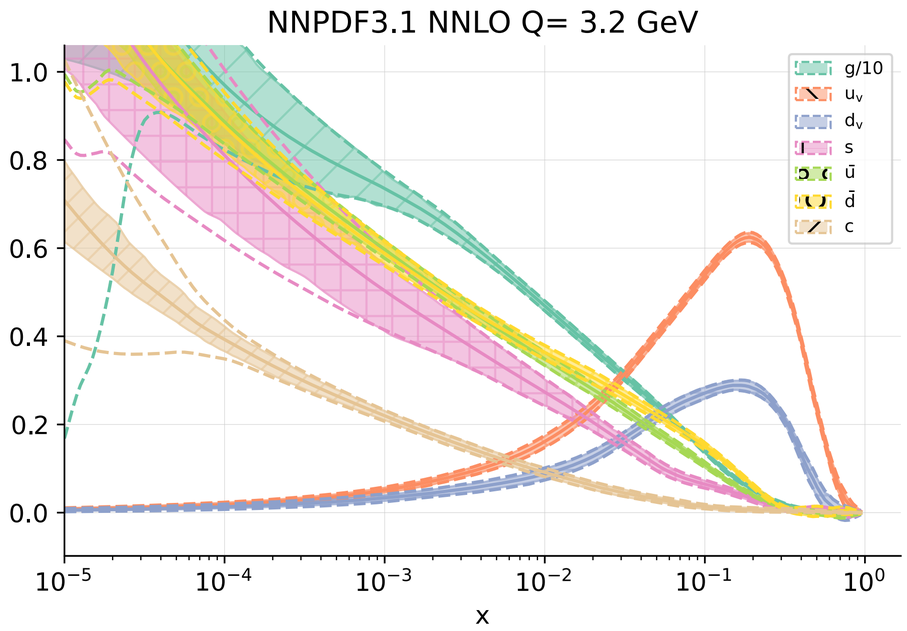
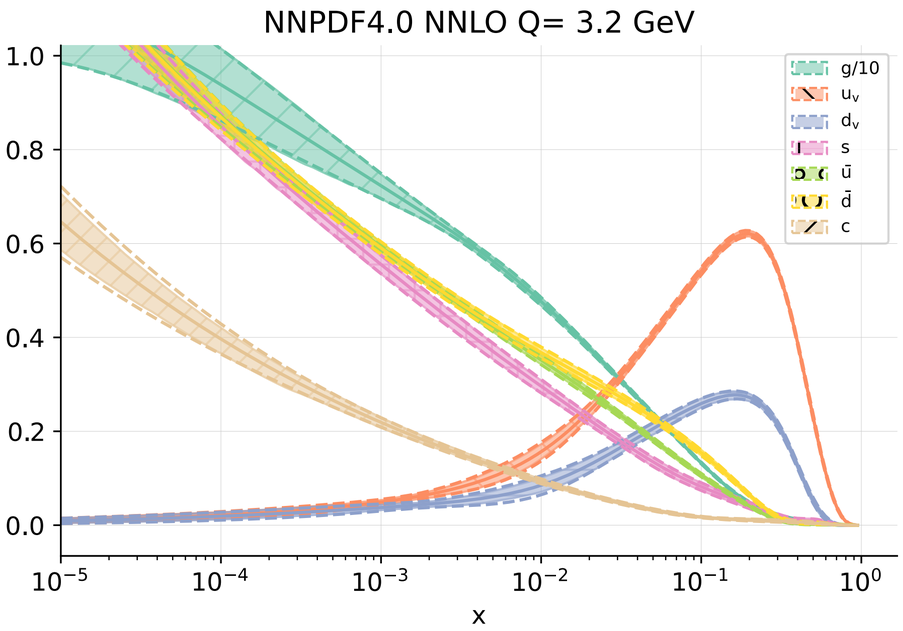
NNPDF4.0

The path to reliable % uncertainties in PDF determination

Juan M. Cruz Martinez - CERN QCD Seminar, November 2022

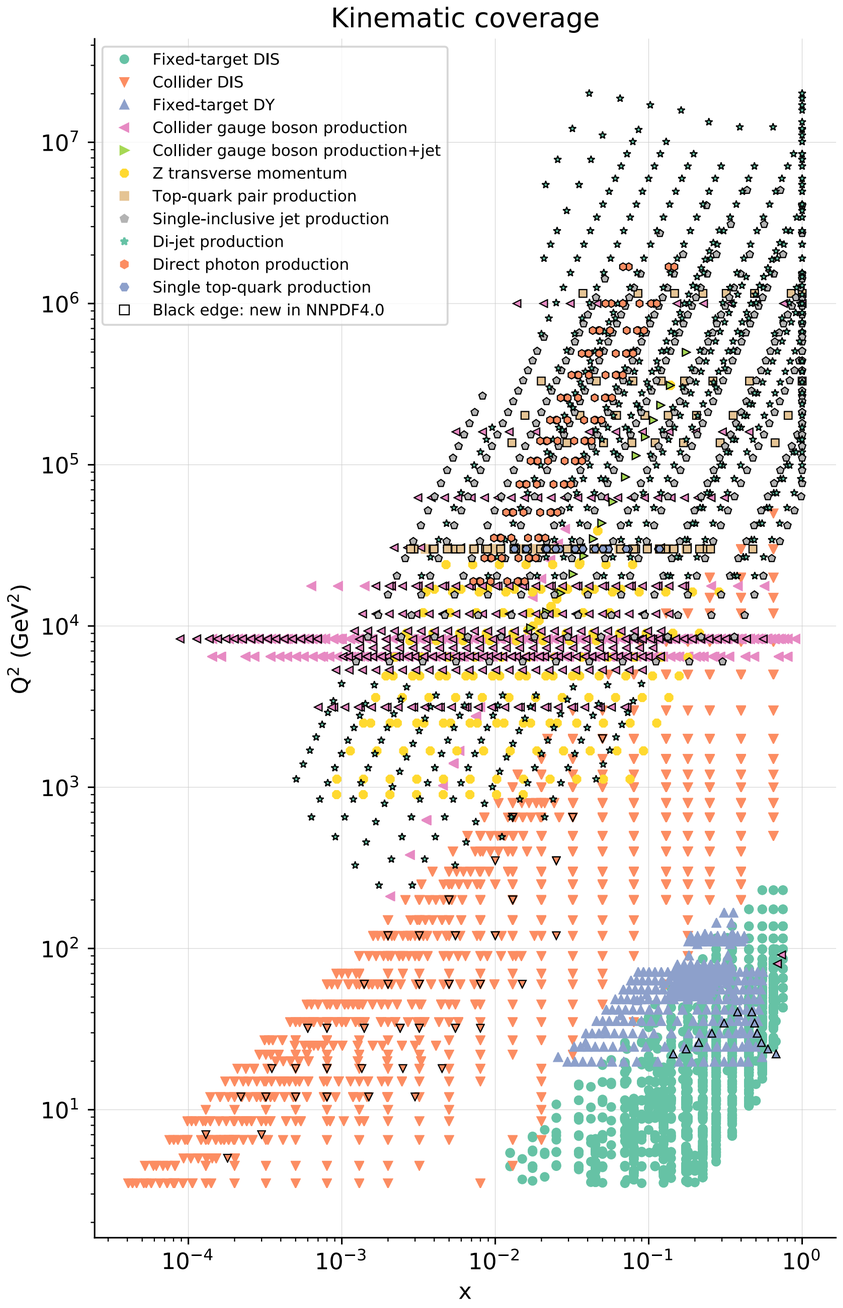
The latest NNPDF: NNPDF4.0



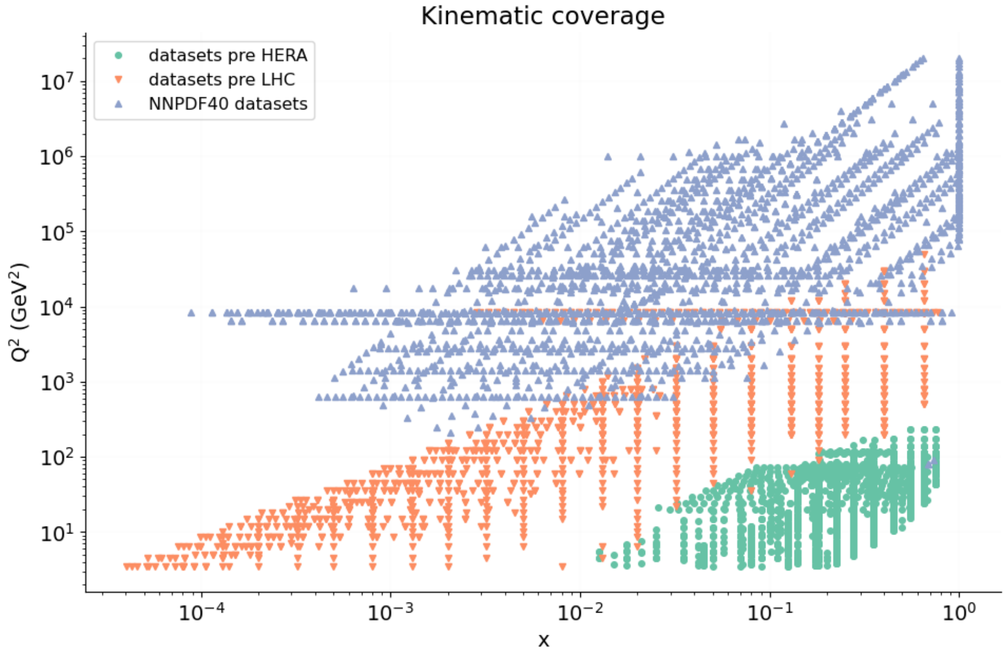


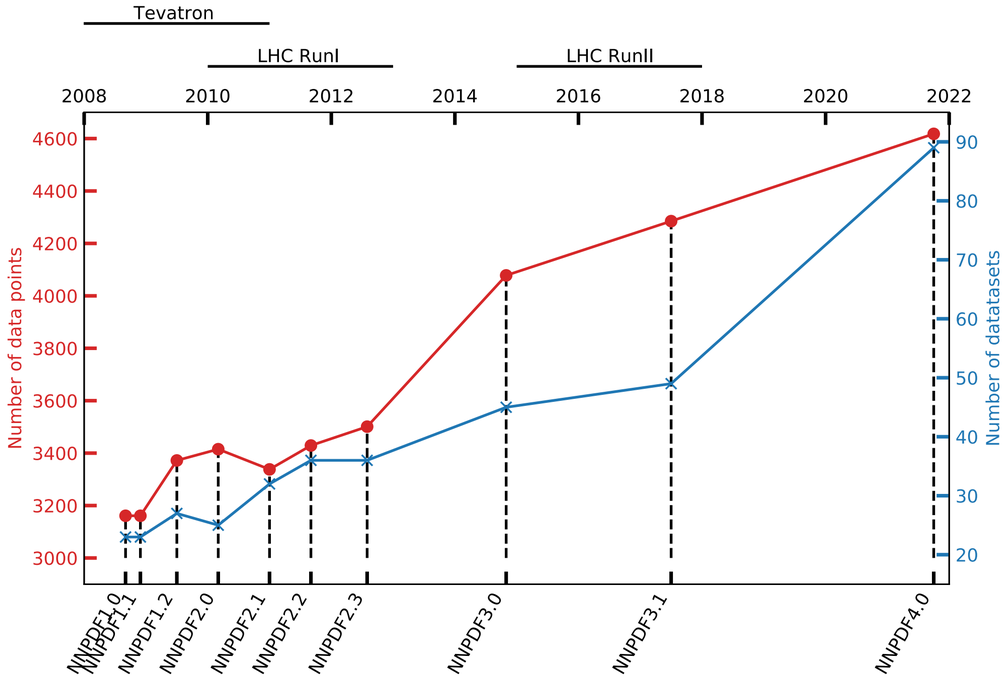
- 4285 datapoints (~50 datasets)
- 15 h per replica
- Private code
- Constraints: positivity, sum rules
- Closure tests

- 4618 datapoints (~90 datasets, many from LHC)
- Under 1 h per replica
- Open source
- Constraints: (+) positivity, sum rules, integrability
- Closure and future tests



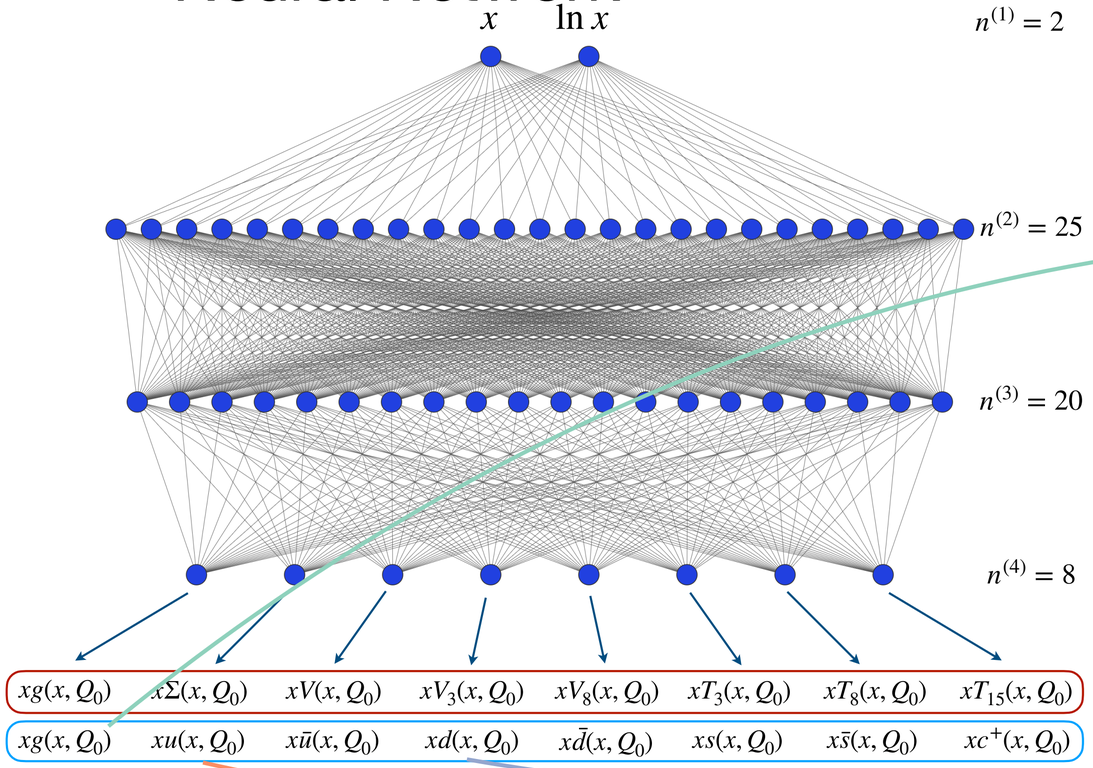
The full list of datasets used can be checked in Appendix B of the NNPDF4.0 paper: link

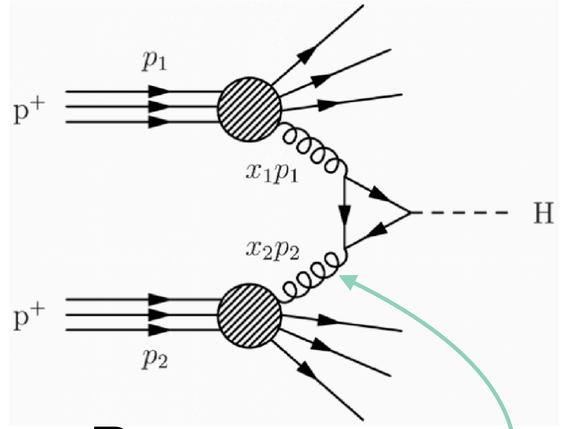




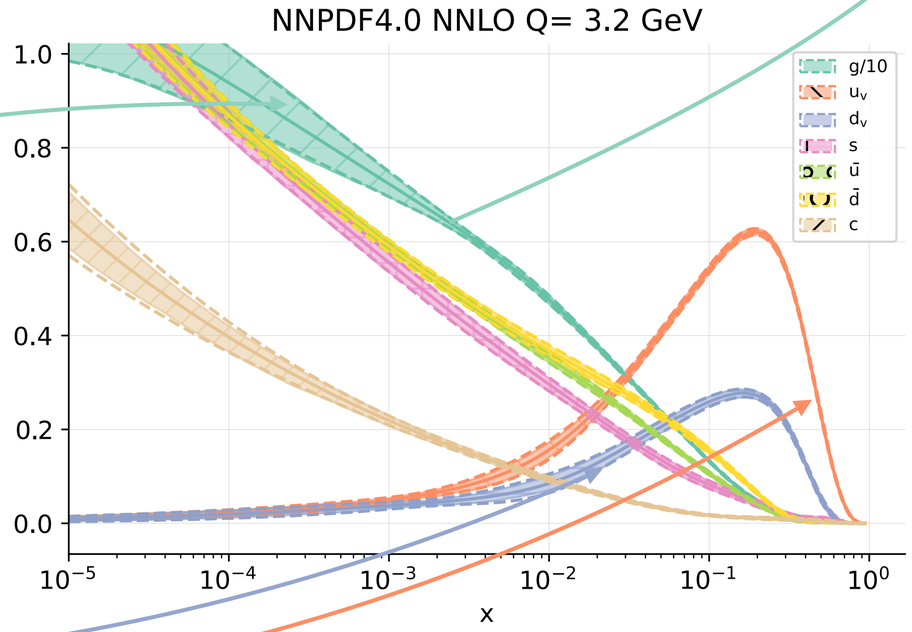
The NNPDF methodology

Neural Network





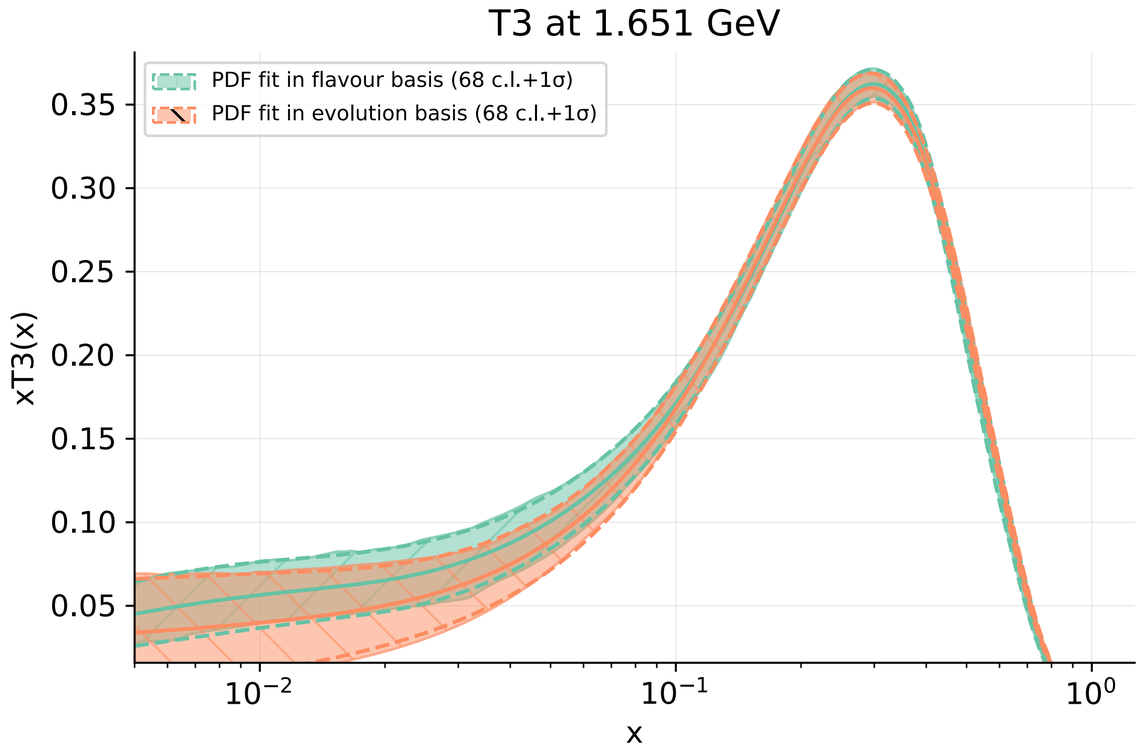
Parton Distribution Function



PDF Parametrization

To ease the fit, the output is parametrized by default in the “evolution” basis at the input scale Q_0

$$
\begin{aligned}
g &= g, \\
\Sigma &= u + \bar{u} + d + \bar{d} + s + \bar{s} + 2c\,, \\
T_3 &= (u + \bar{u}) - \left(d + \bar{d}\right)\,, \\
T_8 &= \left(u + \bar{u} + d + \bar{d}\right) - 2\left(s + \bar{s}\right)\,, \\
T_{15} &= \left(u + \bar{u} + d + \bar{d} + s + \bar{s}\right) - 3\left(c + \bar{c}\right)\,, \\
V &= (u - \bar{u}) + \left(d - \bar{d}\right) + \left(s - \bar{s}\right)\,, \\
V_3 &= (u - \bar{u}) - \left(d - \bar{d}\right)\,, \\
V_8 &= \left(u - \bar{u} + d - \bar{d}\right) - 2\left(s - \bar{s}\right)\,.
\end{aligned}
$$



Different parametrization achieve similar results: basis independence

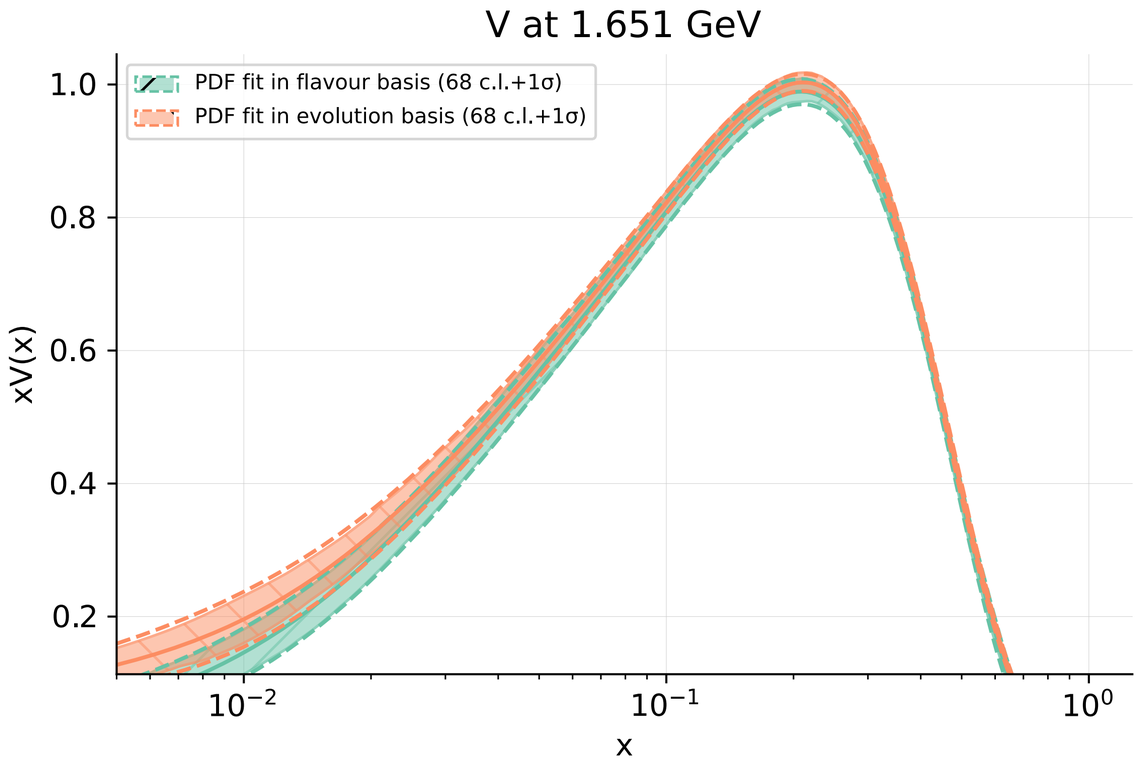
$$xV\left(x, Q_0\right) \propto \mathrm{NN}_V(x)$$

$$xV\left(x, Q_0\right) \propto \left(\mathrm{NN}_u(x) - \mathrm{NN}_{\bar{u}}(x) + \mathrm{NN}_d(x) - \mathrm{NN}_{\bar{d}}(x) + \mathrm{NN}_s(x) - \mathrm{NN}_{\bar{s}}(x)\right)$$

PDF Parametrization

To ease the fit, the output is parametrized by default in the “evolution” basis at the input scale Q_0

$$
\begin{aligned}
g &= g, \\
\Sigma &= u + \bar{u} + d + \bar{d} + s + \bar{s} + 2c\,, \\
T_3 &= (u + \bar{u}) - \left(d + \bar{d}\right)\,, \\
T_8 &= \left(u + \bar{u} + d + \bar{d}\right) - 2\left(s + \bar{s}\right)\,, \\
T_{15} &= \left(u + \bar{u} + d + \bar{d} + s + \bar{s}\right) - 3\left(c + \bar{c}\right)\,, \\
V &= (u - \bar{u}) + \left(d - \bar{d}\right) + \left(s - \bar{s}\right)\,, \\
V_3 &= (u - \bar{u}) - \left(d - \bar{d}\right)\,, \\
V_8 &= \left(u - \bar{u} + d - \bar{d}\right) - 2\left(s - \bar{s}\right)\,.
\end{aligned}
$$



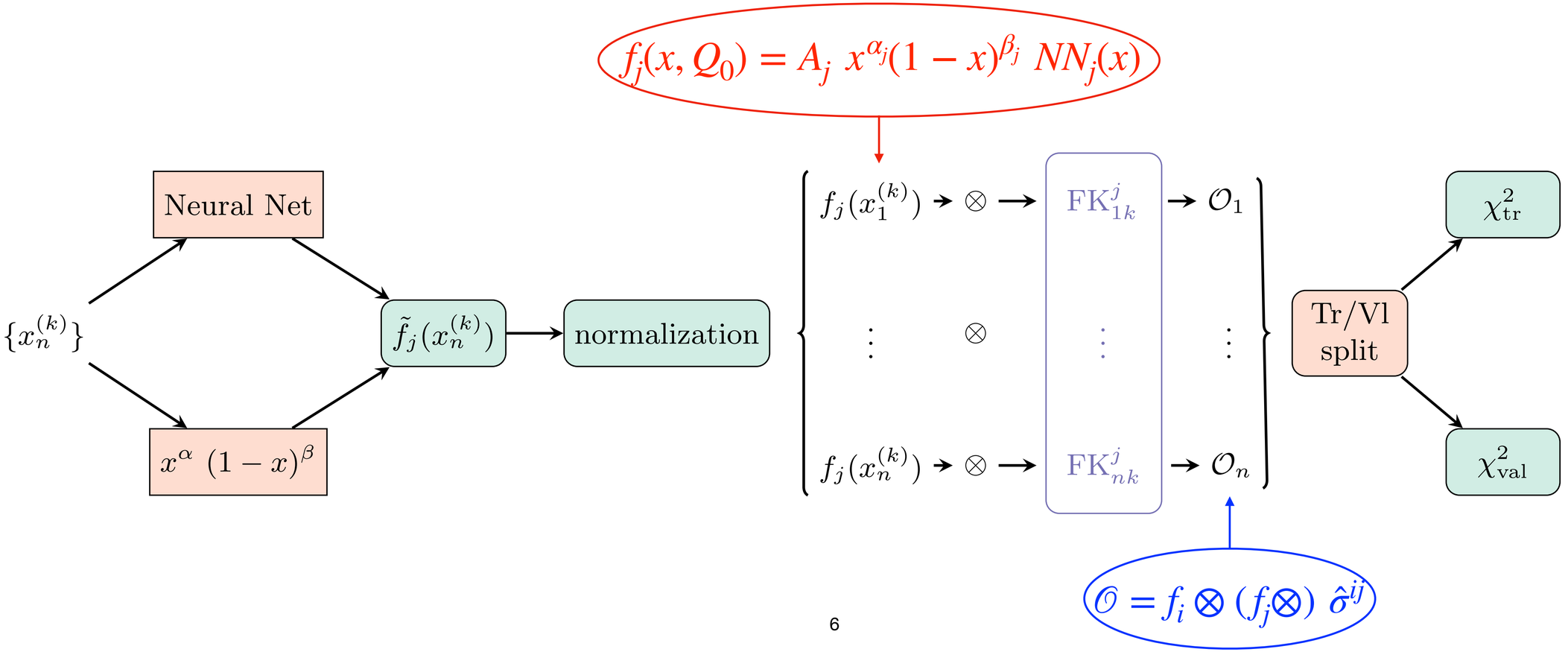
Different parametrization achieve similar results: basis independence

$$xV\left(x, Q_0\right) \propto \mathrm{NN}_V(x)$$

$$xV\left(x, Q_0\right) \propto \left(\mathrm{NN}_u(x) - \mathrm{NN}_{\bar{u}}(x) + \mathrm{NN}_d(x) - \mathrm{NN}_{\bar{d}}(x) + \mathrm{NN}_s(x) - \mathrm{NN}_{\bar{s}}(x)\right)$$

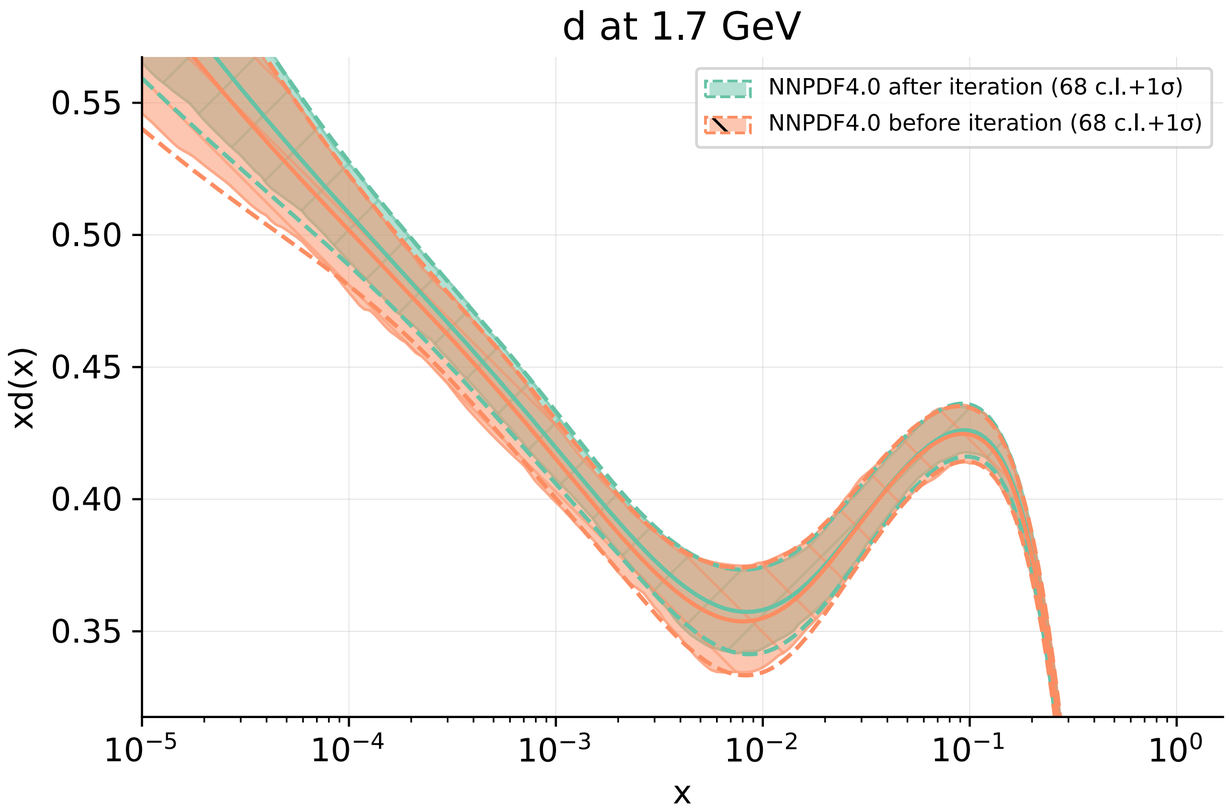
PDF fitting as a Machine learning problem

$$\mathcal{O} = \sum_{ij} \int dx_1 dx_2 \; f_i(x_1, \mu_F) \; f_j(x_2, \mu_F) \; \hat{\sigma}_{ij}(x_1, x_2, \mu_R, \mu_F)$$



The loss function

The t_0 prescription, $\chi^2 \to \chi^2_0$



$$\chi^2 = \sum_N (\mathcal{O}_i - D_i)\ cov^{-1}\ (\mathcal{O}_j - D_j)$$

$$cov_{ij} \longrightarrow cov_{ij} + t_{0i} t_{0j} s_i s_j$$

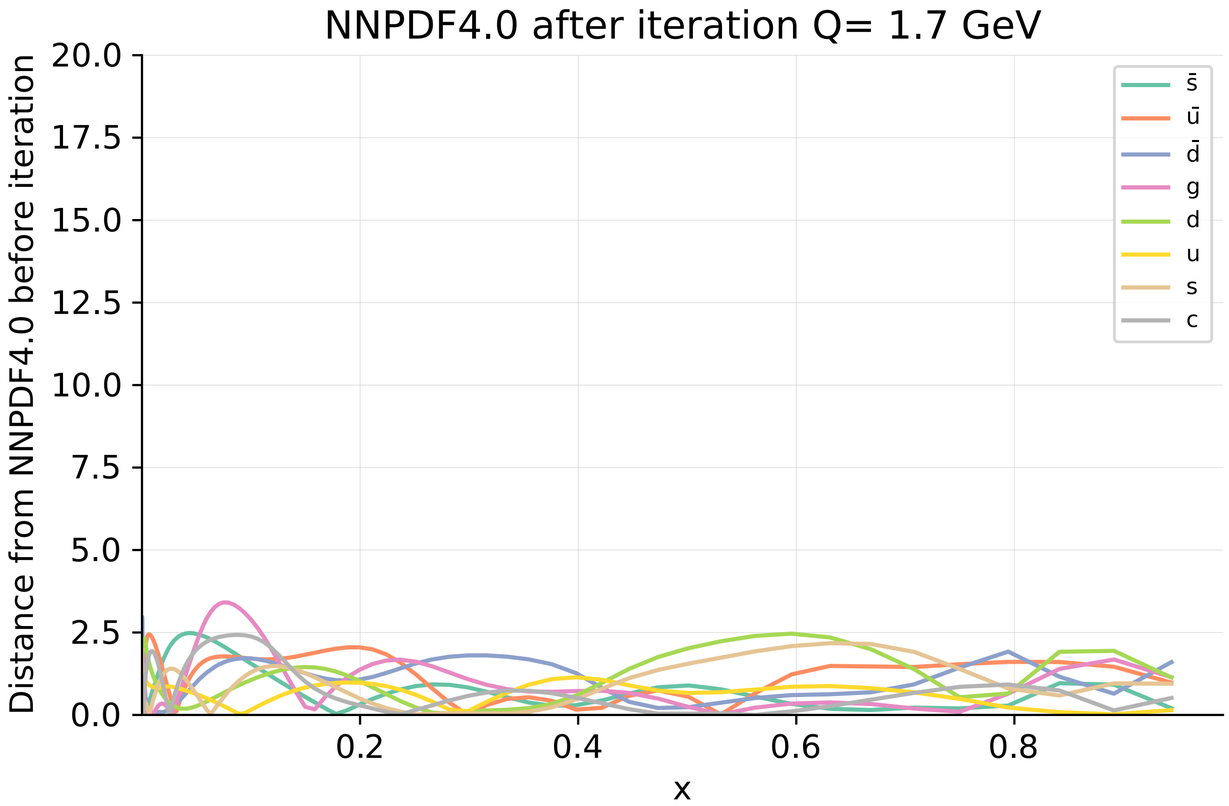
educated guess for $\mathcal{O}$

The χ^2 in the loss function optimized in NNPDF fits is not the χ^2 to the experimental data, but a modified form to account for multiplicative uncertainties

arXiv:0912.2276

The loss function

The t_0 prescription, $\chi^2 \to \chi^2_0$



$$\chi^2 = \sum_N (\mathcal{O}_i - D_i)\ cov^{-1}\ (\mathcal{O}_j - D_j)$$

$$cov_{ij} \longrightarrow cov_{ij} + t_{0i} t_{0j} s_i s_j$$

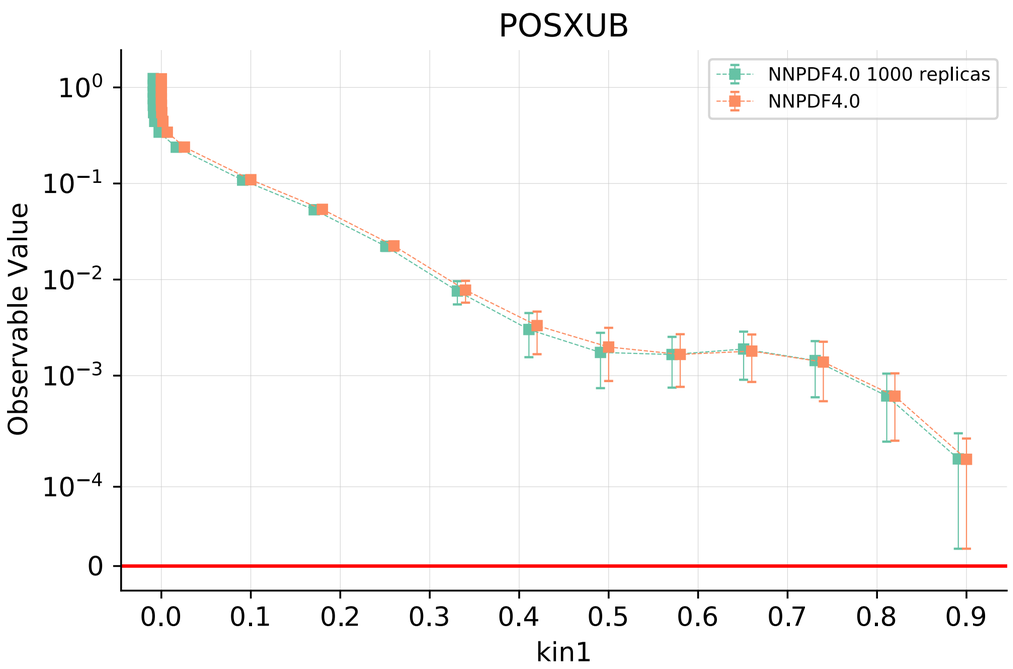
educated guess for $\mathcal{O}$

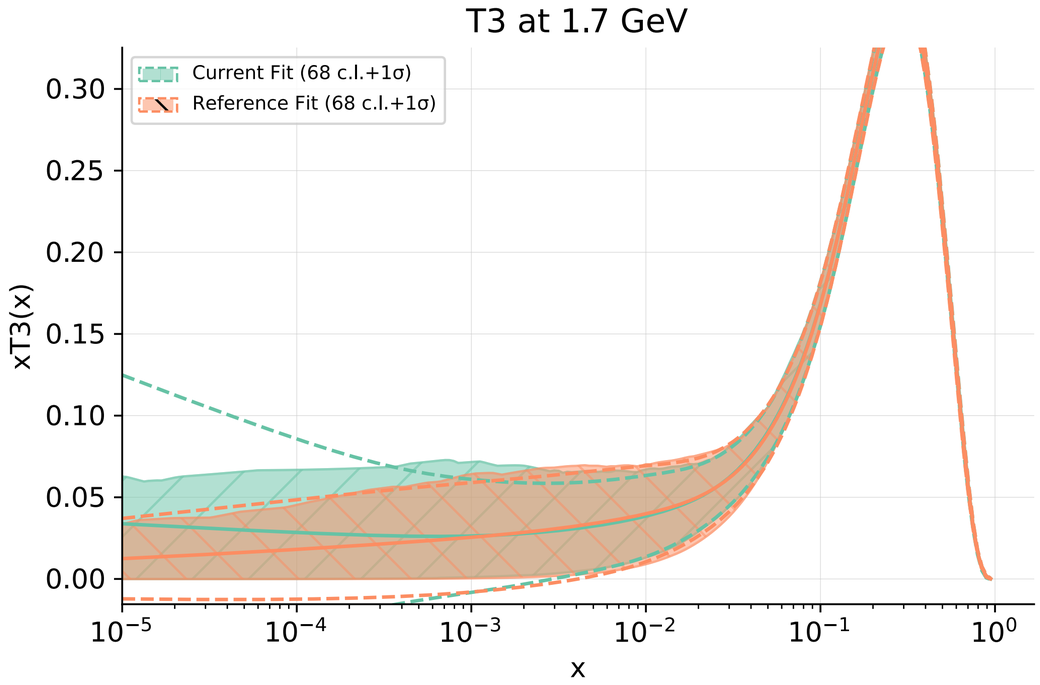
The χ^2 in the loss function optimized in NNPDF fits is not the χ^2 to the experimental data, but a modified form to account for multiplicative uncertainties

arXiv:0912.2276

The loss function

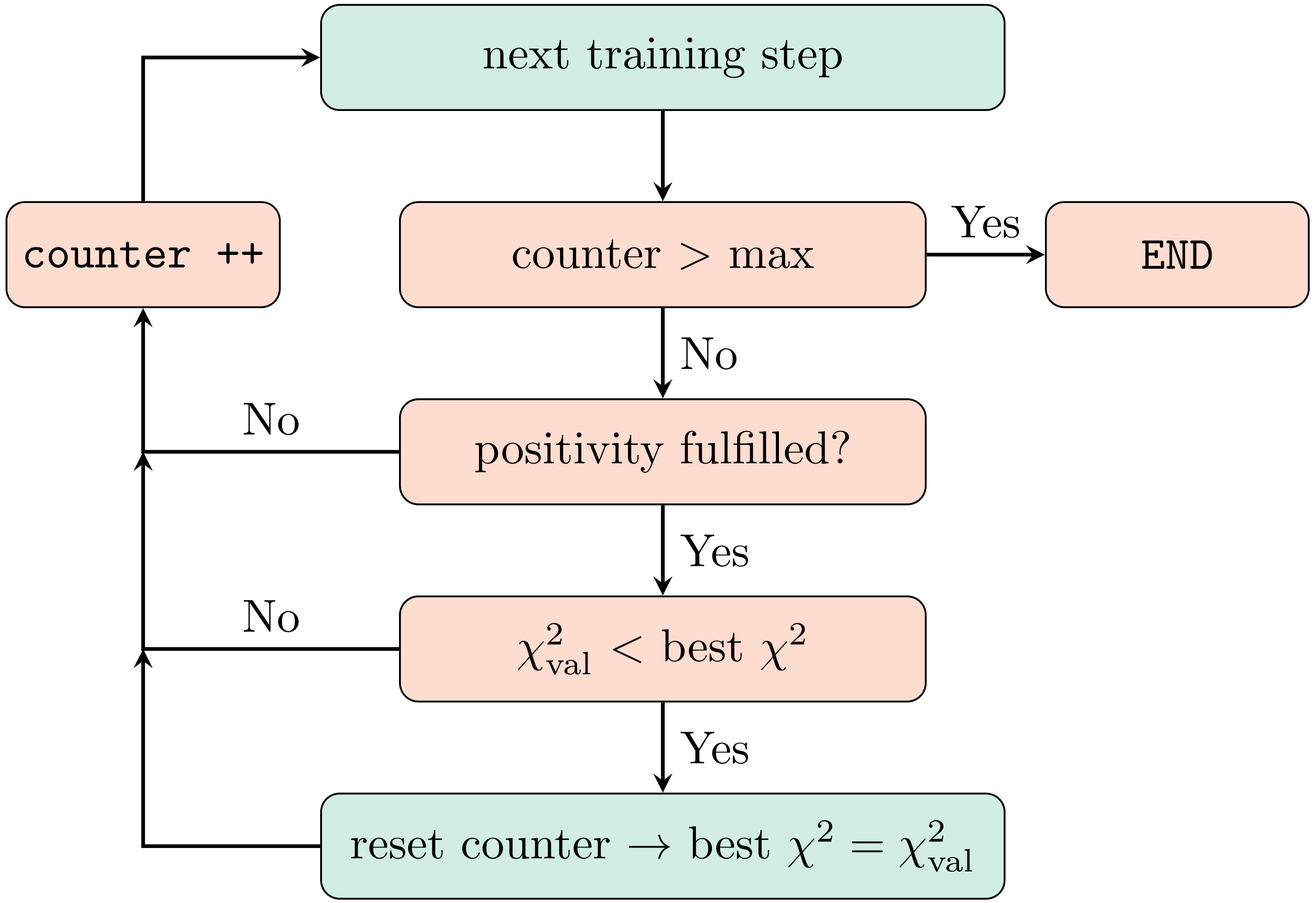
Positivity and integrability

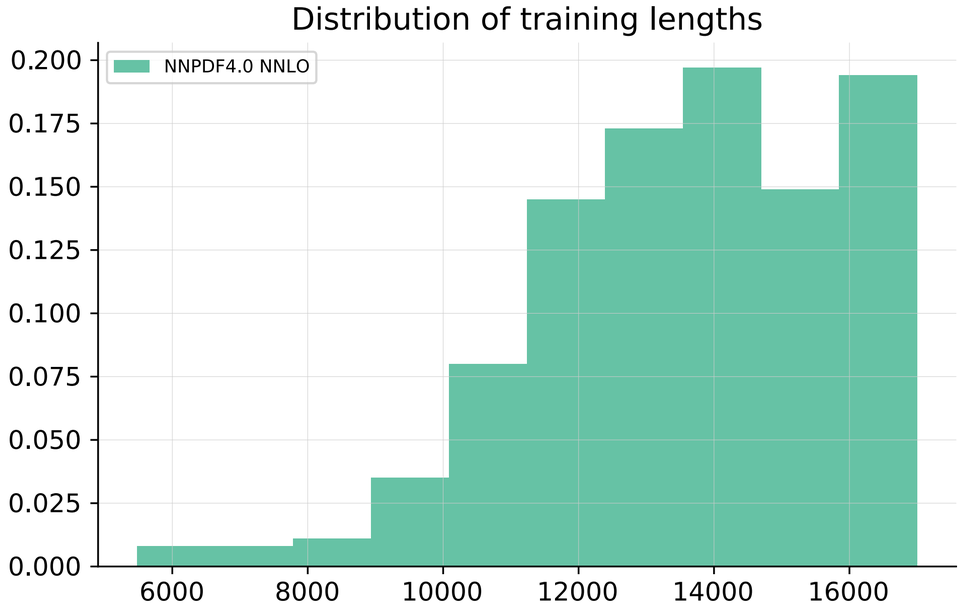




$$\mathscr{L} = \chi_0^2 + \lambda_{pos}\Theta(\sigma < 0) + \lambda_{int}\Theta(\sigma > \text{th})$$

Stopping algorithm

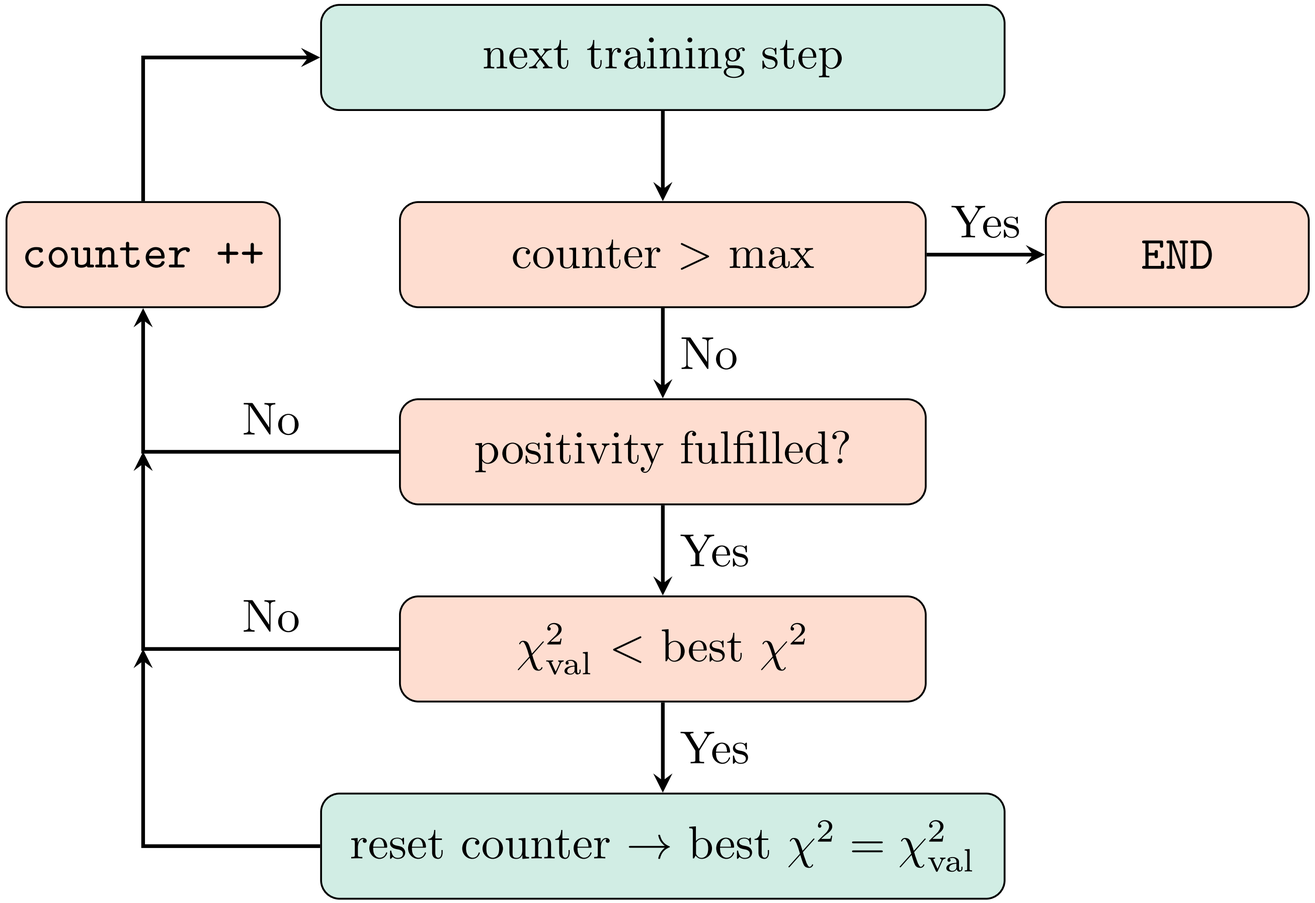


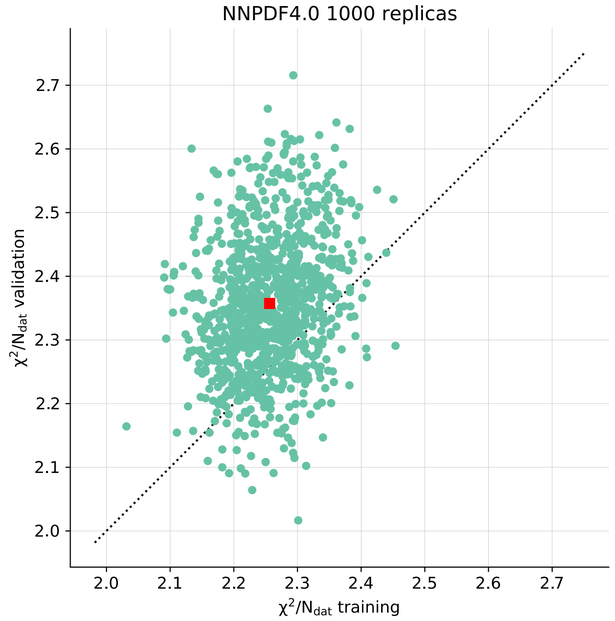


Regardless of the training algorithm or frameworks used the fitting method consist on

1. Reducing the loss function
2. Check the constraints are fulfilled
3. Continue until the validation metric stops improving.

Stopping algorithm

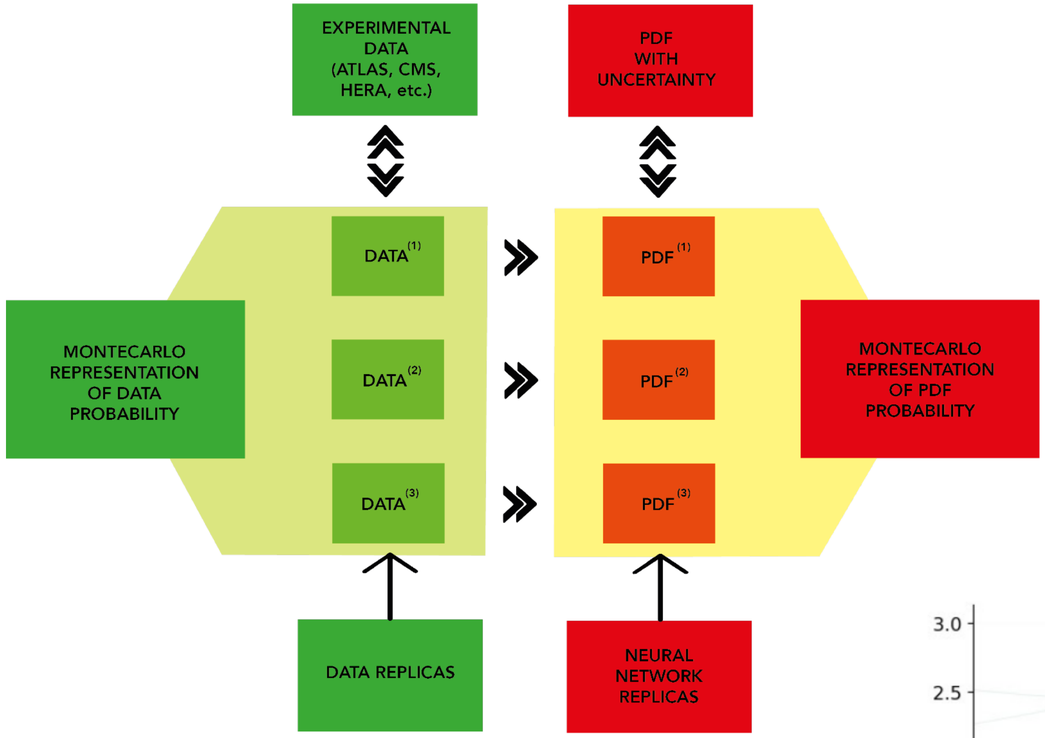


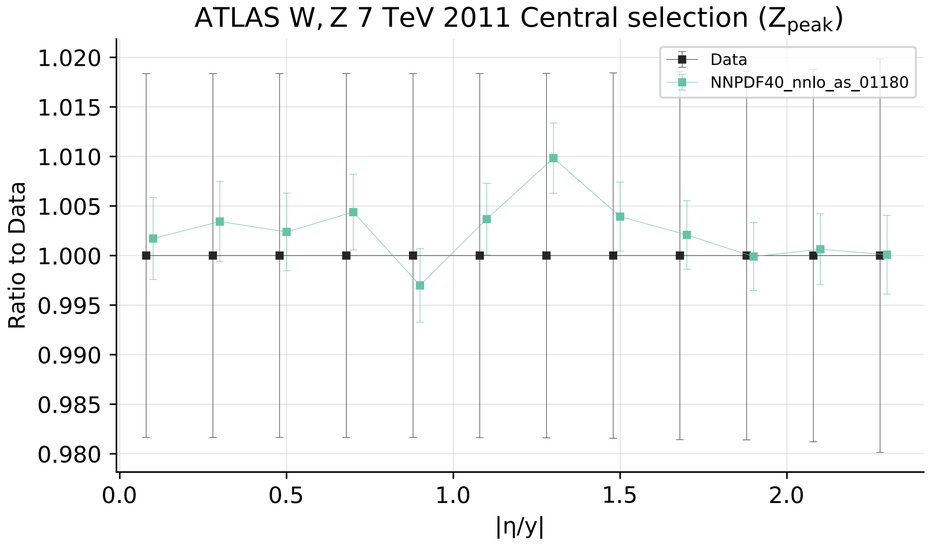


Regardless of the training algorithm or frameworks used the fitting method consist on

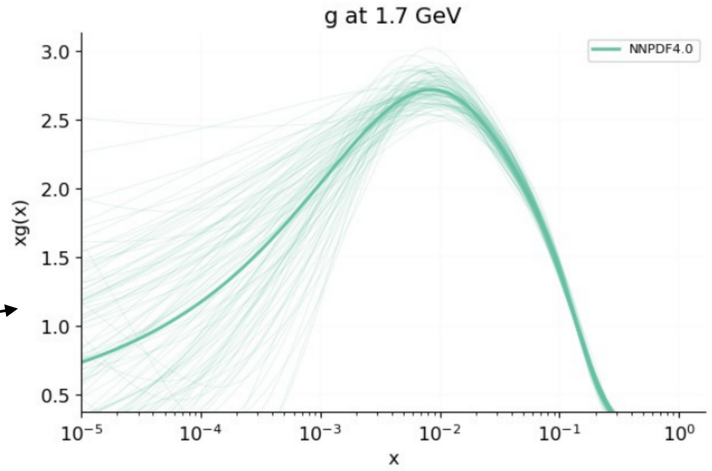
1. Reducing the loss function
2. Check the constraints are fulfilled
3. Continue until the validation metric stops improving.

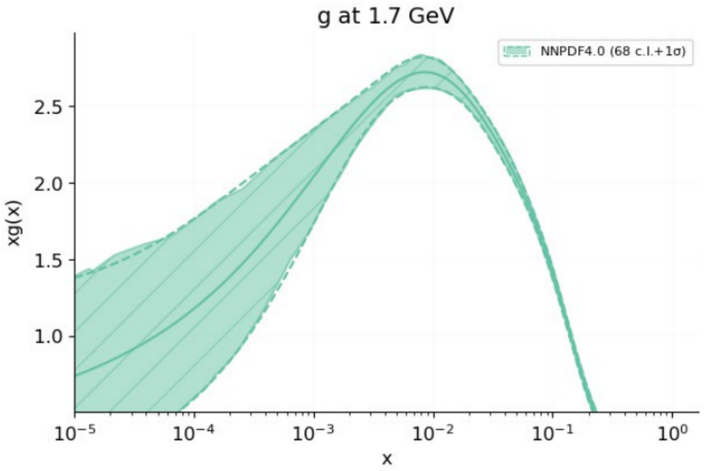
Uncertainties, from data to PDF





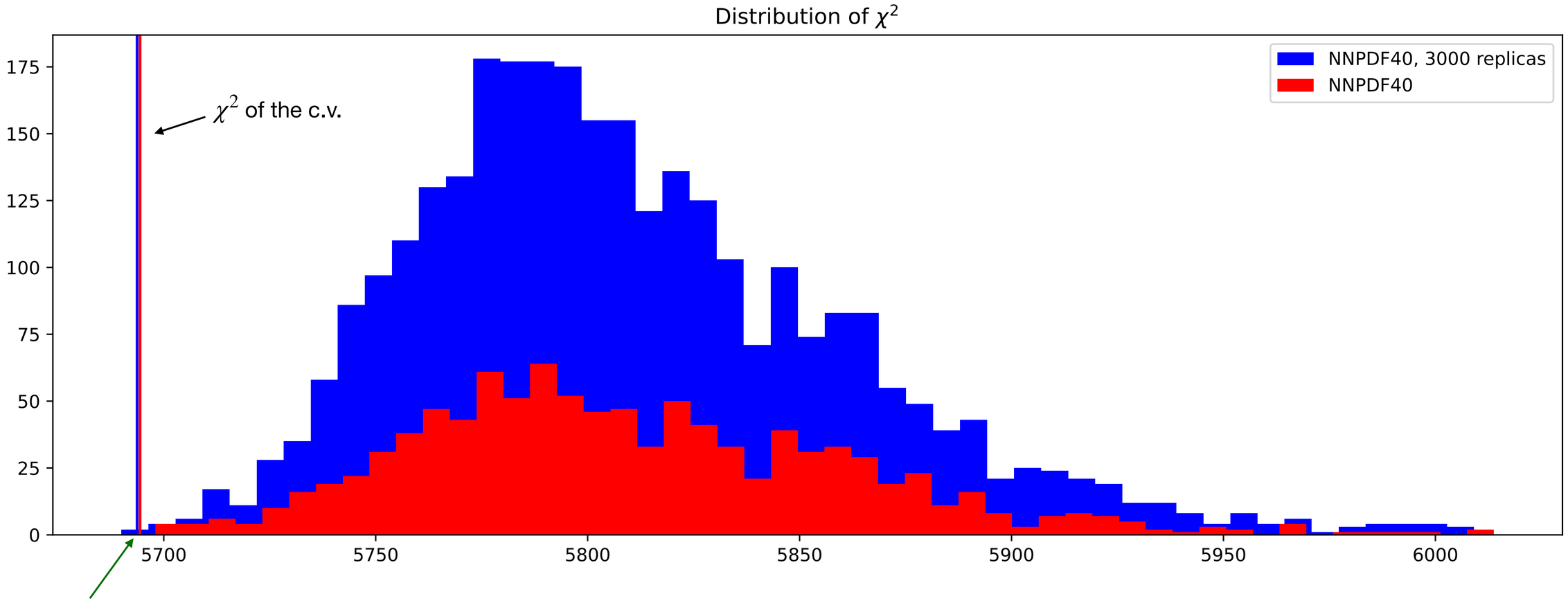
Perform thousand of fits, each to an “new” measure of the experimental data available.



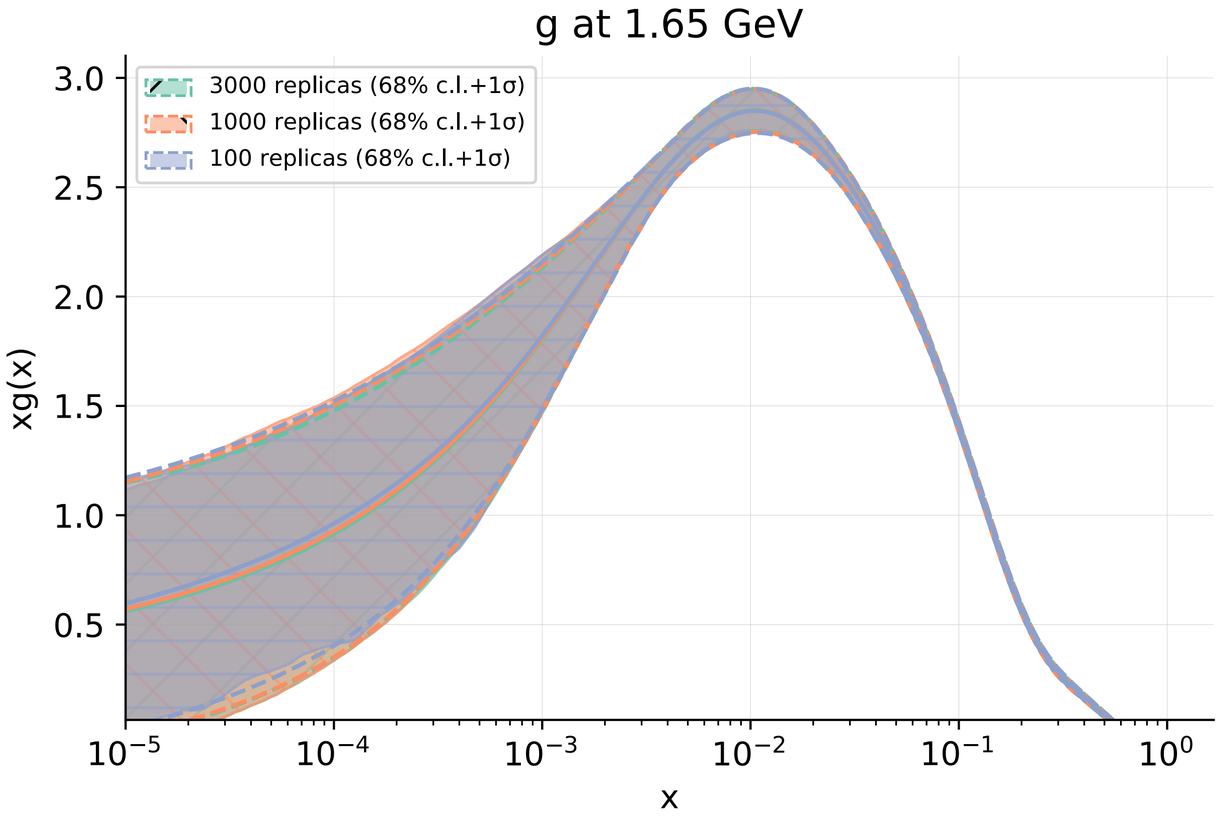


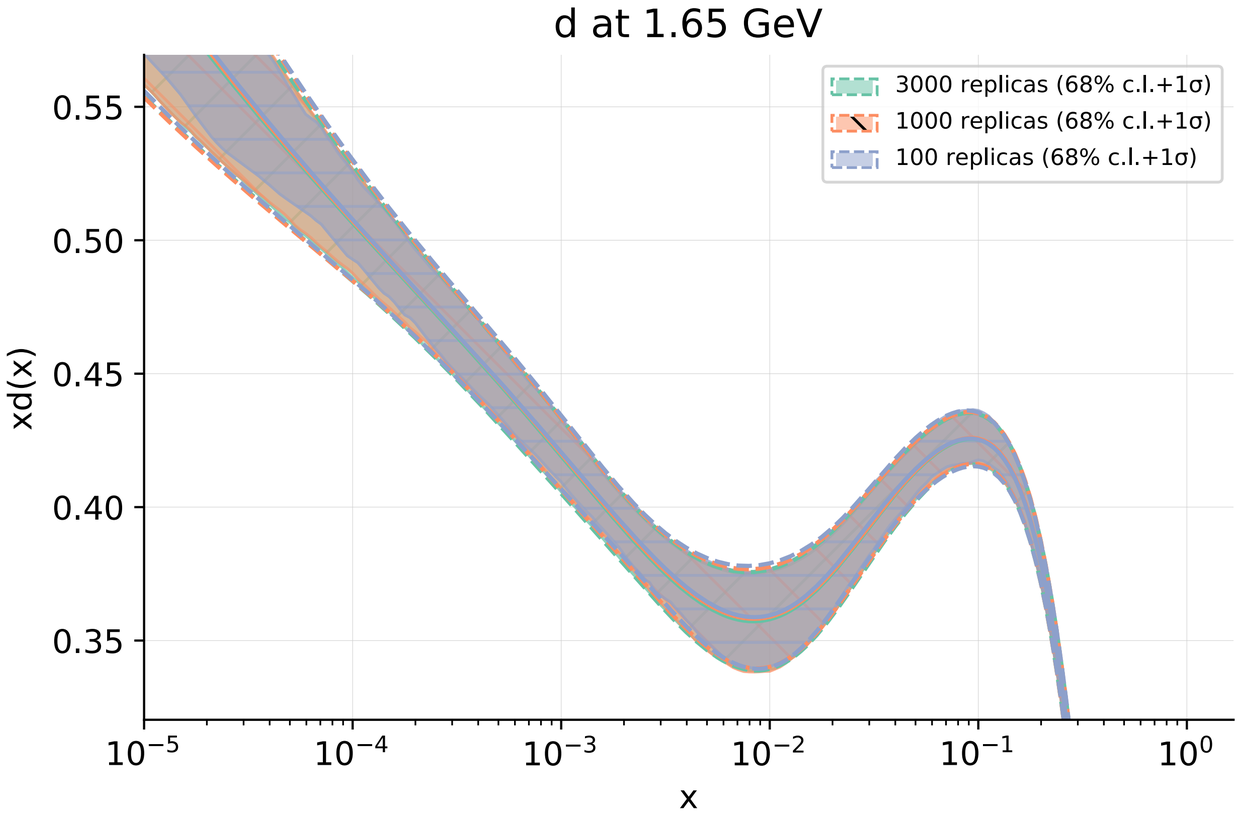
χ^2_0 distribution of the replicas

A common misconception is that the central PDF corresponds to the best fit to the data.



How do the PDFs change with the number of replicas?





Fitting methodology summary

1. Read the target data and associated experimental uncertainties

2. Generate replicas around the central value of the data by varying each datapoint according to the experimental covariance matrix.

3. For each replica, train a machine learning model with the minimization of the $\sim\chi_0^2$ as the target.

4. Use the ensemble of fitted replicas as measurements from which to take central value and uncertainty band.

Automatic hyperparameter selection

The usage of Neural Networks had as primary goal eliminating the biases associated with the choice of a specific functional form.

However, there are still many choices associated with the optimization:

- Number and with of the layers
- Activation functions and initialization
- Optimization algorithm (and associated parameters)
- Training length, stopping patience, etc.
- Strength of lagrange multipliers (positivity, integrability)

Collectively called “hyperparameters”, usually selected manually.

In order to remove any kind of human intervention, better to do them automatically.

Automatic hyperparameter selection

The usage of Neural Networks had as primary goal eliminating the biases associated with the choice of a specific functional form.

However, there are still many choices associated with the optimization:

- Number and with of the layers
- Activation functions and initialization
- Optimization algorithm (and associated parameters)
- Training length, stopping patience, etc.
- Strength of lagrange multipliers (positivity, integrability)

Collectively called “hyperparameters”, usually selected manually.

In order to remove any kind of human intervention, better to do them automatically.

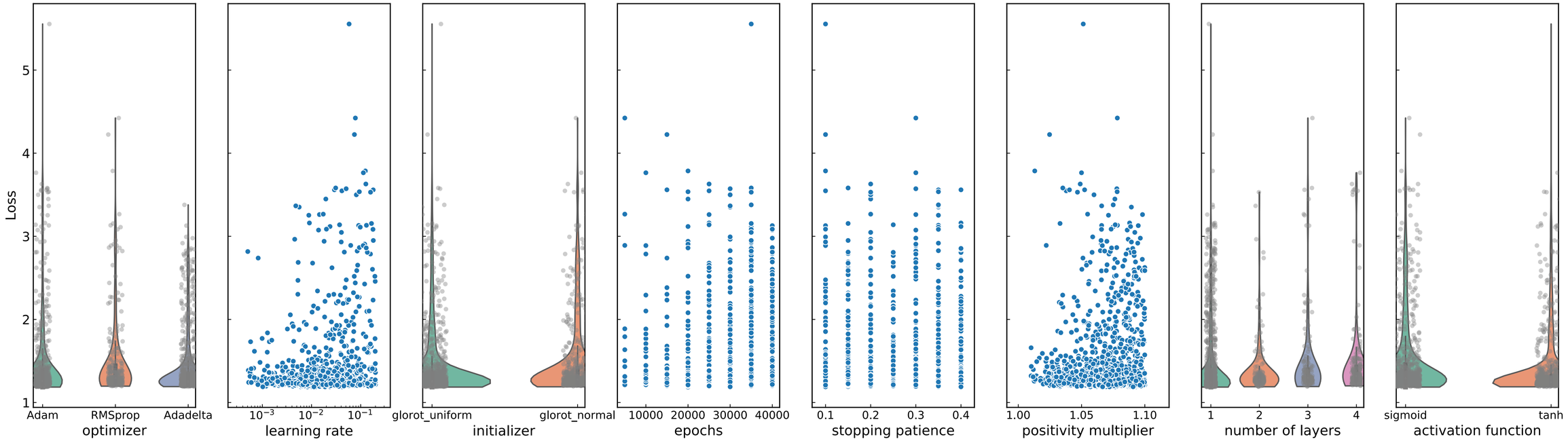
Automatic hyperparameter selection

Model selection

Select a model such that:

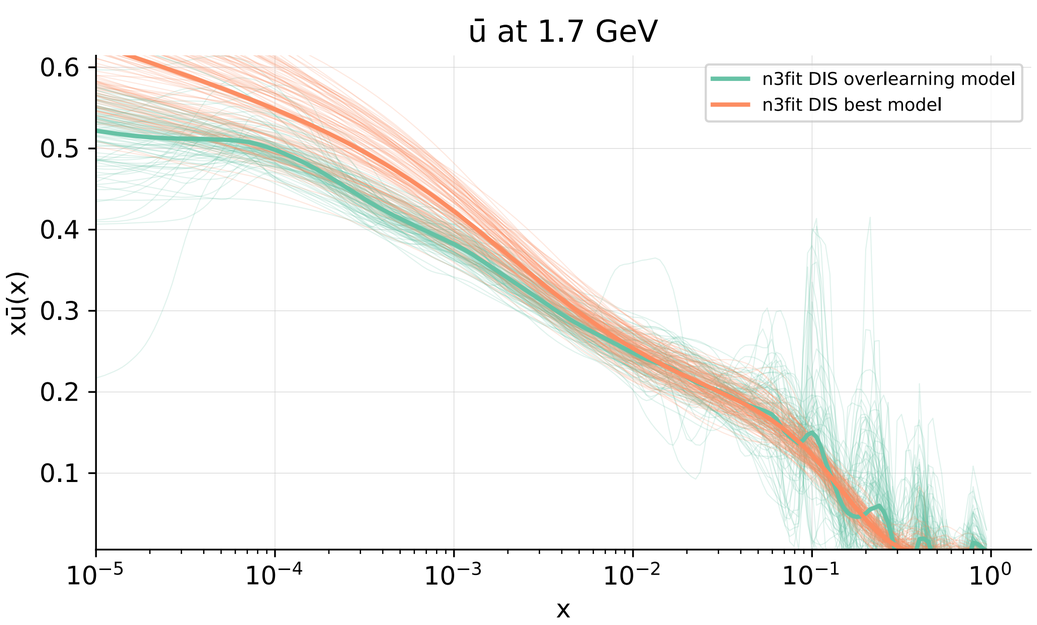
- The χ^2 is minimized (so the data is well described)
- Generalizable
- Fast (choose the faster methodology for the same quality!)
- Stable upon variations (we don't want to redo it too often!)

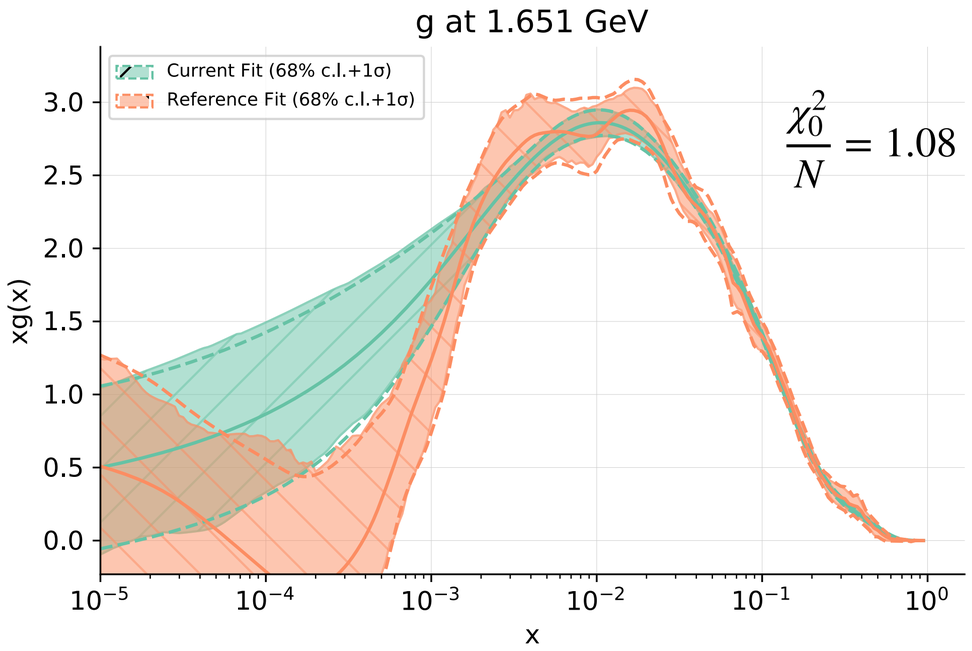
Perform many fits with many different hyperparameter choices and select the absolute best



Automatic hyperparameter selection

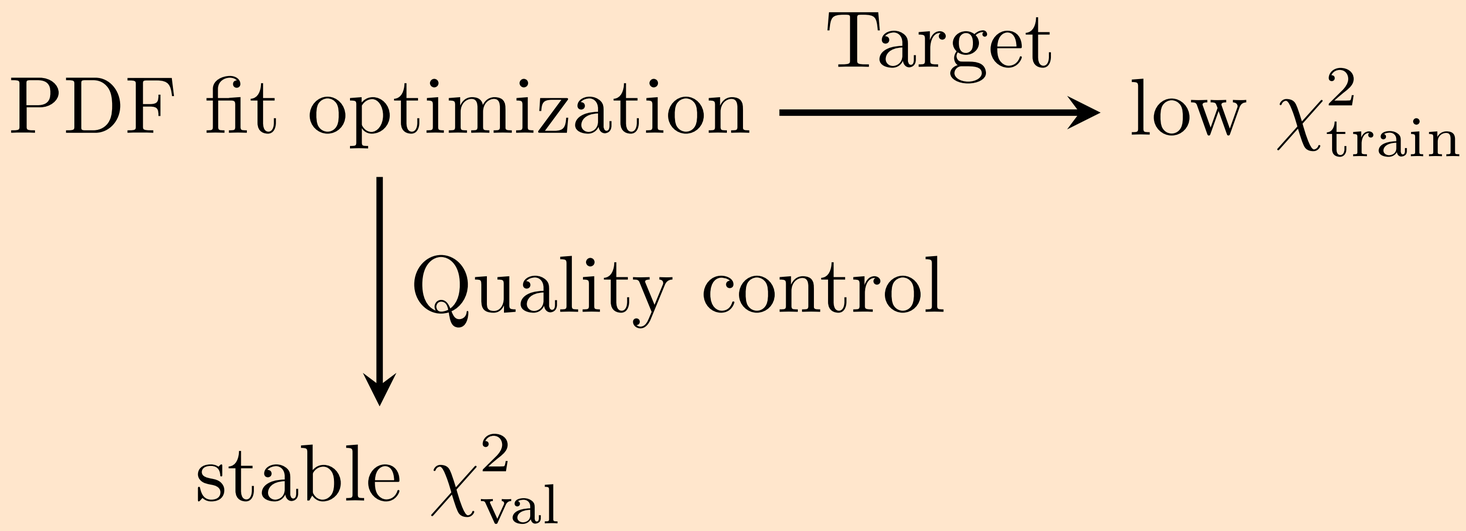
With great power comes great responsability





Getting a χ^2 many units below the nominal NNPDF4.0 (~1.16) is relatively “easy”, but that doesn’t mean it is a good fit.

Automatic hyperparameter selection

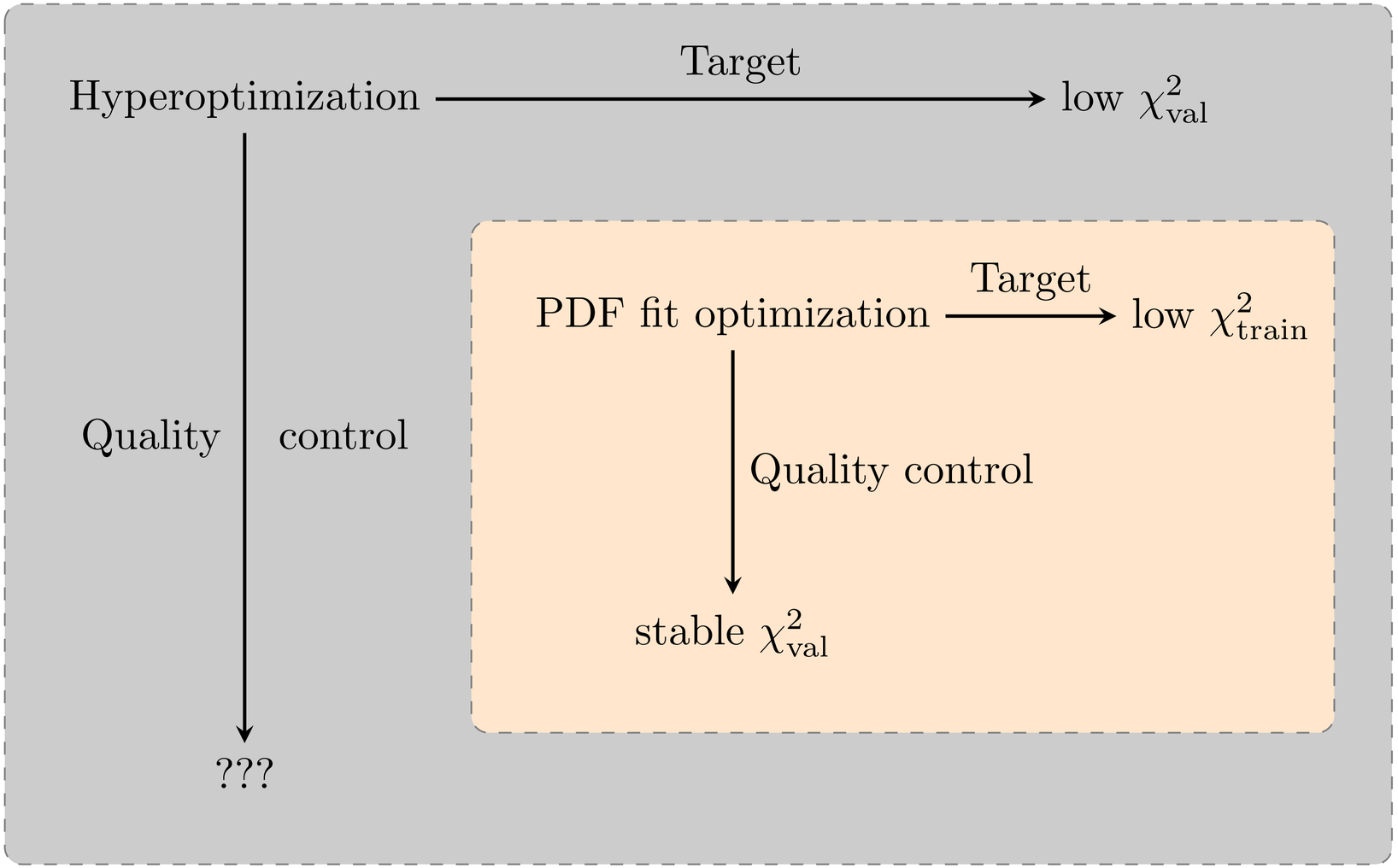


K-folding cross validation:

1. Divide data into k sets
2. Leave one out and fit using the union of the k-1 sets that are still in
3. Compute a reward/loss function on the datasets that are left out

$$\mathscr{L}(\text{parameters}) = \frac{1}{k}\sum_{k}^{i}\frac{\chi_i^2}{N_i}$$

Automatic hyperparameter selection

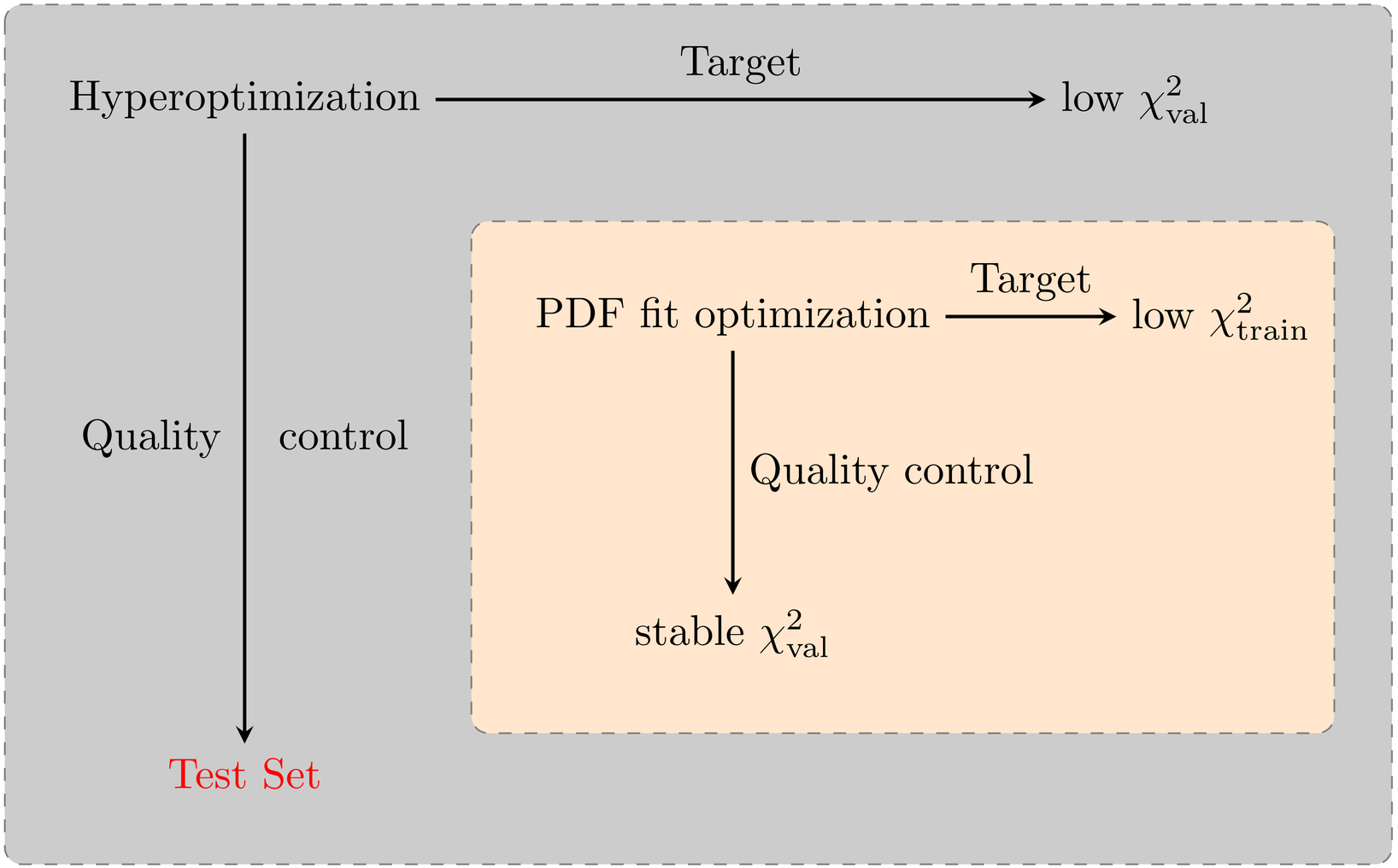


K-folding cross validation:

1. Divide data into k sets
2. Leave one out and fit using the union of the k-1 sets that are still in
3. Compute a reward/loss function on the datasets that are left out

$$\mathscr{L}(\text{parameters}) = \frac{1}{k}\sum_{k}^{i}\frac{\chi^2_i}{N_i}$$

Automatic hyperparameter selection



K-folding cross validation:

1. Divide data into k sets
2. Leave one out and fit using the union of the k-1 sets that are still in
3. Compute a reward/loss function on the datasets that are left out

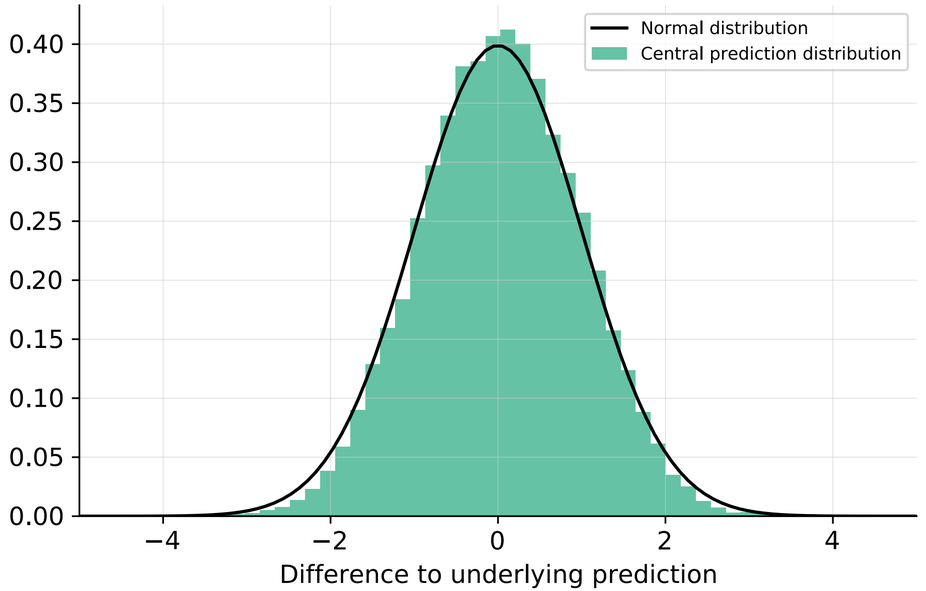
$$\mathscr{L}(\text{parameters}) = \frac{1}{k}\sum_{k}^{i}\frac{\chi_i^2}{N_i}$$

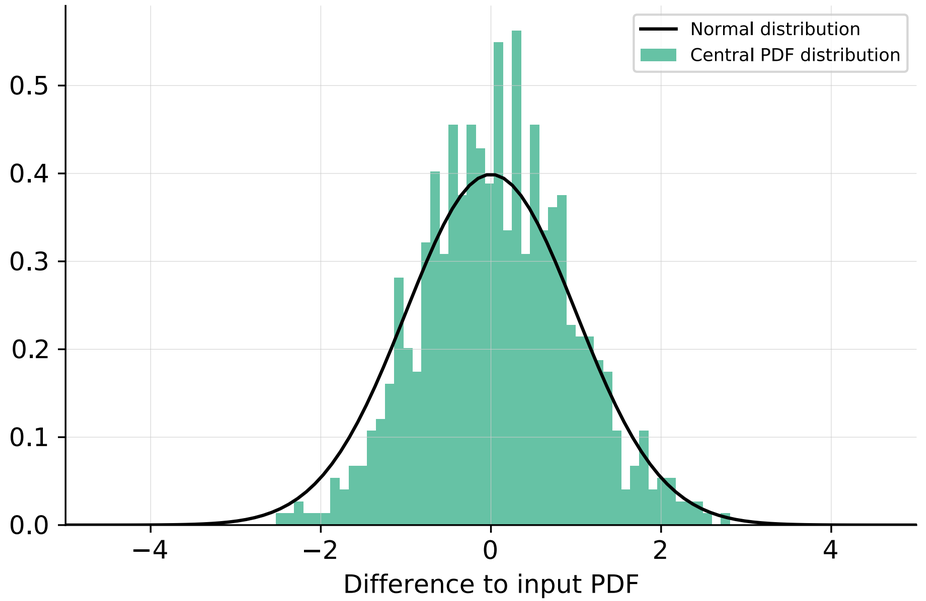
Validation and testing

Closure tests

1. Select some other PDF as the truth (an NNPDF replica or a fit from another group)
2. Generate fake data according to the theoretical predictions used in the fit
3. Generate variations of the data using the experimental uncertainties

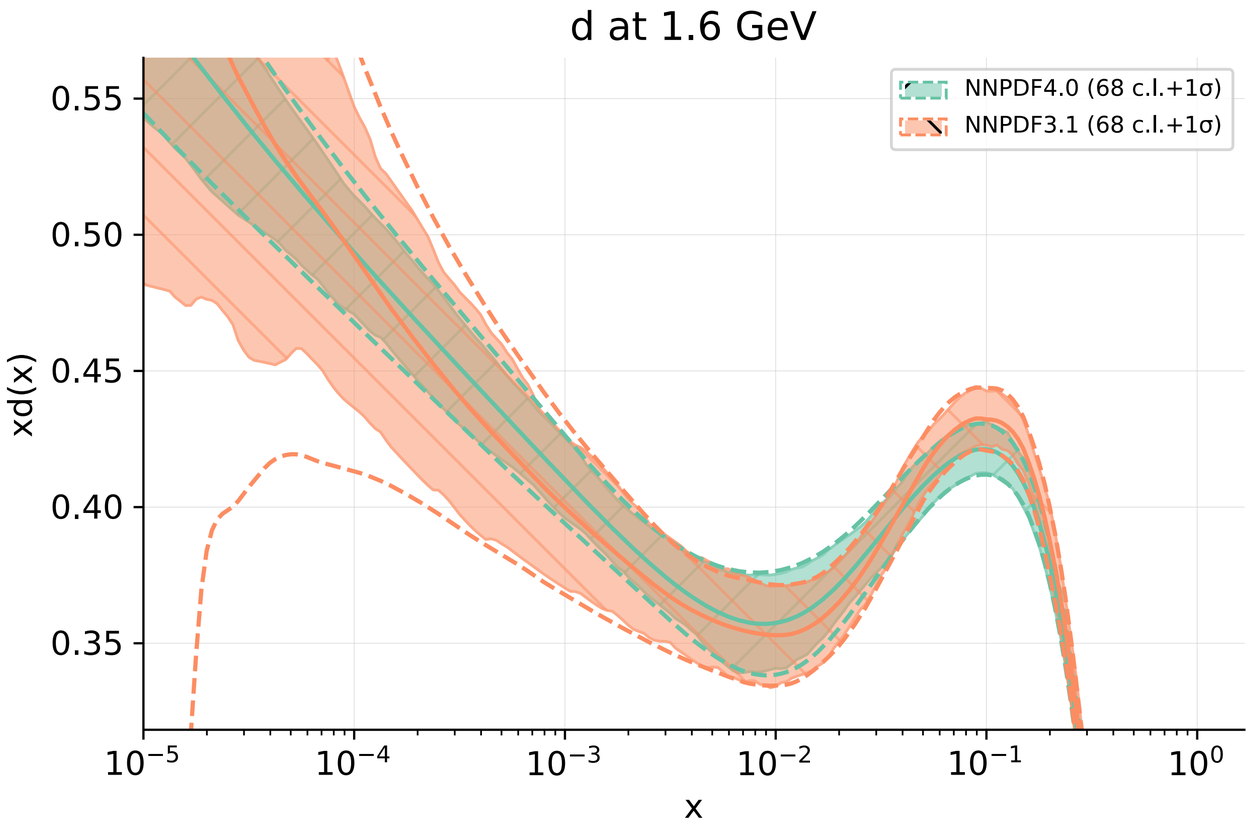
- ➡ Check whether the parametrization is flexible enough
- ➡ Check whether we can reproduce the “true” PDF if it were known
- ➡ Do all of that in an environment in which everything is consistent and no theoretical knowledge is missing (no MHOU)





Validation and testing

Future tests

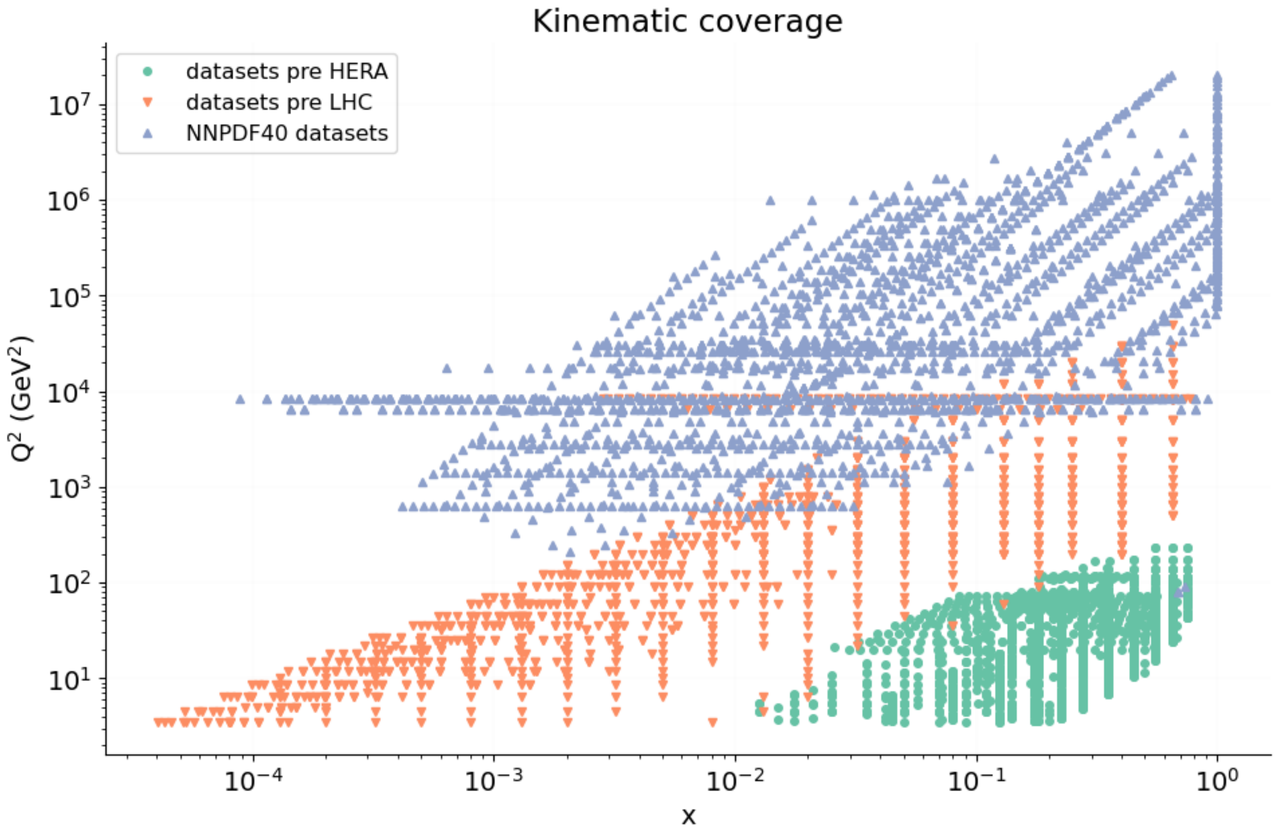


Going from NNPDF3.1 to 4.0, the error bands of the PDFs have shrinked considerably, is that related only to data?

Would NNPDF4.0 datasets be able to “predict” (or rather, accommodate) new data from future experiments?

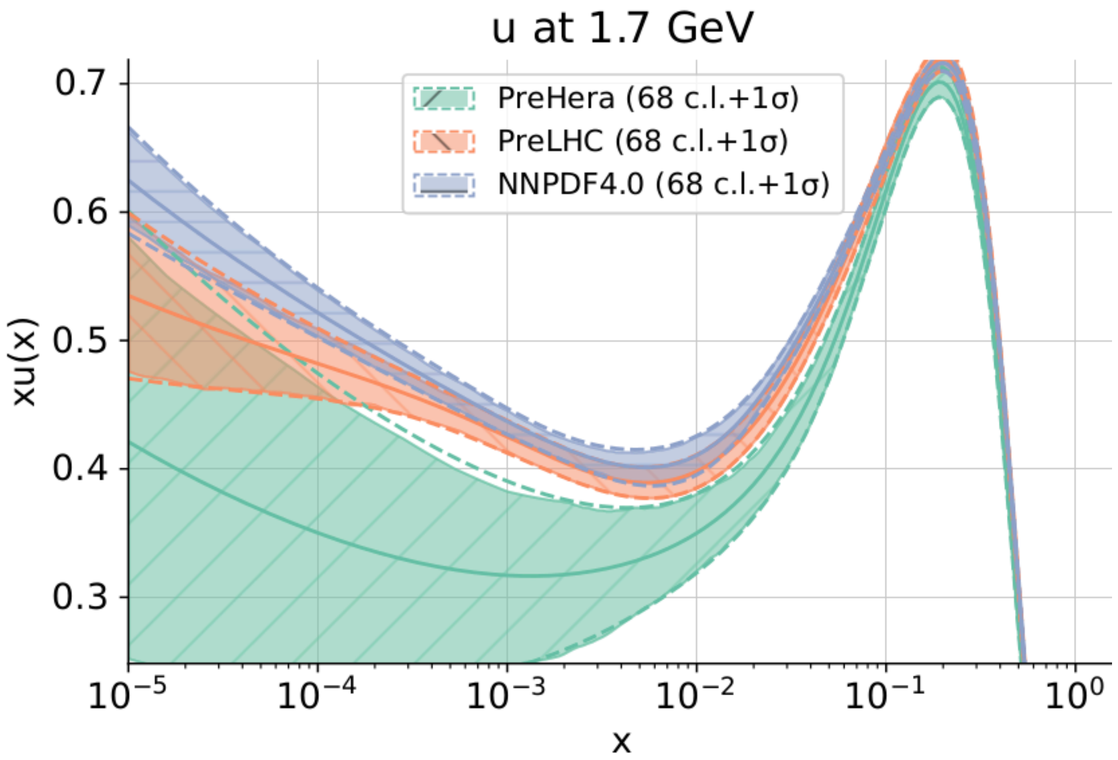
Validation and testing

Future tests



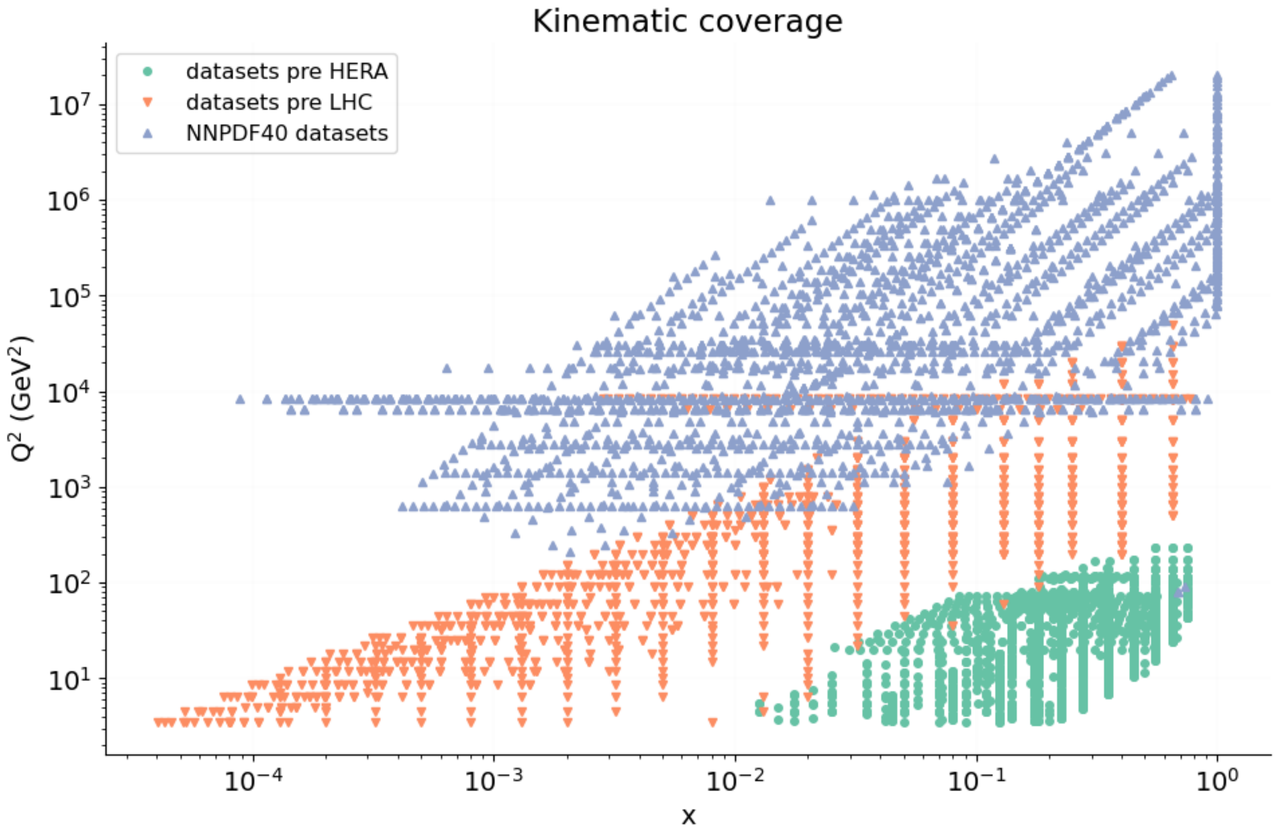
dataset \ fit	NNPDF4.0	pre-LHC	pre-HERA
pre-HERA	1.06	1.01	0.91
pre-LHC	1.20	1.21	**26.1**
NNPDF4.0	1.29	**2.15**	**22.57**

χ^2/N



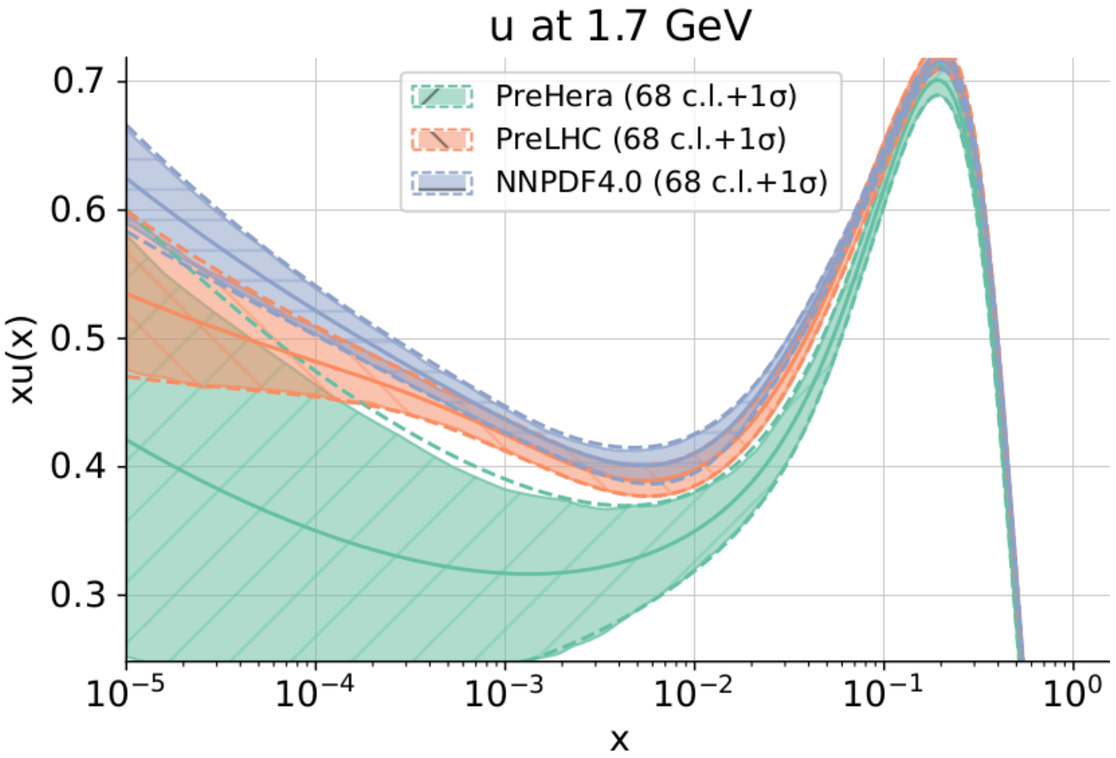
Validation and testing

Future tests



dataset \ fit	NNPDF4.0	pre-LHC	pre-HERA
pre-HERA			0.87
pre-LHC		1.18	**1.22**
NNPDF4.0	1.12	**1.30**	**1.38**

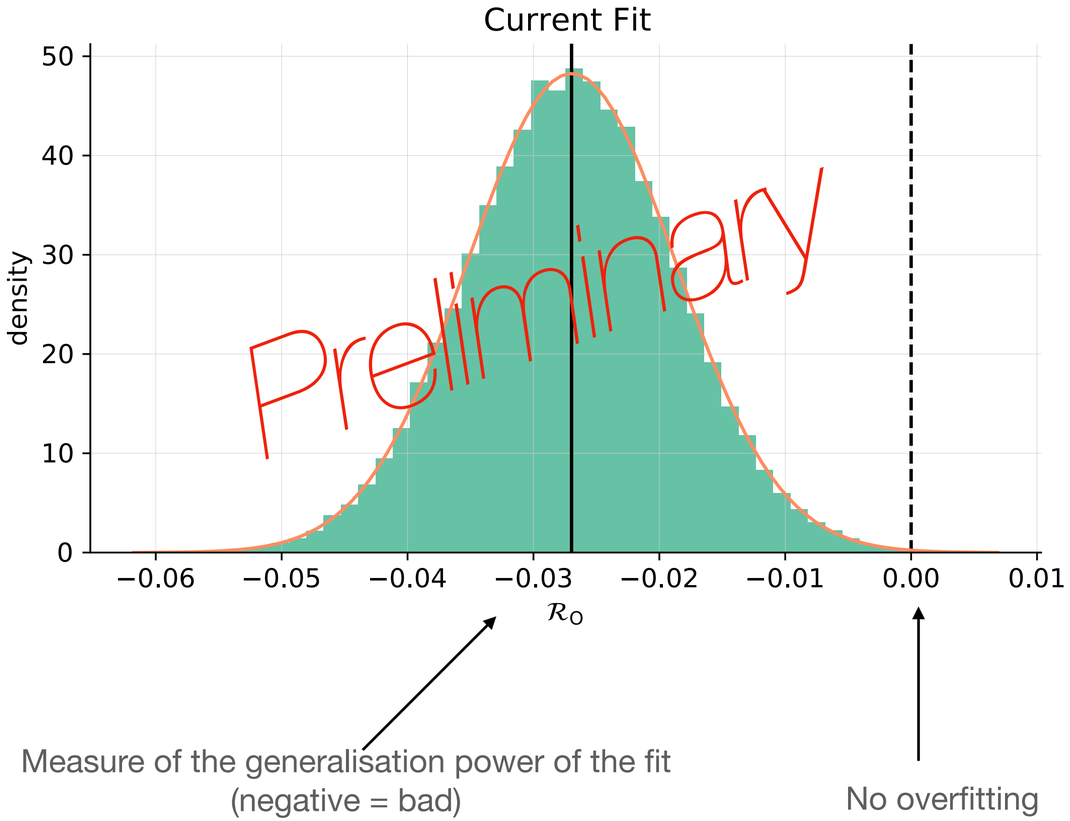
χ^2/N

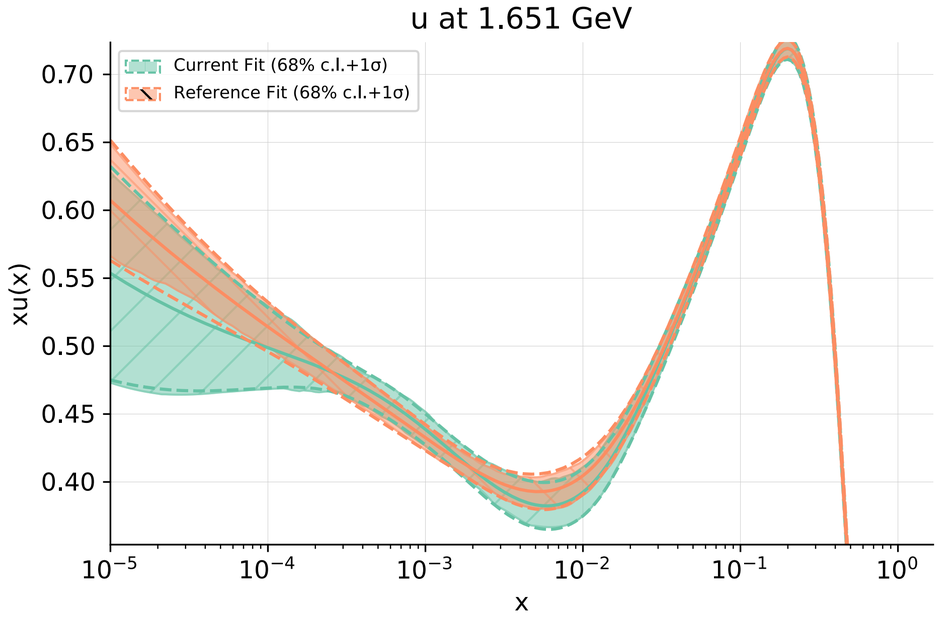


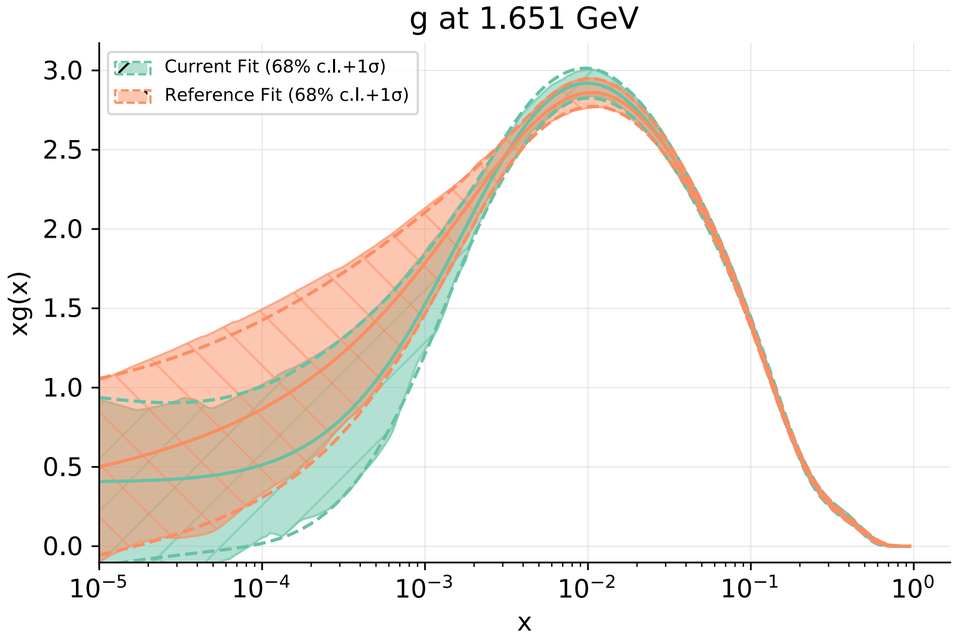
Validation and testing

Avoiding overfitting

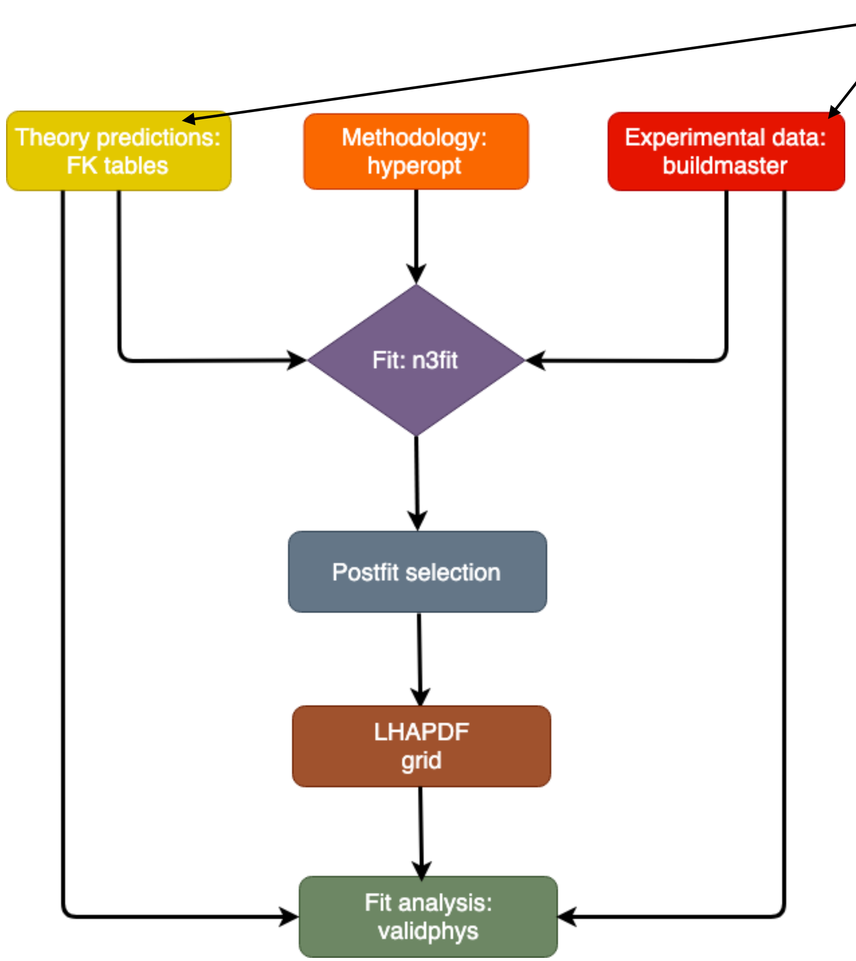
In the hyperopt section we saw an obviously overfitted PDF, what to do when it is not that obvious?







NNPDF fitting framework summary

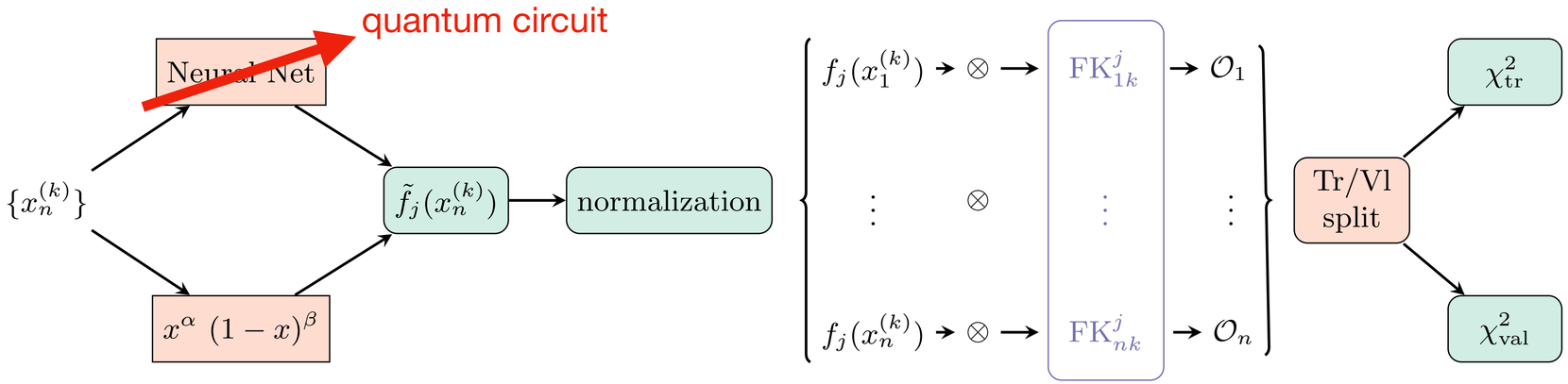


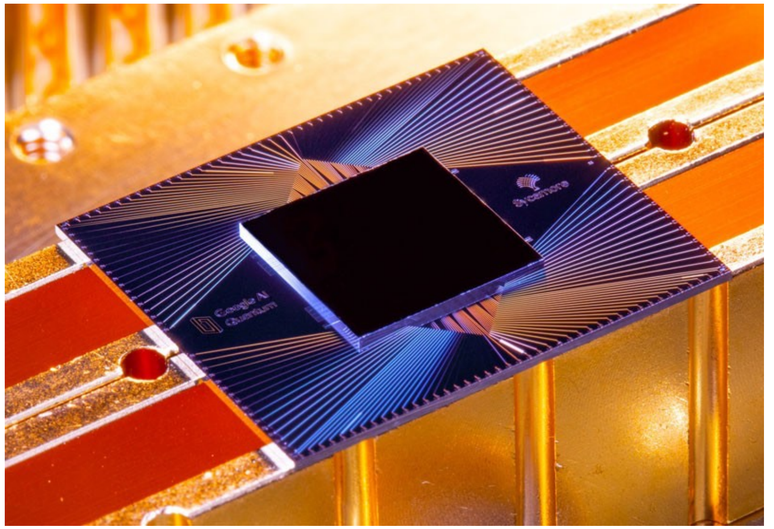
The ingredients for a complete fit:

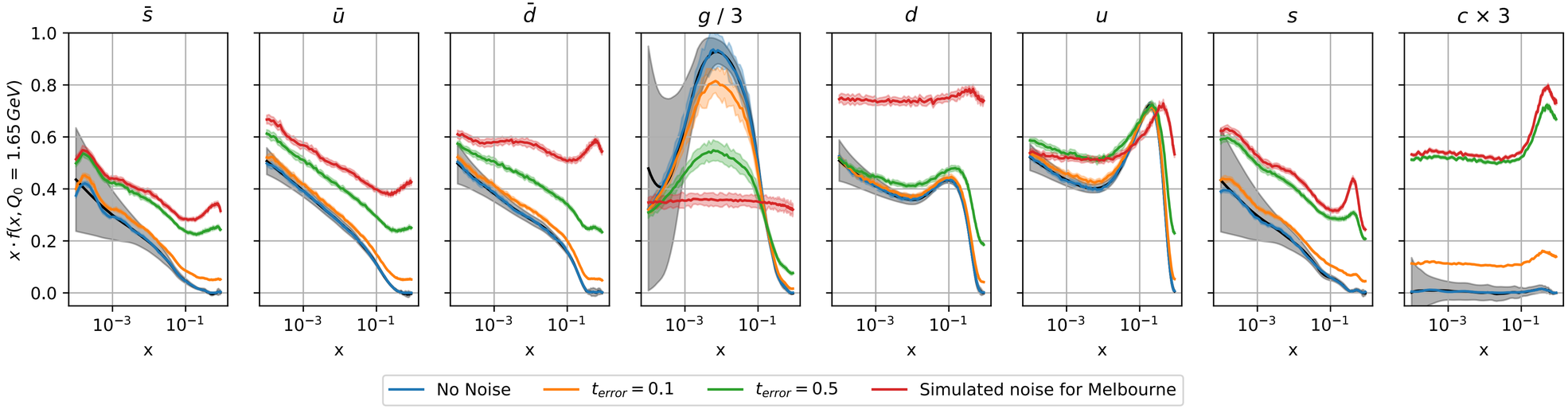
- Experimental data and uncertainties (hepdata)
- Theory predictions in the form of interpolation tables (plougshare, madgraph): Fast Kernel Tables
- Fitting framework (n3fit*) -> PDF at scale Q_0
- DGLAP evolution for any value of Q (Apfel, EKO)
- Postfit selection (eliminate outliers, underlearnt or wiggly replicas and double-check physical constraints)
- Final output: LHAPDF grid
- (optional) an analysis framework to facilitate creating nice plots and presentations

Some exotic possibilities

Quantum PDFs







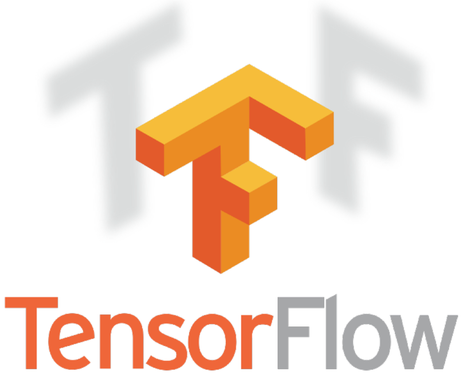
Open source

The whole NNPDF fitting framework is open source, documented and available to be used for all your PDF fitting needs!

- Code
- Data
- Theory Predictions
- Documentation
- Tutorials

https://github.com/NNPDF/nnpdf

https://docs.nnpdf.science/



The importance of Theory Predictions

for PDF fitting and beyond

The whole fitting machinery is based on the ability to perform the following calculation *extremely* fast

$$\mathcal{O} = \sum_{ij} \int dx_1 dx_2 \; f_i(x_1, \mu_F) \, f_j(x_2, \mu_F) \; \hat{\sigma}_{ij}(x_1, x_2, \mu_R, \mu_F)$$

But obviously, performing this integral ~10k times for the ~5k cross sections necessary for a fit would be impossible.

What can we do?

Since the PDF depends only on the values of x an Q: bin the cross section on the relevant variables

$$\frac{d^4\hat{\sigma}_{ij}}{d\mu_F d\mu_R dx_1 dx_2}$$

fastNLO

APPLgrid

PineAPPL

General process, PDF-independent, grid storage

Fast Kernel Tables (FKTables)

The evolution on the μ scales is exact (O(α^2)) so the grid needed during the fit can be further simplified: $\dfrac{d^2\hat{\sigma}_{ij}}{dx_1 dx_2}$

$$\mathcal{O} = \sum_{ij} \int dx_1 dx_2 \; f_i(x_1, \mu_F)\, f_j(x_2, \mu_F)\; \hat{\sigma}_{ij}(x_1, x_2, \mu_R, \mu_F) = f_i^{\alpha} f_j^{\beta} \hat{\sigma}^{ij}_{\alpha\beta}$$

x-grid

flavours

With O(50) points we can get a good representation of most observables, i.e., for each step of the fitting process we *just* need to contract the PDF with a tensor of *only* $4500 \times 50 \times 50 \times 14 \times 14 \simeq 10^9$ elements. Easy!

But actually, the convolution of such a big array with the luminosity takes roughly 1 second!
(and many possibilities for optimization are still available!)

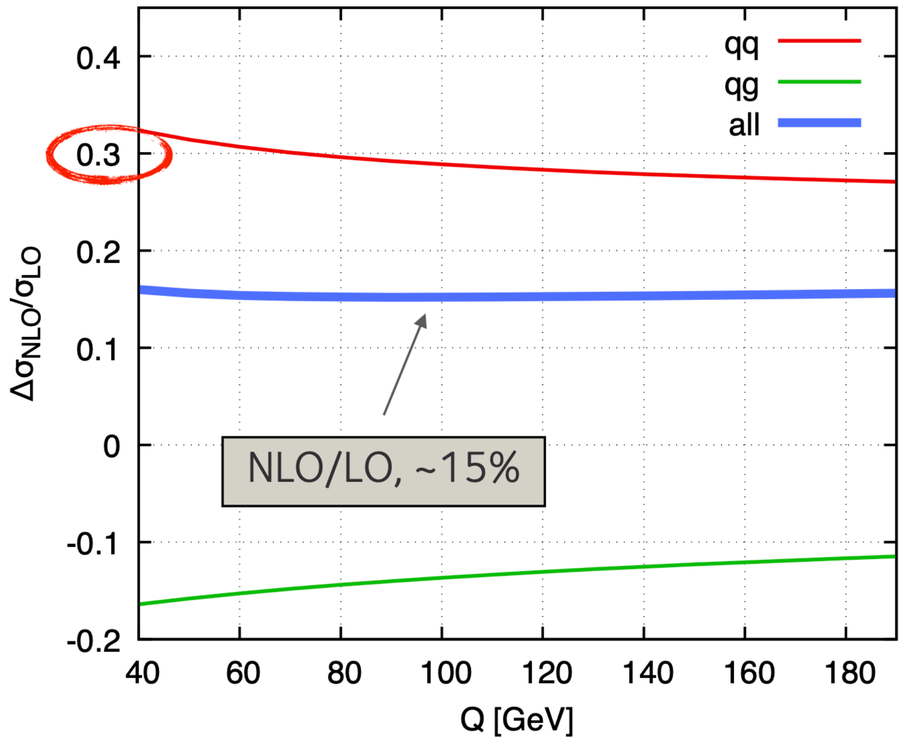
The importance of Theory Predictions

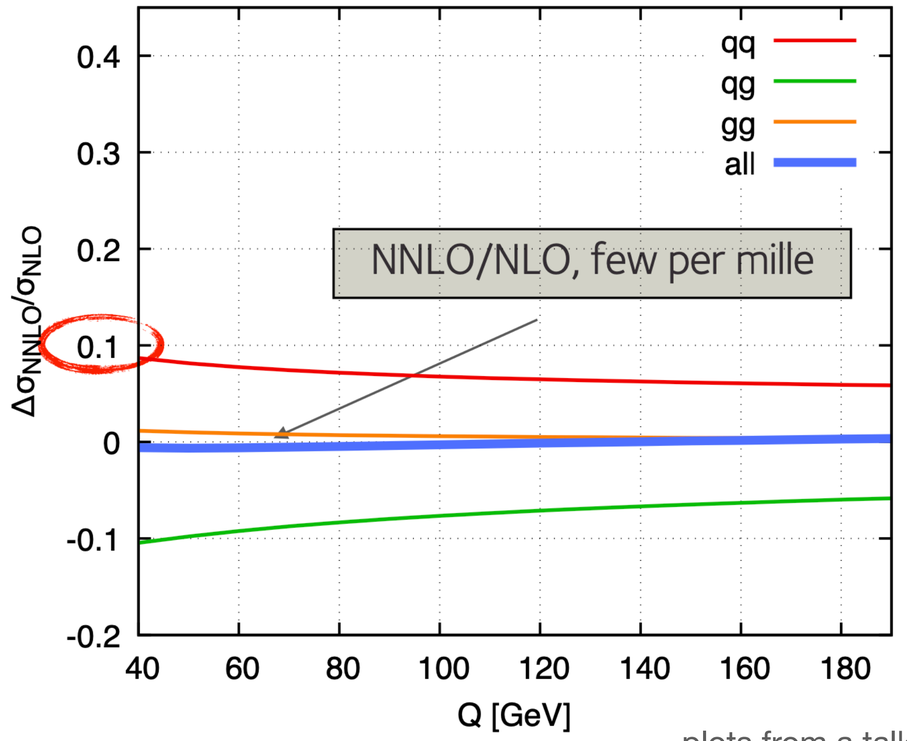
why are we interested in them?

Current frontier: N3LO

PDF Fits, however, are still struggling with implementing (already existing) NNLO corrections (only through k-factors!) or NLO EW

(but we can start thinking about N3LO DGLAP though!)





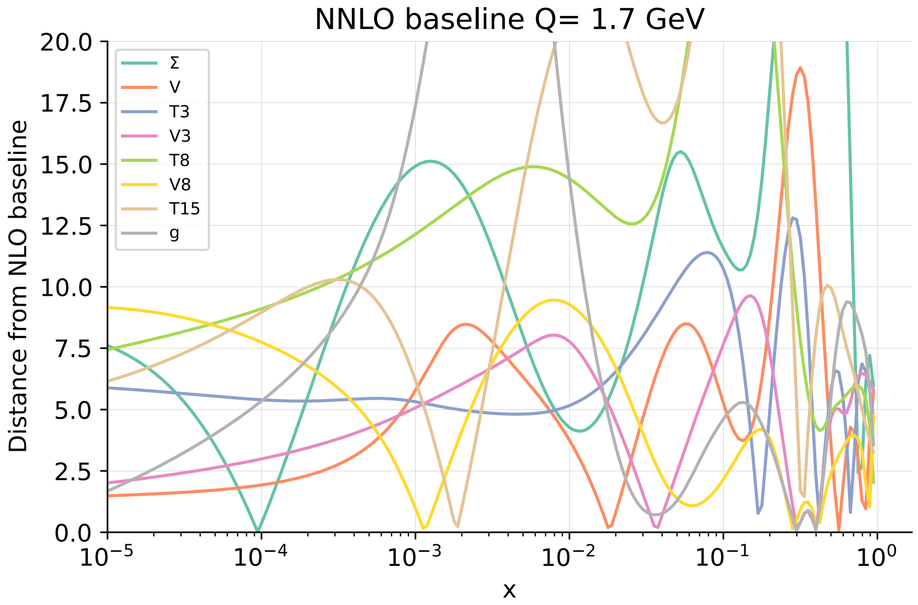
plots from a talk by F. Caola

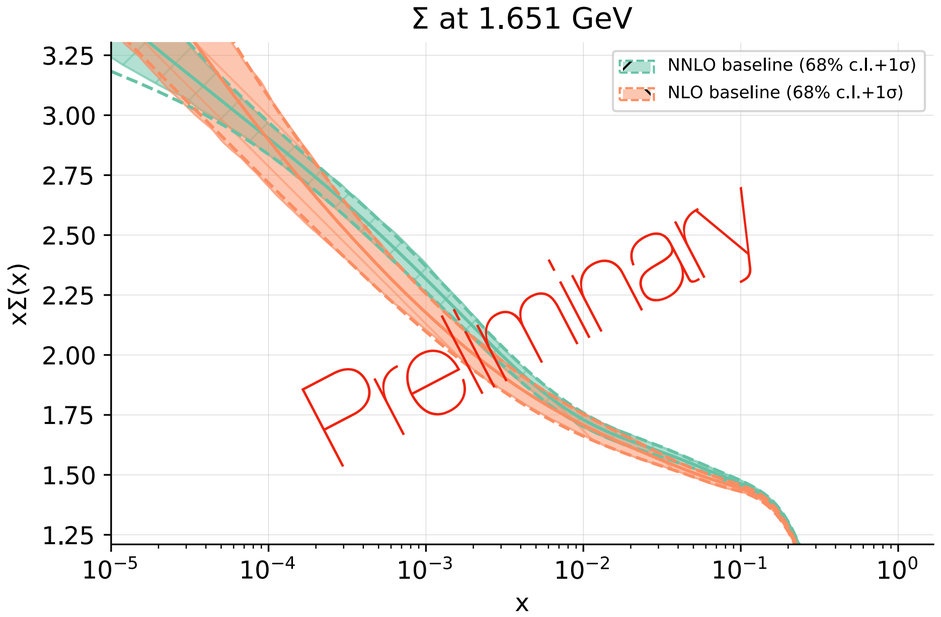
The future of PDF fitting

- N3LO evolution
- NNLO predictions
- NLO EW corrections
- Missing High Order Uncertainties
- Beyond proton PDFs
- Different theory settings

Most future plans for PDF require the generation of new interpolation grids, and generating them can take great amounts of time.

Highly desirable: some framework in which I can run the appropriate settings, let it run in the background and get the necessary interpolation grids maybe a month later.



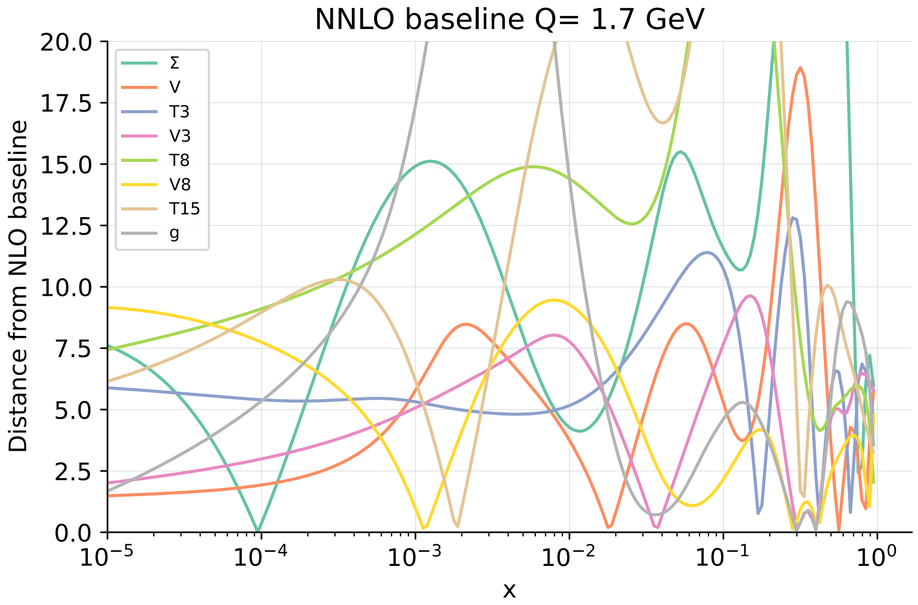


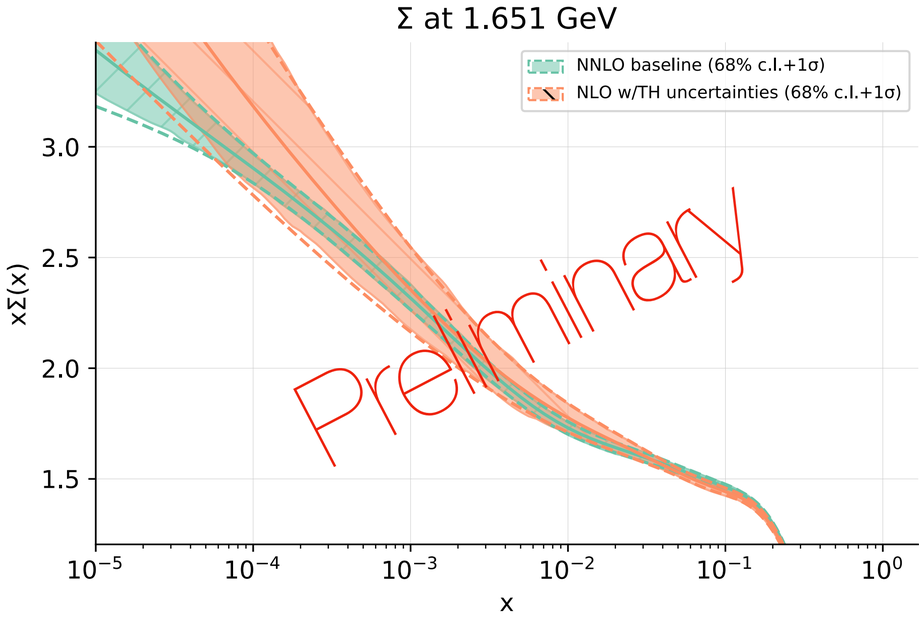
The future of PDF fitting

- N3LO evolution
- NNLO predictions
- NLO EW corrections
- Missing High Order Uncertainties
- Beyond proton PDFs
- Different theory settings

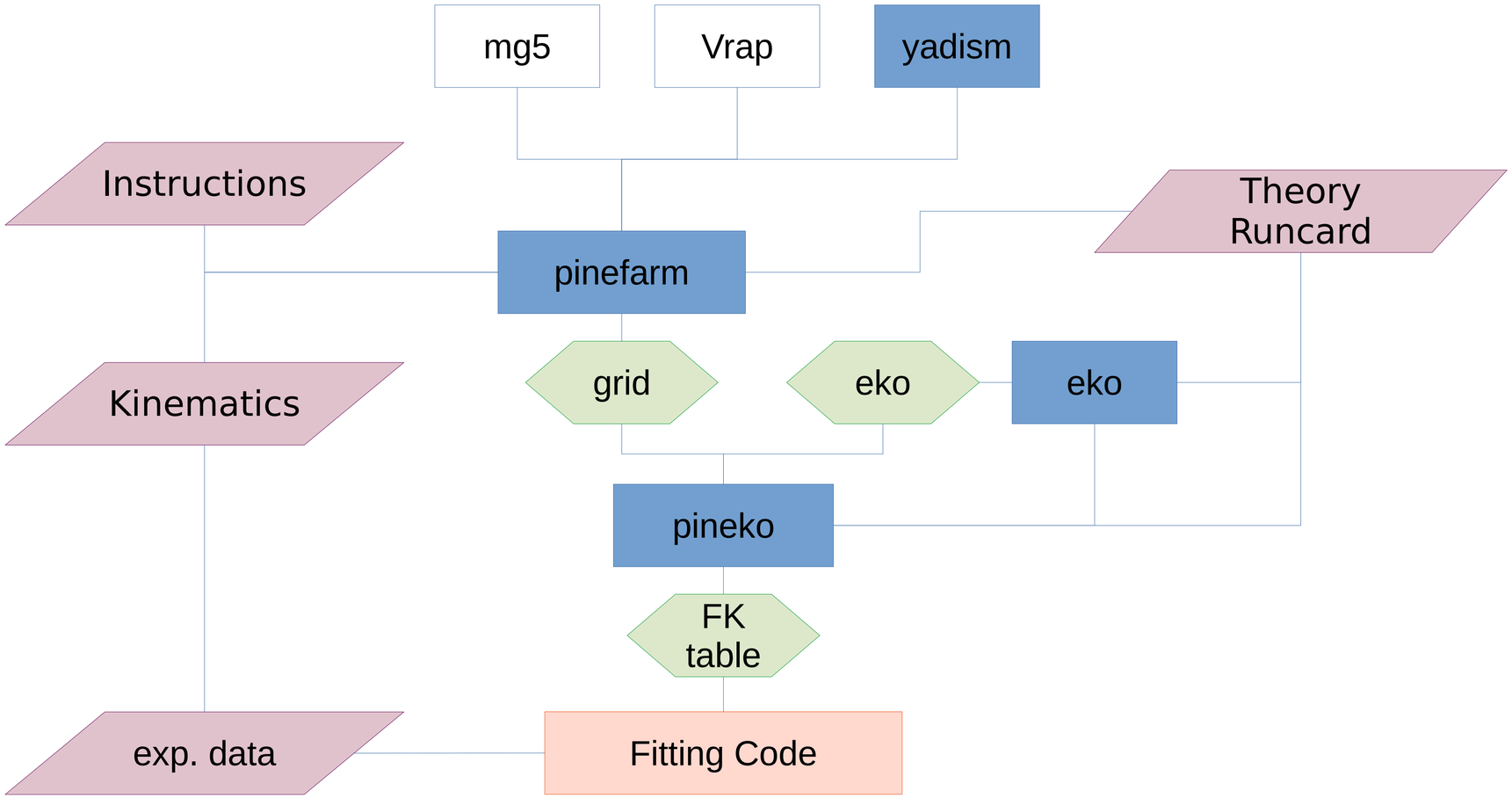
Most future plans for PDF require the generation of new interpolation grids, and generating them can take great amounts of time.

Highly desirable: some framework in which I can run the appropriate settings, let it run in the background and get the necessary interpolation grids maybe a month later.

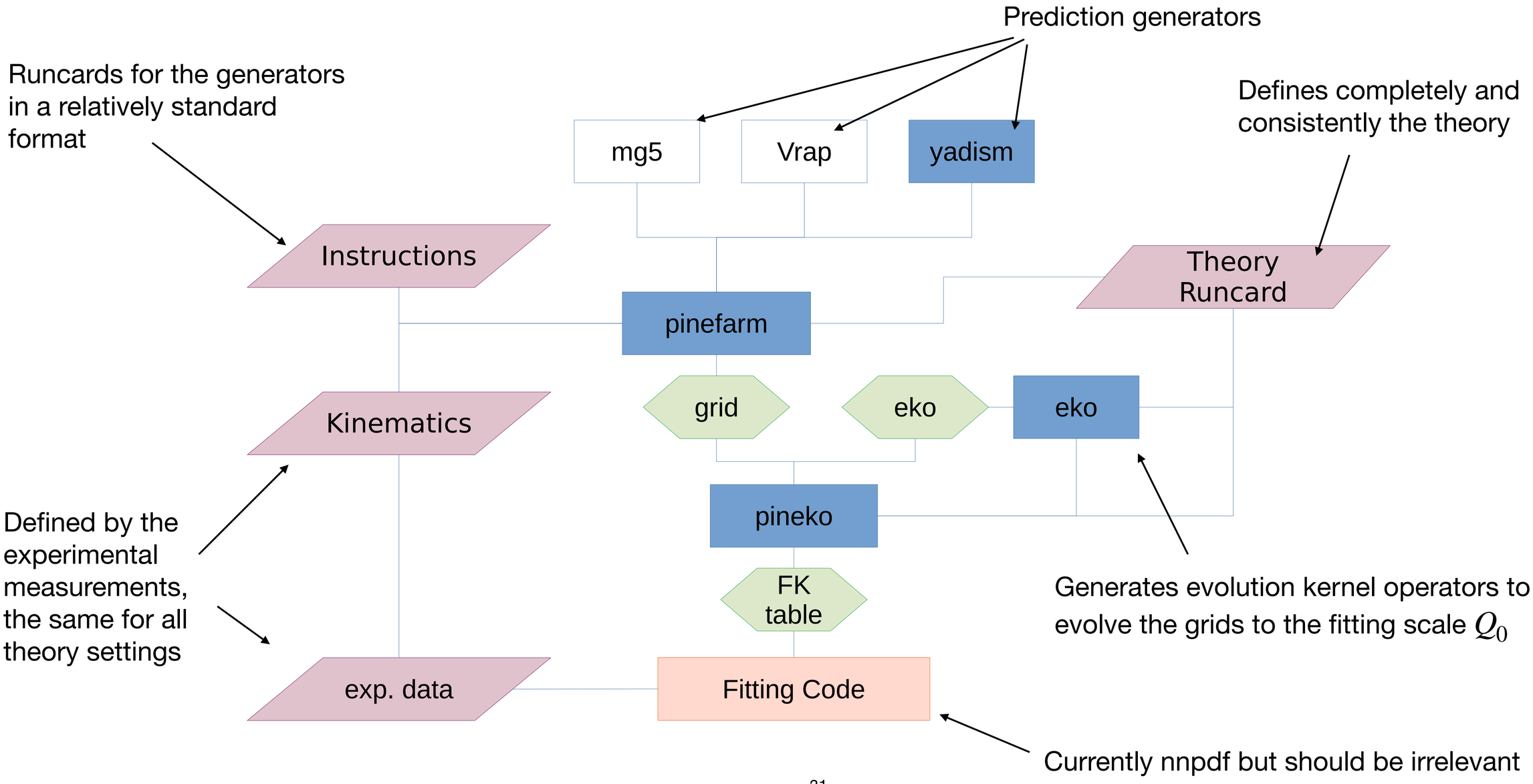




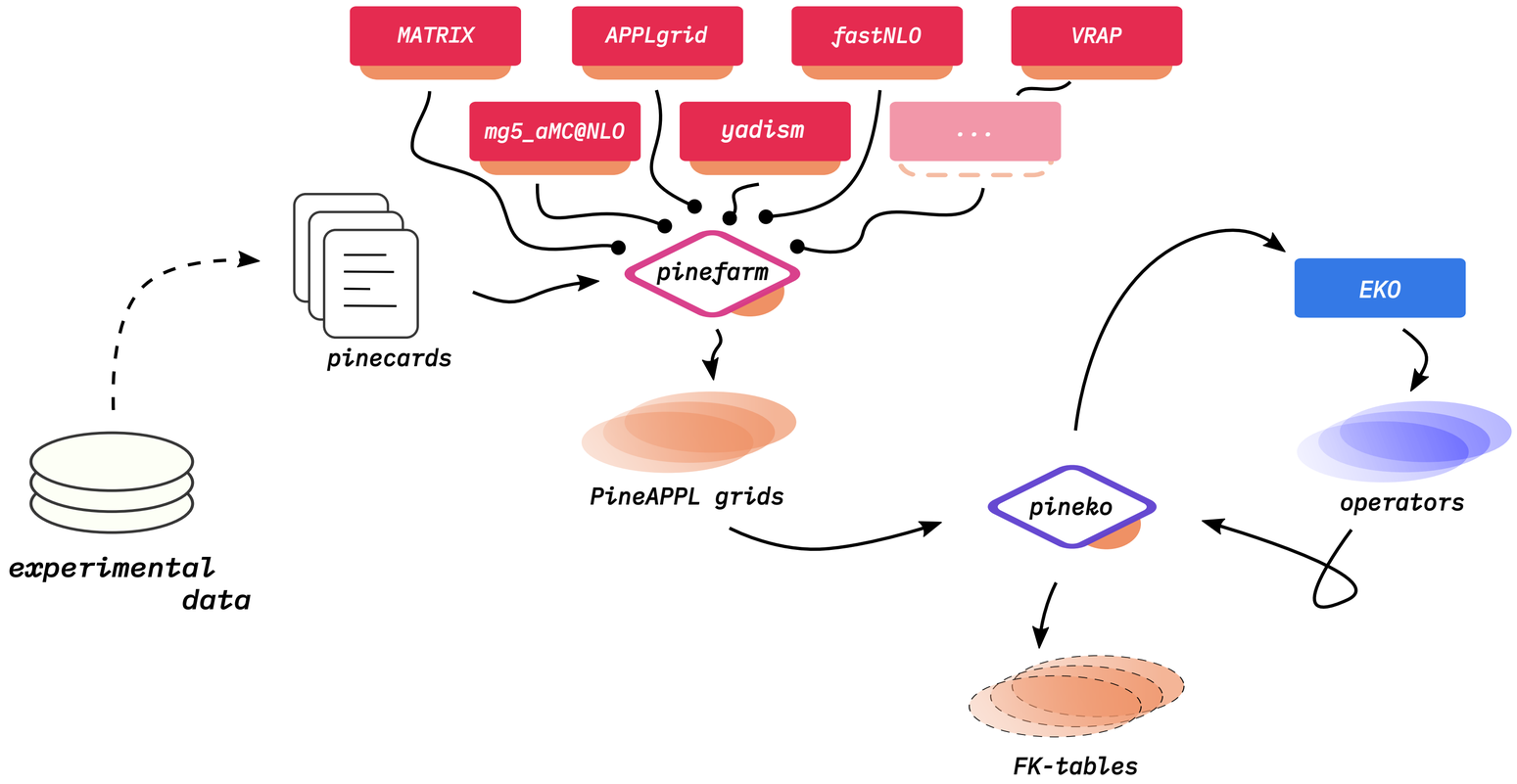
Theory pipeline



Theory pipeline



NNLO Grids from different providers



PineAPPL

General process, PDF-independent, grid storage

+

EKO

Evolution Kernel Operators

=

Pineko

PineAPPL + EKO

Pineappl grids

~$ pineappl convolute CMS_DY_7TEV_2D.pineappl.lz4 NNPDF40_nnlo_as_01180

b	Mll [GeV]		yll []		dsig/dyll [pb]	scale uncertainty [%]	
0	20	30	0	0.1	1.6056289e1	-19.35	17.18
1	20	30	0.1	0.2	1.6036726e1	-19.35	17.18
2	20	30	0.2	0.3	1.5990756e1	-19.33	17.16
3	20	30	0.3	0.4	1.5973514e1	-19.32	17.16
4	20	30	0.4	0.5	1.5921352e1	-19.30	17.13

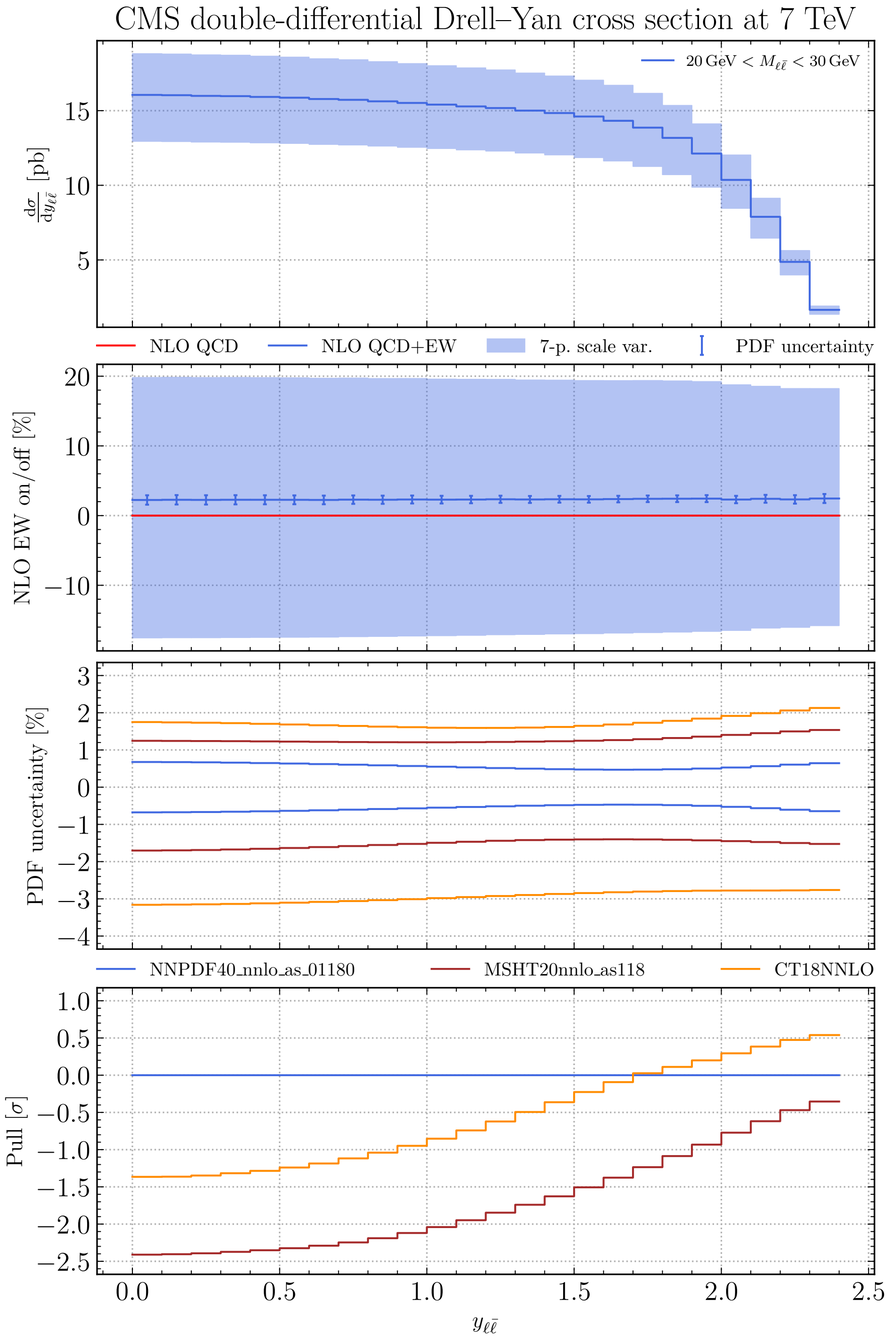
...

```python
import lhapdf
import pineappl

# Load a PDF
pdf = lhapdf.mkPDF("NNPDF40_nnlo_as_01180", 0)

# Load a pineappl grid or FKTable
grid = pineappl.grid.Grid.read("CMS_DY_7TEV_2D.pineappl.lz4")

# Convolute
result = grid.convolute_with_one(2212, pdf.xfxQ2,)
```



Try it out!

https://nnpdf.github.io/pineline