Systematic Uncertainties with Deep Sets Neural Network (DSNN)

Fang-Ying Tsai (Stony Brook University)
Computational HEP Traineeship Summer School
July 27 2023

Motivation

- In particle physics analysis, we compare the observed data with predictions from Monte Carlo simulations. However these MC simulations often have limited precision and can't fully capture all aspects of the data.
- The likelihood function quantifies how likely it is to observe the data given the model's prediction (S+B), and the uncertainties associated with the model.

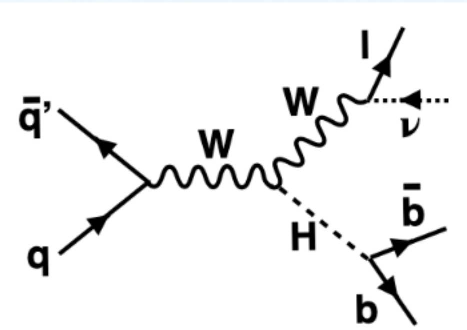
$$\mathcal{L}(n|\boldsymbol{\mu},\boldsymbol{\theta}) = \prod_{i \in bins} \mathcal{P}(n_i|\boldsymbol{\mu} \cdot S_i(\boldsymbol{\theta}) + B_i(\boldsymbol{\theta})),$$

μ: signal strength

NPs (θ): affecting the total signal or background are called normalization factors (NFs), affecting the corresponding probability distribution function (PDFs) are called shape uncertainties.

Signal process

1-lep



+ **Background processes (Diboson, single top, W+jets…)**

- We, analyzers, provide variations of histograms that represent the systematic uncertainty for specific kinematic distributions (e.g. mBB). The LH fit will determine the best-fit values for μ and NPs that minimize the discrepancy between the data and MCs.

Motivation

Uncertainties in the W + Jets Background Process

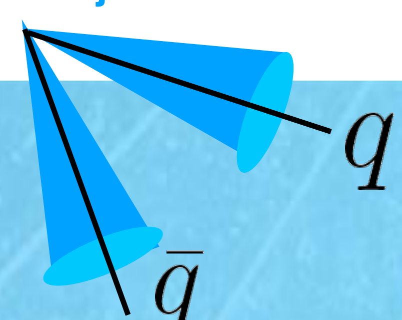
➤ One common way to estimate shape uncertainties is to make MC-to-MC comparisons on kinematic variables in 1-dimension and take the differences as shape uncertainties. 🙅‍♀️
- Many variables are correlated, and uncertainties in one variable can affect the others.

➤ An alternative approach that aims to capture the interdependencies and correlations between input variables using neurons networks.
- The comparison between the BDT and the DSNN may shed light on the relative strengths and weaknesses of these different ML techniques.

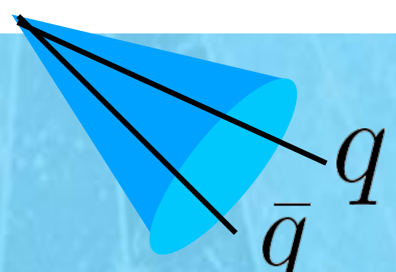
➤ Goals of the DSNN: Allowing for independence from specific analysis techniques and reconstruction schemes.

➤ To achieve a classifier trained inclusively, the DSNN framework replaces all higher level input variables with the 4-vectors of the final state particles' momenta.
- Avoiding the need for separate training for small-radius or large-R jet algorithms used in resolved or boosted regions.

Small-R jet: Anti-kt R=0.4

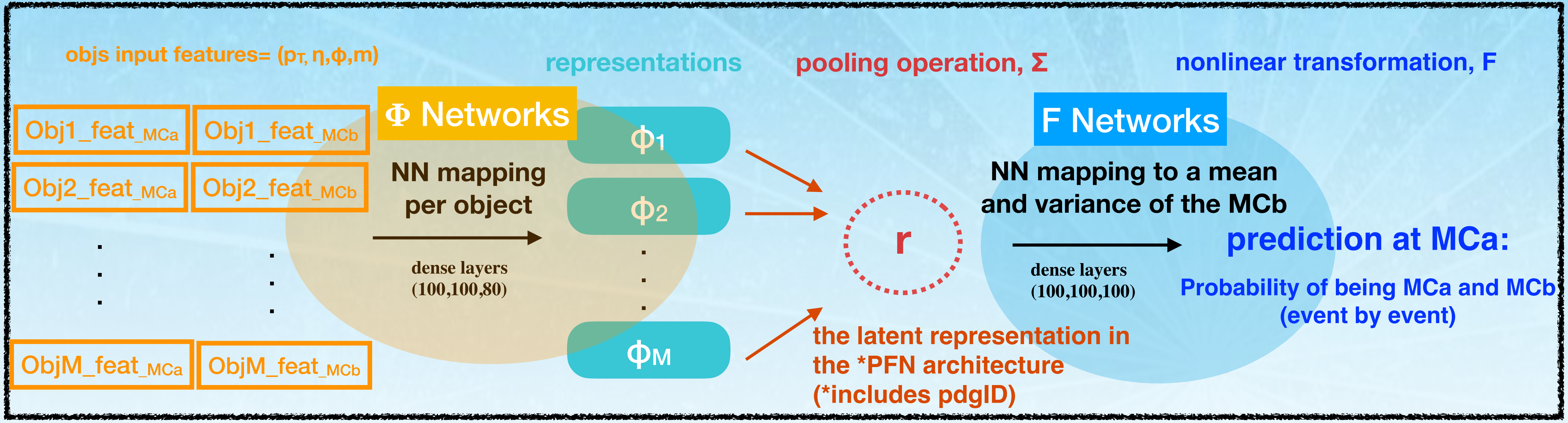


Large-R jet: Anti-kt R=1.0



Intro to the DSNN $\mathcal{O}(p_1,\ldots,p_M) = \mathrm{F}(\Sigma_{i=1}^{M}\Phi(p_i))$

- ➤ The DSNN architecture (ref.)
- \- to analyze collections of data points generated by Sherpa and Madgraph.



➤ The inputs to the framework are fixed-length vectors, which are then passed to a deep-set neural network.

➤ The deep-set neural network (Φ) is used to handle unordered sets of data, allowing the DSNN to analyze particle collision data without the need for a specific ordering of particles.

➤ Another deep neural network (F) is used to predict the behavior of particles event by event.

Comparison **BDT v.s. DSNN**

➤ BDT

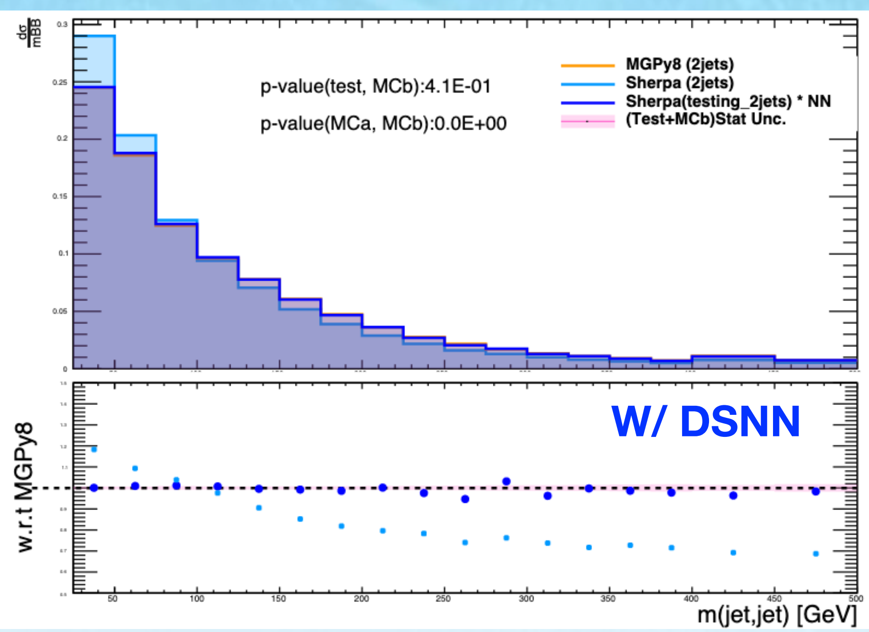
- Collection of decision trees trained on high-level observables (see backup).
- Optimized using a gradient boosted decision trees algorithm.
- Works by building trees that ask a series of questions based on input variables.
- Hyper-parameters control the learning process (see backup).
- Events are categorized into four folders based on remainder when divided by 4.
- Training for this study is done without requiring b-tagging, truth flavor info and the BDT.

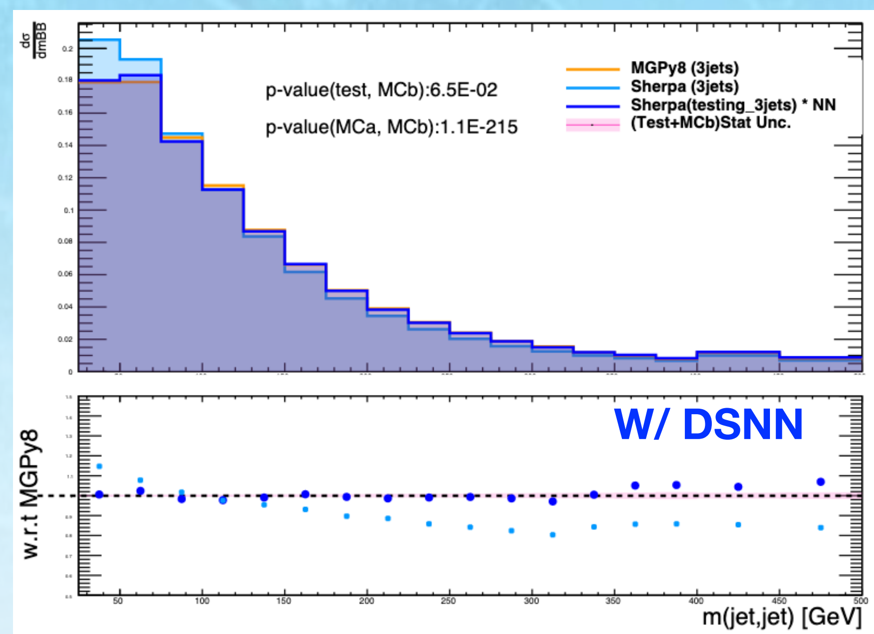
➤ DSNN

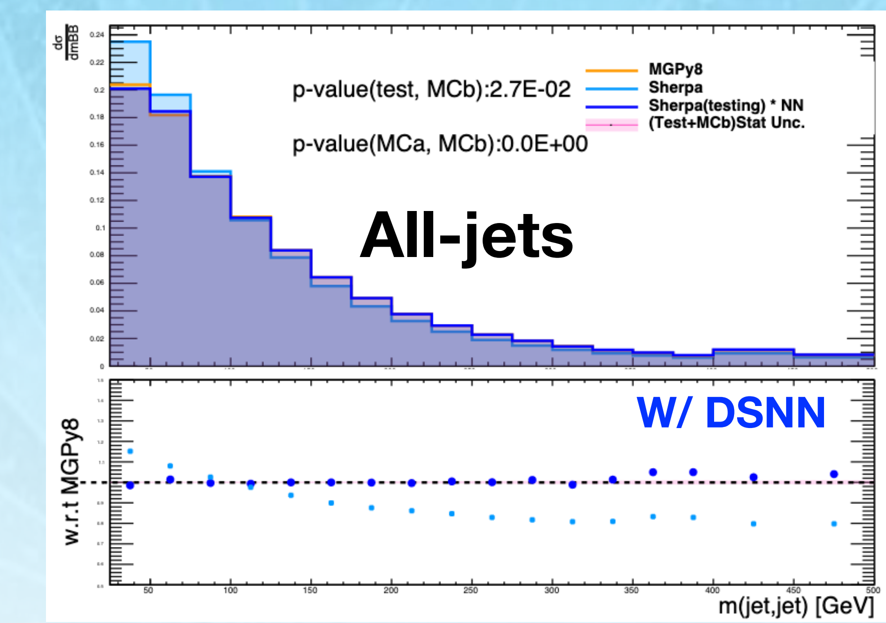
- Deep neural network trained on four-vector of input 6 objects.
- Optimized using backpropagation algorithm.
- Works by passing the input through multiple layers of interconnected nodes.
- Number of nodes, layers, and activation functions control the learning process.
- Events are split 50-50 using the Sklearn train_test_split function.
- without requiring b-tagging but with truth flavor info.

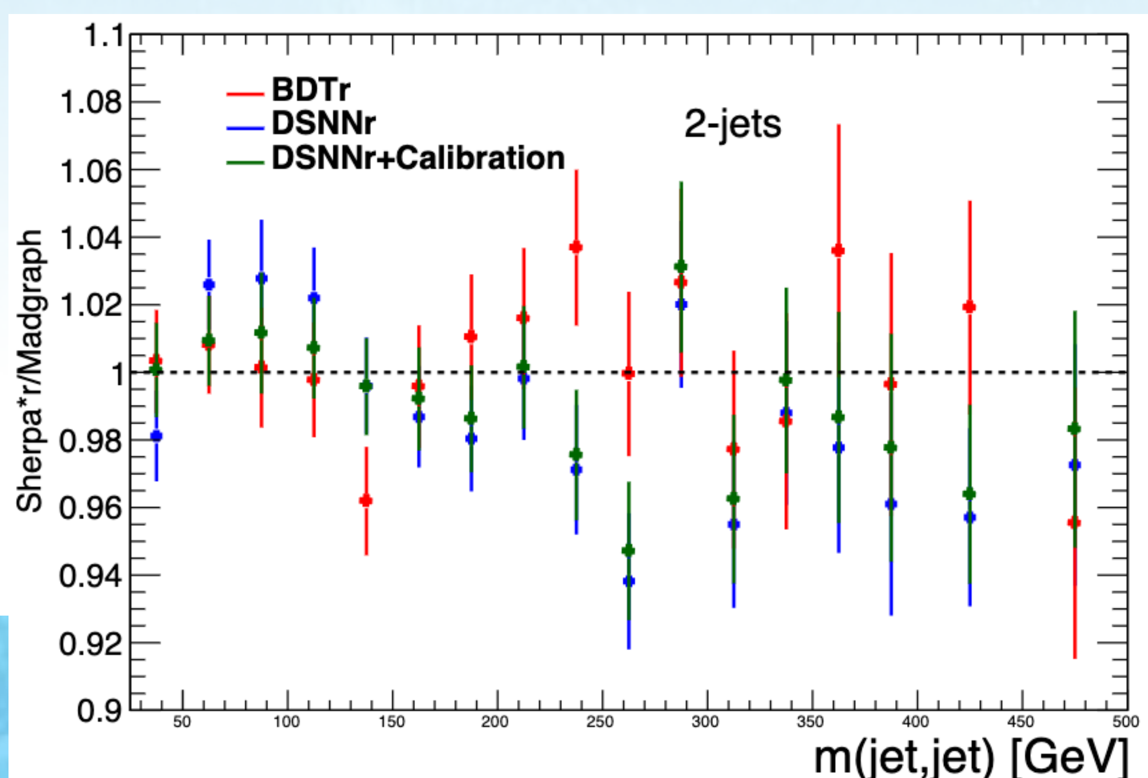
Performance & Demo

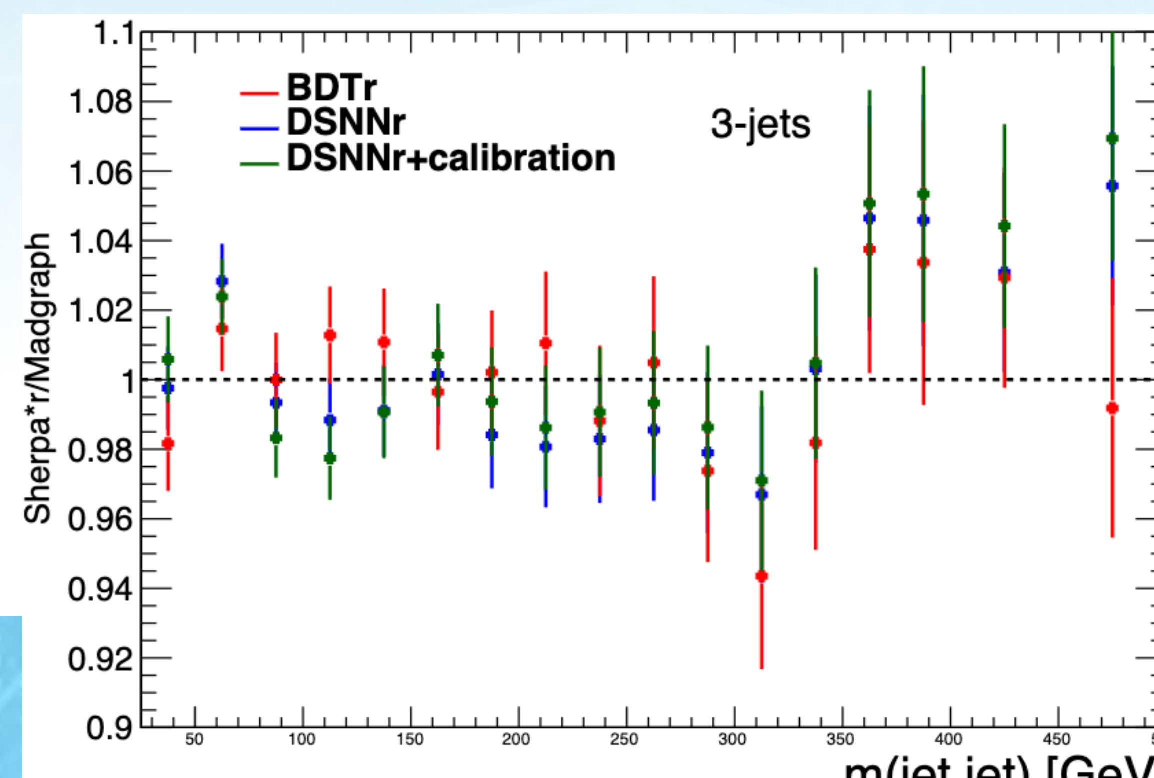
```
(data >= 0.4) & (data <= 0.48): -0.05;  (data > 0.55) & (data <= 0.84): -0.05;
(data > 0.32) & (data <= 0.38): + 0.05

P-value (2jets): 4.1E-01
P-value (3jets:): 6.5E-02
P-value (all-jets): 2.7E-2
```

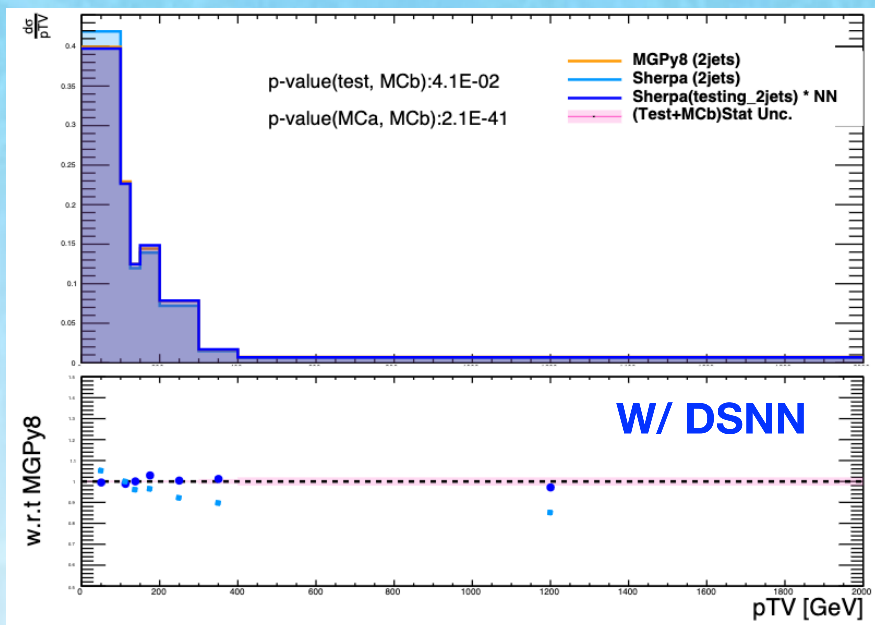


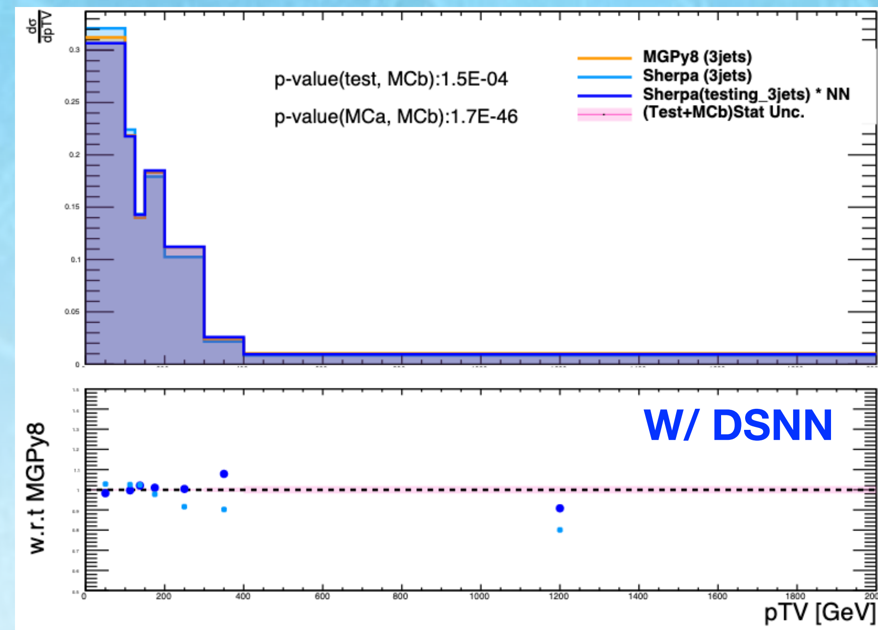


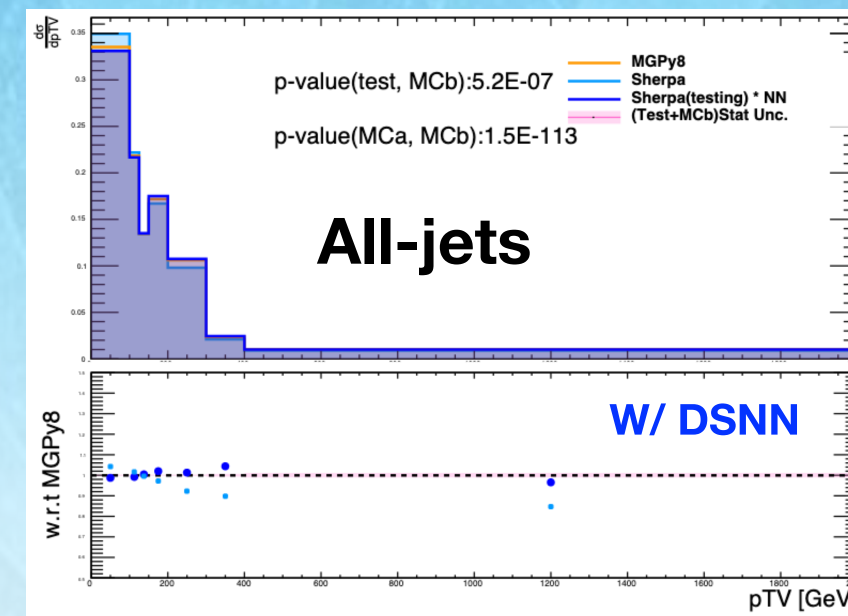


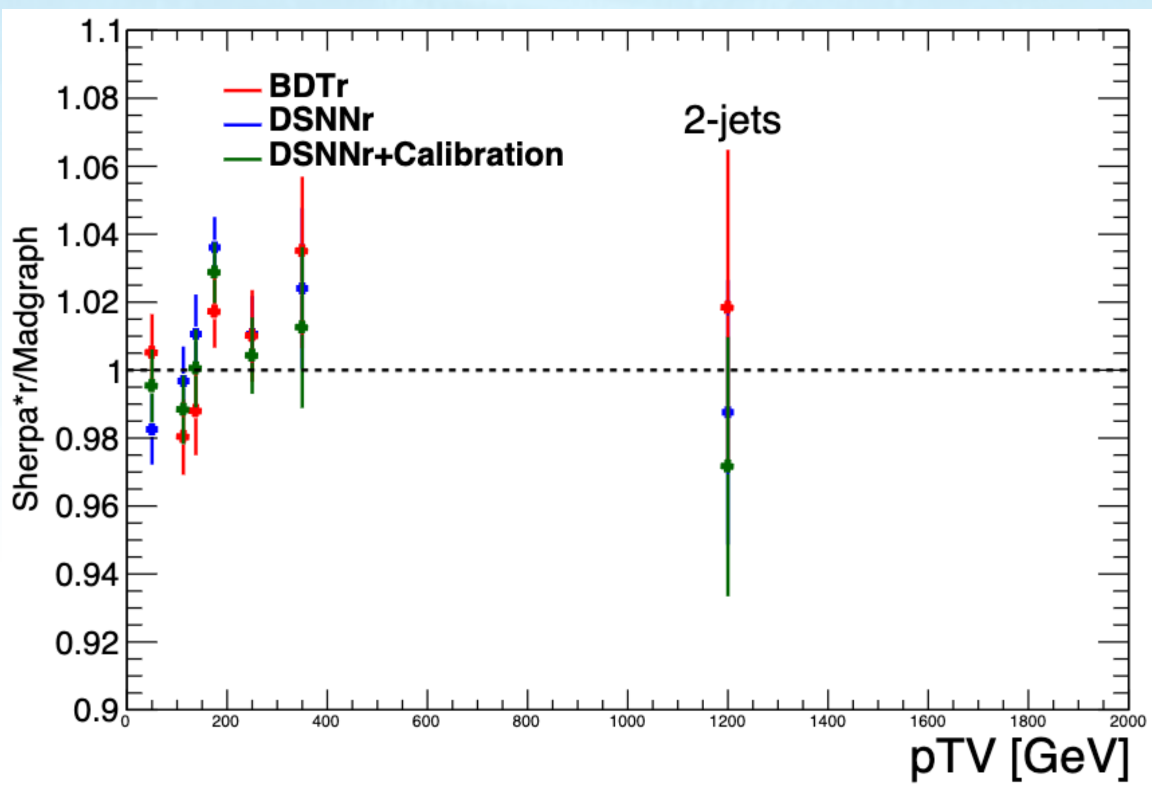


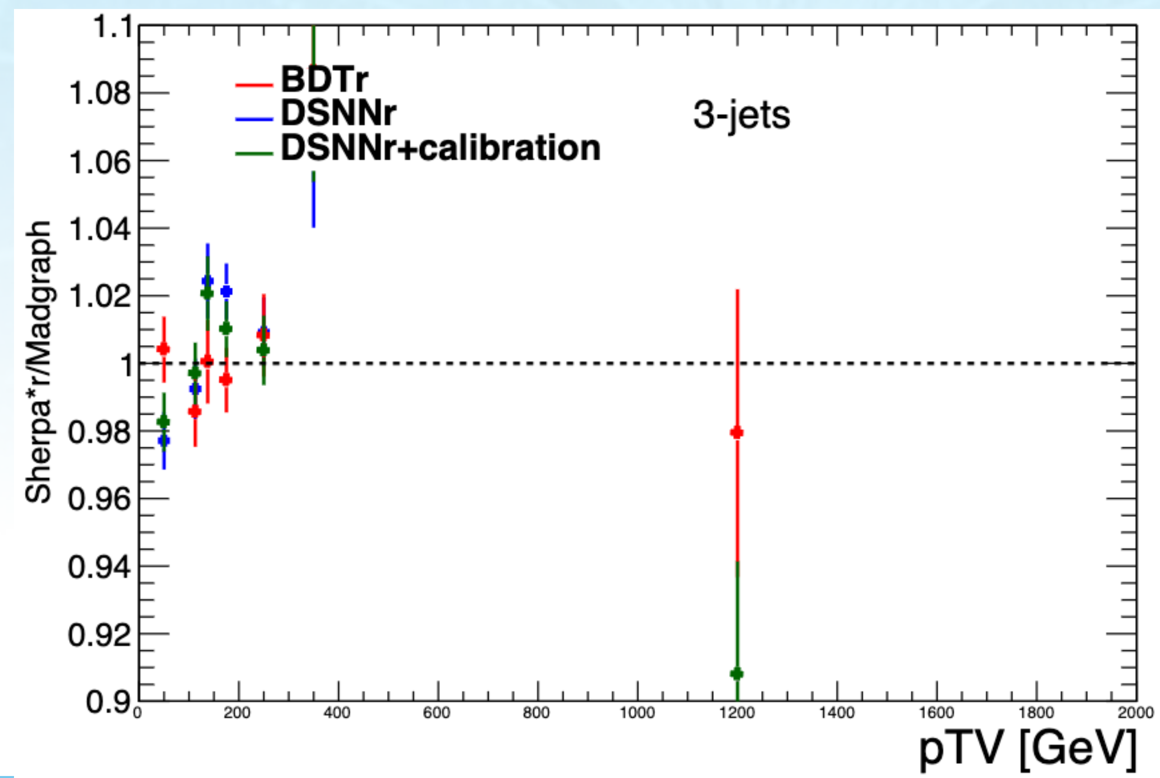



p-value(test, MCb):4.1E-02
p-value(MCa, MCb):2.1E-41
MGPy8 (2jets)
Sherpa (2jets)
Sherpa(testing_2jets) * NN
(Test+MCb)Stat Unc.
W/ DSNN
w.r.t MGPy8
pTV [GeV]


p-value(test, MCb):1.5E-04
p-value(MCa, MCb):1.7E-46
MGPy8 (3jets)
Sherpa (3jets)
Sherpa(testing_3jets) * NN
(Test+MCb)Stat Unc.
W/ DSNN
w.r.t MGPy8
pTV [GeV]


p-value(test, MCb):5.2E-07
p-value(MCa, MCb):1.5E-113
MGPy8
Sherpa
Sherpa(testing) * NN
(Test+MCb)Stat Unc.
All-jets
W/ DSNN
w.r.t MGPy8
pTV [GeV]


BDTr
DSNNr
DSNNr+Calibration
2-jets
Sherpa*r/Madgraph
pTV [GeV]


BDTr
DSNNr
DSNNr+calibration
3-jets
Sherpa*r/Madgraph
pTV [GeV]

Backup

Data Preprocessing

```
ResultVHbb1lep selectionResult =
((VHbb1lepEvtSelection*)m_eventSelection)->result();
const xAOD::Electron *el = selectionResult.el;
if (el) {
    m_tree->el_pt = el->pt()/1000;
    m_tree->el_eta = el->eta();
    m_tree->el_phi = el->phi();
    m_tree->el_m = el->m() /1000;
    m_tree->el_charge = el->charge();
    m_tree->el_pdgid = 0.1;
  }
```

➤ Store 4-vector sets of interested objects that pass certain criteria from CxAOD.

➤ Convert information from CxAOD to Numpy arrays in a tensor format: (events(N) x objects(6) x features(5)) dimension.

pT Eta Phi Mass

1st event

ID

```
MCa:[[[ 1.78720997e-02  7.32235032e-01  5.40055261e-01  1.09313274e-06
    1.00000001e-01]
  [-9.90000000e+01 -9.90000000e+01 -9.90000000e+01 -9.90000000e+01
   -9.90000000e+01]
  [ 2.30312683e-02  6.29122032e-01  8.74925878e-01  1.42243351e-02
    0.00000000e+00]
  [ 1.05225705e-02  6.78216278e-01  8.98270289e-01  5.61454207e-03
    0.00000000e+00]
  [-9.90000000e+01 -9.90000000e+01 -9.90000000e+01 -9.90000000e+01
   -9.90000000e+01]
  [ 2.93246533e-02 -9.90000000e+01  2.40396801e-01 -9.90000000e+01
    6.00000024e-01]]
```

2nd event

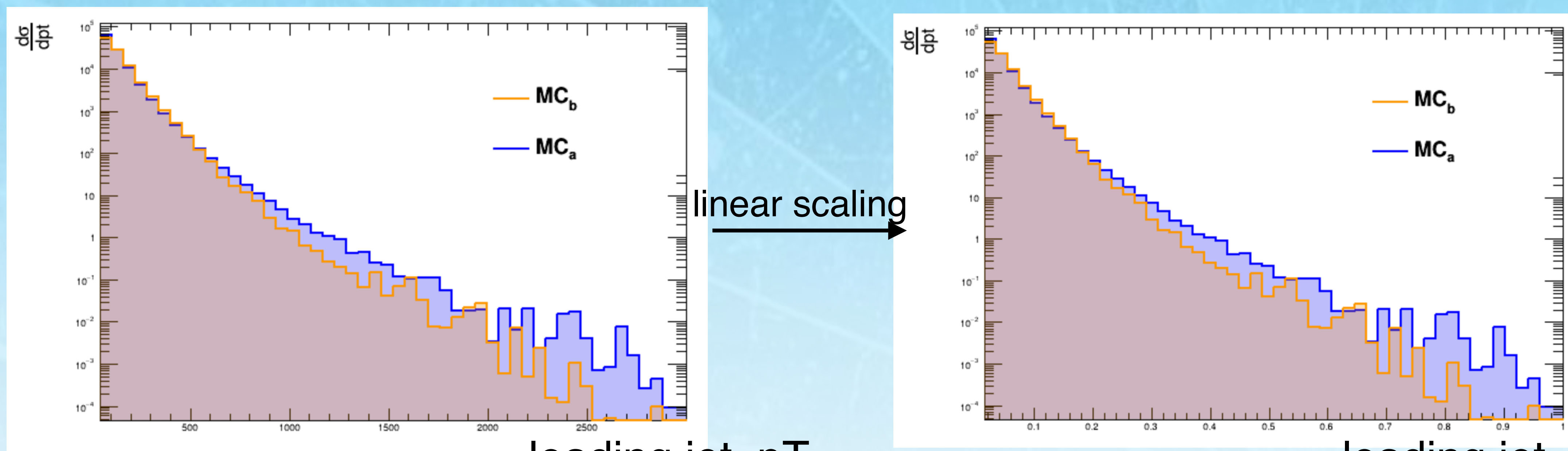
```
 [[ 2.23052587e-02  5.08907581e-01  2.15625185e-01  1.09313274e-06
    1.00000001e-01]
  [-9.90000000e+01 -9.90000000e+01 -9.90000000e+01 -9.90000000e+01
   -9.90000000e+01]
  [ 2.19909302e-02  5.73897966e-01  8.50040311e-01  1.51478814e-02
    4.00000000e-01]
  [ 9.07409307e-03  4.96559685e-01  1.09516810e-01  6.66544071e-03
    0.00000000e+00]
  [ 9.32545214e-03  4.62520496e-01  7.02413019e-01  8.69327710e-03
    0.00000000e+00]
  [ 2.24808316e-02 -9.90000000e+01  4.70176510e-01 -9.90000000e+01
    6.00000024e-01]]
```

electron (0.1)
Muon (0.2)
jet1
jet2
3rd
MET(0.6)

Props::HadronConeExclTruthLabelID.get()
jet flavors (0.4, 0.5 or 0) are used as input features for 2 jets and the third jets

Data Scaling

➤ The feature scaling technique: Normalization. $X' = \frac{x - x_{min}}{x_{max} - x_{min}}$



leading jet_pT leading jet_pT

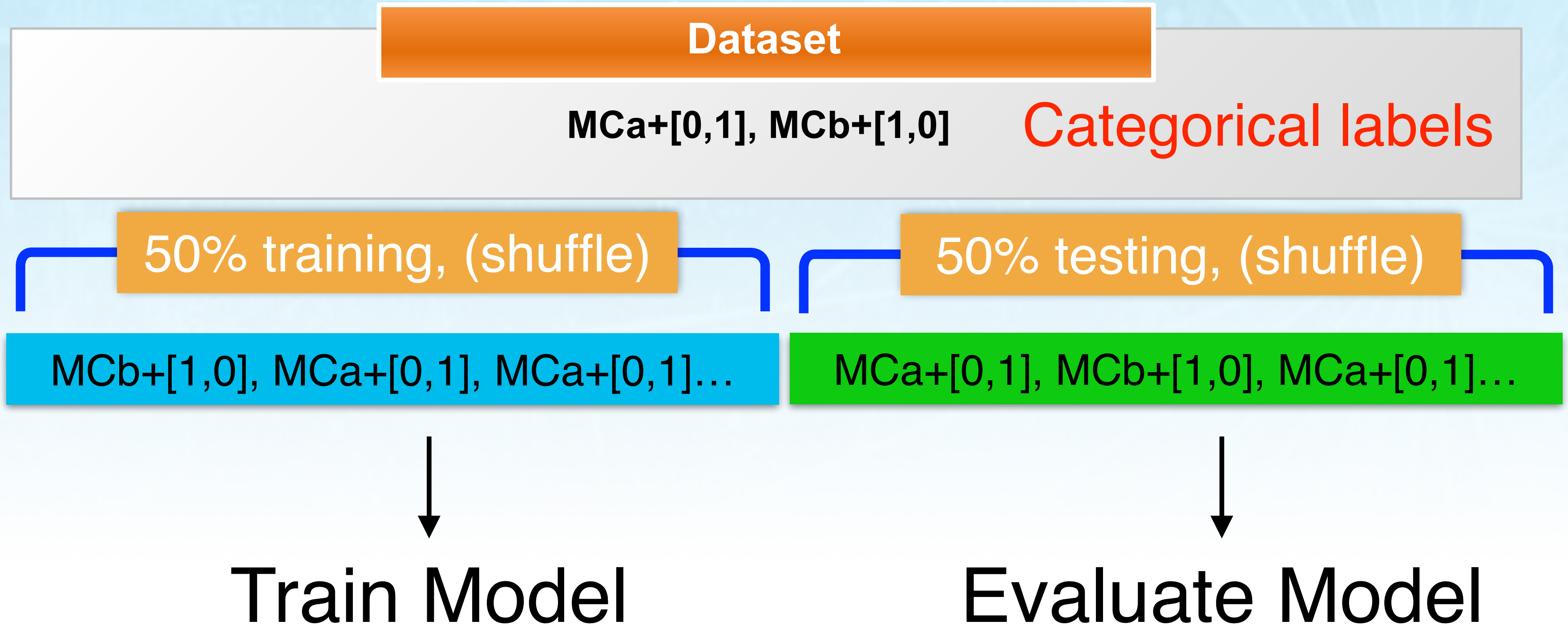
➤ Remove outliers; e.g. pT > 3000 GeV.

➤ Particle IDs are scaled to be between 0 and 1 (e.g. 4 → 0.4, 5 → 0.5). However, some events may not have a third jet, in which case the jet flavor became -9.9. This value of -9.9 is likely to be misleading for the model.

→ jet flavors that are missing in some events are modified to be NAN before scaling, and then reassigned to -99 after scaling and masked during training.

Train & Test

➤ The NN is trained in a supervised way.
- Both the training and testing datasets assigned categorical labels [0,1] and [1,0] representing MCa and MCb.
- The labels are represented using one-hot encoding, which ensures that there is no ranking between the category values and makes it easier to determine the prior probability of each category.



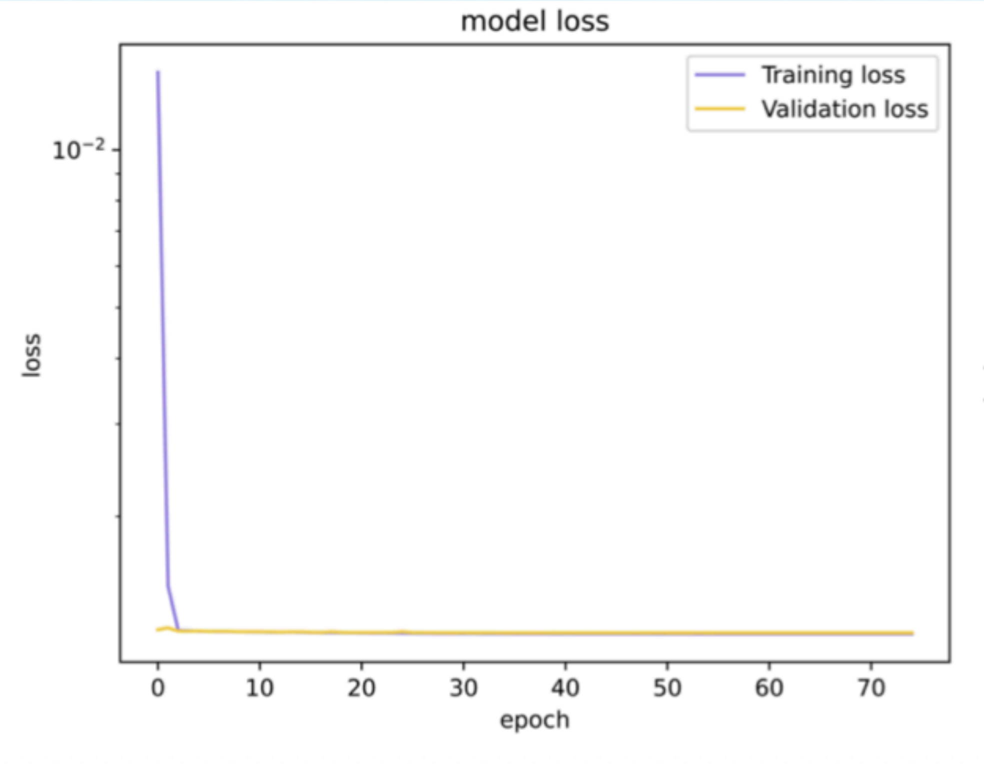
Hyper-parameters

- Finding the optimal combination of hyper-parameters can be challenging!
- Understanding hyper parameters in the DSNN:

  - batch size: if events in a single batch are not enough to represent the full dataset, resulting in poor network performance.

  - optimization algorithm: Adam (default), Adamax, RMSprop, etc., can be used to modify the learning rate.

  - learning rate: A linear decrease in the learning rate after the first 2 epochs is seen to improve stability and reduce the gap in the loss function, leading to better network performance.
- The behavior of the cross-entropy/loss suggests that it is a good fit that results in the generalization ability of the DSNN model.



Hyper-parameters and Deeper Residual Learning

➤ Neural networks with many layers have shown great potential for improving the accuracy of various tasks [ref].

- However, **the gradient vanishing or exploding problem** can occur when training deep neural networks meaning the system is not easy to optimize.

➤ A solution to the problem of training very deep neural networks with many layers:

paper, here

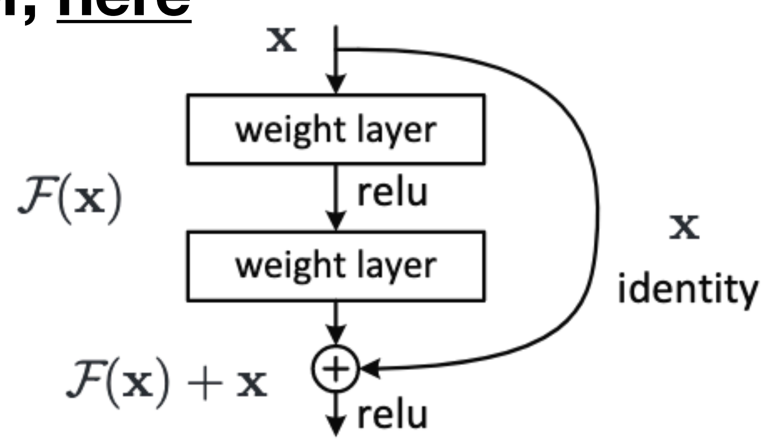


Figure 2. Residual learning: a building block.

The residual **F(x)** = H(x) - x
-> H(x) is a “truth function”, x is the input
To ensure we get the desired/truth mapping:
-> output = F(x) + x
If the identity mapping is optimal, then F(x) ~ 0:
-> The output (F(x)+x) ~ the input (x)

If the input is directly added to the output, the gradient can flow directly through the network.

➤ The best combination:
Phi(110,105,100)+F(95,95,95,95,95,95,95)

Prediction

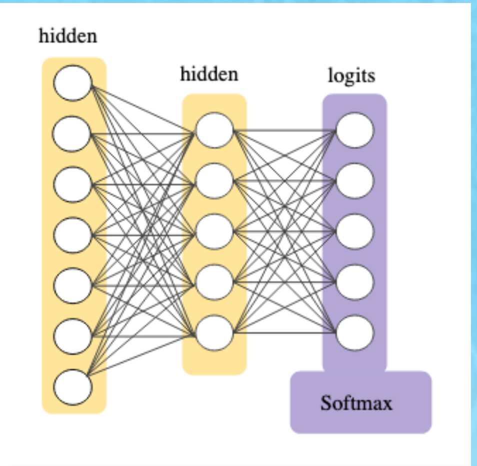
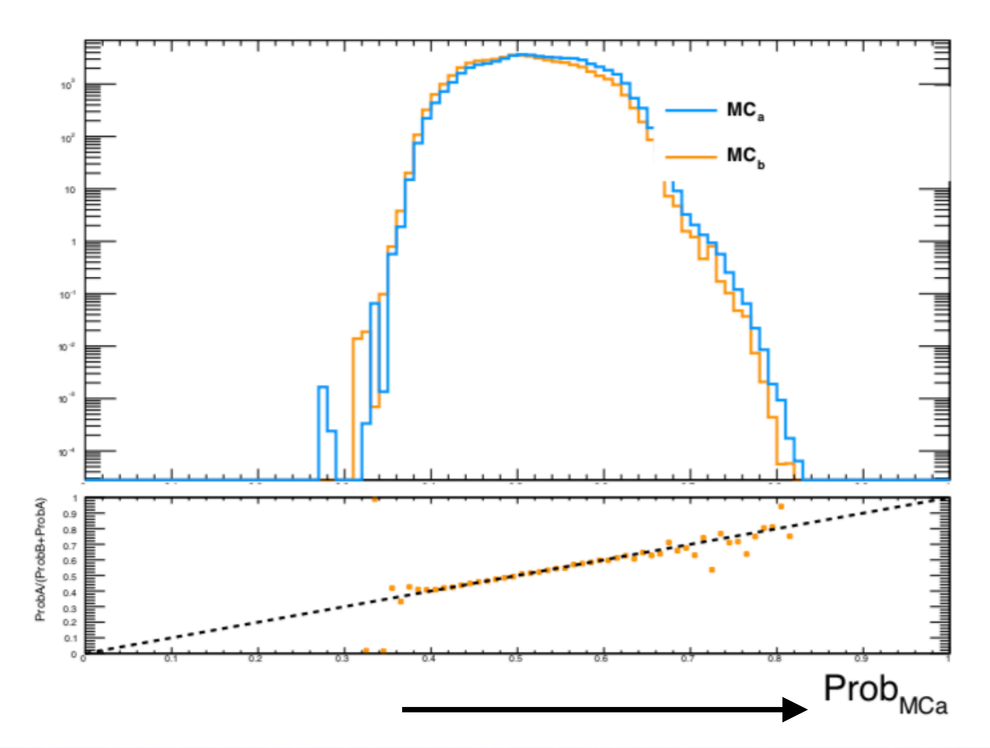


fig. source here
Common DSNN architectures options, here.

➤ The final NN layer returns the raw values for the predictions (= logits), classifier output.
➤ Softmax is used as a default recommended activation function: $Func_{softmax}$ (logits) => Probabilities for each class.



higher likelihood of being MCa

➤ Two distributions intersect at Prob_{MCa} = 0.5.
➤ The probability metric $ProbA/(ProbA+ProbB)$ is monotonically related to the predicted probability for class MCa. However, a calibration is needed!

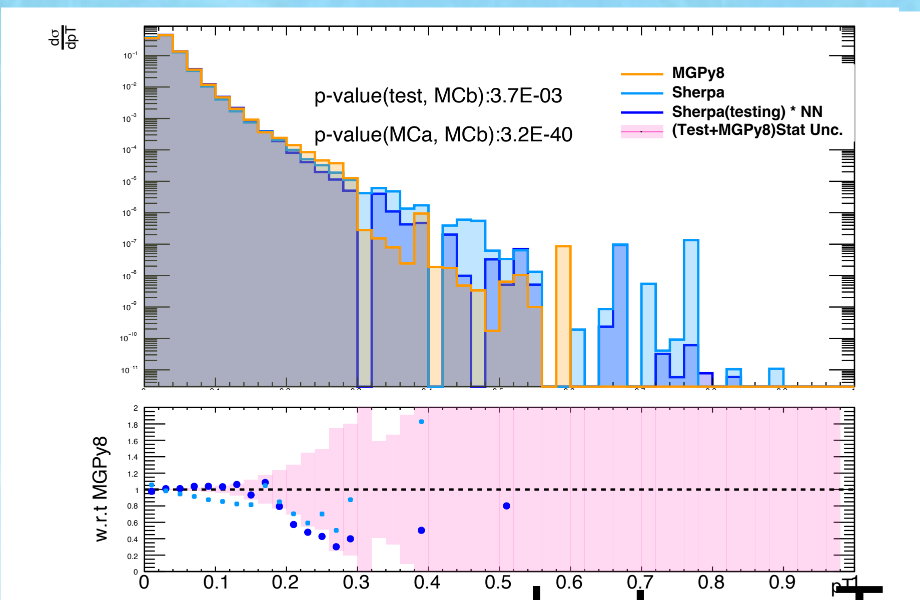
➤ Morphing between samples. $$NN_{weight} = \frac{Prob_{MC_b}}{Prob_{MC_a}}$$

➤ Reweight the W + jets production process as predicted by Sherpa (nominal sample) with a ratio provided by the DSNN algorithm event-by-event.

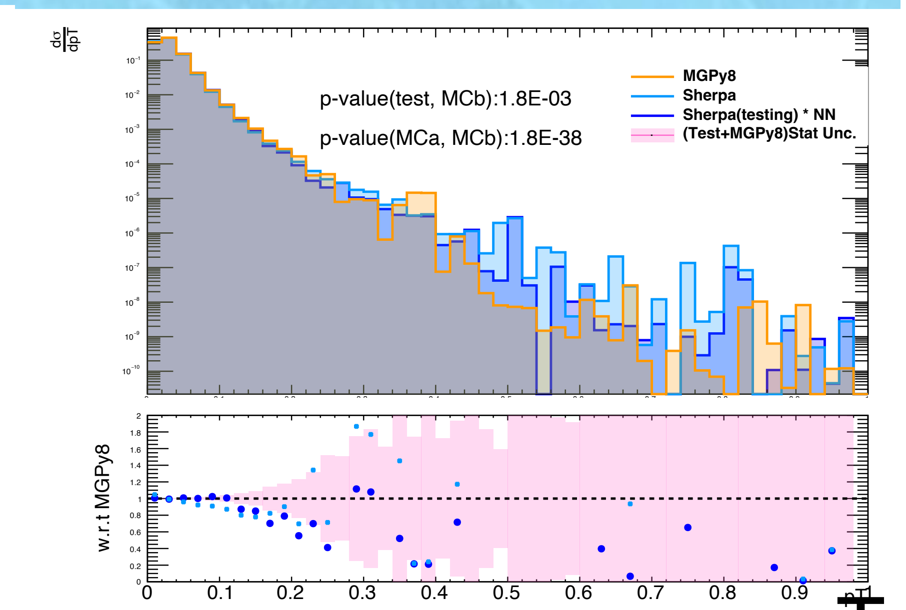
Features Performance

W/o corrections

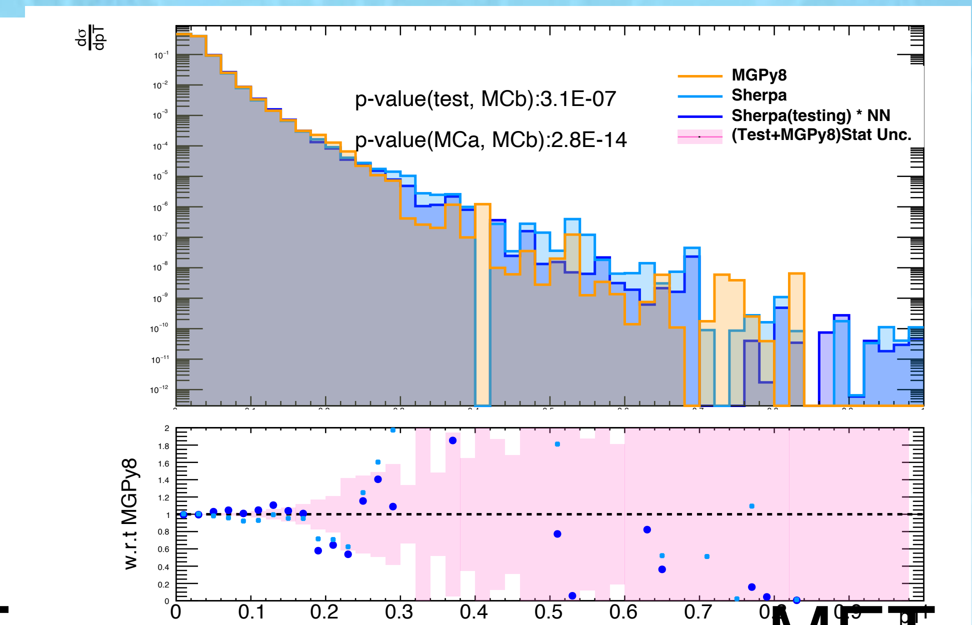
- ➤ Checking the algorithm is using the informative input features.
- ➤ The non-closure can mostly be explained by statistical fluctuations.



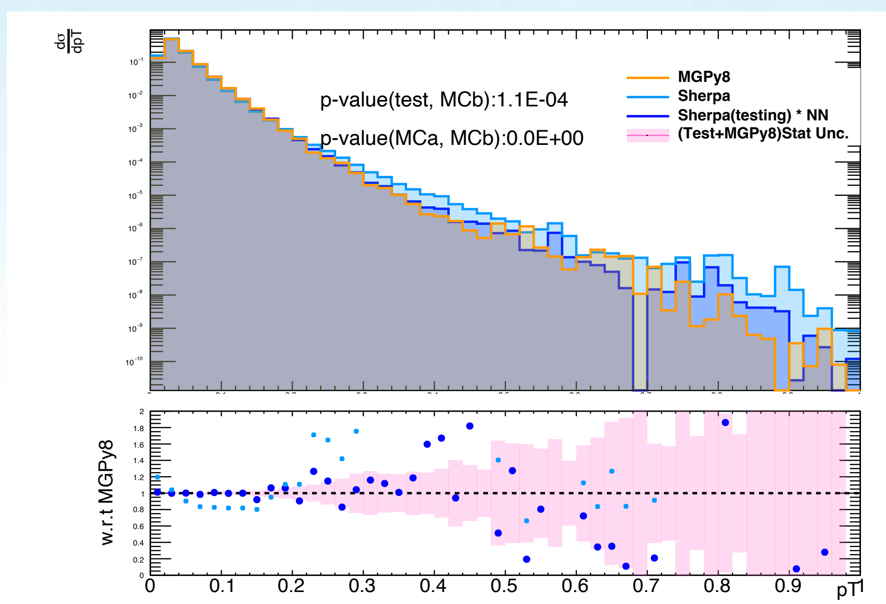
electron pT



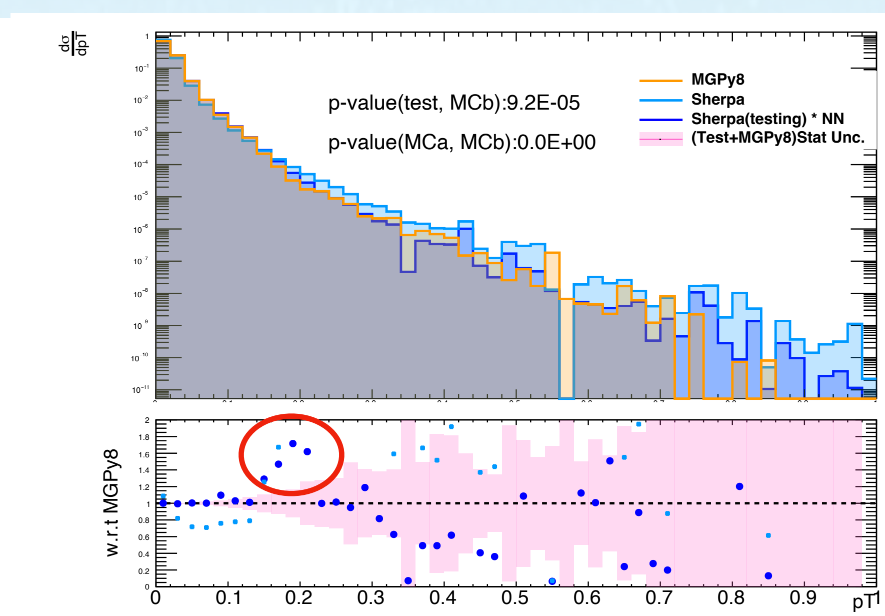
muon pT



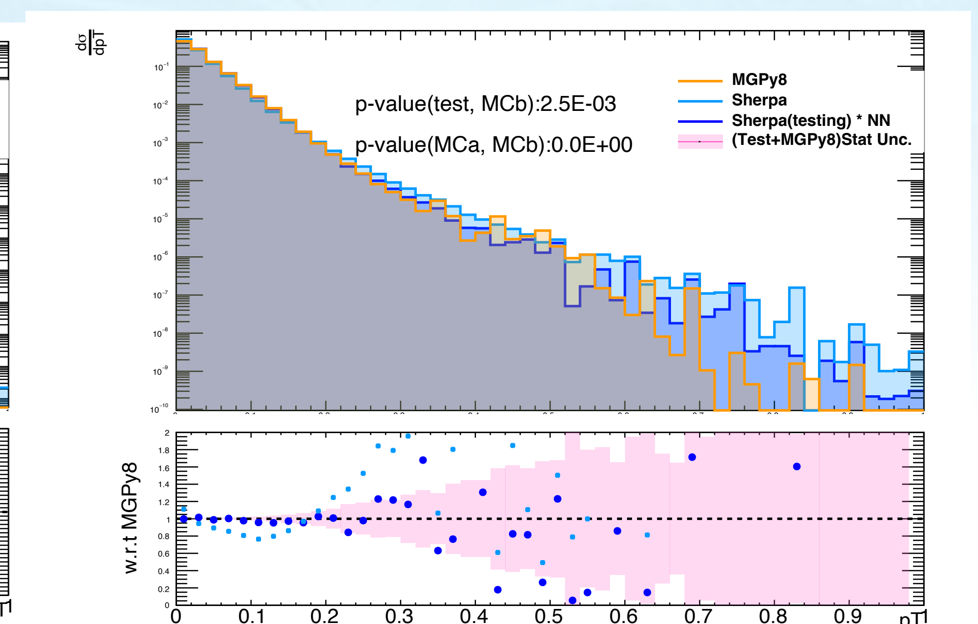
MET



leading jet pT



sub-leading jet pT



3rd jet pT

perhaps we can optimize the Φ network for better mapping of the objects

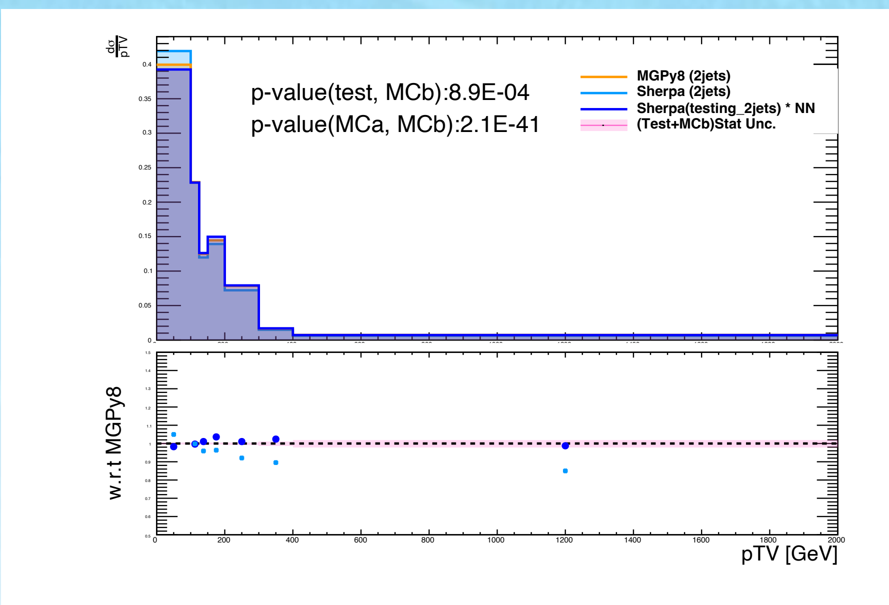
Observables Performance

W/o corrections

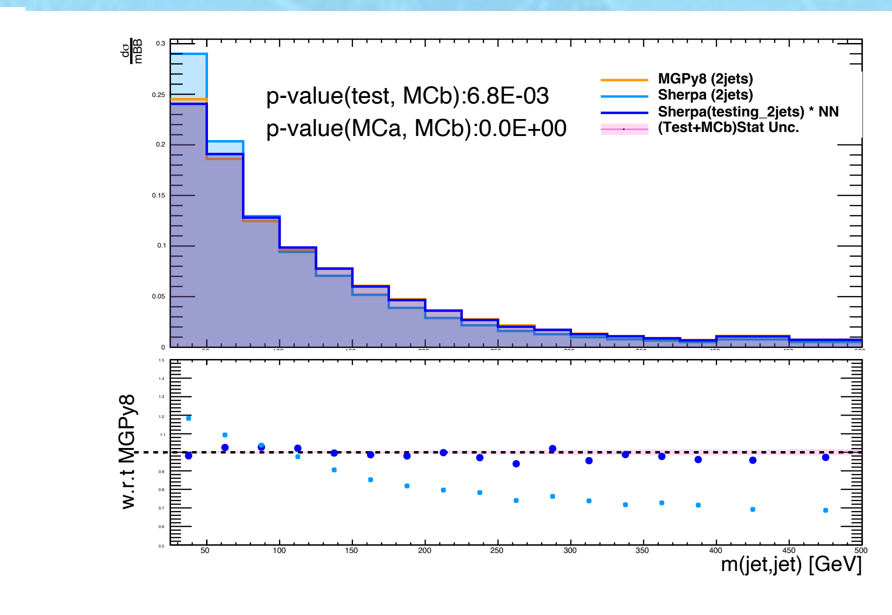
➤ These are evaluated by applying the NN weight to each events and looking at the values of the observables stored in the NTuple.

2-jets

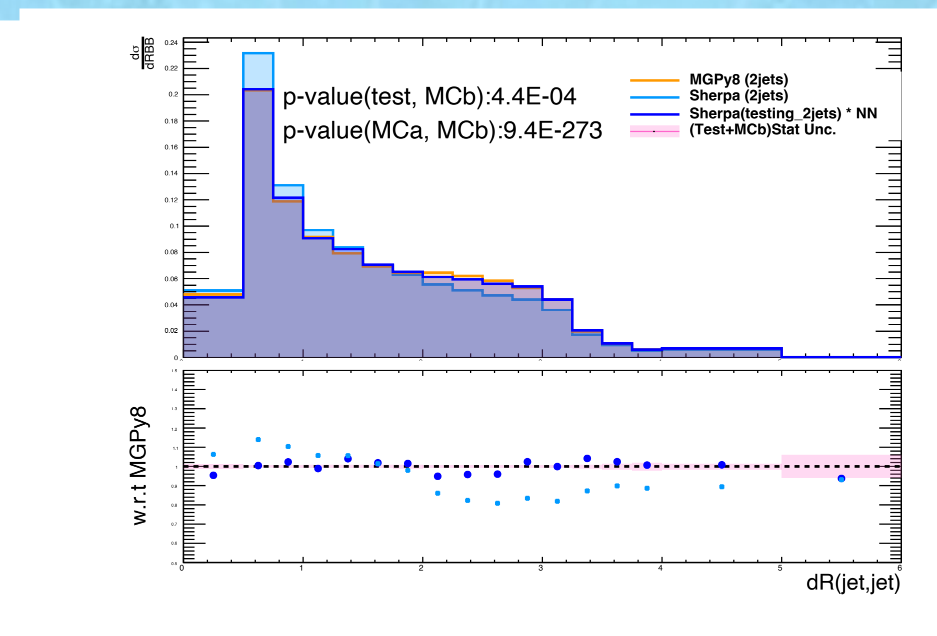
pTV



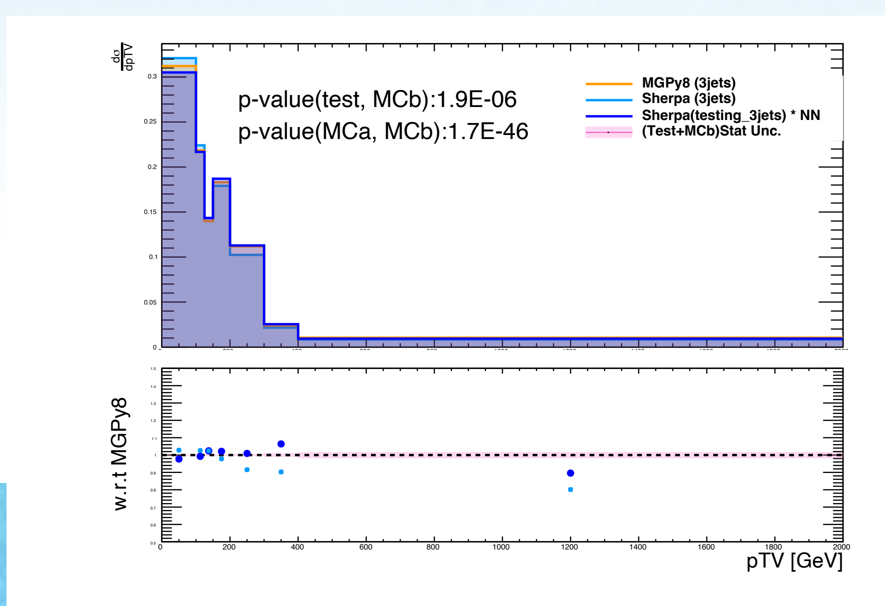
m(jet, jet)

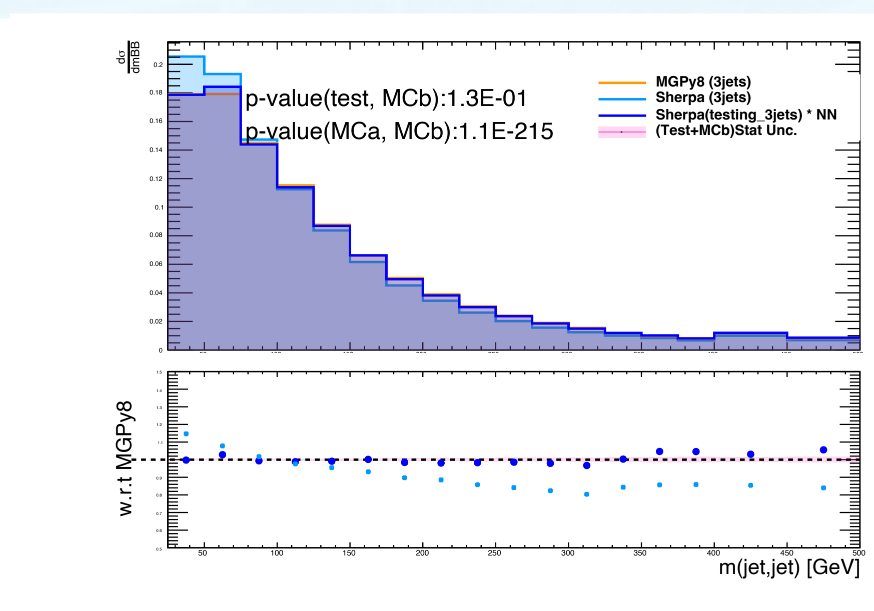


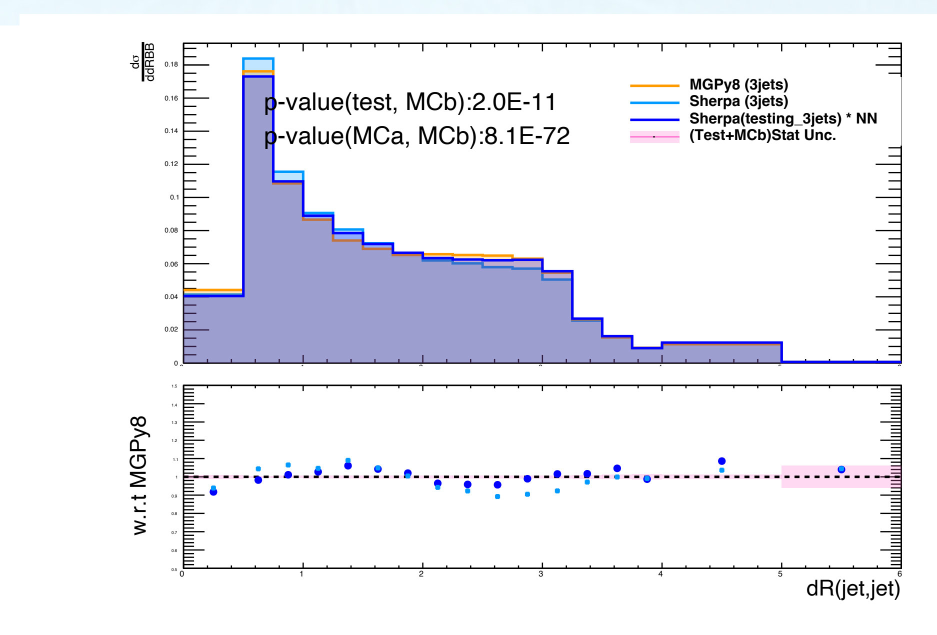
dR(jet, jet)



3-jets





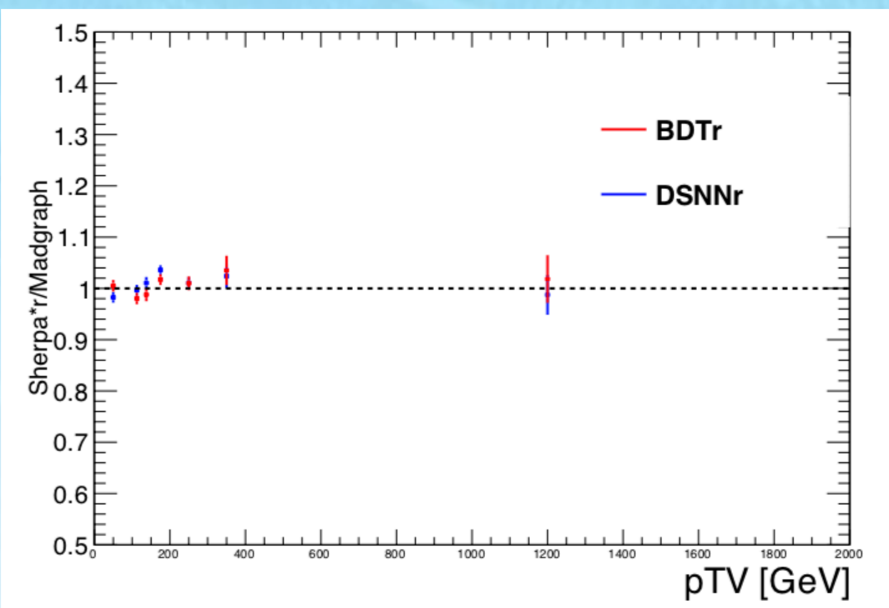


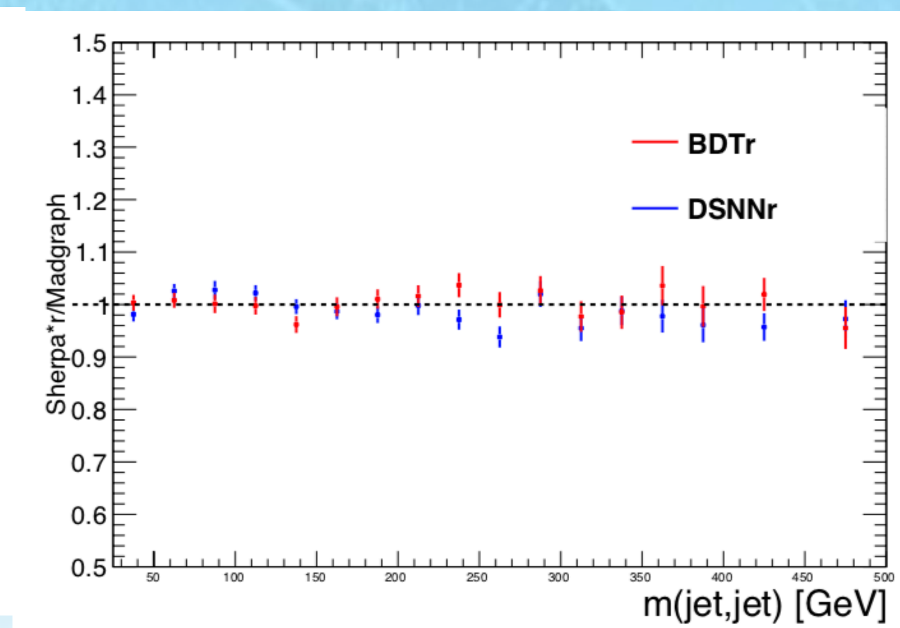
Comparison BDT v.s. DSNN

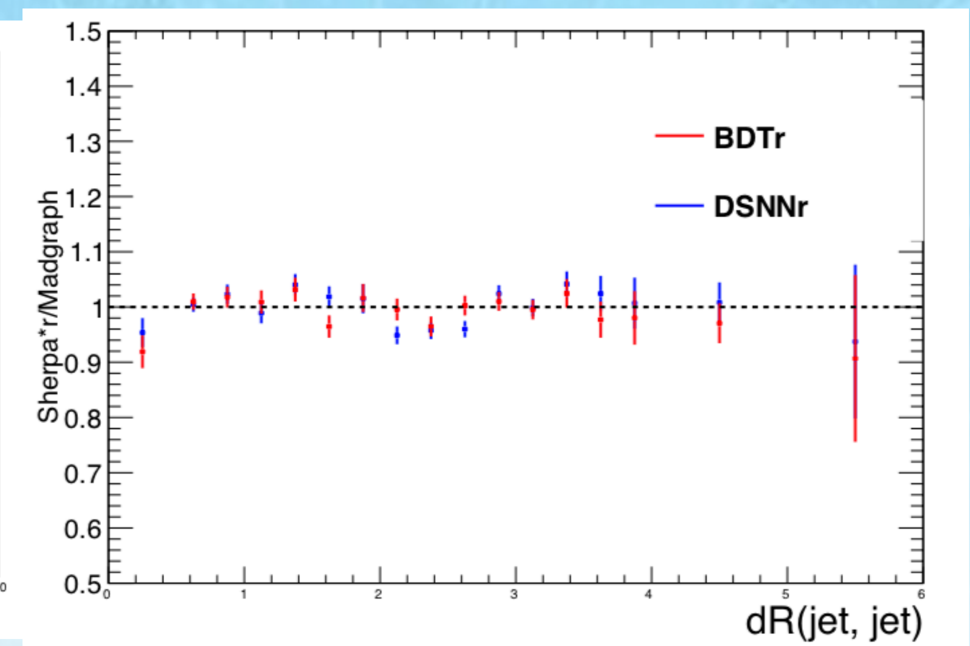
W/o corrections

➤ The performance of each method depended on the kinematics and bins being considered, with the DSNN sometimes outperforming the BDT and vice versa.

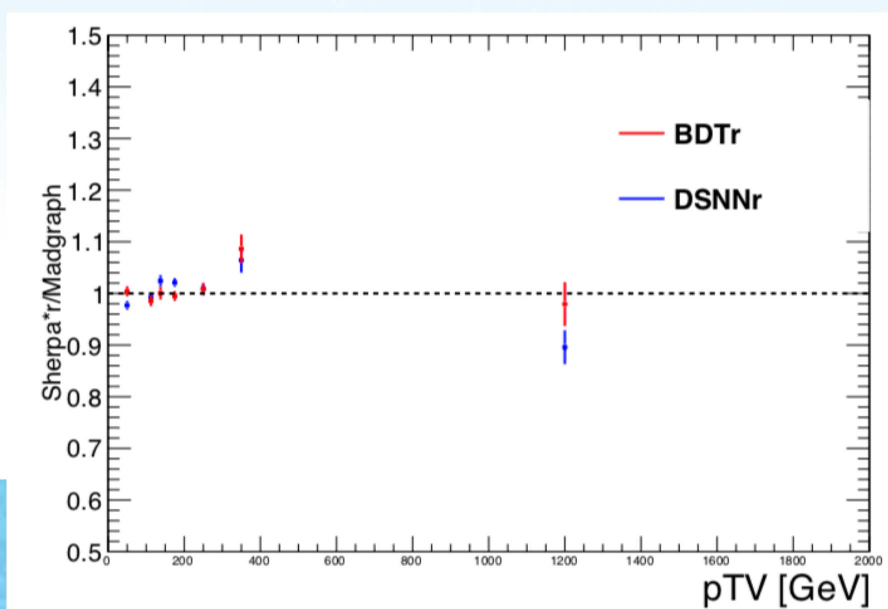
2-jets

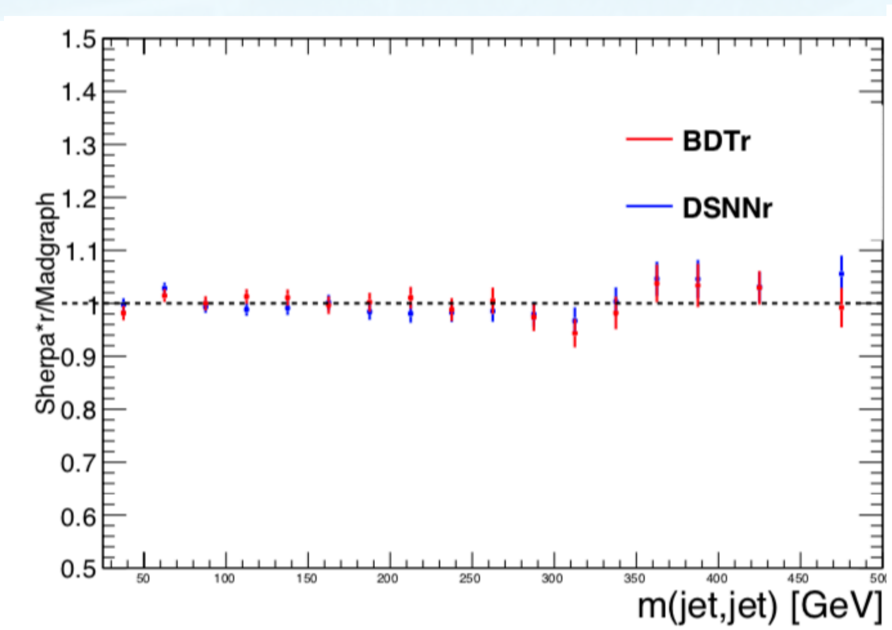


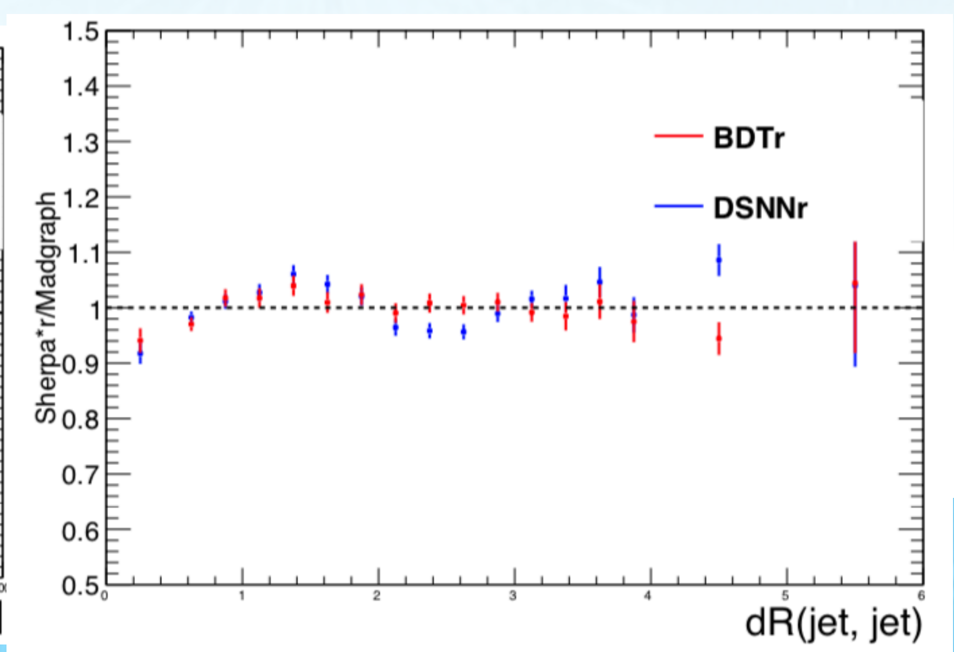




3-jets







Summary

Internal Note, here.

➤ We evaluate the W+jet shape uncertainty using the Deep Sets neural network technique.

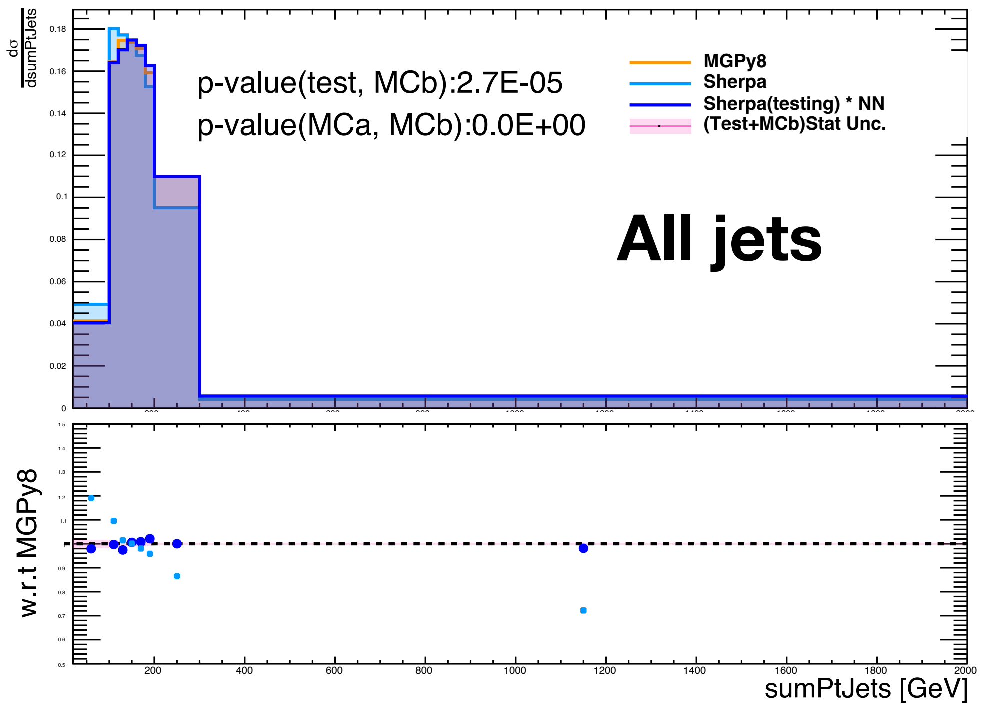
➤ The DSNN showed improved performance when additional particle-level information was incorporated, and residual learning through the use of residual blocks further improved performance.

➤ The performance is similar to the BDT, but with a significant advantage.
- the ability to define analysis independent weights and incorporate them into the nominal MC sample without the need to run on the entire alternative sample.

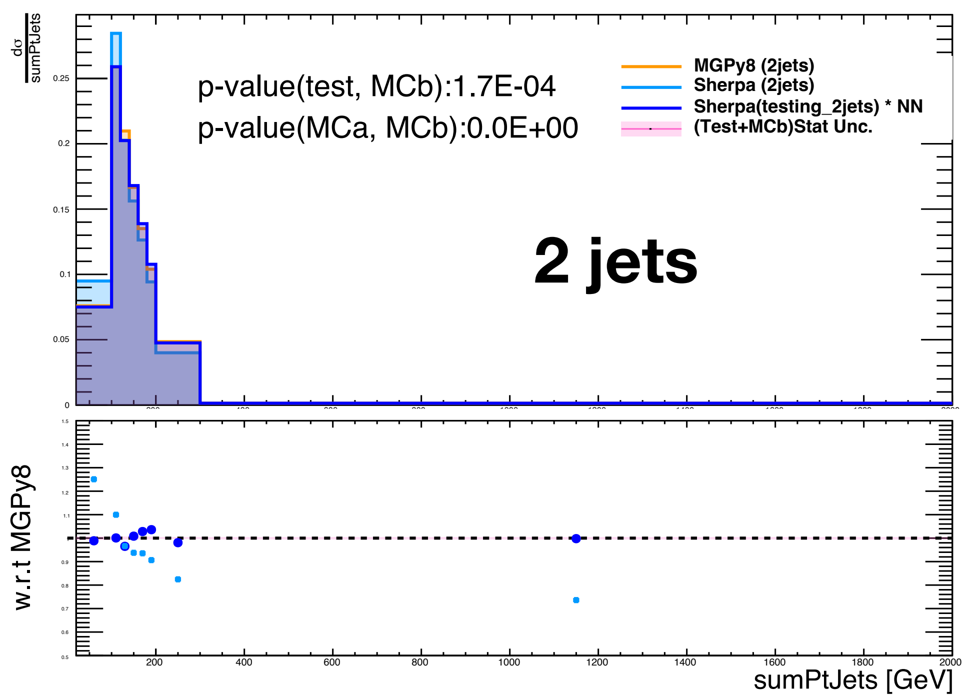
➤ Outlines the next steps to take:
- The results are promising, and further improving the performance of a DSNN is possible (e.g. calibration)
- derive one round of these weights and validate/test it further in additional analyses that are subject to the same kind of systematics.
- if that works well, the approach could be extended to other samples as well.
- We are welcome you to join us to build upon the work!

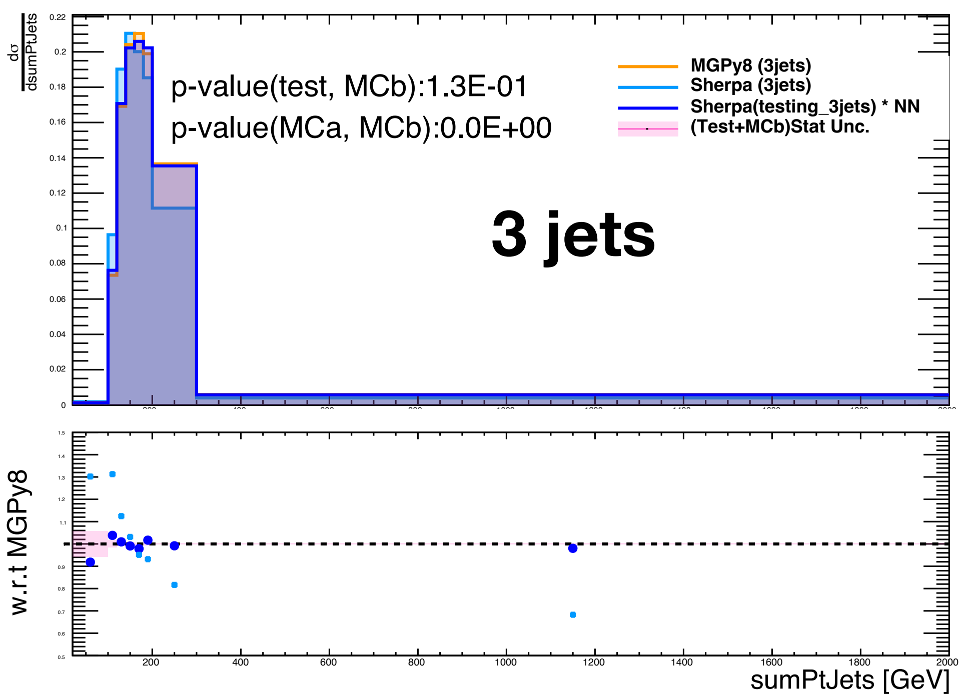
SumPtJets

Scalar sum of the pT of jets



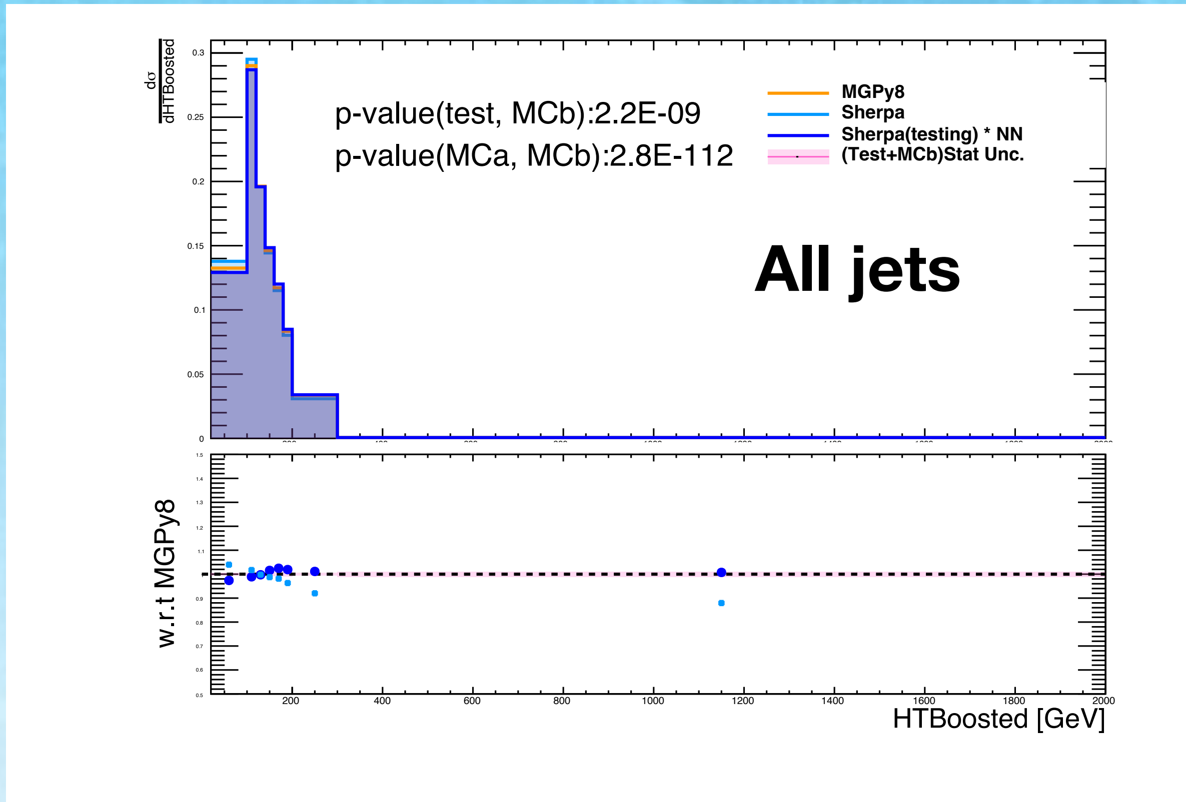

```
//CxAODReader_VHbb
AnalysisReader_VHQQ::computeSumPt(std::vector<const xAOD::Jet *> signalJets,
                                  std::vector<const xAOD::Jet *> forwardJets) {
  double sumPt = 0;
  for (unsigned int s_i = 0; s_i < signalJets.size(); s_i++) {
    sumPt += signalJets.at(s_i)->pt();
  }
  for (unsigned int f_i = 0; f_i < forwardJets.size(); f_i++) {
    sumPt += forwardJets.at(f_i)->pt();
  }
  return sumPt;
}

// Attention: this is before b-jet corrections
  double sumpt = computeSumPt(signalJets, forwardJets);
  m_tree->sumPtJets = sumpt / 1e3;
```





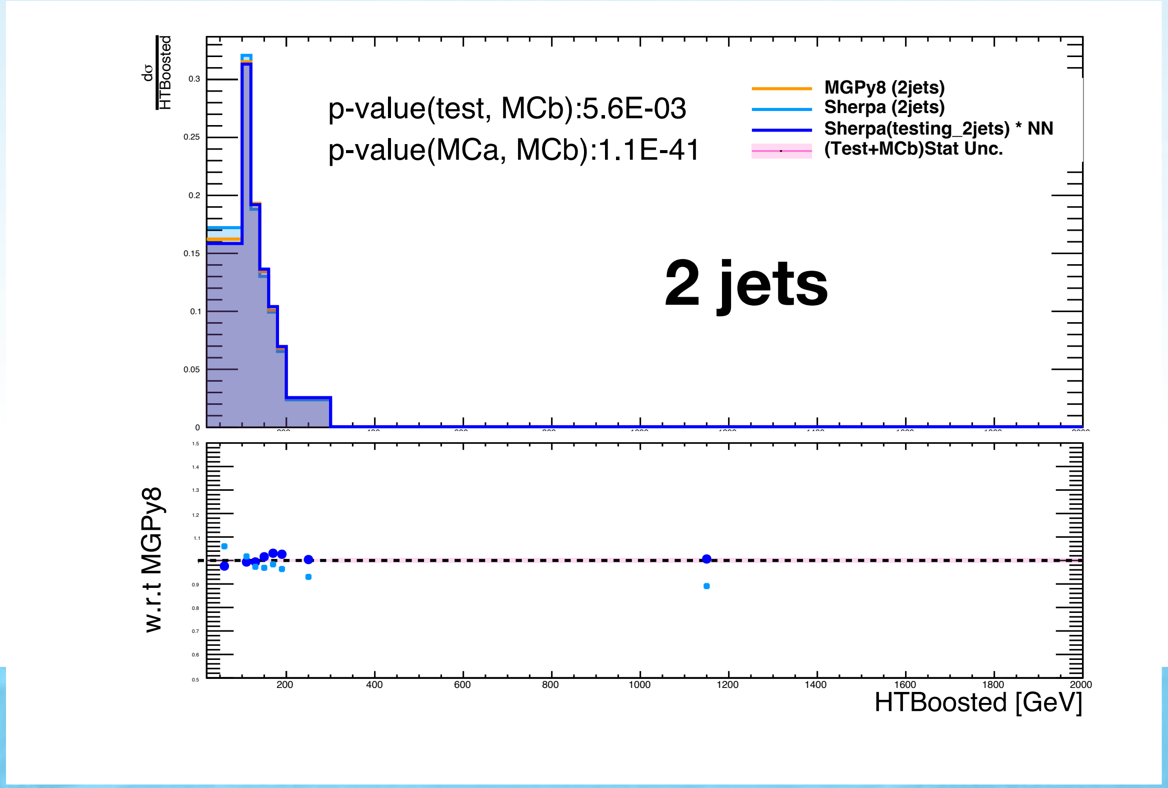
HTBoosted

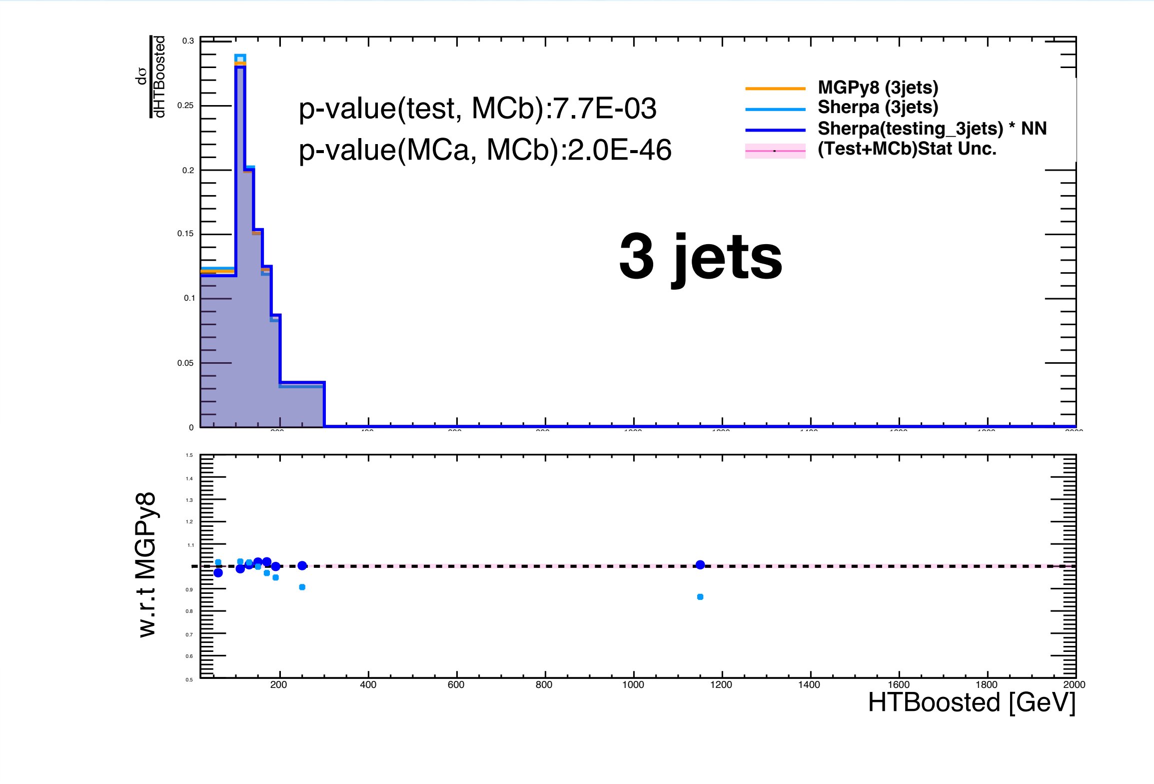
Scalar sum of the pT of all the objects



//CxAODReader_VHbb
//vector boson pt: VVec.Pt()
//const std::vector<const xAOD::Jet *> fatJets
//const int nAdditionalCaloJets, const float pTAddCaloJets

m_tree->HTBoosted = VVec.Pt() / 1e3 + fatJet.Pt() / 1e3 + pTAddCaloJets;





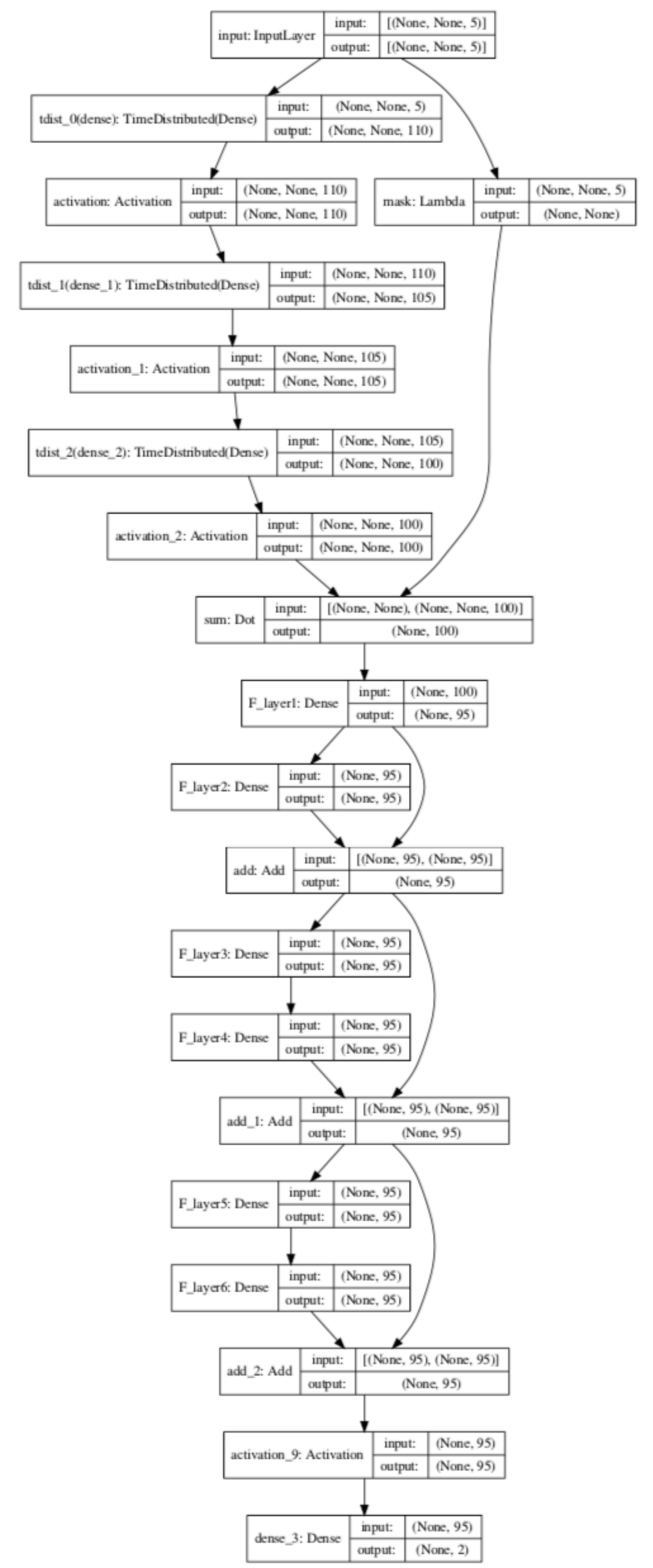
Variables	Description
$m_{j,j}$	Invariant mass of two Higgs boson candidate jets
$\Delta R(j,j)$	Distance between the two Higgs boson candidate jets
p_T^{j1}	Transverse momentum of the leading jet
p_T^{j2}	Transverse momentum of the sub-leading jet
p_T^V	Transverse momentum of the vector boson
$E_{\mathrm{T}}^{\mathrm{miss}}$	Missing transverse energy
$\Delta\ \phi$(V,H)	Distance in ϕ between the vector boson and the Higgs boson candidate
$min(\Delta\phi(l,jet))$	Distance in ϕ between the lepton and the closest jet
m_T^W	Transverse mass of the W boson
$\Delta Y(W,H)$	Difference in rapidity between the W boson and the Higgs boson candidate
m_{top}	Mass of the top quark decaying leptonically
	Only in 3-jet events
$p_T^{jet_3}$	Transverse momentum of the leading un-tagged jet
m_{jjj}	Invariant mass of the two tagged jets and the leading un-tagged jet

Table 4.1: List of variables used in the BDT 1-lepton channel. [15]

Training Setting	Value	Definition
BoostType	adaboost	Boost procedure
Shrinkage	1.0	Learning rate
SeparationType	Gini index	Node separation gain
PruneMethod	No Pruning	Pruning method
NTrees	200	Number of trees
MaxDepth	5	Maximum tree depth
nCuts	250	Number of equally spaced cuts tested per variable per node
nEventsMin	0%	Minimum number of events in a node (% of total events)

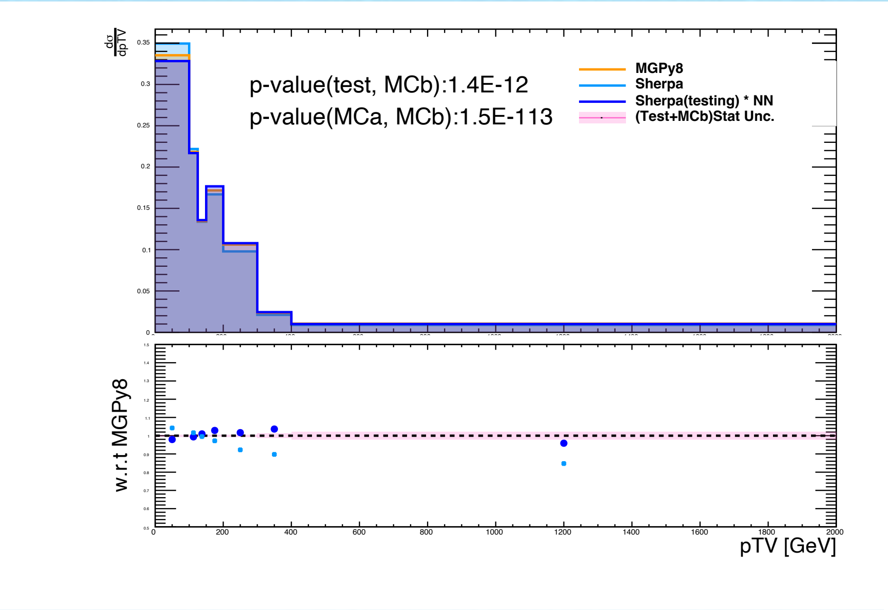
Table 4.2: List of hyper-parameters used in the BDT 1-lepton channel. [15]

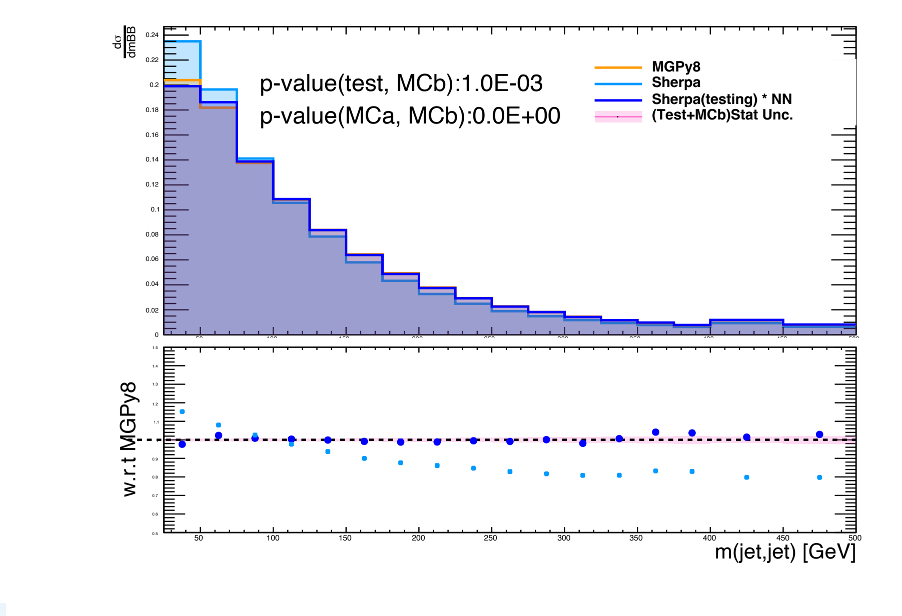
The display of the DSNN model structure

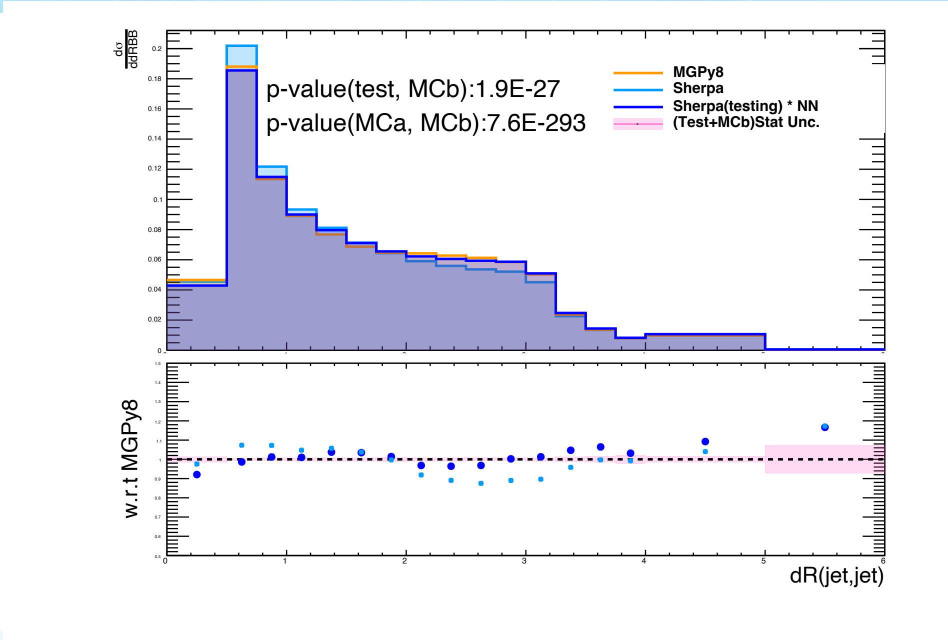


Observables Performance

all-jets

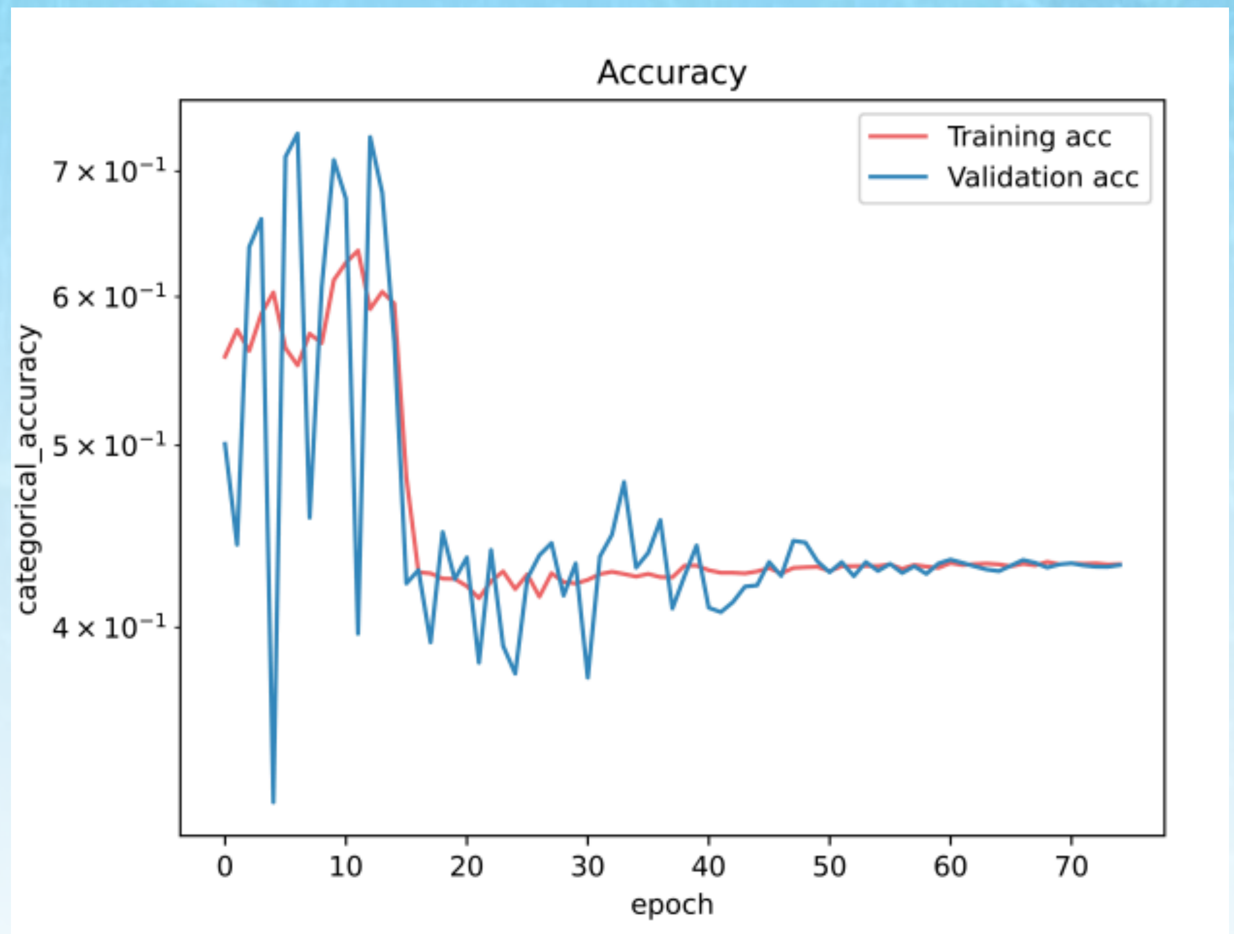






Accuracy

➤ Compare the predicted labels with the true labels event by event to calculate the accuracy for each epoch.



$$Y_{True} = \begin{bmatrix} [0,1] \\ [0,1] \\ [1,0] \end{bmatrix}$$

$$Y_{Pred} = \begin{bmatrix} [0.45, 0.55] \\ [0.65, 0.35] \\ [0.53, 0.47] \end{bmatrix} \xrightarrow{\text{argmax()}} [[0,1], [1,0],[1,0]]$$

Acc = the number of correctly predicted / total number of events

➤ argmax() function may be the reason causing accuracy drops because it may not be the best way to convert the predicted probabilities given Sherpa and Madgraph used to describe physics processes are similar.