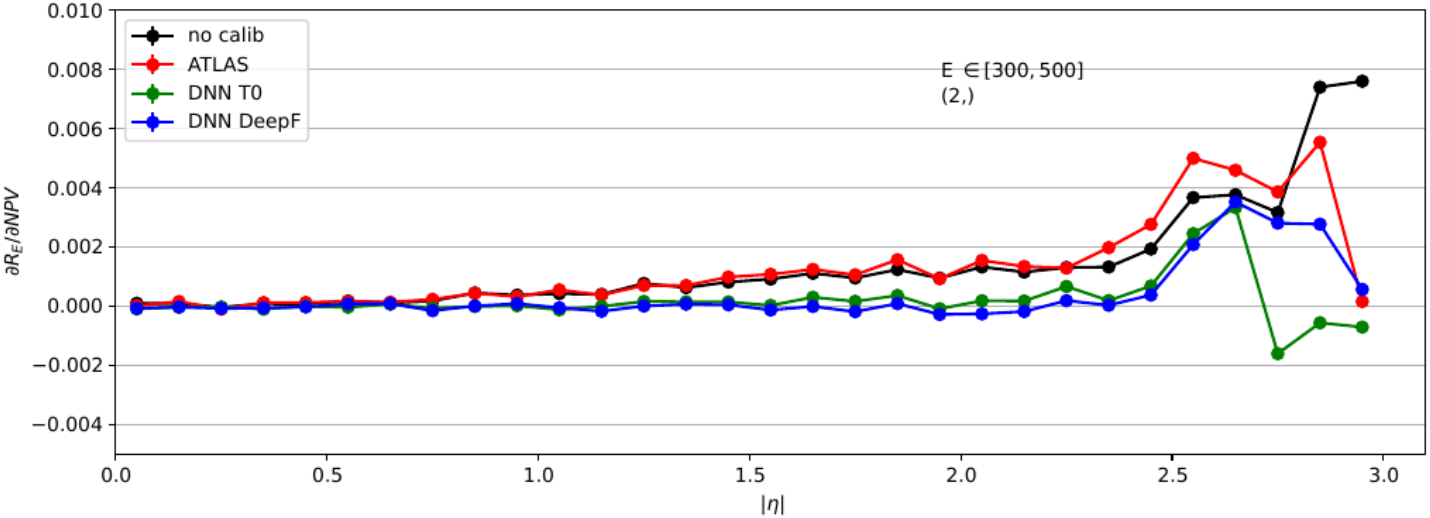
M resolution vs p_T





E response, NPV dependence vs η





M response, NPV dependence vs η