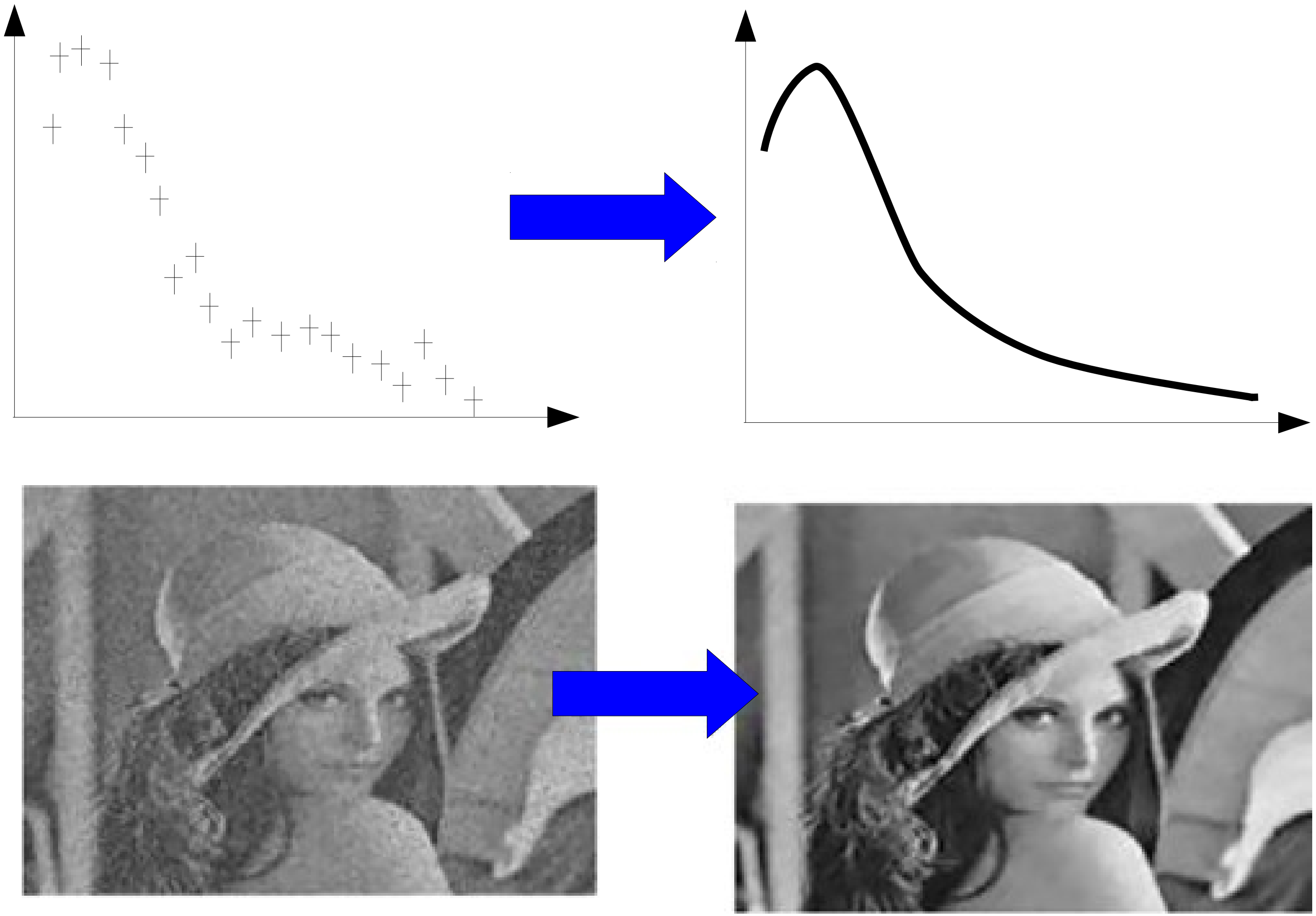
Fitting a spectrum using ML

Samuel Calvet

March 16th 2021

Changing the perspective...

- Fitting a spectrum can also be seen as **removing the poisson noise**



Changing the perspective...

- Wanting also the NN to produce realistic bgd fit
  (ie signal is seen as “noise” and is subtracted)

“signal” not subtracted

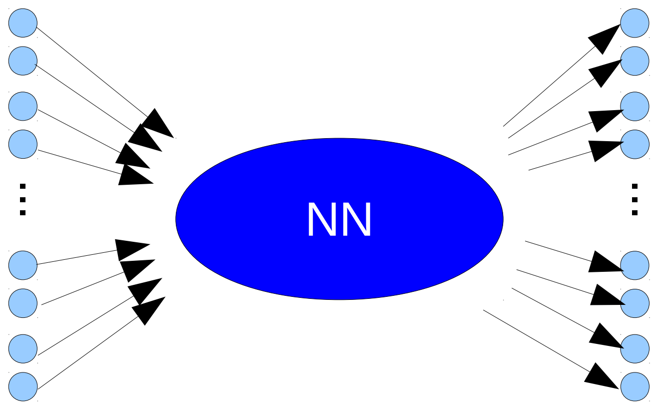
“signal” subtracted producing realistic “bgd” shape

The idea

- The concept would be naively:

N-bins data histogram →

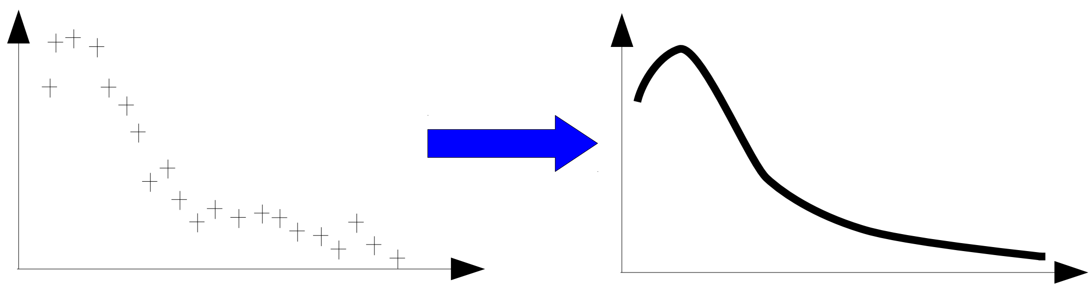
(truth + Poisson noise)



→ N-bins “fitted” histogram
(~ truth)

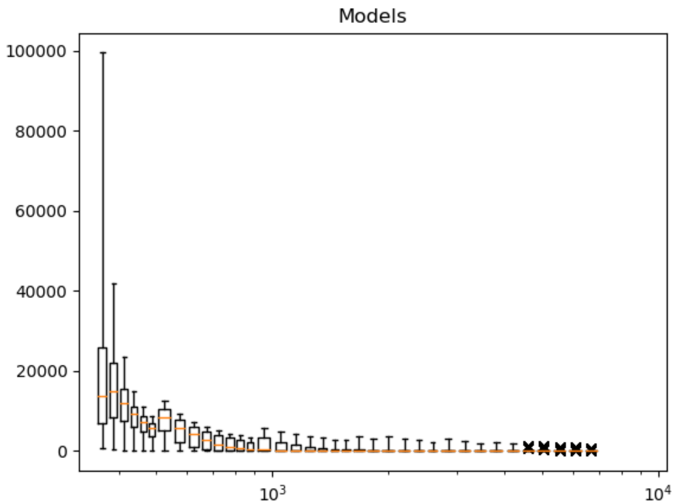
N inputs

N outputs

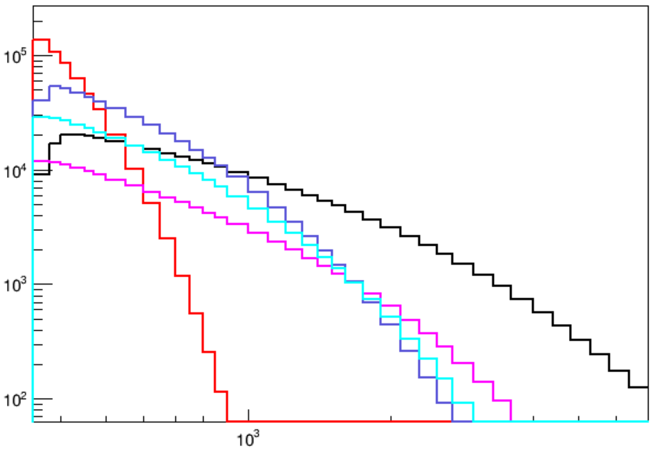


Training/test samples

- Truth PDF:
  - $PDF(x) = \sum f_i . Erf(x, \sigma_i, \mu_i) * exp(- x^{\rho_i} / \alpha_i)$
  - Number of backgrounds can vary, aim to have N_{bgd}=5
  - 1e5 ±80% events per histogram
  - O(1e6) shapes



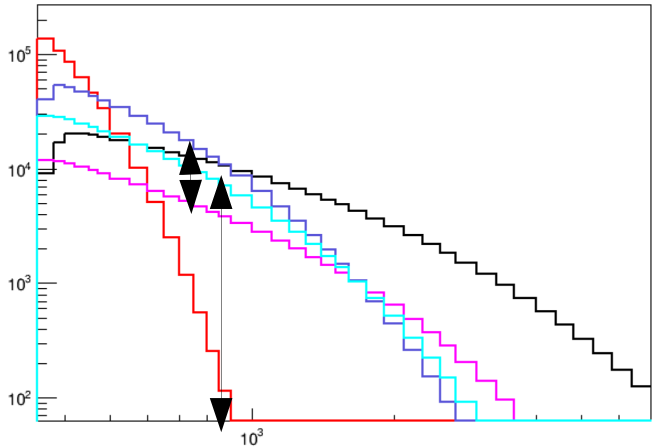
Distribution of bin contents for N=5



Large variety of shapes

Issue

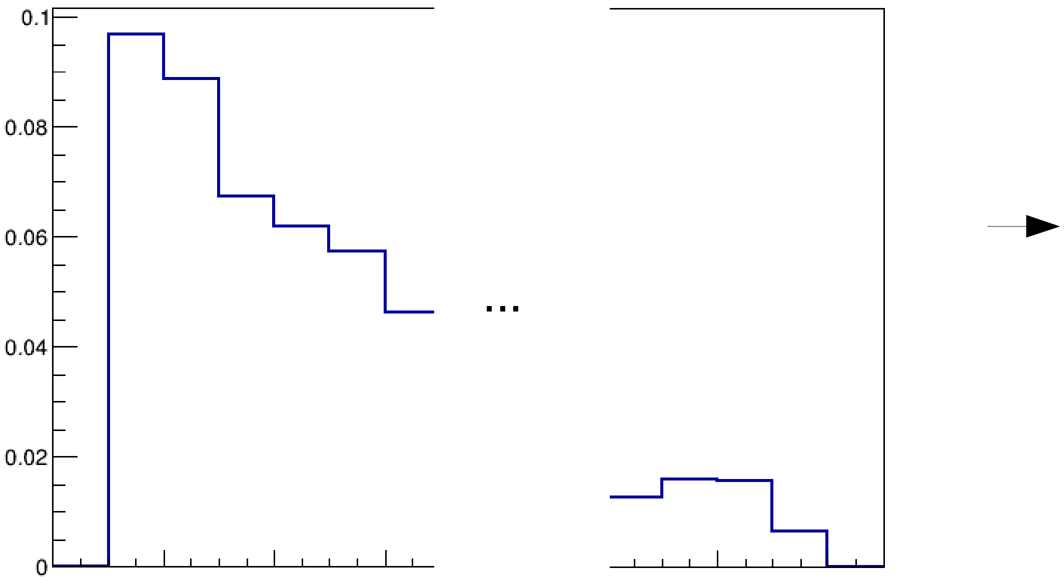
- Large variety of shape → large variation in order of magnitude
  - Activation functions can not work properly if all the order of magnitude of the input are significant
    → Very difficult to train

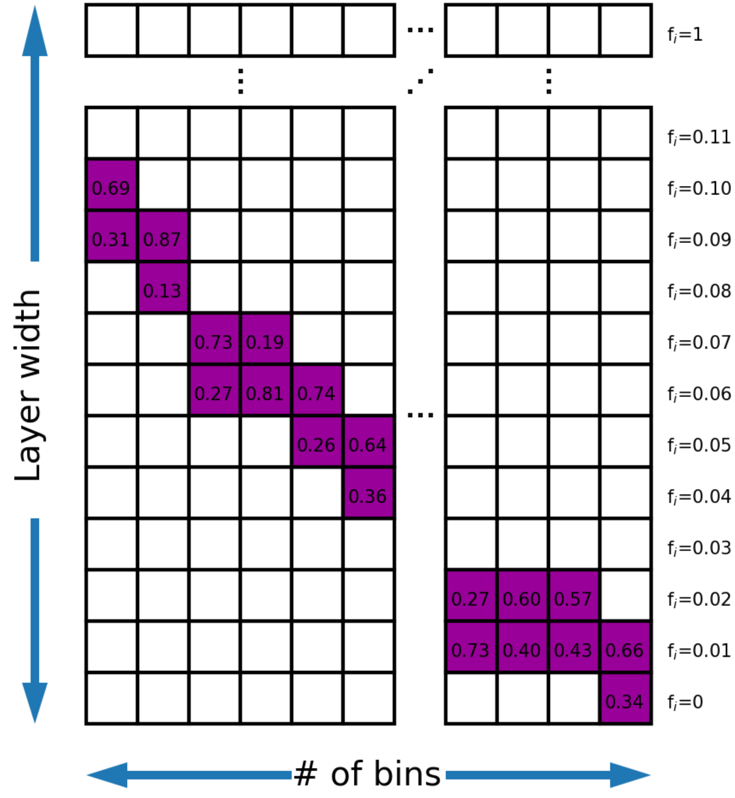


- Is there a way to get rid of this ?
  - Yes ! ‘Pixelize’ the histogram!

FlattenHistogramAndNormalize layer

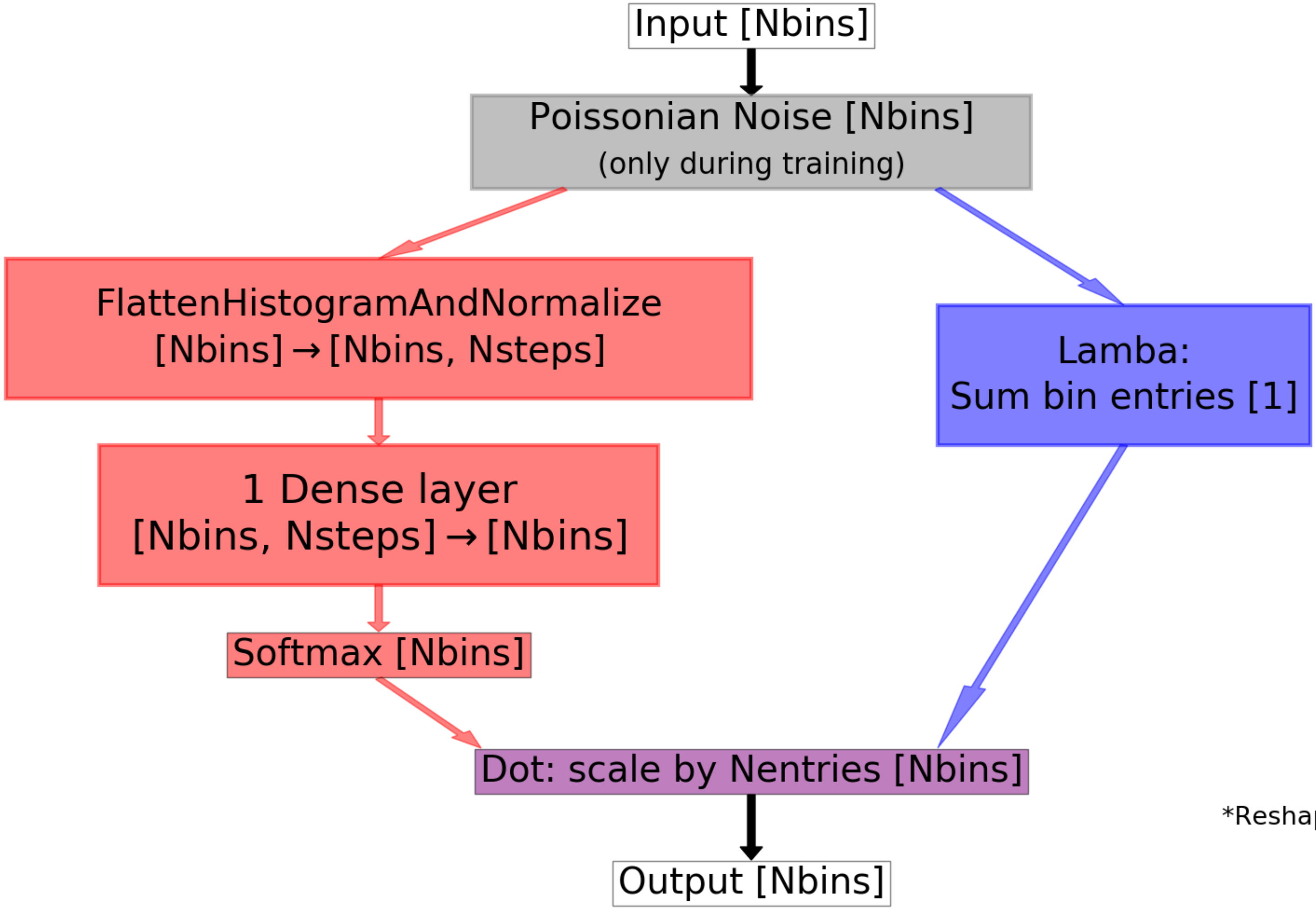
- Convert 1D histo into 2D tensor





- Introduce **structure and invariance**
  - Only an handful of elements activated per histogram
    → simplify the convergence of the following layers

A simple NN



*Reshaping steps not shown

Training of NN

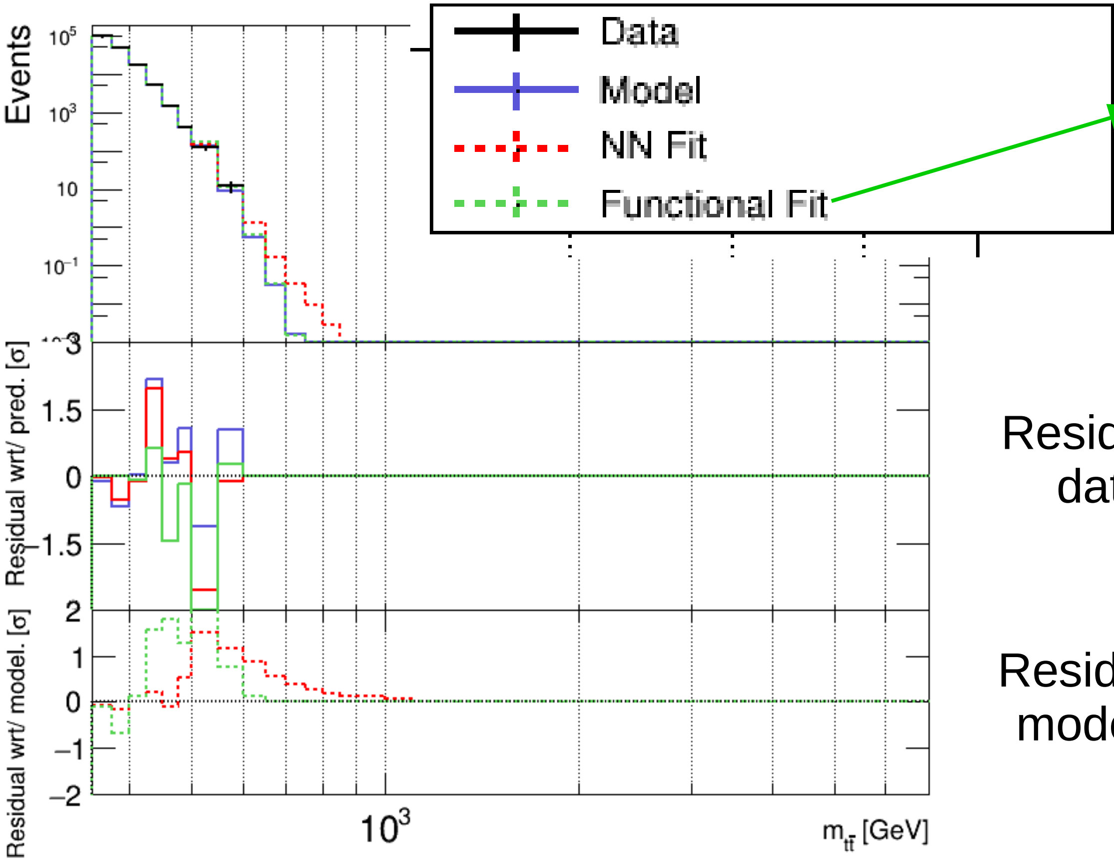
- 1.024e6 models, N_{bgd}=5
- Batch size: 128
- FlattenHistogramAndNormalize width: 101
- Loss: AtlasSignificanceNormalized (ASN)
  - Normalize Y^{true} to Y^{pred}
    then compute: mean(Y_i^{true} * log(Y_i^{true} / Y_i^{pred}) - (Y_i^{true} - Y_i^{pred}))
  - From Atlas recommendation https://cds.cern.ch/record/2643488/?

In the case of $n \geqslant b$, this formula corresponds to equation 25 in [1]. In the case of vanishing uncertainty ($\sigma = 0$) this formula reduces to:

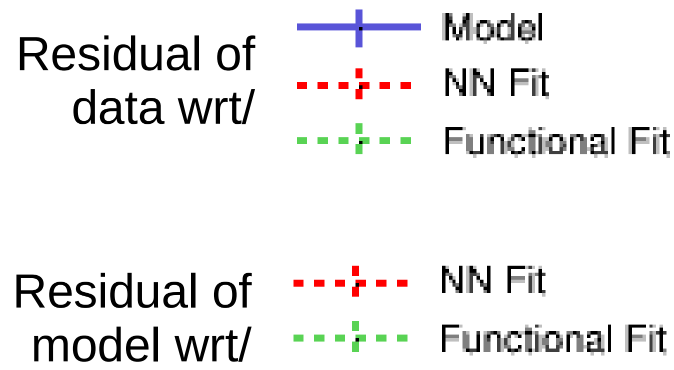
$$Z = \begin{cases} +\sqrt{2\left(n \ln\left[\frac{n}{b}\right] - (n-b)\right)} & \text{if } n \geqslant b \\ -\sqrt{2\left(n \ln\left[\frac{n}{b}\right] - (n-b)\right)} & \text{if } n < b. \end{cases} \quad (2)$$

- Early stopping on val_loss, min_delta=0, patience=3
- Optimizer: RMSprop (learning rate: 1e-6)
- Initialization of weights: random uniform, bias set to 0

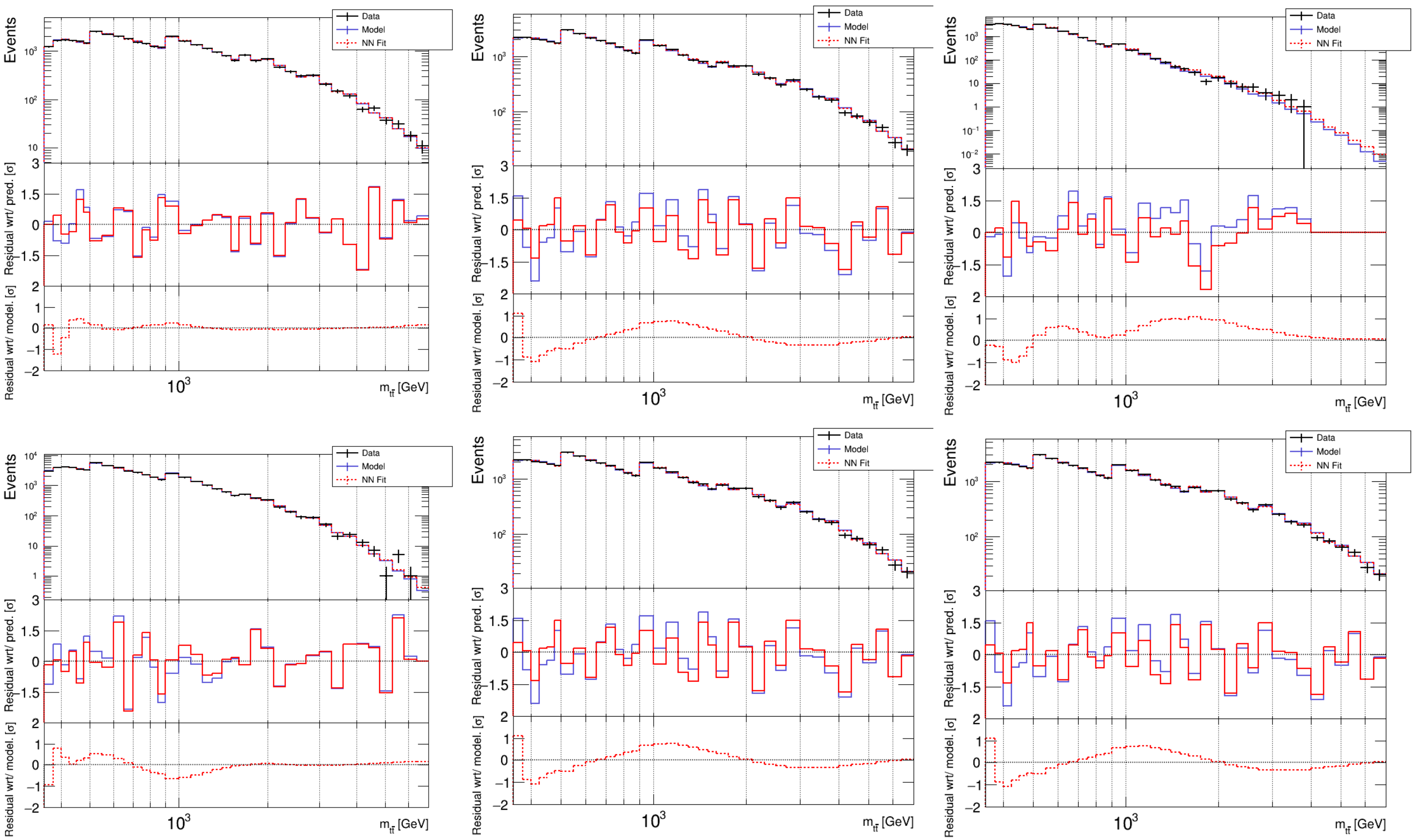
Results



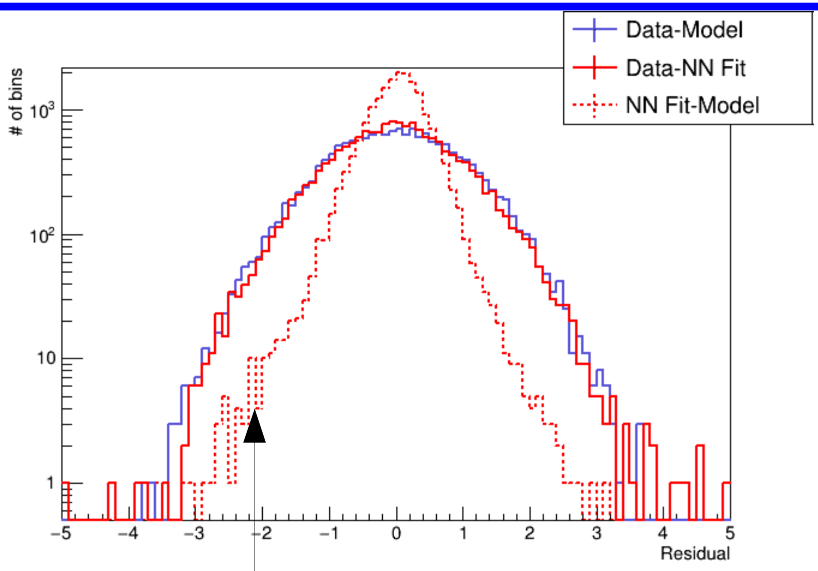
- Fitted parameters **constrained to be within the range** used during generation
- **iterative fit**: the "center" of the bins is recomputed, for each iteration based on the result of the previous iteration

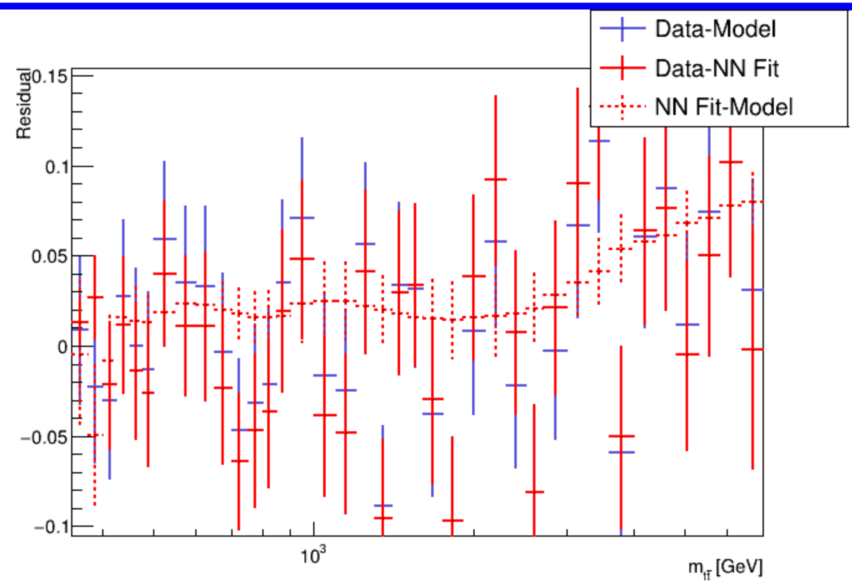


Result: test sample (6 first models as an illustration)



Result: test sample

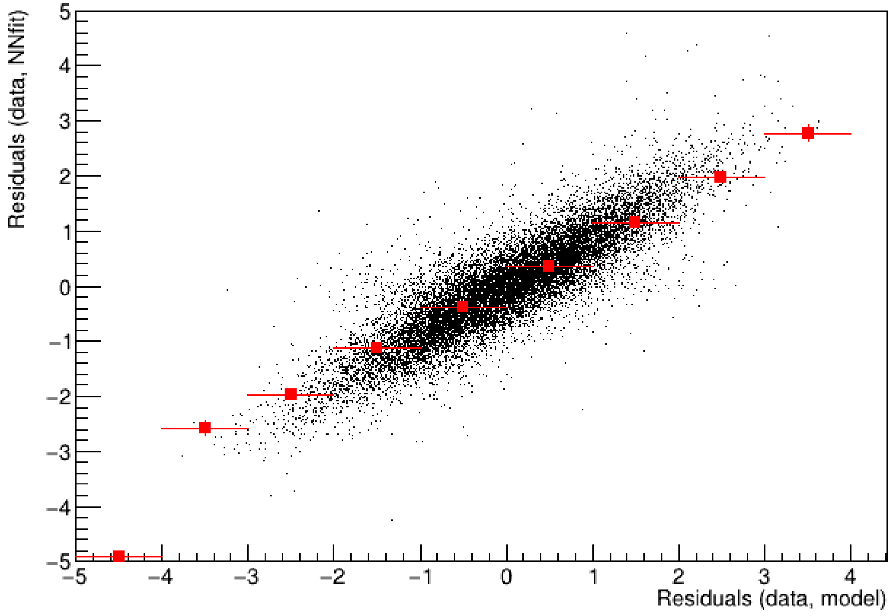


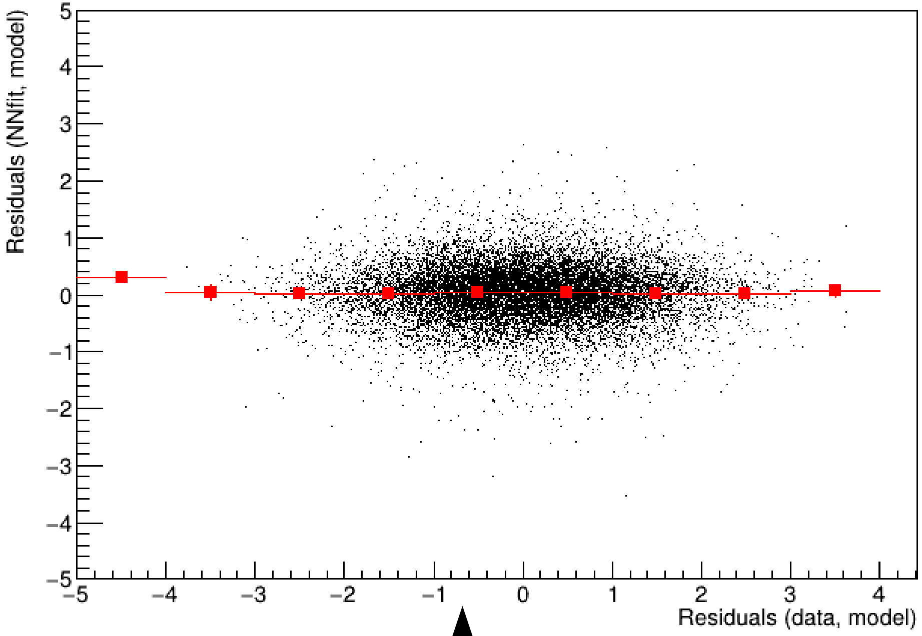


RMS=0.47

Very small bias, in average

Result: test sample





Bin-by-bin residual correlations:

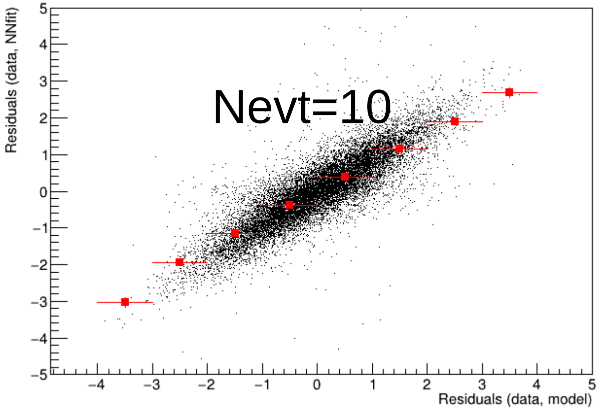
- residual(data, NNfit) ~ residual(data, model)
- residual(NNfit, model) does not depend on residual(data, model) ie data fluctuation

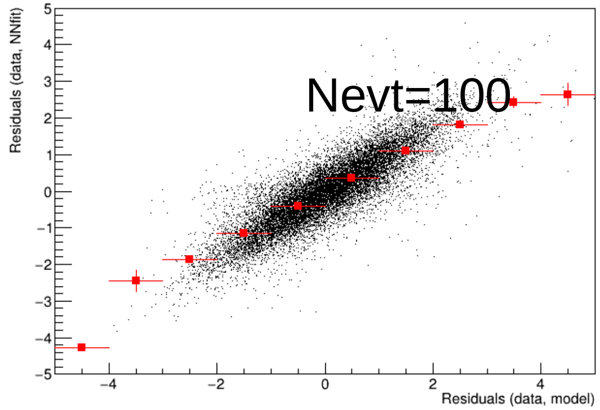
Let's now inject signal in the pseudo-data

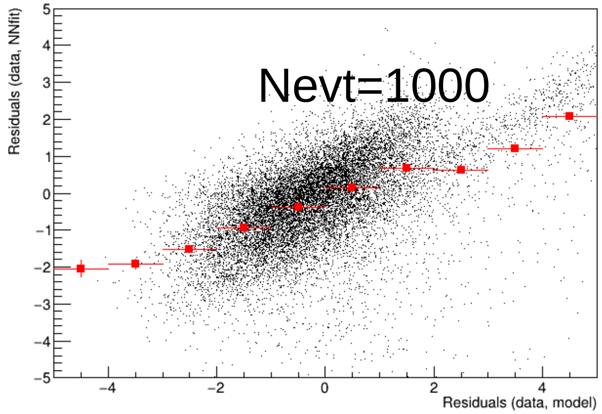
Signal=Nevt*Gaus(1TeV, 10%)

Nevt=10, 1e2, 1e3, ...

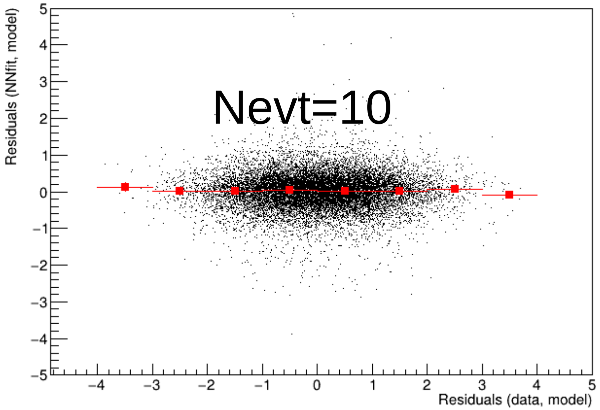
Mass=1TeV, width=10%, Nevt=...

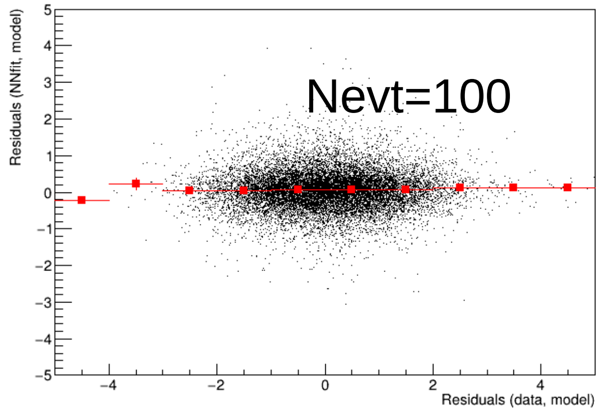


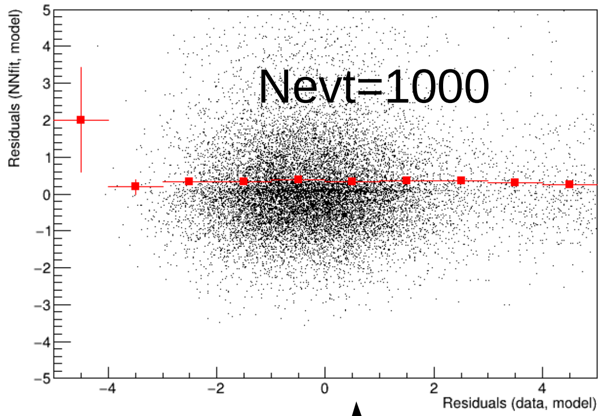




residual(data, NNfit) vs residual(data, model)





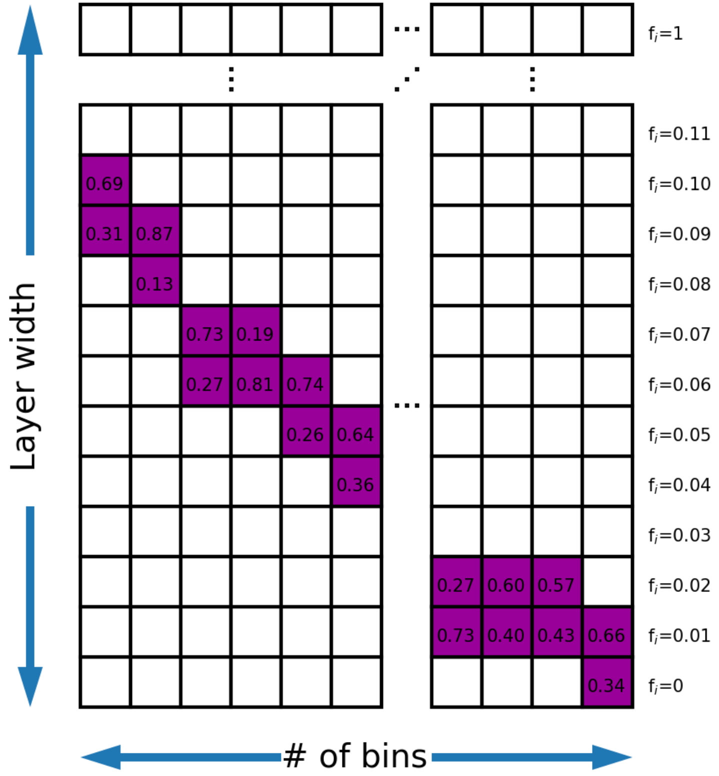


residual(NNfit, model) vs residual(data, model)

Bias appear when strong signal

Conclusion

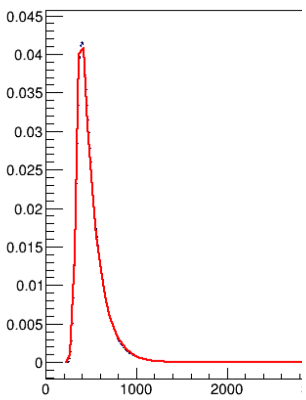
- Defined a new activation layer
  - Convert 1-D histogram into 2D ‘image’
    - Can also produce several ‘images’ to reflect the stat uncertainty
  - Simplify a lot the convergence of the NN
    - Only an handful of nodes are activated for a given histogram
- Promising performances on tested spectra
  - Just adding a simple and small dense layer !
    (much more tests in backup and on my disks...)

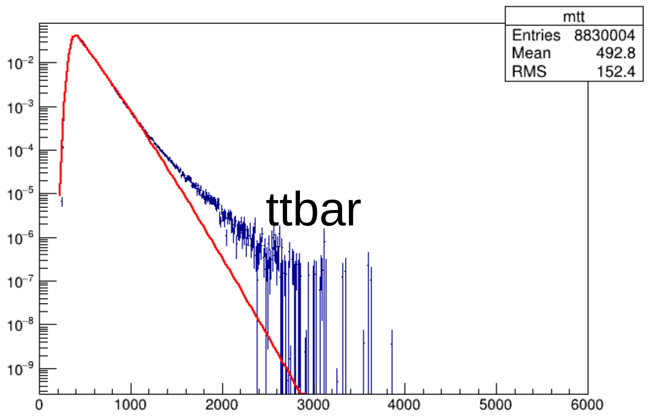


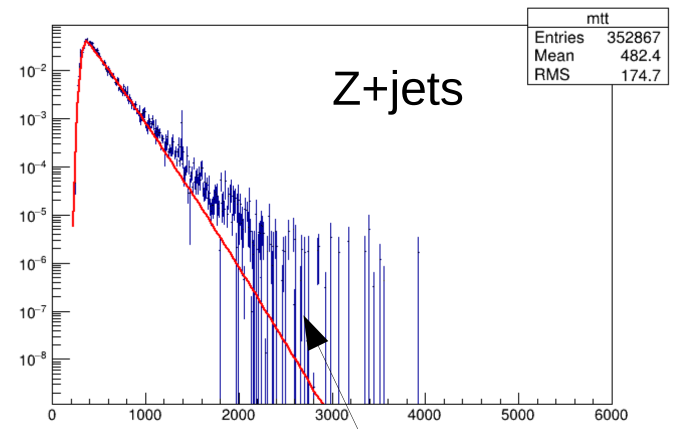
Backup

Training/test samples

- Use MC samples [ttbar, W, Z, VV, single-t, dijet] (thanks to Souad!) to know typical values of functional shape:

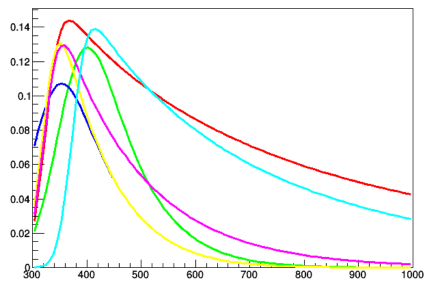






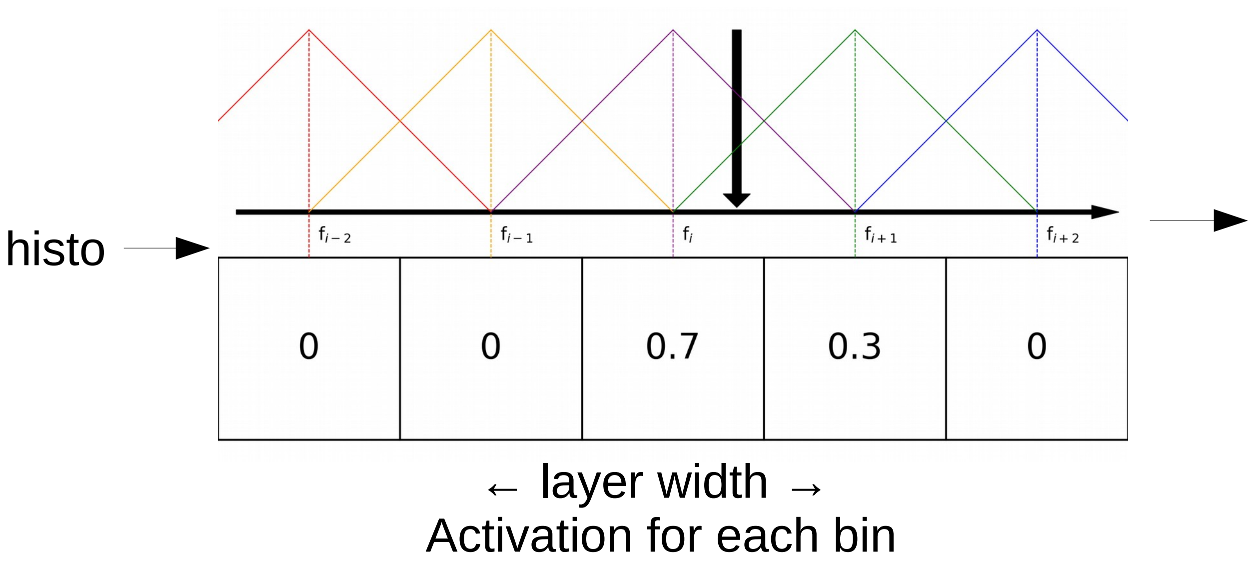
Not perfect in the tail…
but hopefully it will be fine
(see next slides)

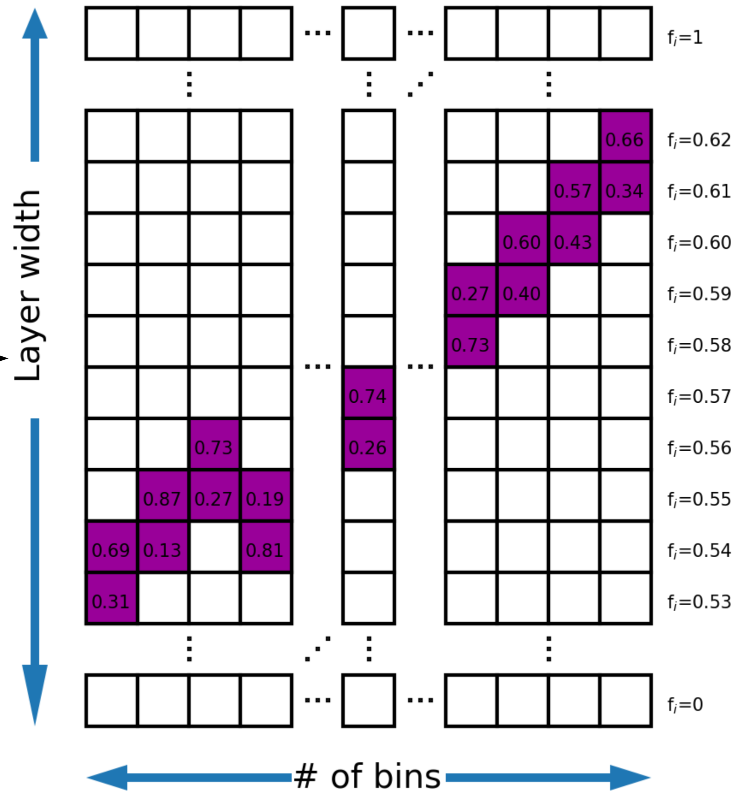
- PDF(x) = Erf(σ, μ) * exp(- x^{ρ}/ α)
- "Measured" intervals → Used intervals
  - $\rho \in$ [0.68, 1.43] → [0.5, 1.5]
  - $\alpha \in$ [36, 275] → [25, 200]
  - $\sigma \in$ [16.7, 23.3] → [10, 30]
  - $\mu \in$ [325, 349] → [325, 375]



FlattenHistogramAndNormalize layer

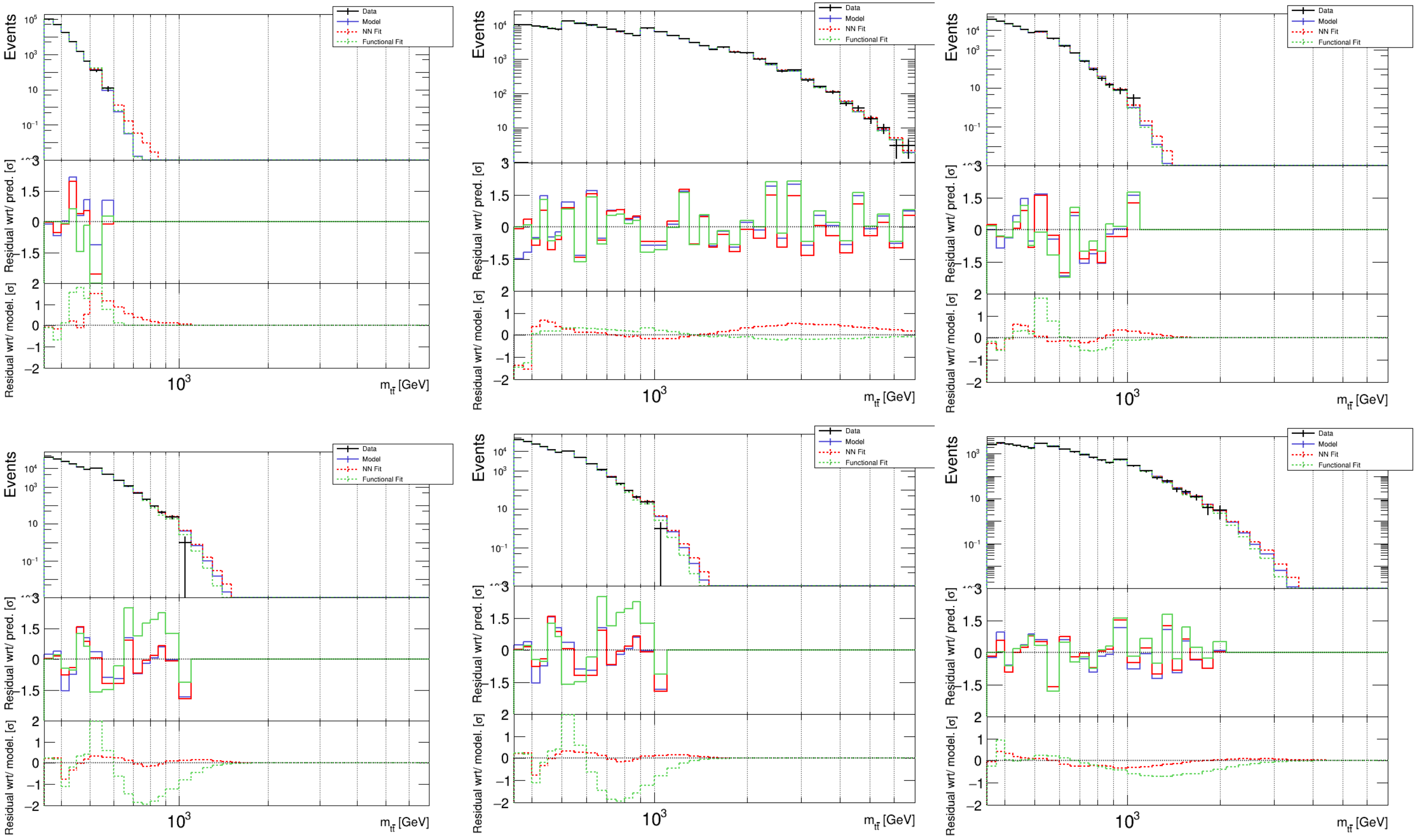
- Convert 1D histo into 2D tensor



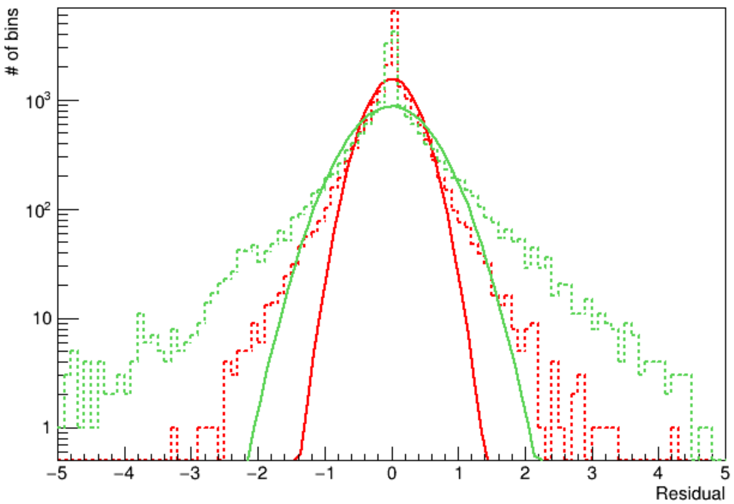


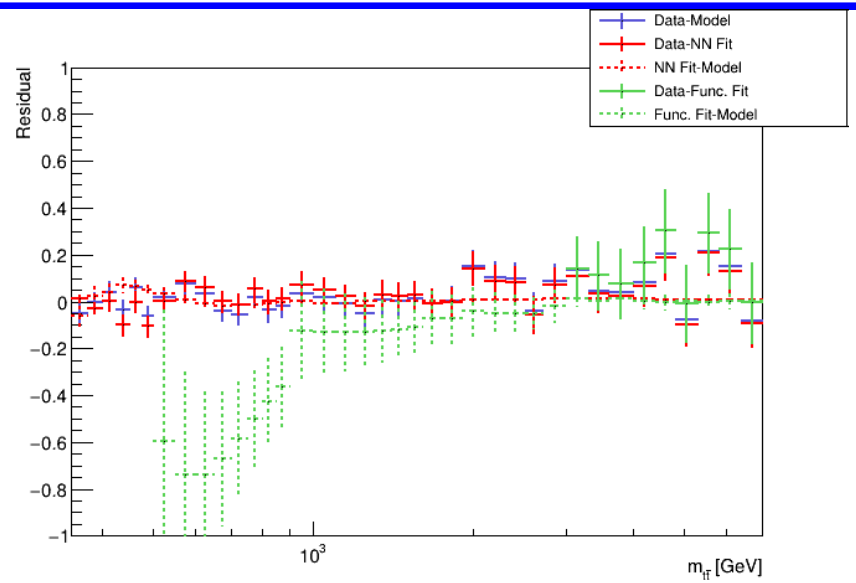
- Introduce **structure and invariance**
  - Works properly even if its input vary by several order of magnitude
  - Only an handful of elements activated → simplify the convergence of the following layers

Result: test sample, N_{bgd}=1 (6 first models as an illustration)



Result: test sample, N_{bgd}=1

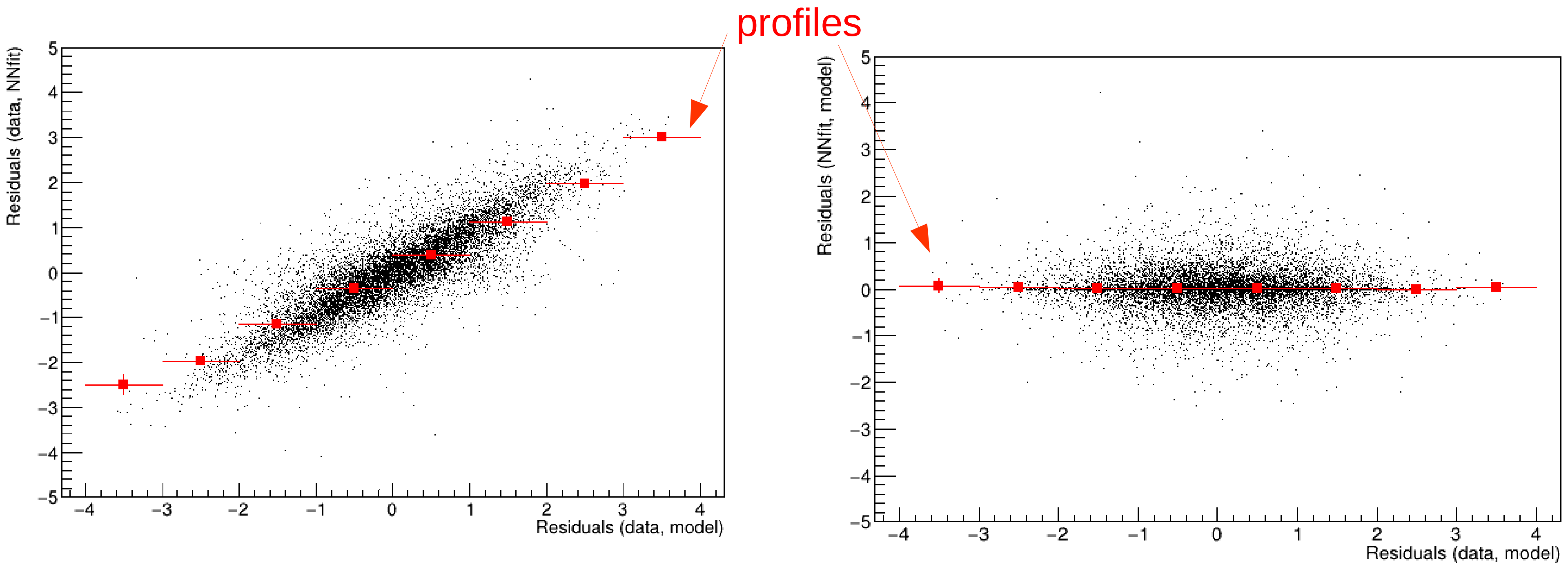




NNfit performs better than a functional fit

Few remarks:
- func. fit would do better if it was allowed to use a wider range of parameters
- func. fit performance is very sensitive to binning (variable size bins!).

Result: test sample, N_{bgd}=1



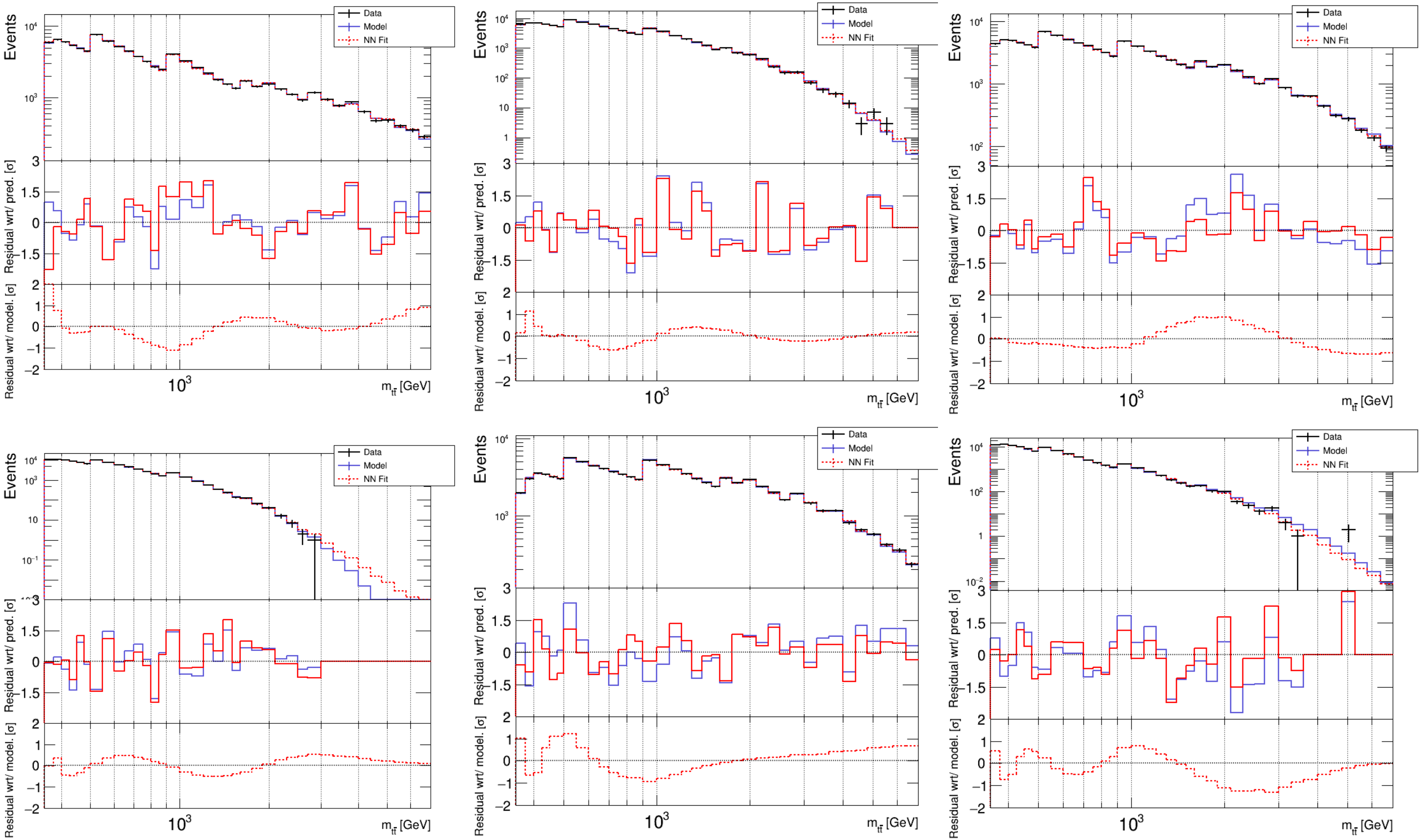
Bin-by-bin residual correlations:

- residual(data, Nnfit) ~ residual(data, model)
- residual(NNfit, model) does not depend on residual(data, model) ie real data fluctuation

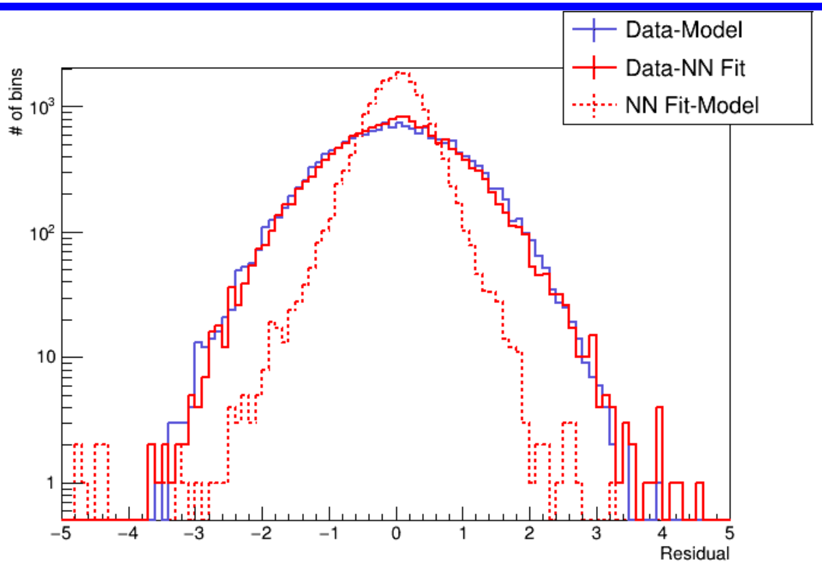
What if the test sample has N_{bgd}=7 ?

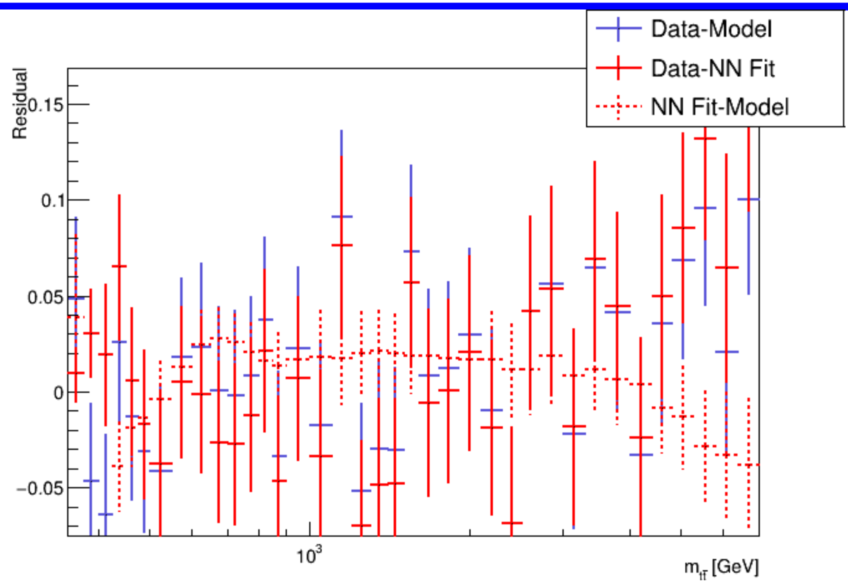
Is the NN able to generalize ?

Result: test sample, N=7 (6 first models as an illustration)



Result: test sample, N_{bgd}=7

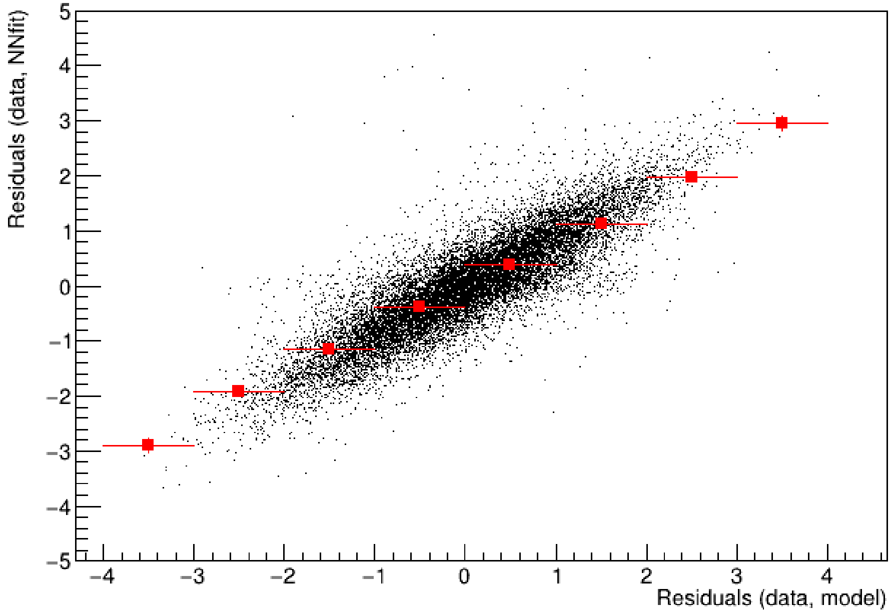


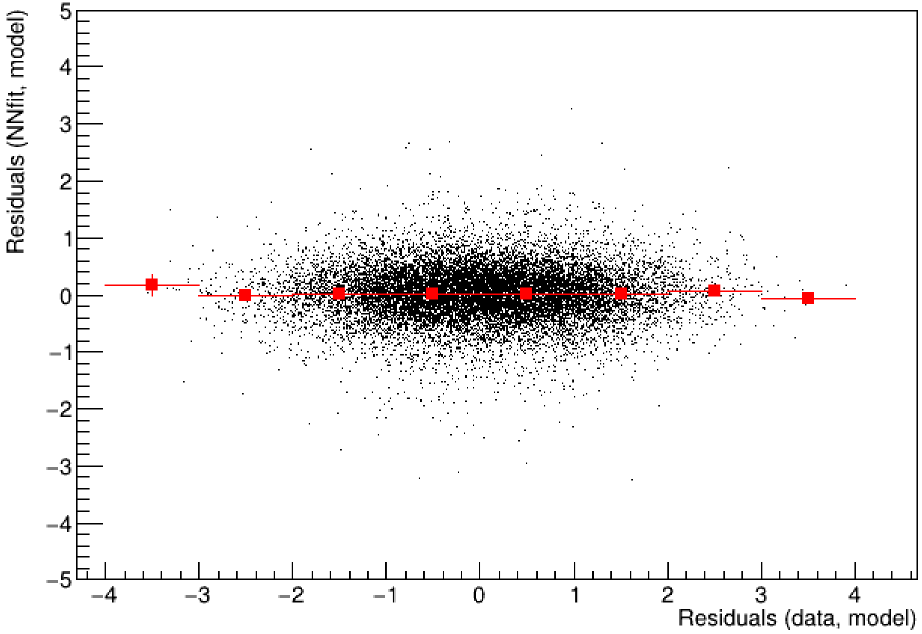


RMS=0.49 (was 0.47/0.42 for N_{bgd}=5/1)

Very small bias

Result: test sample, N_{bgd}=7





Bin-by-bin residual correlations:

- residual(data, NNfit) ~ residual(data, model)
- residual(NNfit, model) does not depend on residual(data, model) ie data fluctuation

Same results as before!

Re-fit the data

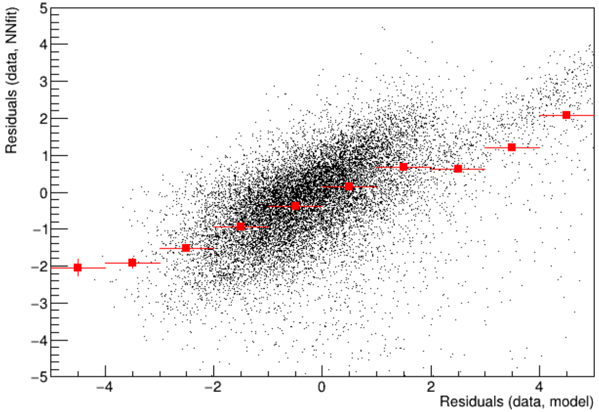
- Results are not bad for weak signal

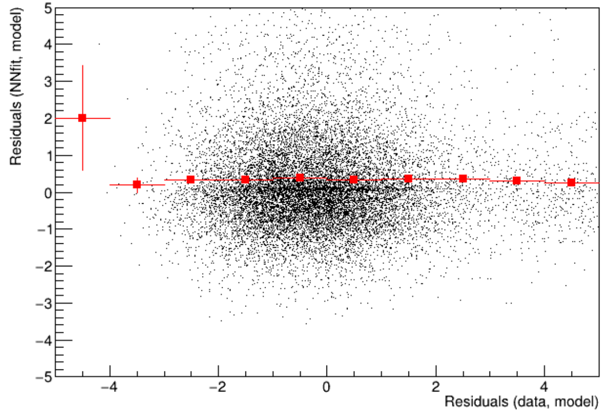
  but far to be perfect when there the signal is strong

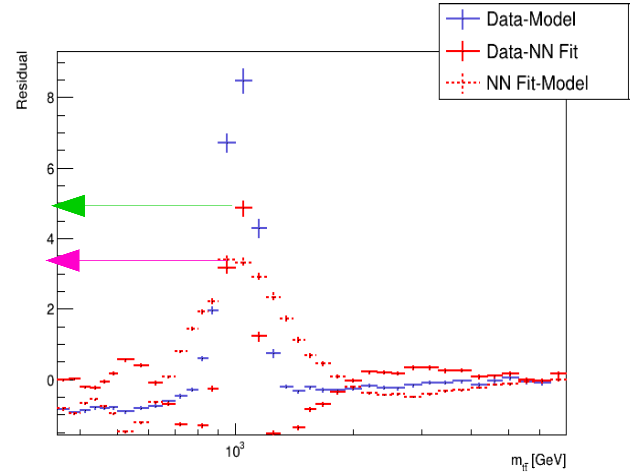
- What if we refit the data, but correcting the most discrepant bins ?
  - Apply NN on data once → bgd histo
  - MaxSignif = maximum bin significance(data, bgd)

    this bin is added to BinToBeCorrected list
  - While MaxSignif>1:
    - data' = data
    - for i in BinToBeCorrected: data'[i]= bgd[i]
    - Apply NN on data' → bgd histo

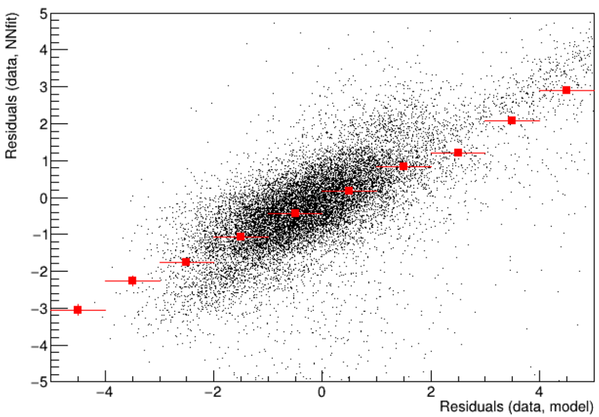
Mass=1TeV, width=10%, Nevt=1000

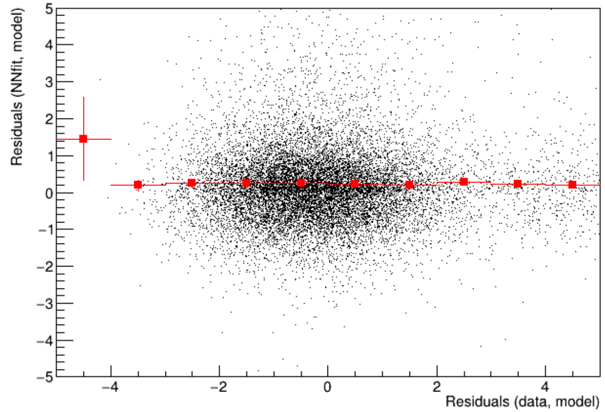


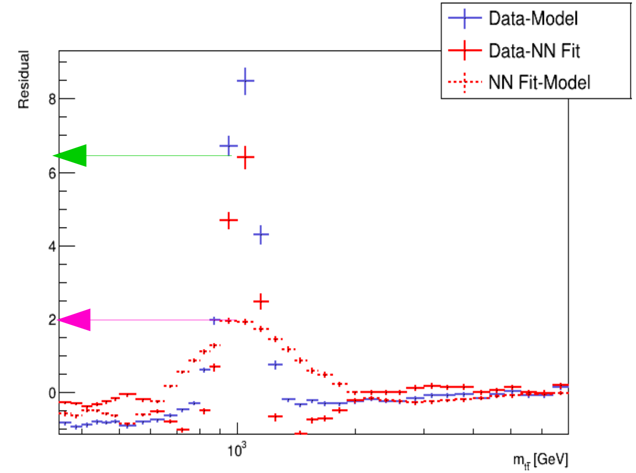




No refit



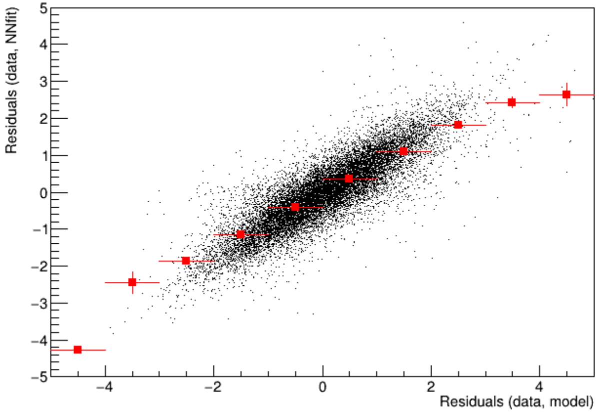


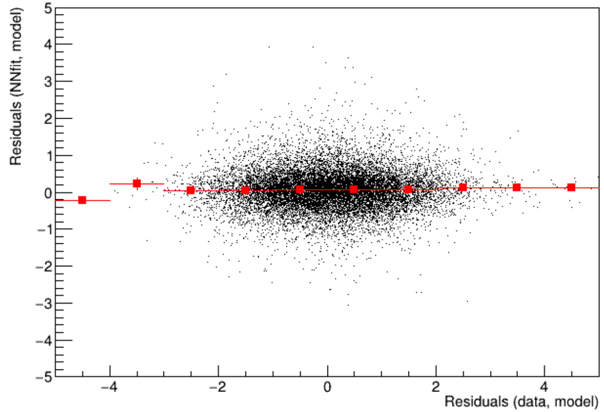


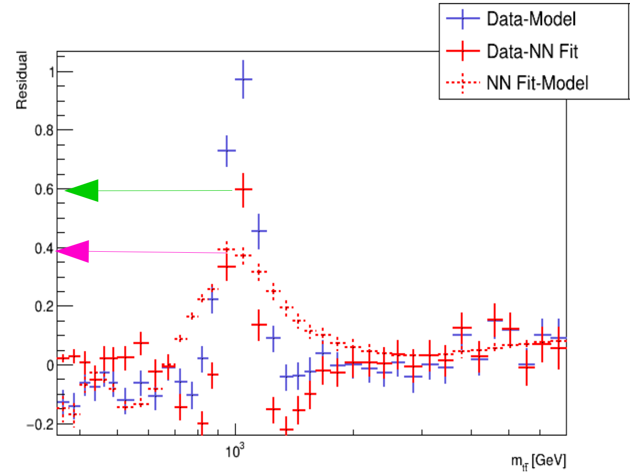
With refit

Reduce a lot the bias !

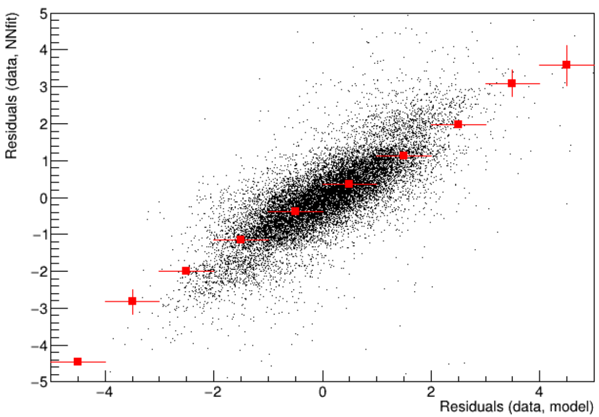
Mass=1TeV, width=10%, Nevt=**100**

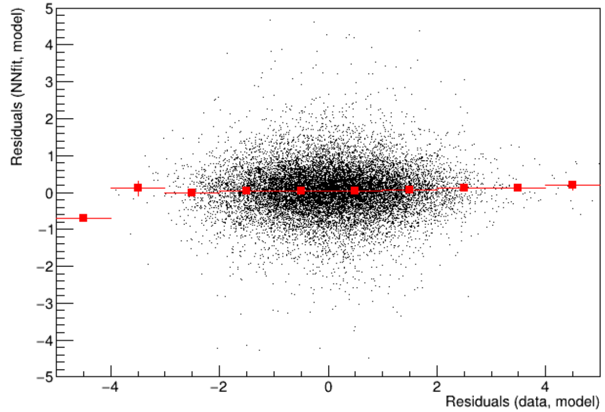


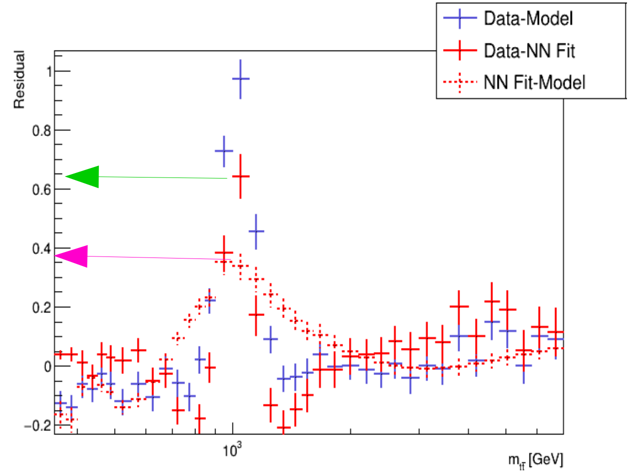




No refit







With refit

Smaller excess, refit less efficient