DNNs in W mass and W pT measurement

Mykola Khandoga

IN2P3/IRFU Machine Learning workshop

January 23, 2020

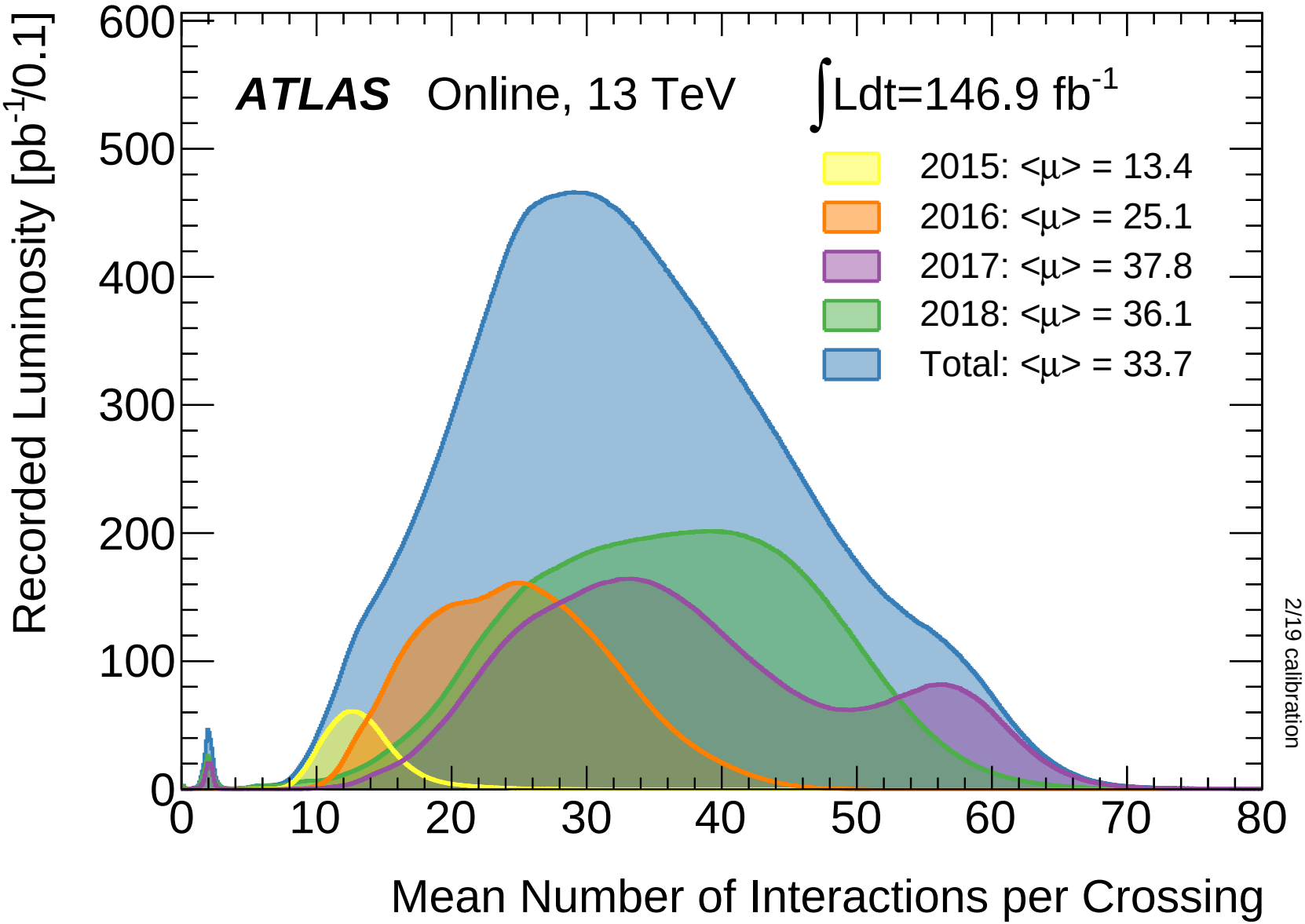
Introduction

W mass and W pT measurements in ATLAS

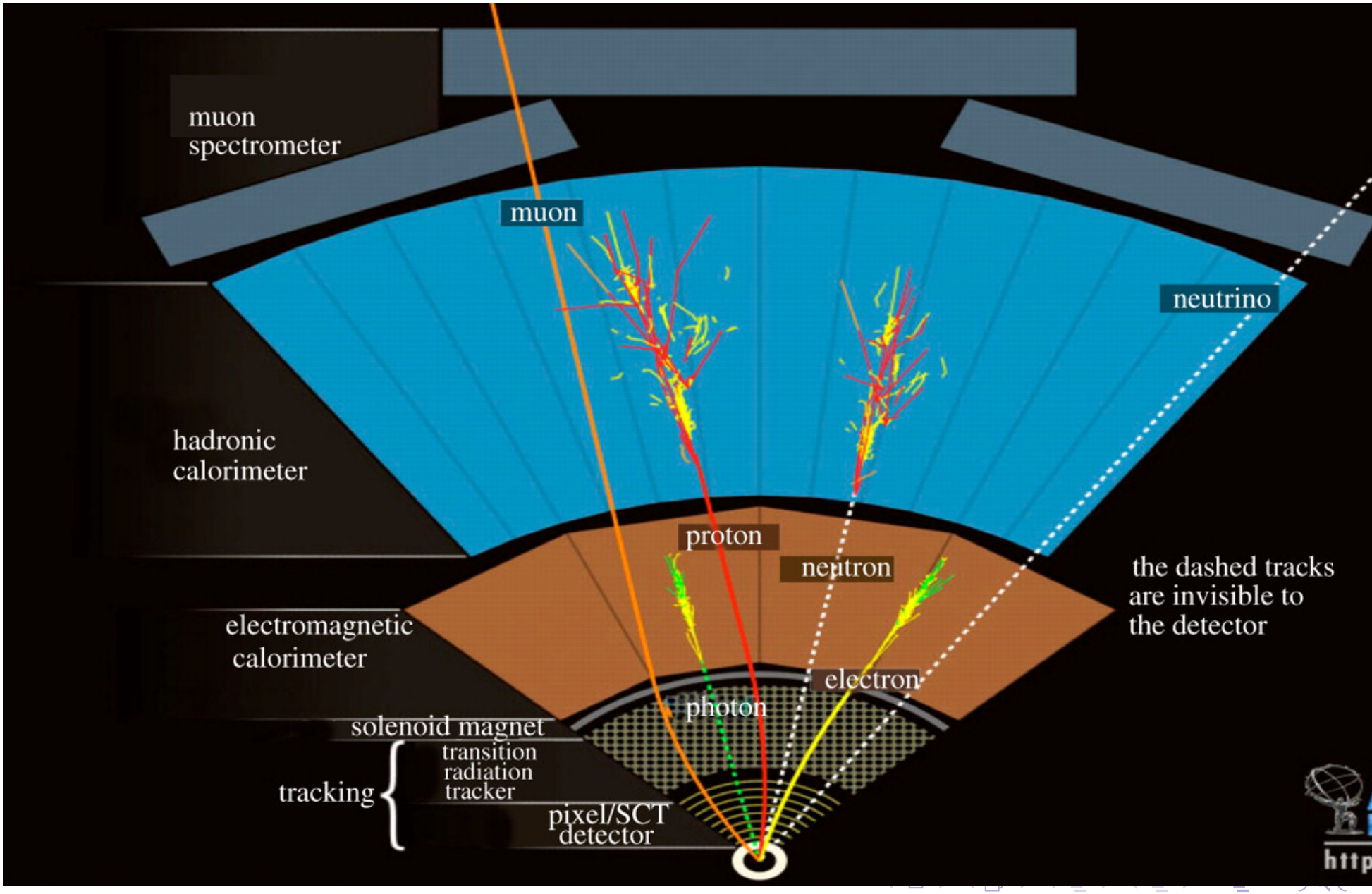
W boson has proven to be a challenging object for study in hadron colliders.

$$W^{\pm} \rightarrow l^{\pm} + \nu$$

So far the only measurements of W pT spectrum at LHC are ATLAS and CMS results from 2010 at 7 TeV.
The only LHC measurement of W boson mass is the ATLAS analysis of 2011 data.
Special low pile-up LHC runs of 2017 and 2018 have opened new possibilities for the measurement of W boson mass and pT spectrum.


ATLAS Online, 13 TeV
∫Ldt=146.9 fb-1
2015: <μ> = 13.4
2016: <μ> = 25.1
2017: <μ> = 37.8
2018: <μ> = 36.1
Total: <μ> = 33.7
Recorded Luminosity [pb-1/0.1]
Mean Number of Interactions per Crossing
2/19 calibration
0
100
200
300
400
500
600
0
10
20
30
40
50
60
70
80

The ATLAS detector



Observables

The transverse momentum of the vector boson comes from the initial state radiation in the transverse plane:

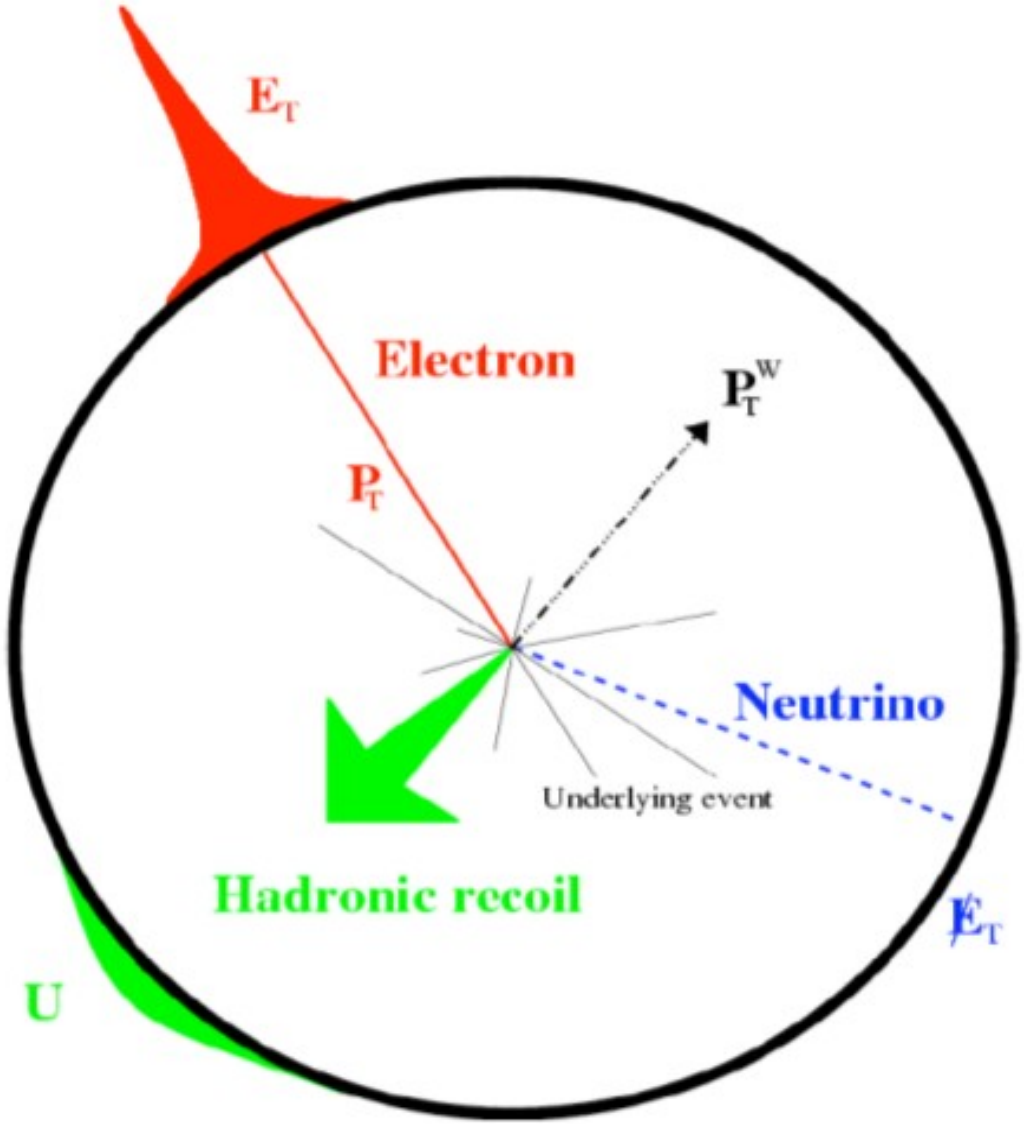
$$\vec{p}_T(W) = \vec{p}_T(l^{\pm}) + \vec{p}_T(\nu) = -\Sigma\vec{p}_T^{ISRpartons}$$

We can only measure the W pT indirectly. The term $\Sigma\vec{p}_T^{ISRpartons}$ stands for the vector sum of all reconstructed PFOs (charged and neutral) excluding the decay leptons and is called the **hadronic recoil** (HR) vector, $\vec{u} = \Sigma\vec{p}_T^{PFOs}$.

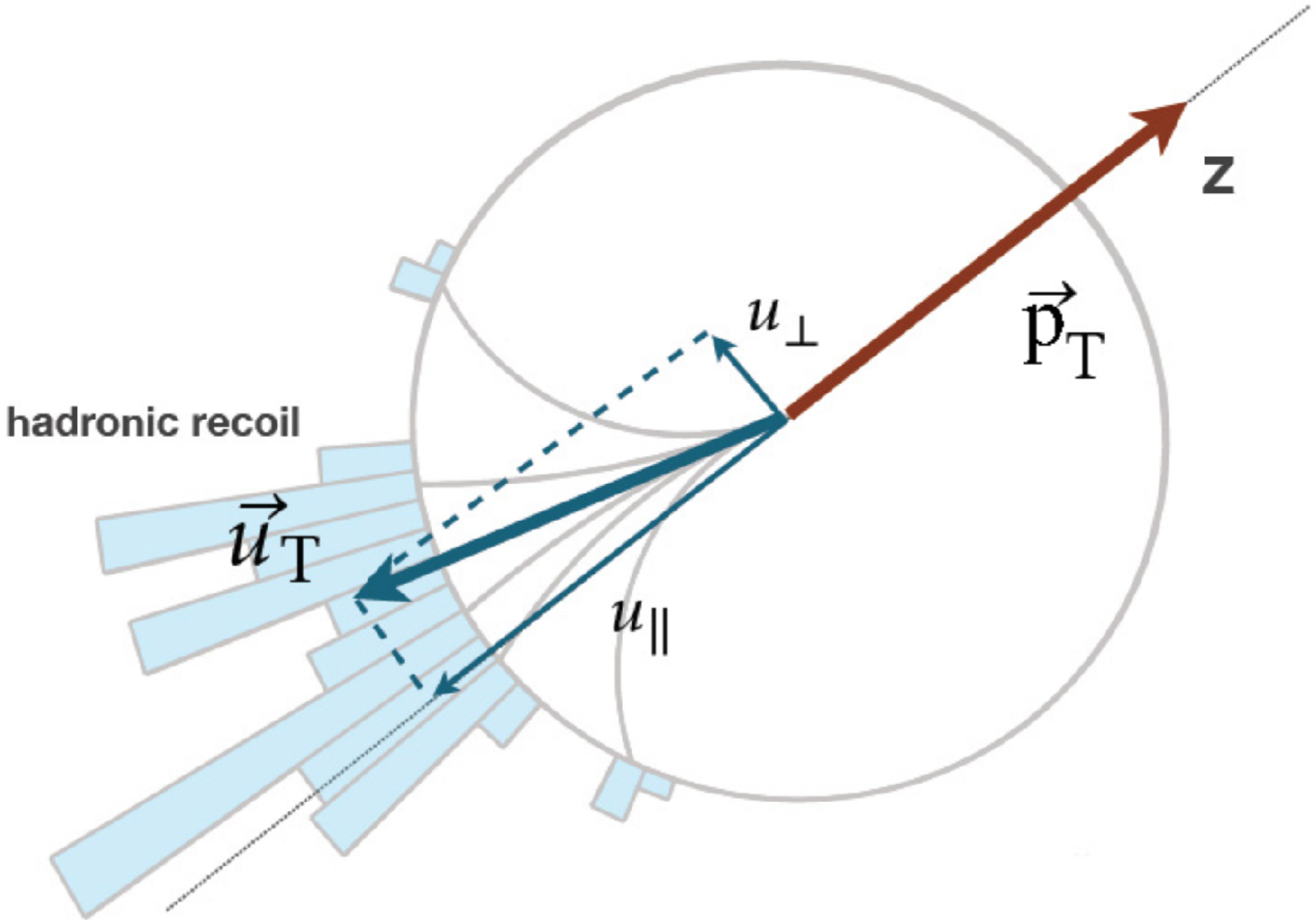
HR is a high-level observable that uses many inputs from lower-level parameters. u_T regression looks like a suitable task for ML.

A 10-15% improvement in HR resolution would be a notable improvement of the precision.

W boson decay



Hadronic recoil



Benchmarking HR measurement

Ideally uT should have the same magnitude as the p_T^W and directed in the opposite way:

$$\vec{u_T} = -\vec{p_T^W},$$

but in reality due to detector effects this is not true (see picture on the previous slide).
In order to estimate bias and the resolution of the HR we introduce the following observables:

$$u_{\parallel} = \vec{p_T^W} \cdot \vec{v_T}$$

$$u_{\perp} = u_x v_y - u_y v_x$$

$u_{\parallel}$ is the detector response.
$1 - \frac{u_{\parallel}}{|p_T^W|}$ represents the bias of our pT measurement.
$\sigma(u_{\perp})$ represents the resolution.

Input features

Features:

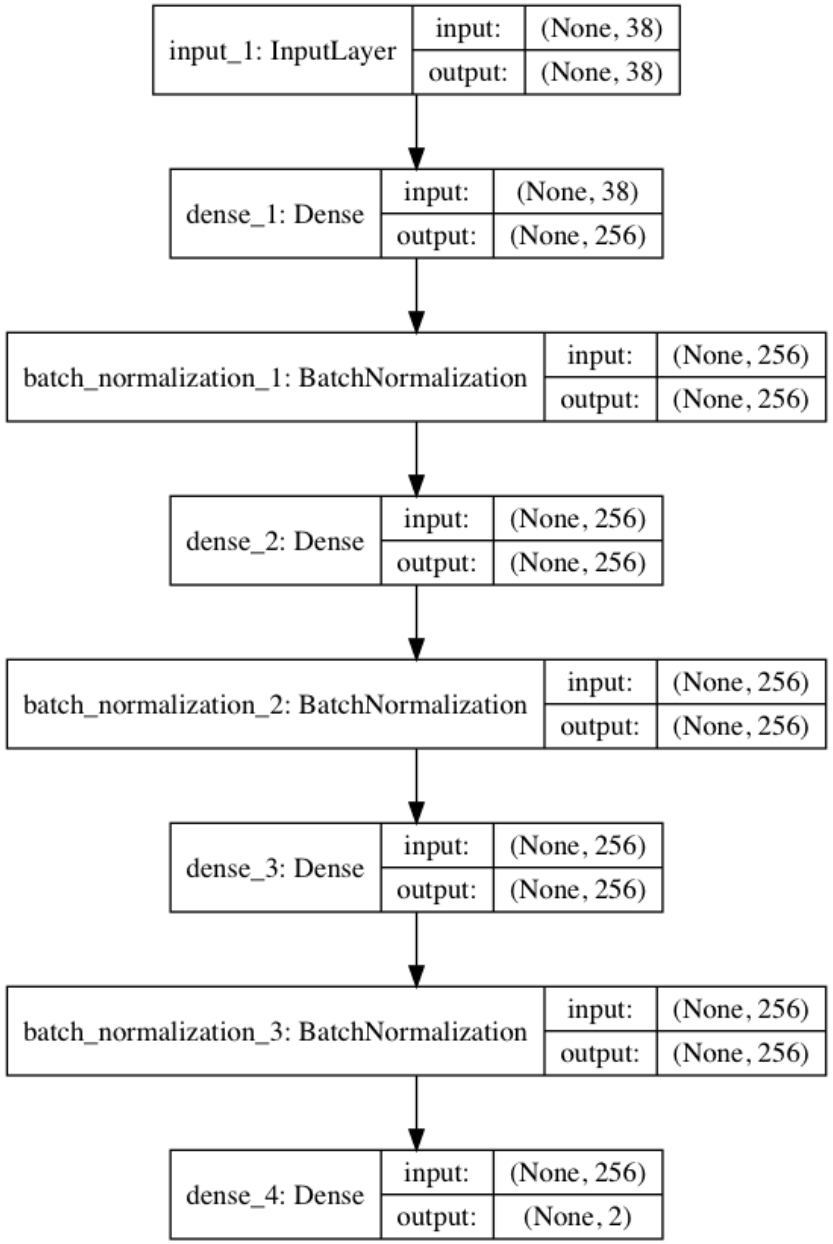
- u_T - 2 components
- u_T_*charged* - 2 components
- u_T_*neutral* - 2 components
- *SumET*
- *SumET*_*charged*
- *SumET*_*neutralEM*
- *lead_jet_px* - leading jet pT, 2 components, cut below 20 GeV
- *ntlead_jet_px* - second to leading jet pT, 2 components, cut below 20 GeV
- *NPV* - number of primary vertices
- *u_nPFO_charged* and *u_nPFO_neutral* - number of charged and neutral PFOs
- *neut_nPFO_pT* , *ch_nPFO_pT* - pTs of the first 5 charged and 5 neutral PFOs, 2 components each

Target values: pT_{truth}_*x*, pT_{truth}_*y*.

The setup

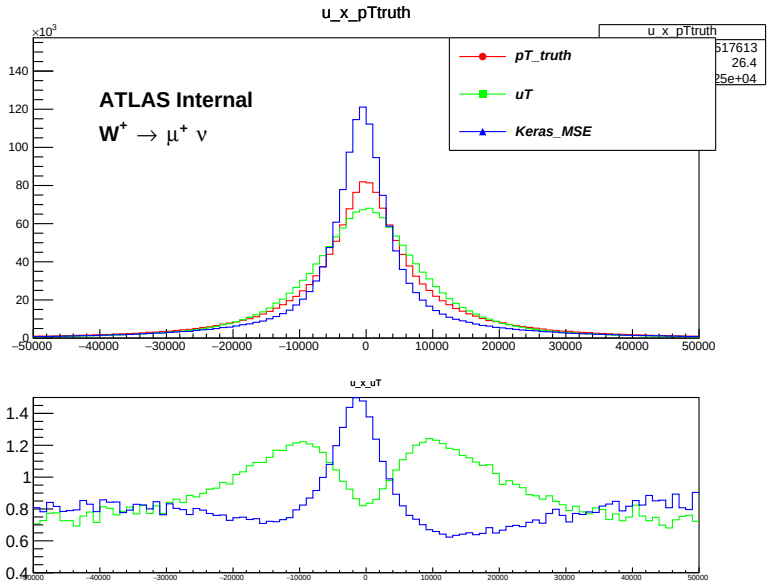
The setup

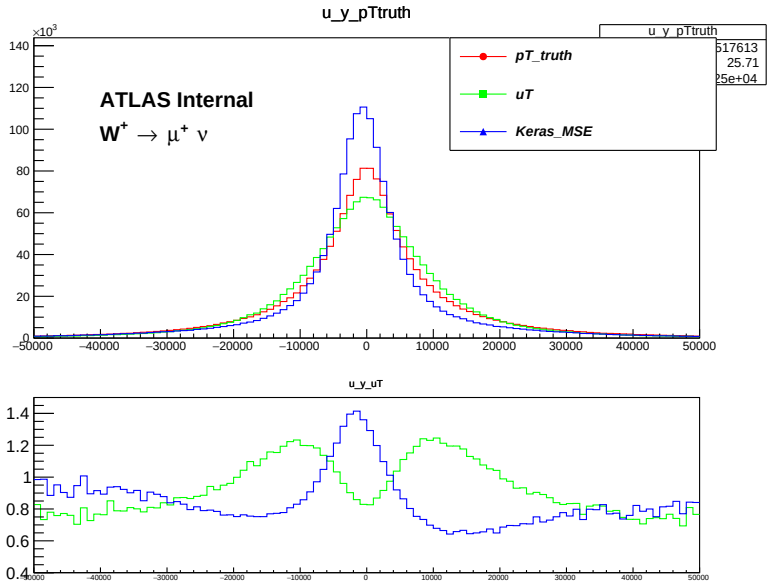
- Training sample: $1.5 \cdot 10^6$ $W \to \mu\nu$ MC events, validation sample: $5 \cdot 10^5$ MC events
- Framework: Keras/TensorFlow
- Loss function: MSE
- Optimizer: ADAM with the step of 0.001
- Batch size: 1500 events
- $\approx$ 50 epochs for optimal training (with batch normalization)
- 3 hidden dense layers 256 nodes each
- batch normalization layer between each dense layer
- LWTNN package is used to integrate the trained NN within the analysis framework

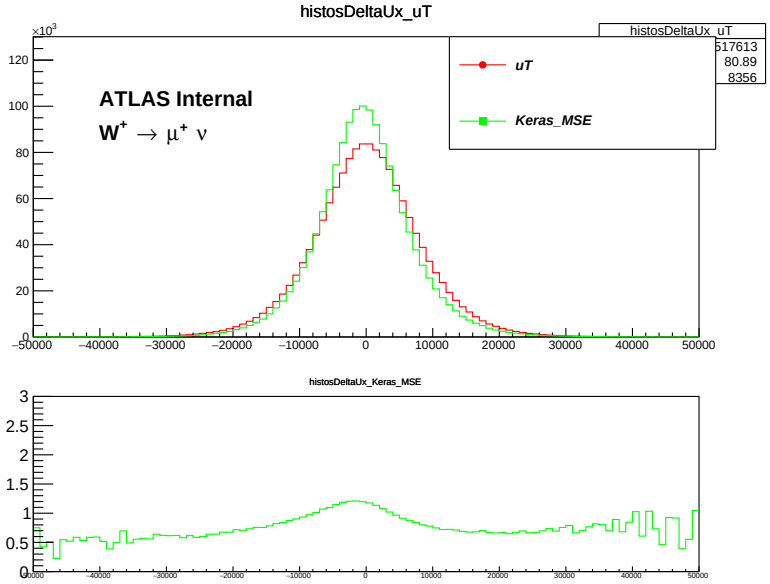

input_1: InputLayer
input: (None, 38)
output: (None, 38)
dense_1: Dense
input: (None, 38)
output: (None, 256)
batch_normalization_1: BatchNormalization
input: (None, 256)
output: (None, 256)
dense_2: Dense
input: (None, 256)
output: (None, 256)
batch_normalization_2: BatchNormalization
input: (None, 256)
output: (None, 256)
dense_3: Dense
input: (None, 256)
output: (None, 256)
batch_normalization_3: BatchNormalization
input: (None, 256)
output: (None, 256)
dense_4: Dense
input: (None, 256)
output: (None, 2)

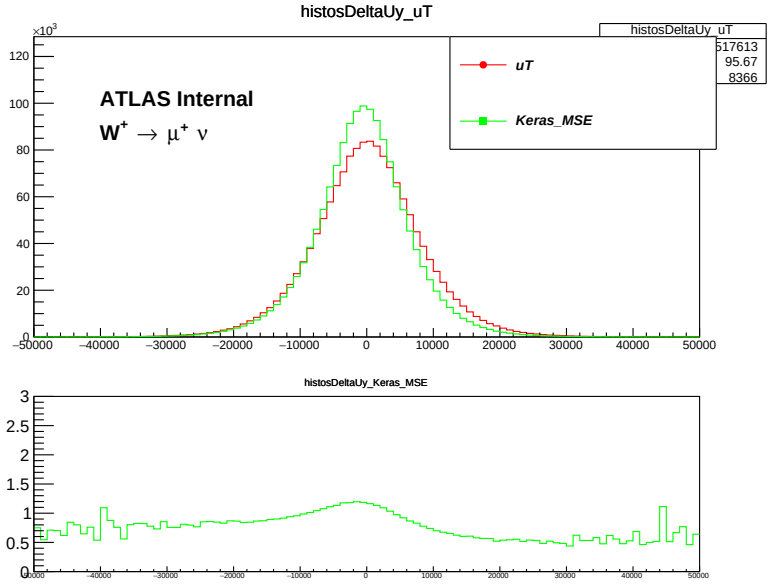
Results

Target comparison – uX, uY

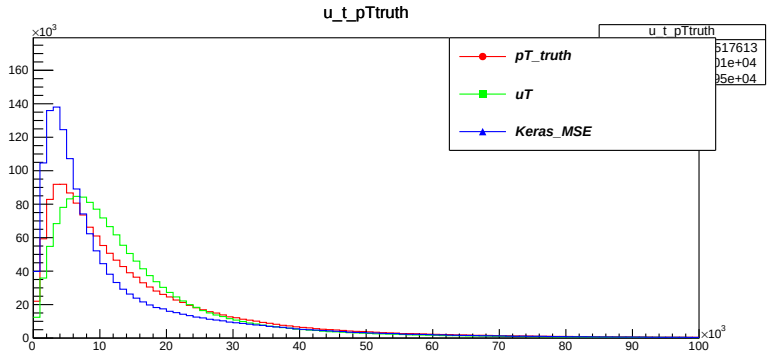


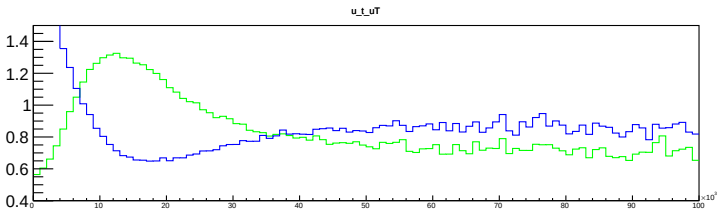


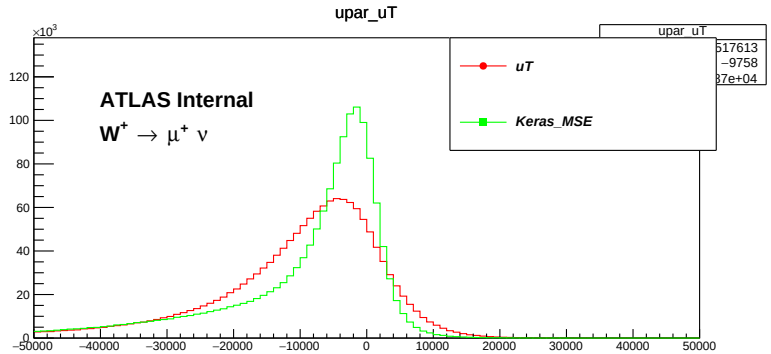


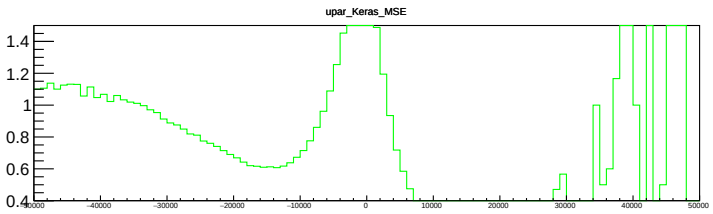


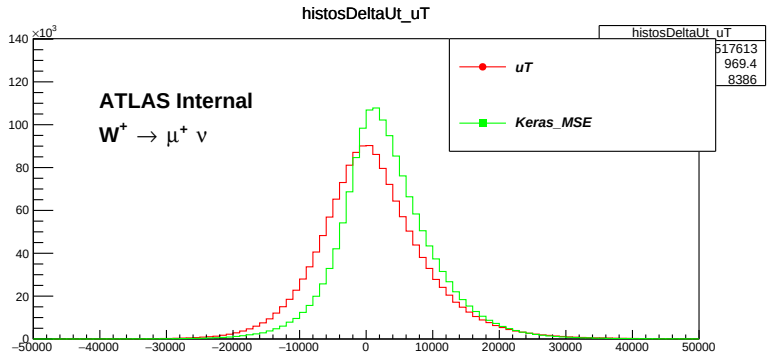
uT, uPar, bias

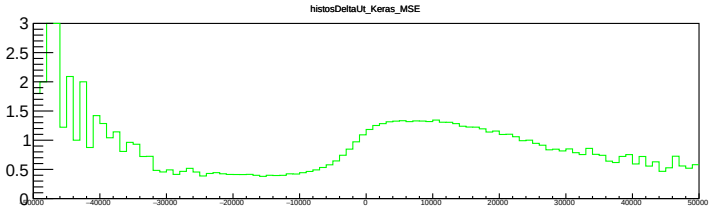

u_t_pTtruth
pT_truth
uT
Keras_MSE

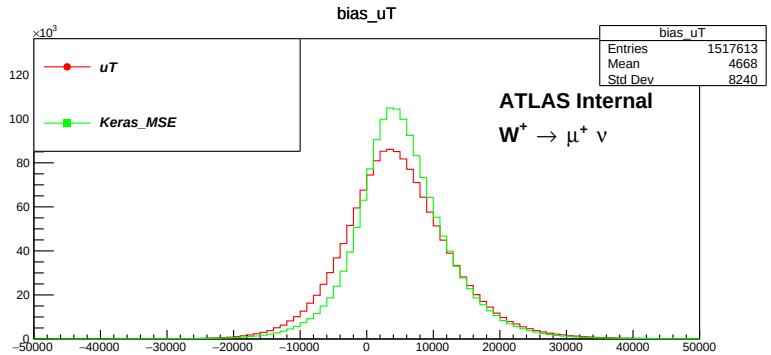

u_t_uT

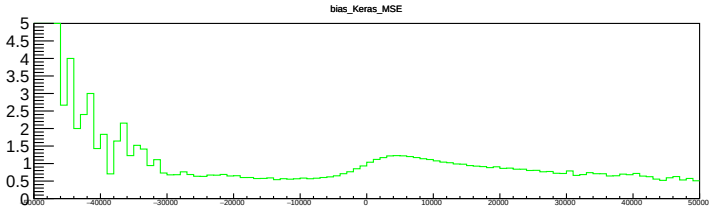

upar_uT
ATLAS Internal
W+ → μ+ ν
uT
Keras_MSE


upar_Keras_MSE

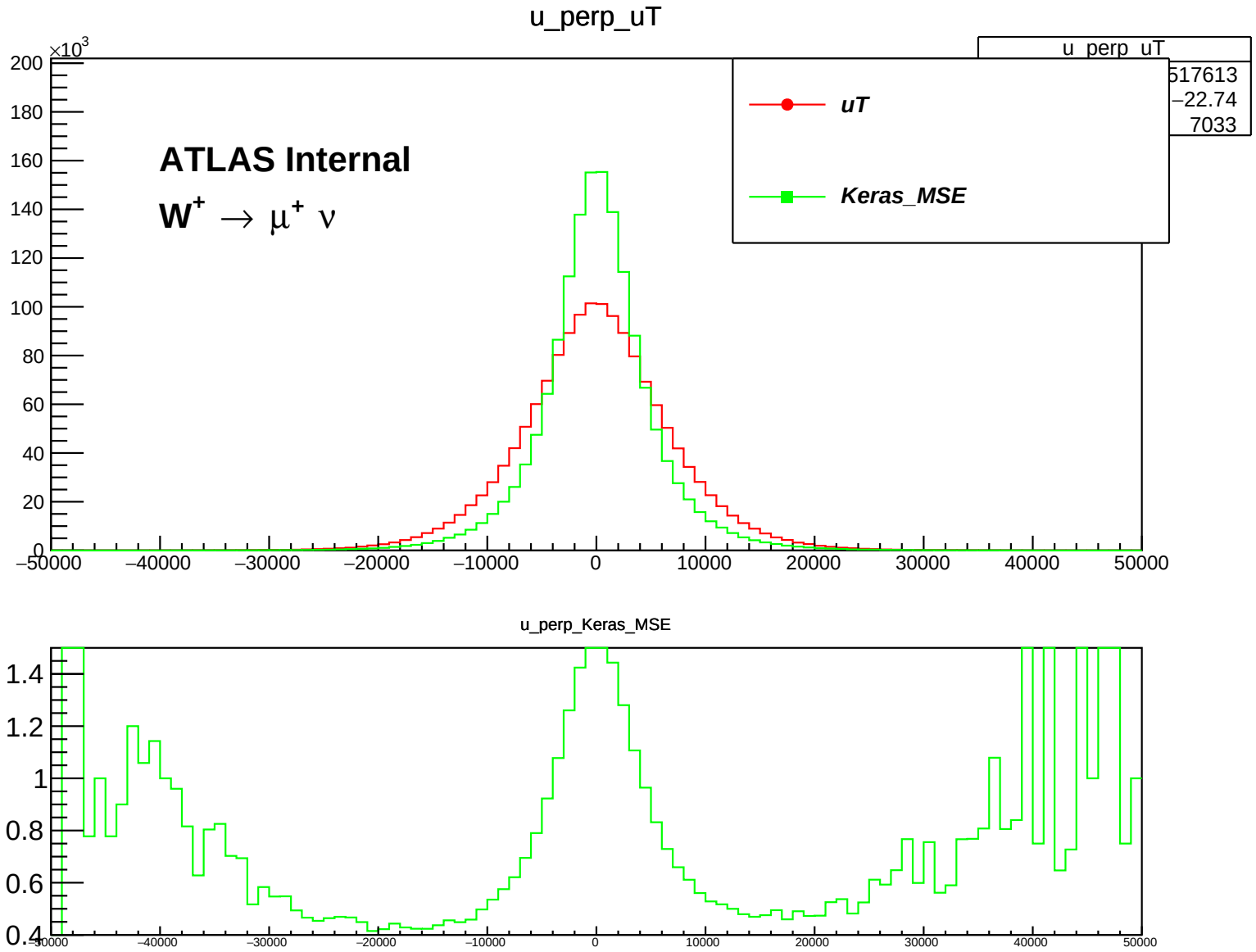

histosDeltaUt_uT
ATLAS Internal
W+ → μ+ ν
uT
Keras_MSE


histosDeltaUt_Keras_MSE

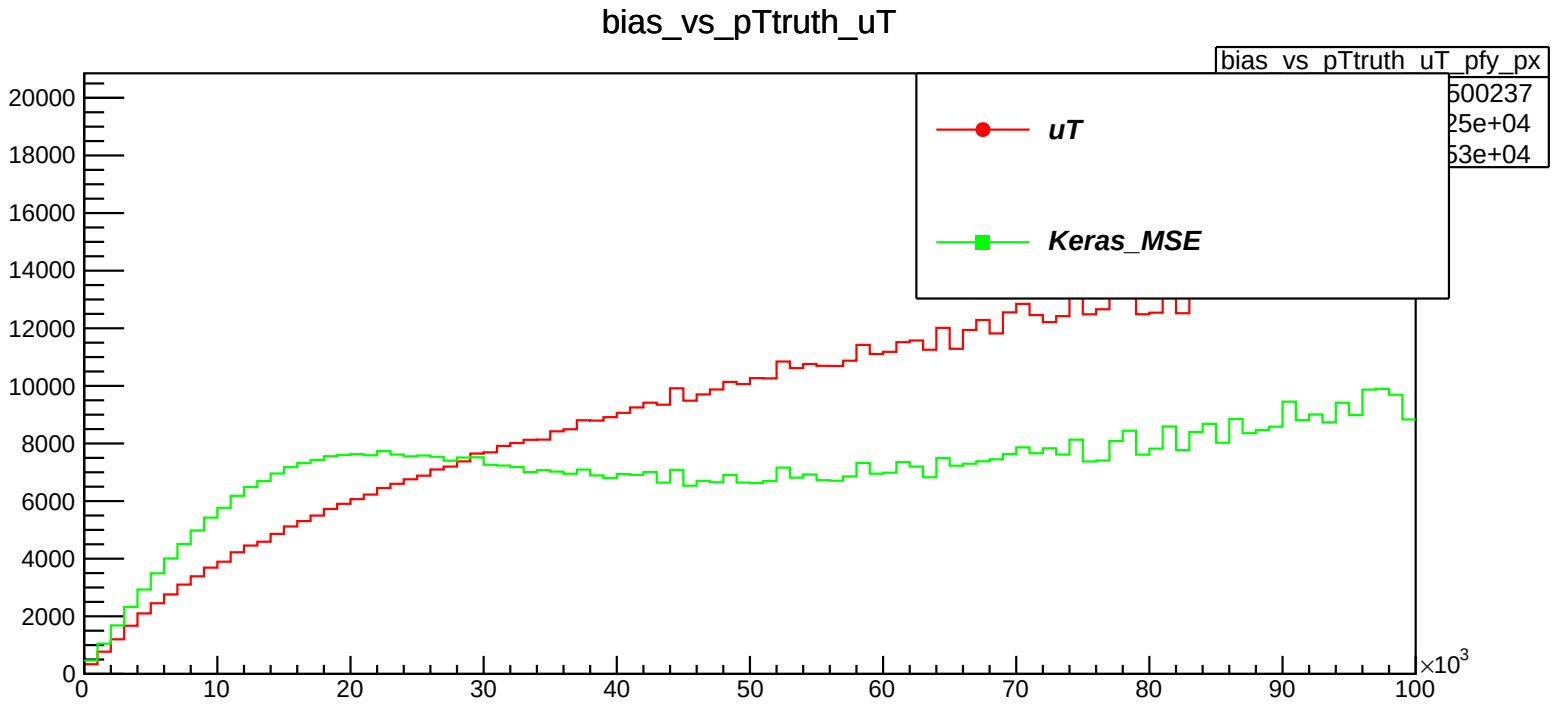

bias_uT
Entries 1517613
Mean 4668
Std Dev 8240
uT
Keras_MSE
ATLAS Internal
W+ → μ+ ν

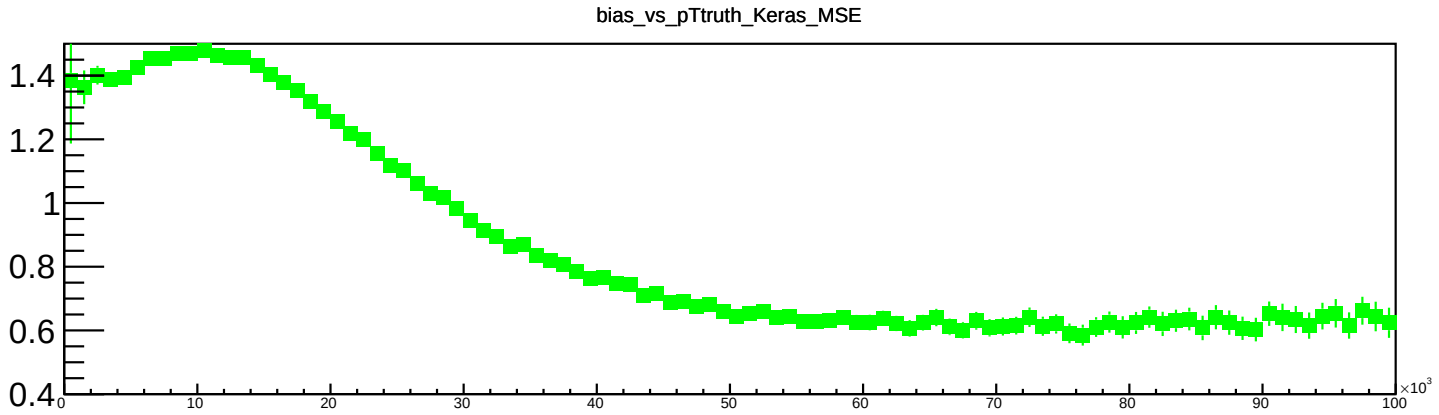

bias_Keras_MSE

uPerp, Integrated

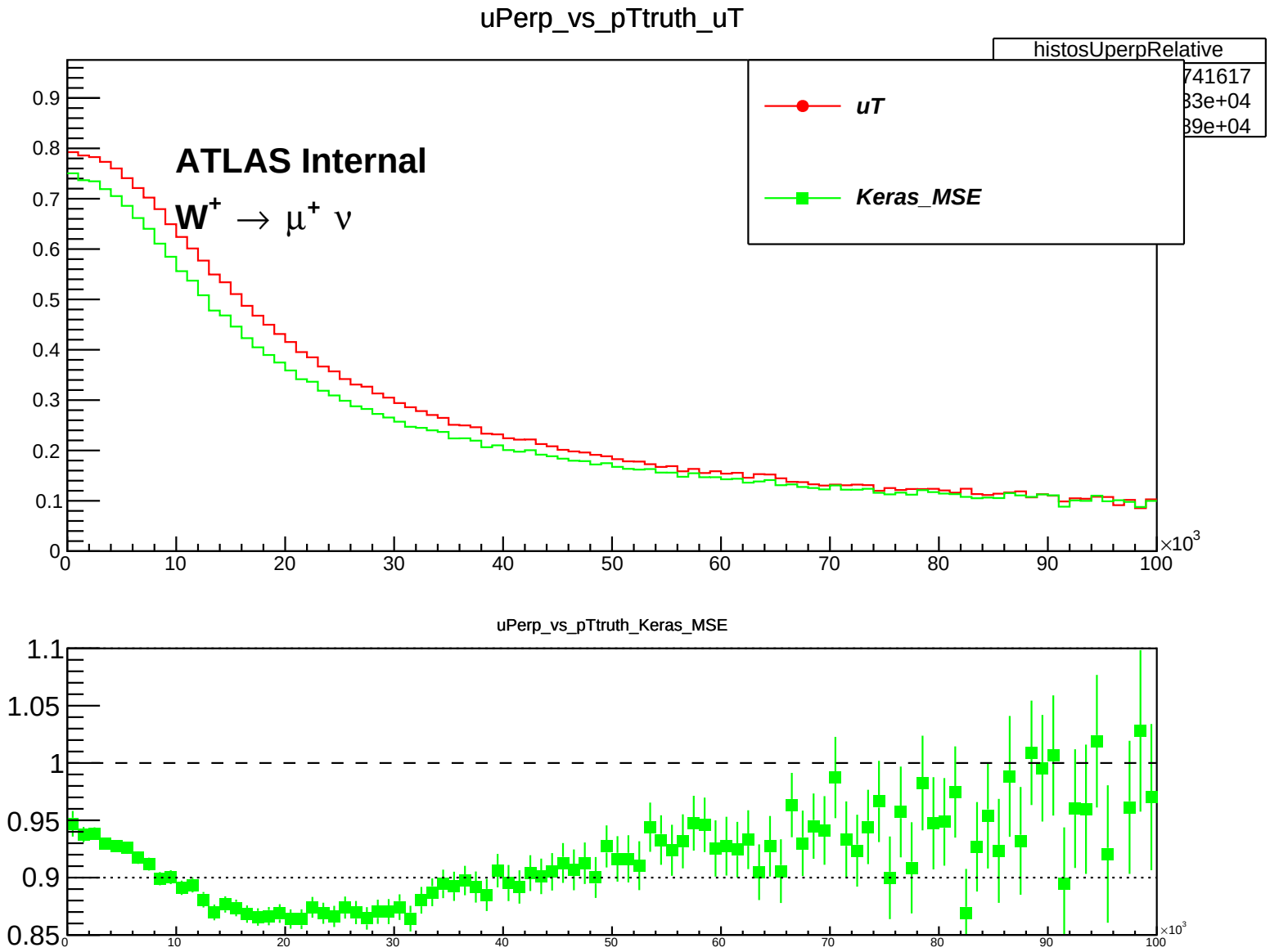


<bias> vs pTtruth





sigma(uPerp)/<uT> vs pTtruth



Attempts to modify the loss function to minimize bias

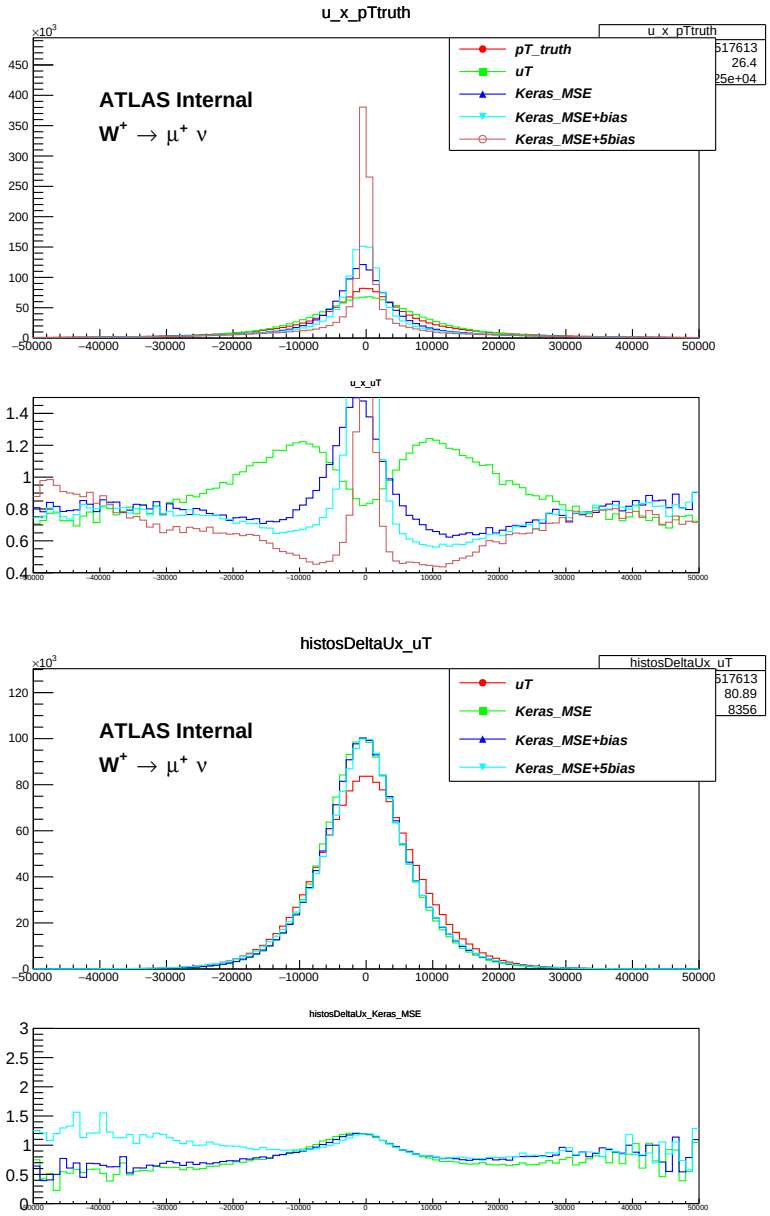
In an attempt to minimize bias I've tried to modify the loss finction, adding a penalizing term for the bias:

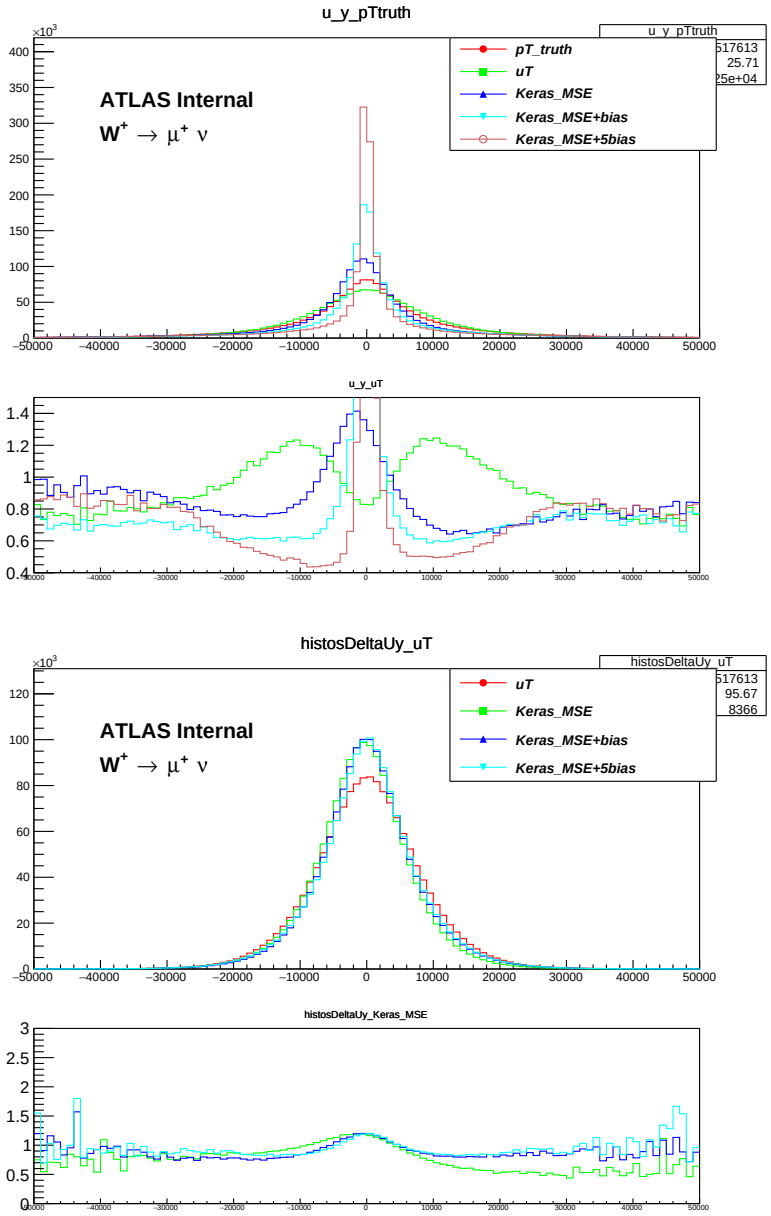
$$\mathcal{L}_1 = < (y_{pred} - y_{true})^2 > + \alpha < (1 - bias)^2 > \quad (1)$$

where α is the normalization coefficient.

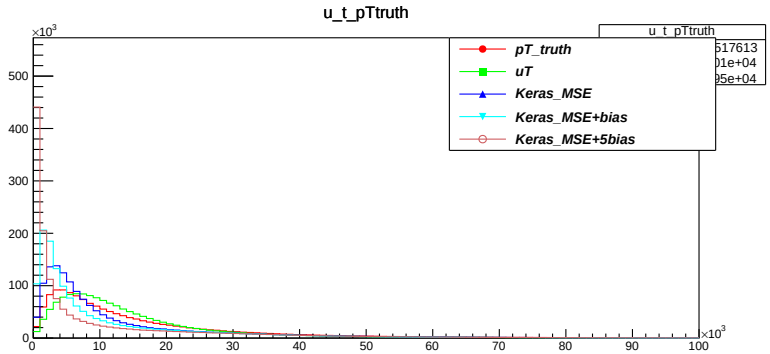
$$\mathcal{L}_2 = < (y_{pred} - y_{true})^2 > + 5\alpha < (1 - bias)^2 > \quad (2)$$

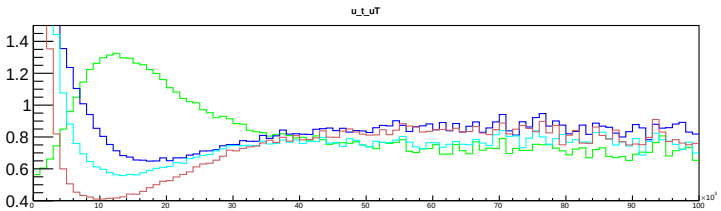
Target comparison – uX, uY

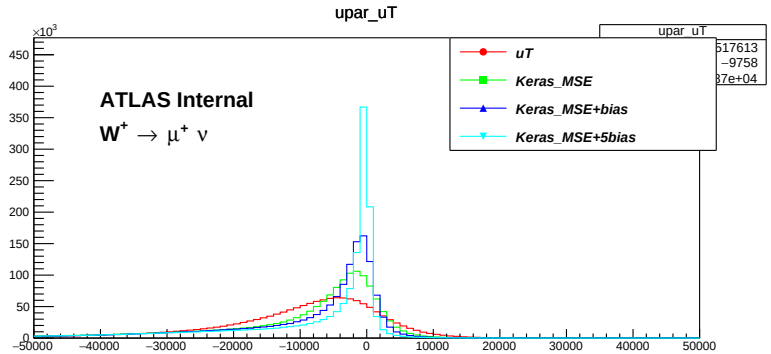


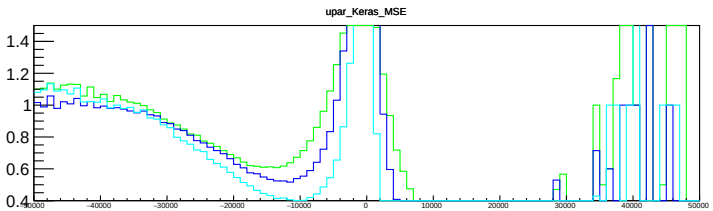


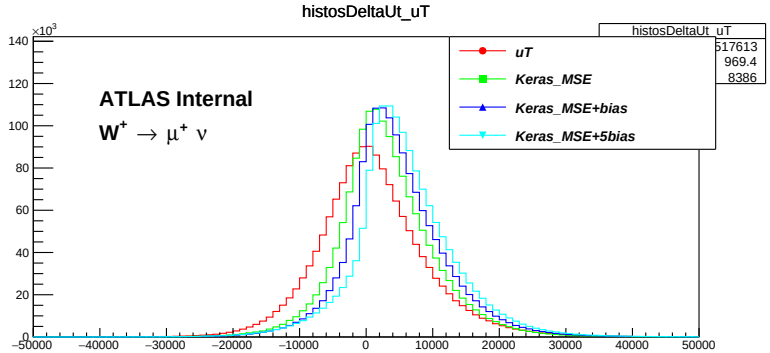
uT, uPar, bias

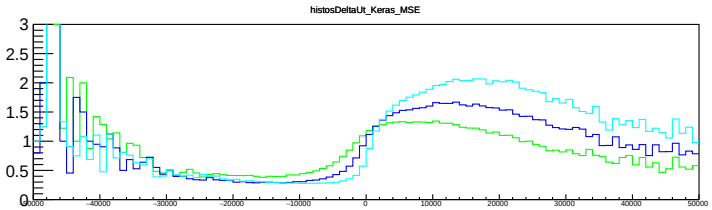

u_t_pTtruth
pT_truth
uT
Keras_MSE
Keras_MSE+bias
Keras_MSE+5bias

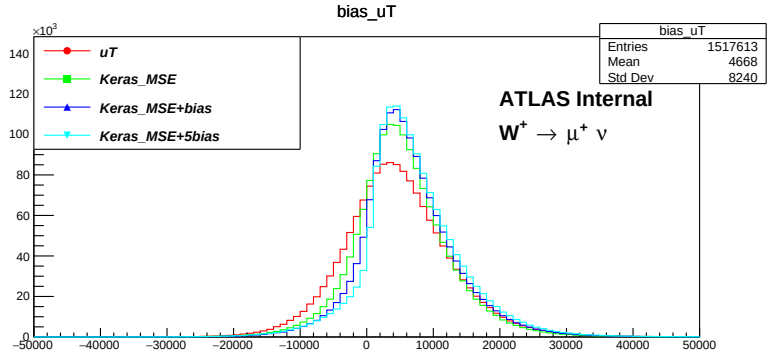

u_t_uT

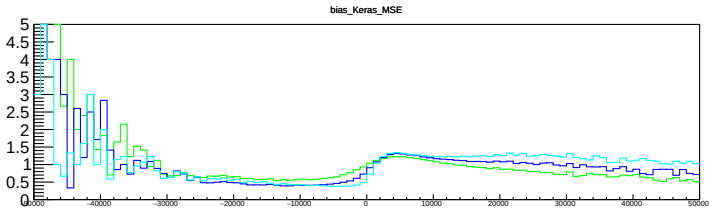

upar_uT
ATLAS Internal
W+ → μ+ ν
uT
Keras_MSE
Keras_MSE+bias
Keras_MSE+5bias


upar_Keras_MSE

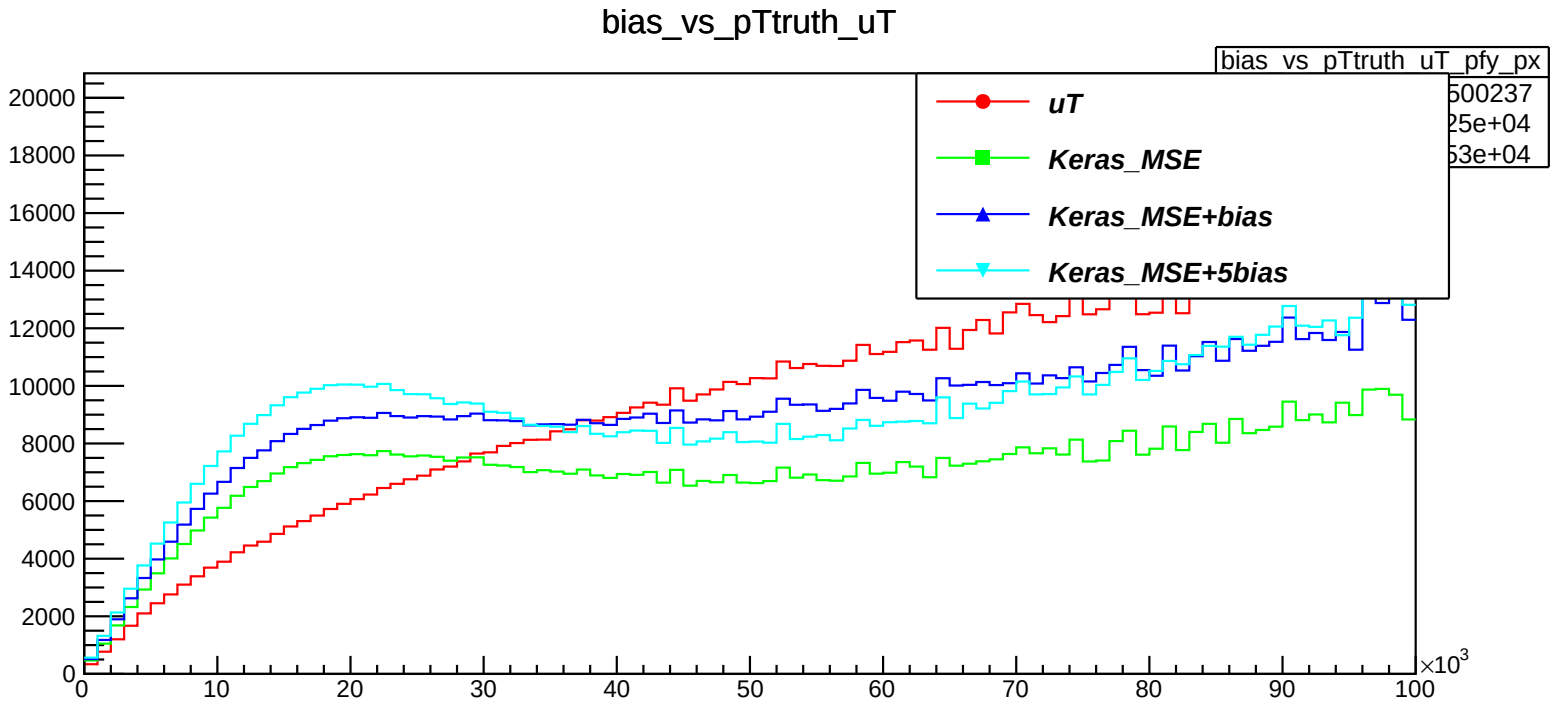

histosDeltaUt_uT
ATLAS Internal
W+ → μ+ ν
uT
Keras_MSE
Keras_MSE+bias
Keras_MSE+5bias

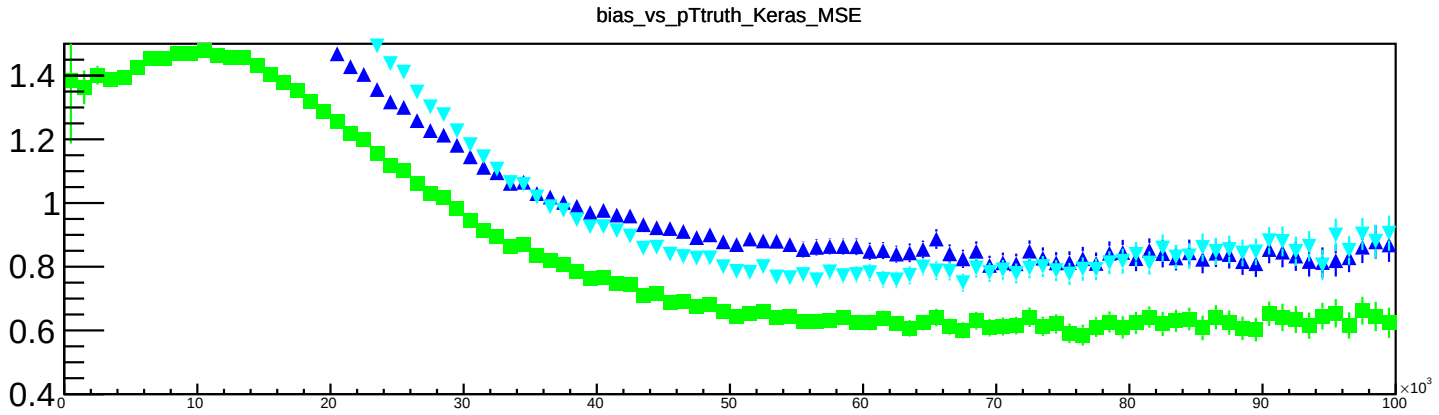

histosDeltaUt_Keras_MSE


bias_uT
Entries 1517613
Mean 4668
Std Dev 8240
ATLAS Internal
W+ → μ+ ν
uT
Keras_MSE
Keras_MSE+bias
Keras_MSE+5bias


bias_Keras_MSE

<bias> vs pTtruth



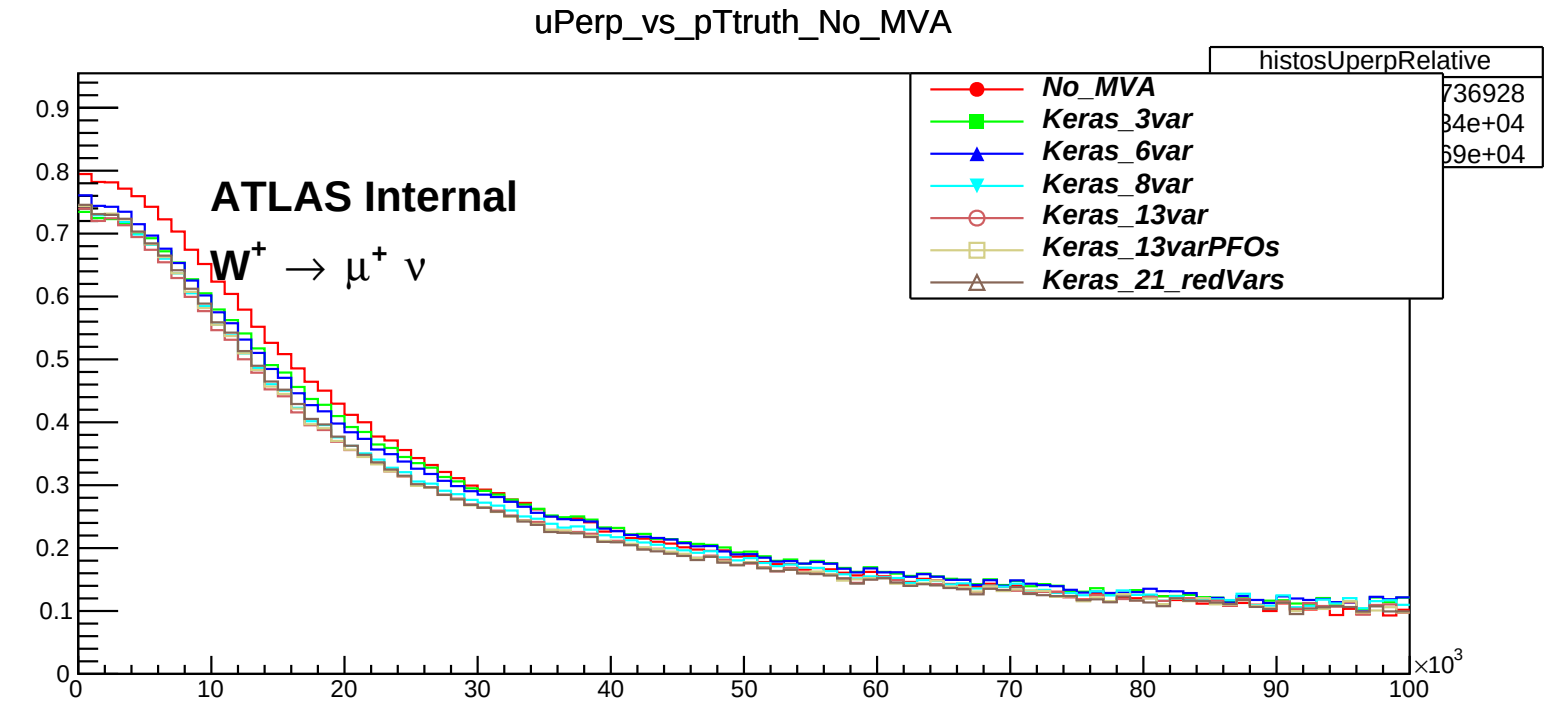


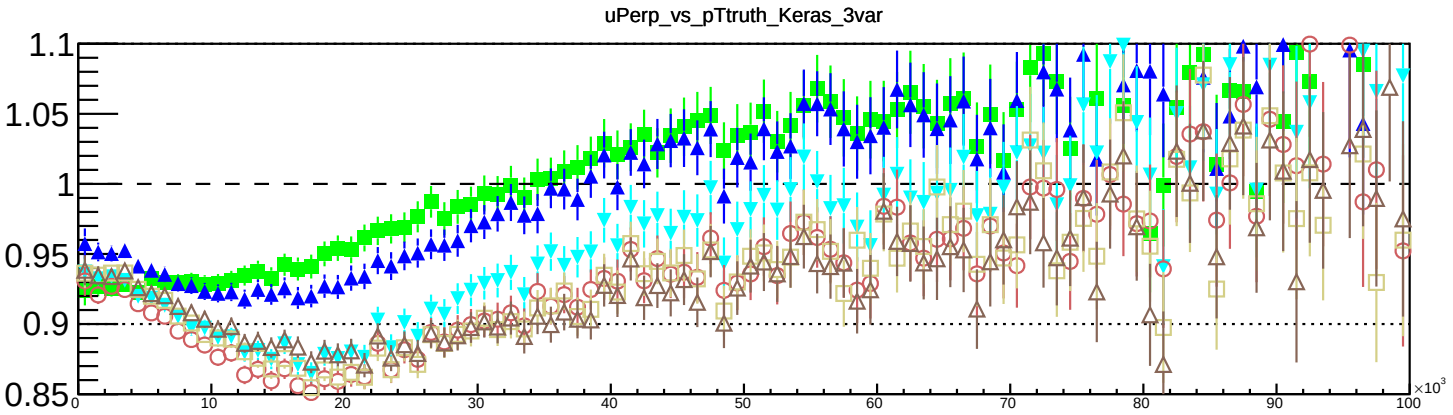
Conclusions

- Even the basic and straightforward approach brings visible results
- There is certainly room for improvement: wiser loss functions, more complicated NN models
- All suggestions for are **very welcome**

Backup

Adding inputs





sigma(uPerp)/<uT> vs pTtruth