Artificial Intelligence to Map the Proton Structure

Juan Rojo

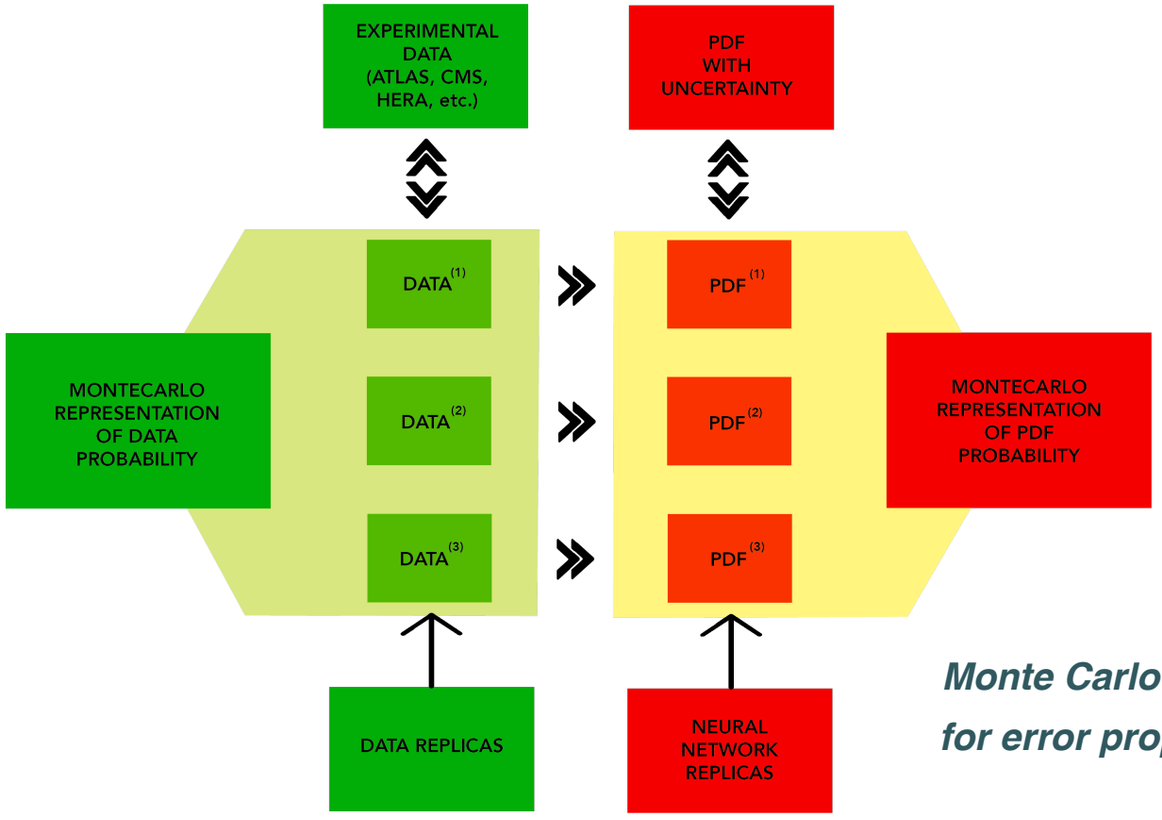
VU Amsterdam & Theory group, Nikhef

AI for Nuclear Physics Workshop

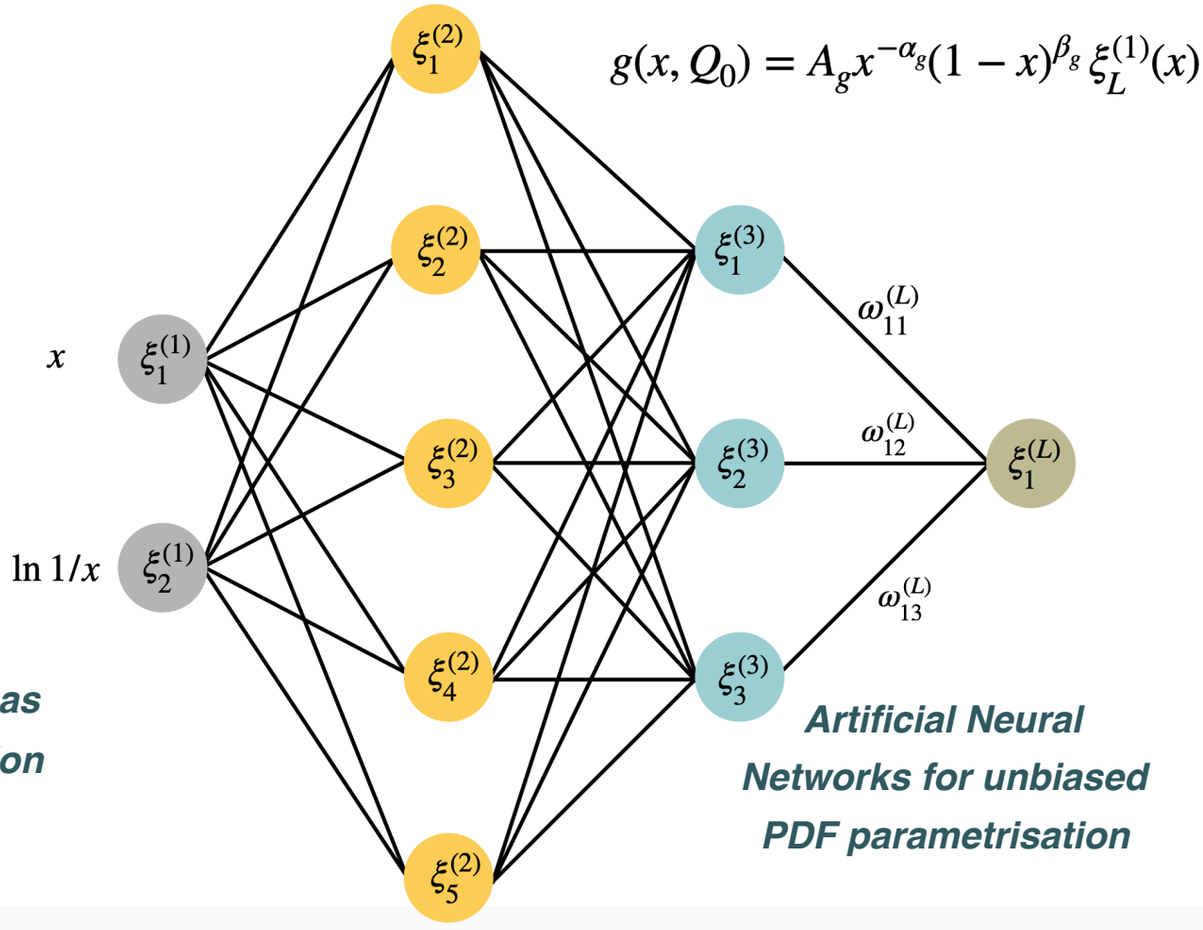
Bayesian Inference for Quantum Correlators workgroup

JLAB, Wednesday 4/3/2020

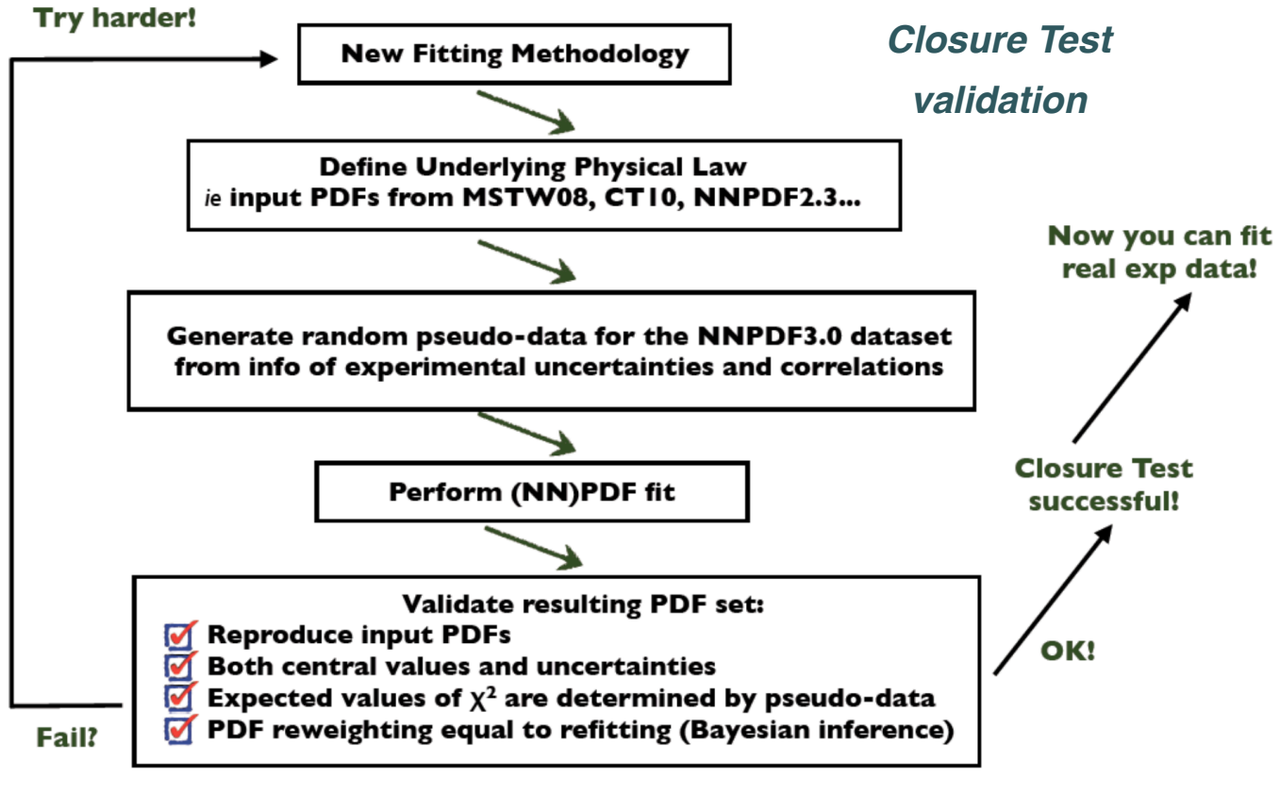
NNPDF in a nutshell



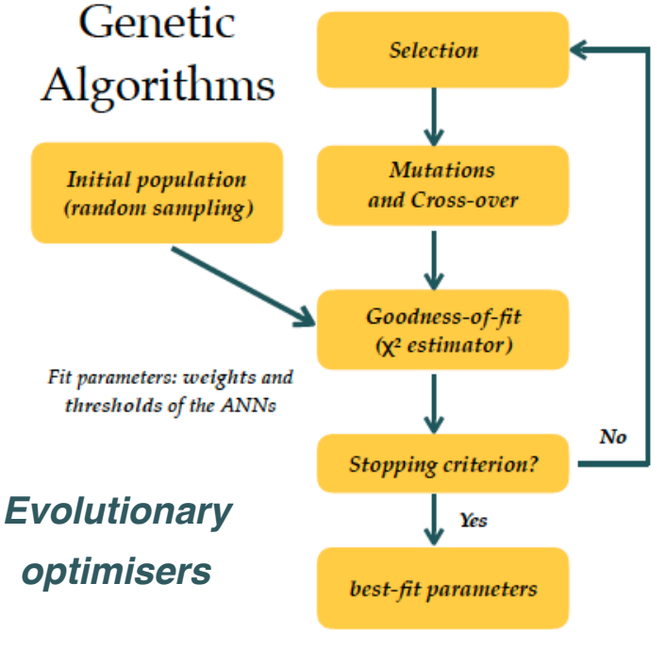
Monte Carlo replicas for error propagation



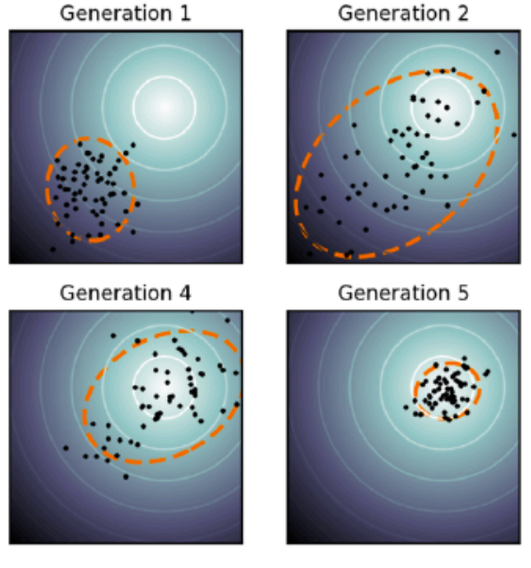
Artificial Neural Networks for unbiased PDF parametrisation



Closure Test validation



Evolutionary optimisers



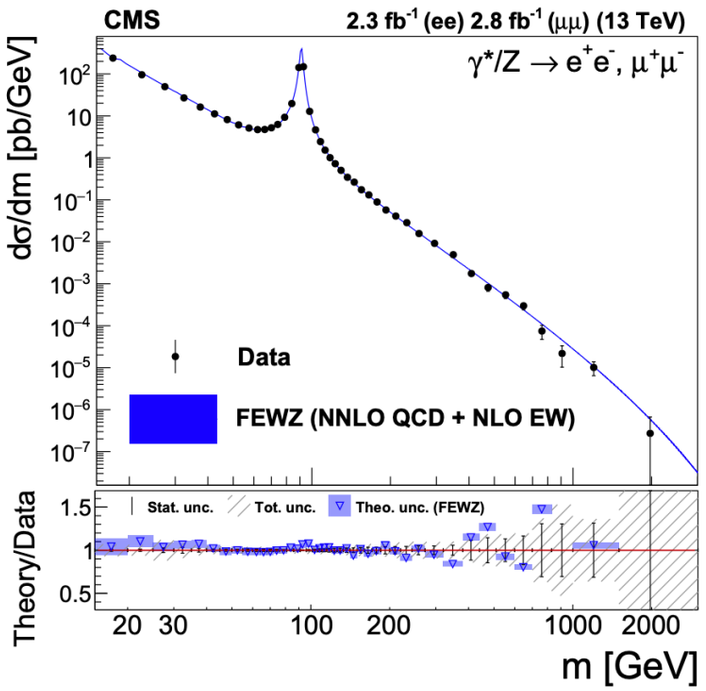
The NNPDF (proton) timeline

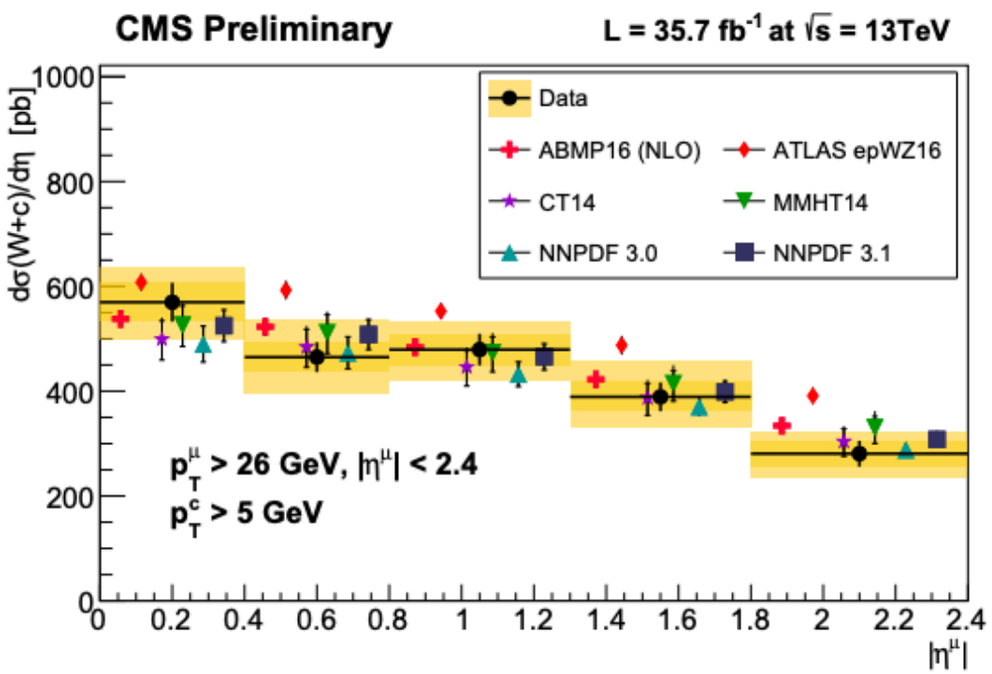
- **NNPDF2.3**: first PDF set with LHC data. LO set used in Monash 2013 Tune of Pythia8. Default (internal) PDF sets in MadGraph_aMC@NLO — 07/12
- **NNPDF2.3QED**: model-independent determination of the photon PDF — 04/13
- **NNPDF3.0**: large amounts of LHC data, methodological improvements closure-tested. Baseline PDF set in CMS MC event generation. — 10/14
- **NNPDF3.1**: most LHC Run I data included (several new processes: ttbar differential, high-mass Drell-Yan, Z pT spectra, …). NNLO fit markedly superior than NLO one — 06/17
- **NNPDF3.1QED**: LuxQED prescription for photon PDF — 12/17
- **NNPDF3.1smallx**: BFKL-improved PDFs for small-x physics (HERA, forward physics) — 10/17
- **NNPDF3.1TH**: first set of parton distributions with theory (MHO) uncertainties — 05/19
- **NNPDF4.0**: work on progress, expected release in summer 2020 — 2020

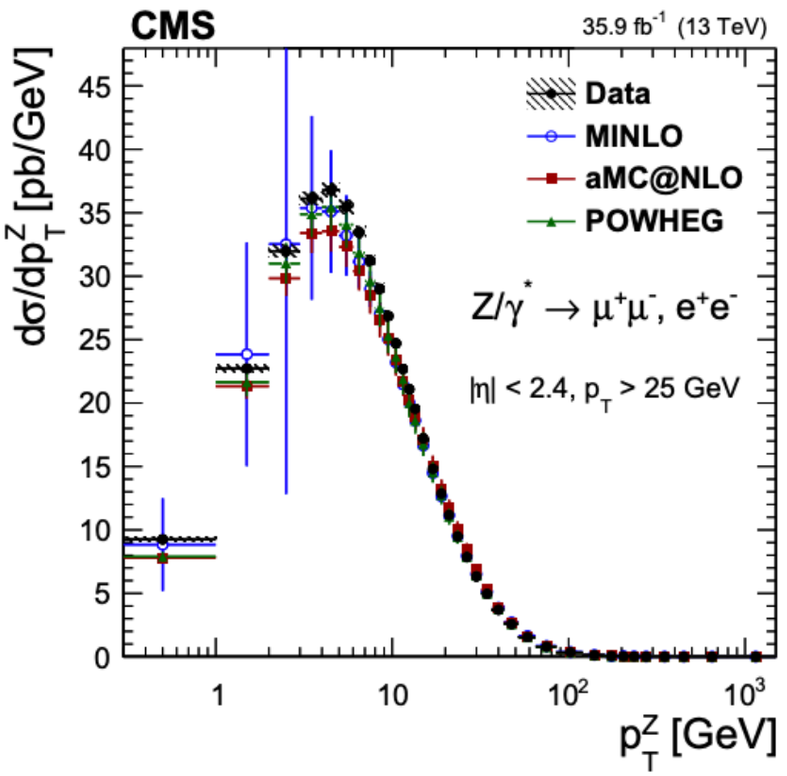
NNPDF4.0: new experimental data

Many new LHC Run I & II data included

- Differential Drell-Yan cross-sections at 13 TeV
- Z and W pT spectra at 8 and 13 TeV **New**
- W+c at 8 and 13 TeV (strangeness)
- Dijet production at 7 and 8 TeV **New**
- Top-quark pair production at 8 and 13 TeV
- Single-top cross-sections **New**



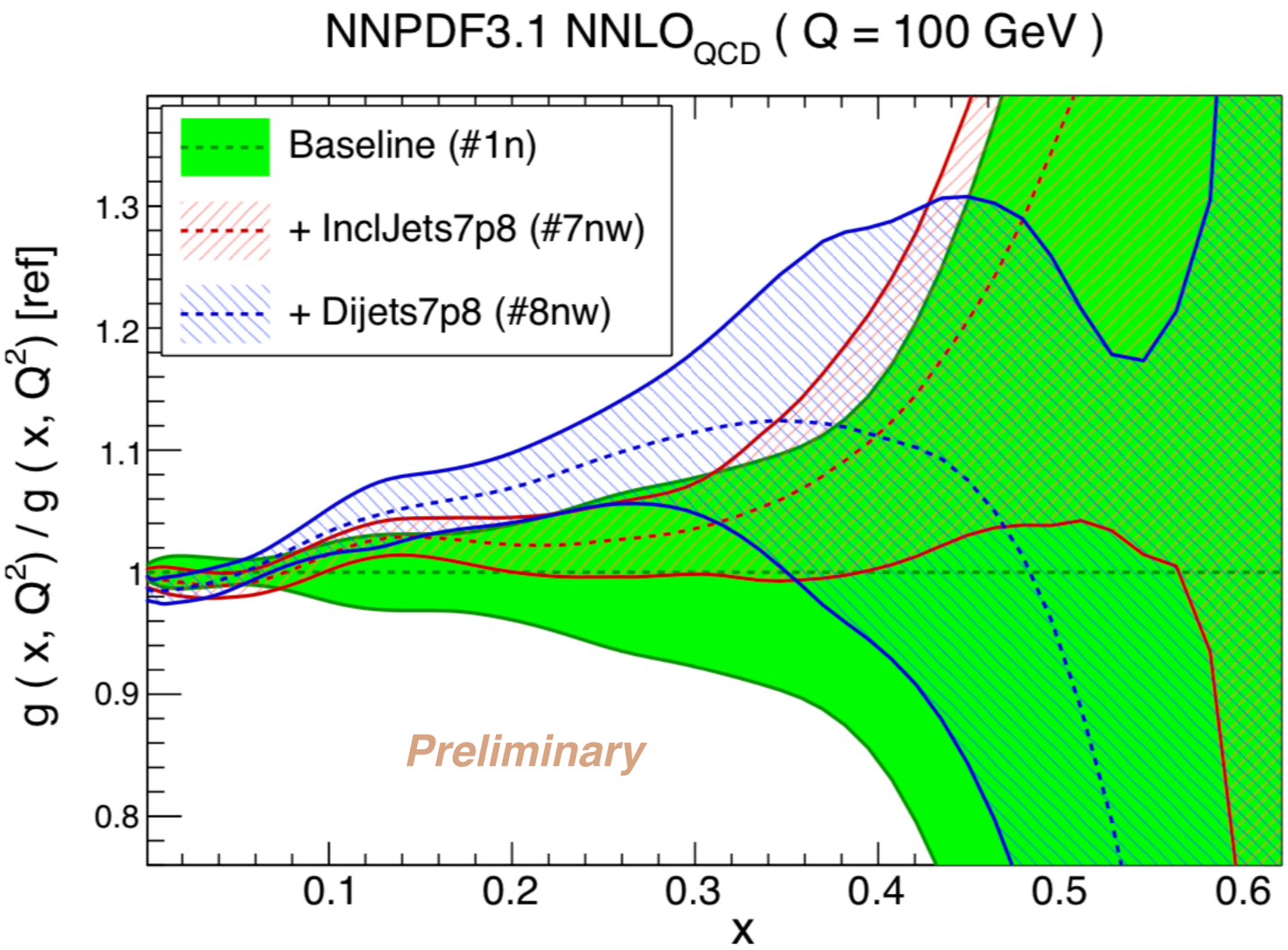




Dijet production

- Added several new measurements of **inclusive jet** and **dijet production** at 7 and 8 TeV, including the CMS 8 TeV 3D dijet distributions, using **NNLO QCD theory**

- Explore **sensitivity to gluon PDF** and robustness wrt experimental correlation models



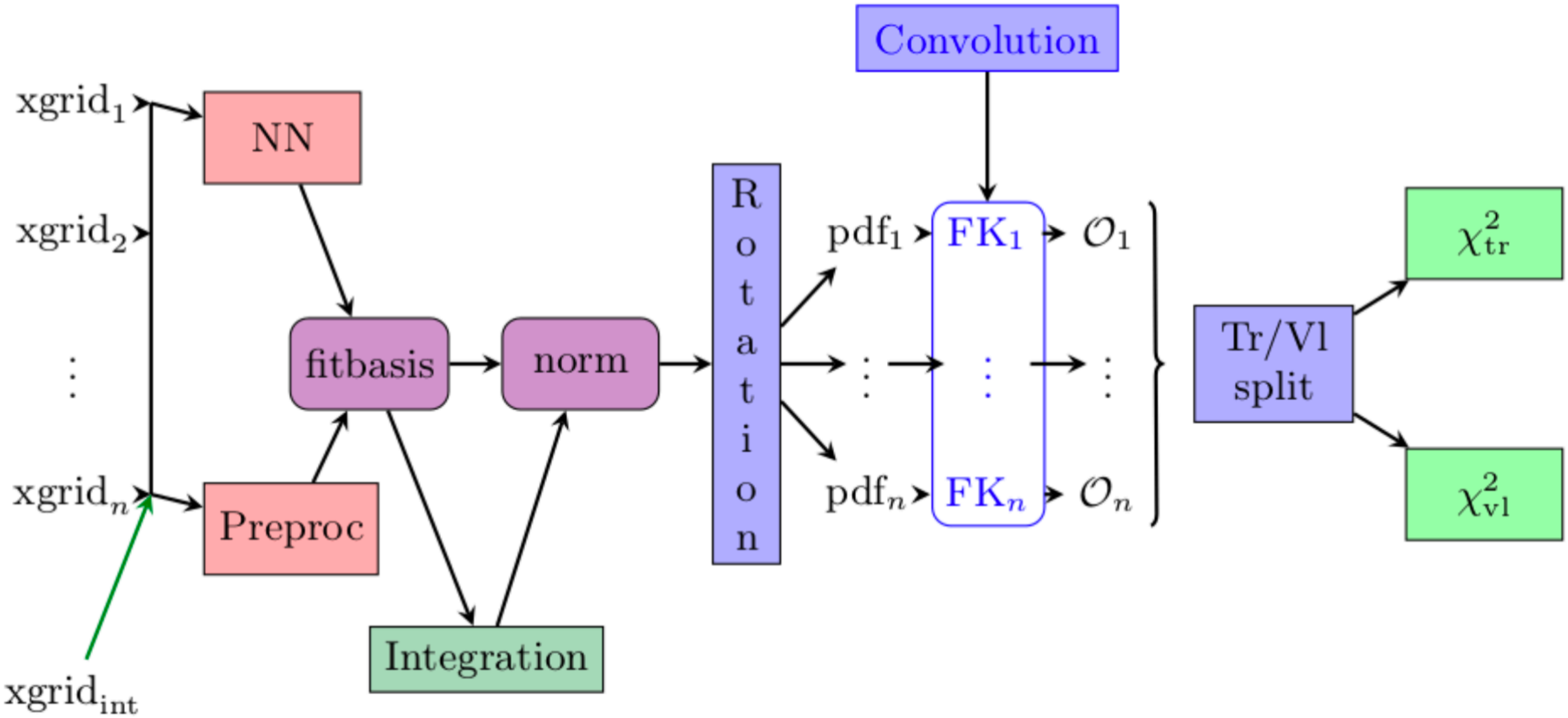
NNPDF + NNLOJET , arXiv:2003:aaaaa

Significant **pull on large-*x* gluon** & satisfactory description of dijet data within NNLO global PDF fit

	$n_{\rm dat}$	$\chi^2/n_{\rm dat}$
ATLAS dijets 7 TeV	90	1.77
CMS dijets 7 TeV	54	1.72
CMS dijets 8 TeV	122	1.82
dijets	266	1.78

first time in PDF fit!

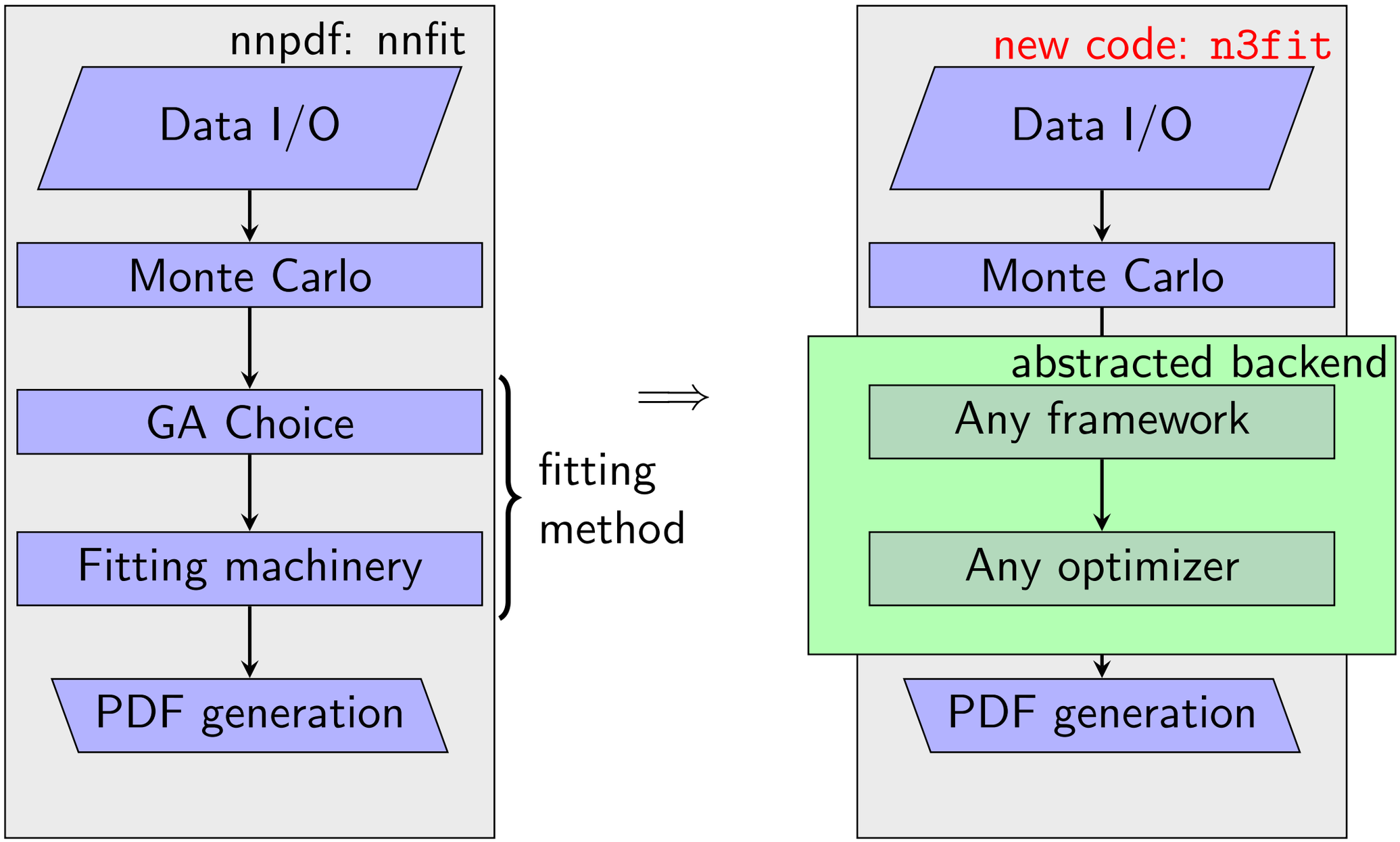
The n3fit project



Complete restructure of the NNPDF fitting framework: enhanced modularity that dramatically improves its flexibility, in particular to exploit **external ML libraries** *eg* Keras, TensorFlow, …

Cruz-Martinez, Carrazza, arXiv:1907.05075

The n3fit project



Complete restructure of the NNPDF fitting framework: enhanced modularity that dramatically improves its flexibility, in particular to exploit **external ML libraries** *eg* Keras, TensorFlow, …

Cruz-Martinez, Carrazza, arXiv:1907.05075

NNPDF4.0: methodology improvements

	NNPDF3.1	***NNPDF4.0***
Random numbers	main seed, closure filter seed	multi seed
Data management	libnnpdf	same as nnfit
Neural net	fixed architecture, per flavour	**single net, flexible architecture**
Preprocessing	random fixed	**fitted in range**
Integration	a posteriori per iteration	**buildin in the model**
Optimizer	genetic optimizer	**gradient descent** ***SGD, ADAM, RMSmom, …***
Stopping	lookback	**patience**
Positivity	penalty and threshold	**dynamic penalty, PDF must fulfill positi**
Postfit	4-sigma chi2 and arclenght	same as nnfit
Fine tuning	manual	**semi-automatic**
Model selection	closure test	closure test, **hyper optimization**

NNPDF4.0: methodology improvements

NNPDF3.1	***NNPDF4.0***
→ Genetic Algorithm optimizer	→ Gradient Descent optimization
→ One network per flavour	→ One network for all flavours
→ Sum rules imposed outside of optimization	→ Sum rules imposed during optimization
→ Preprocessing fixed per each of the replicas	→ Preprocessing fitted within replicas
→ C++ monolithic codebase	→ Python object oriented codebase
→ Fit parameters manually chosen (i.e., manual optimization of hyperparameter)	→ Fit parameters chosen automatically (hyperparameter scan)
→ In-house ML framework	→ Complete freedom for choosing the ML library: i.e., tensorflow.

complete freedom to change ML framework, optimisers, ...

Hyper optimisation

In most Machine Learning applications, the model has several parameters which are typically **adjusted by hand** (trial and error) rather than algorithmically:

- Network architecture: number of layers of neurons per layer, activation functions, …
- Choice of minimiser (which of the Gradient Descent variants?)
- Learning rate, momentum, memory, size of mini-batches, ….
- Regularisation parameters, stopping, dropout rate, patience, …

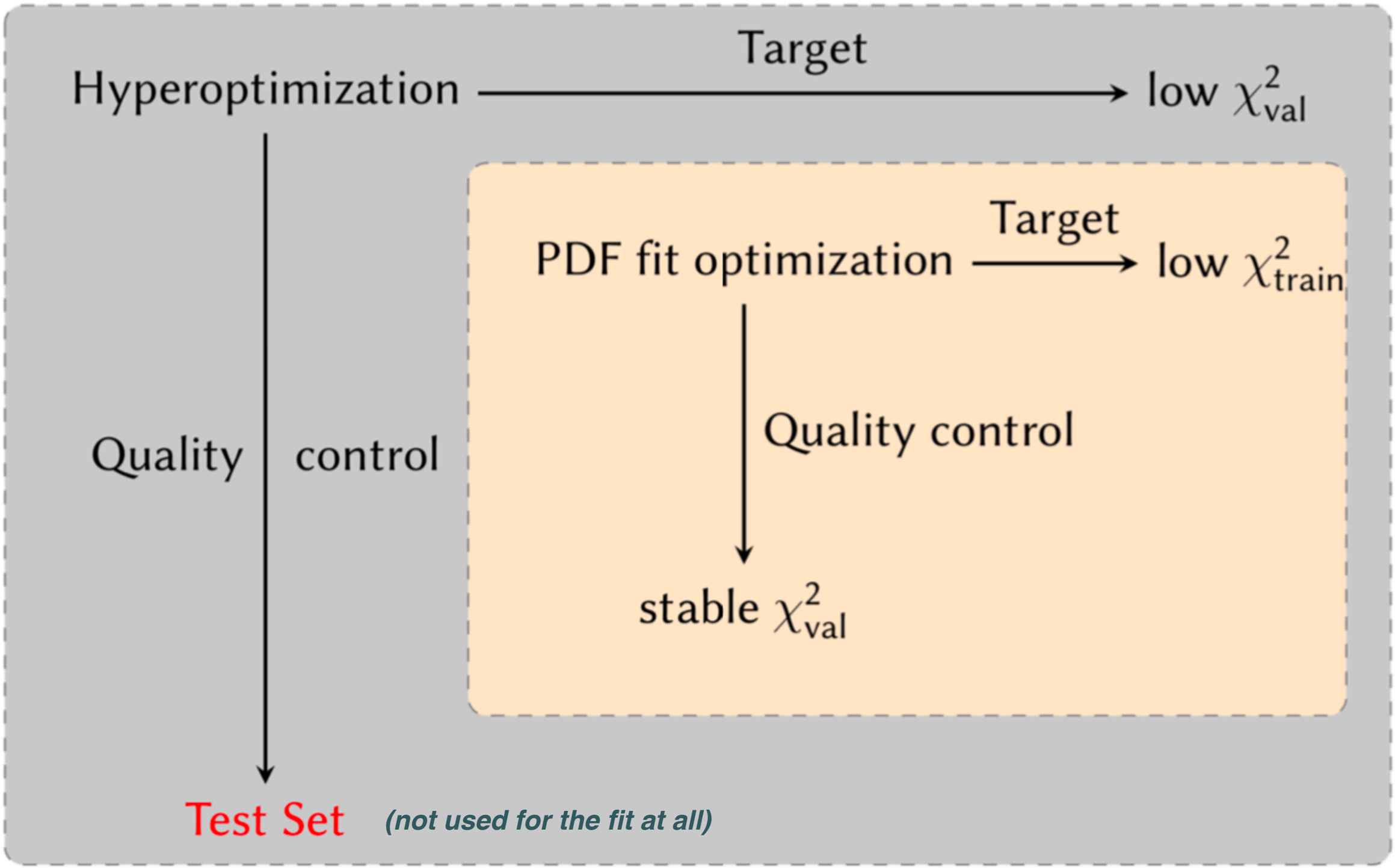
one can avoid the need of subjective choice by means of **an hyperoptimisation procedure**, where all model and training/stopping parameters are determined algorithmically

Such hyperoptimisation requires introducing a **reward function** to grade the model.
Note that this is different from the **cost function:** the latter is optimised separately model by model (e.g. for each NN architecture) while the former compares between all optimised models

e.g. cost function $C = E_{\mathrm{tr}}$

reward function $R = \frac{1}{2}\left(E_{\mathrm{val}} + E_{\mathrm{test}}\right)$

Hyper optimisation



$\chi^2_{\text{test set}}$

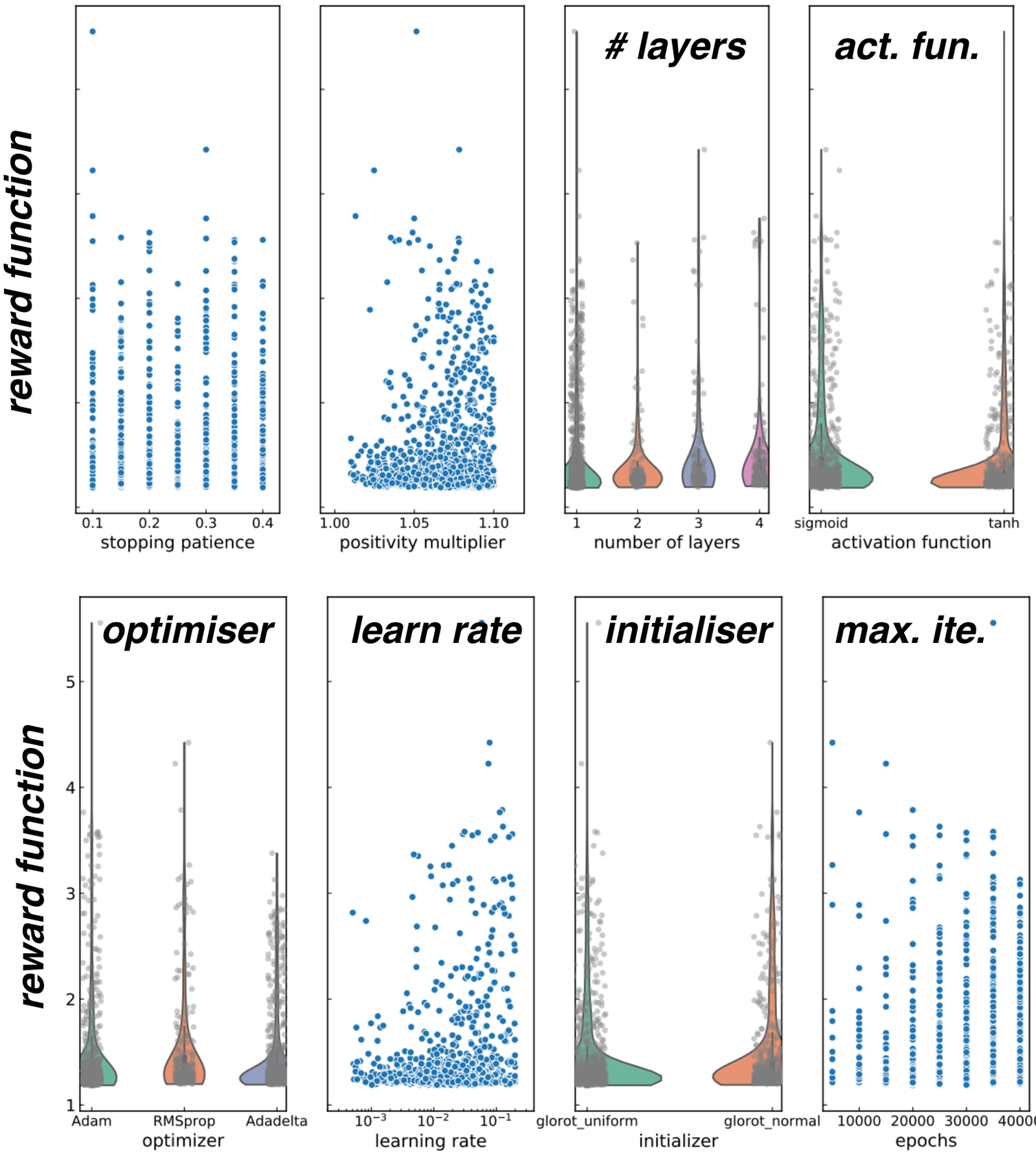
Hyper optimisation

- In a hyperparameter scan one can compare the performance of **hundreds or thousands** of parameter combinations

- Some choices are **discrete** (type of minimiser, # of layers) others are **continuous** (learning rate)

- One can also **visualise** which choices are more crucial and which ones less important

- The violin plots are the **KDE-reconstructed probability distributions** for the hyper-parameters

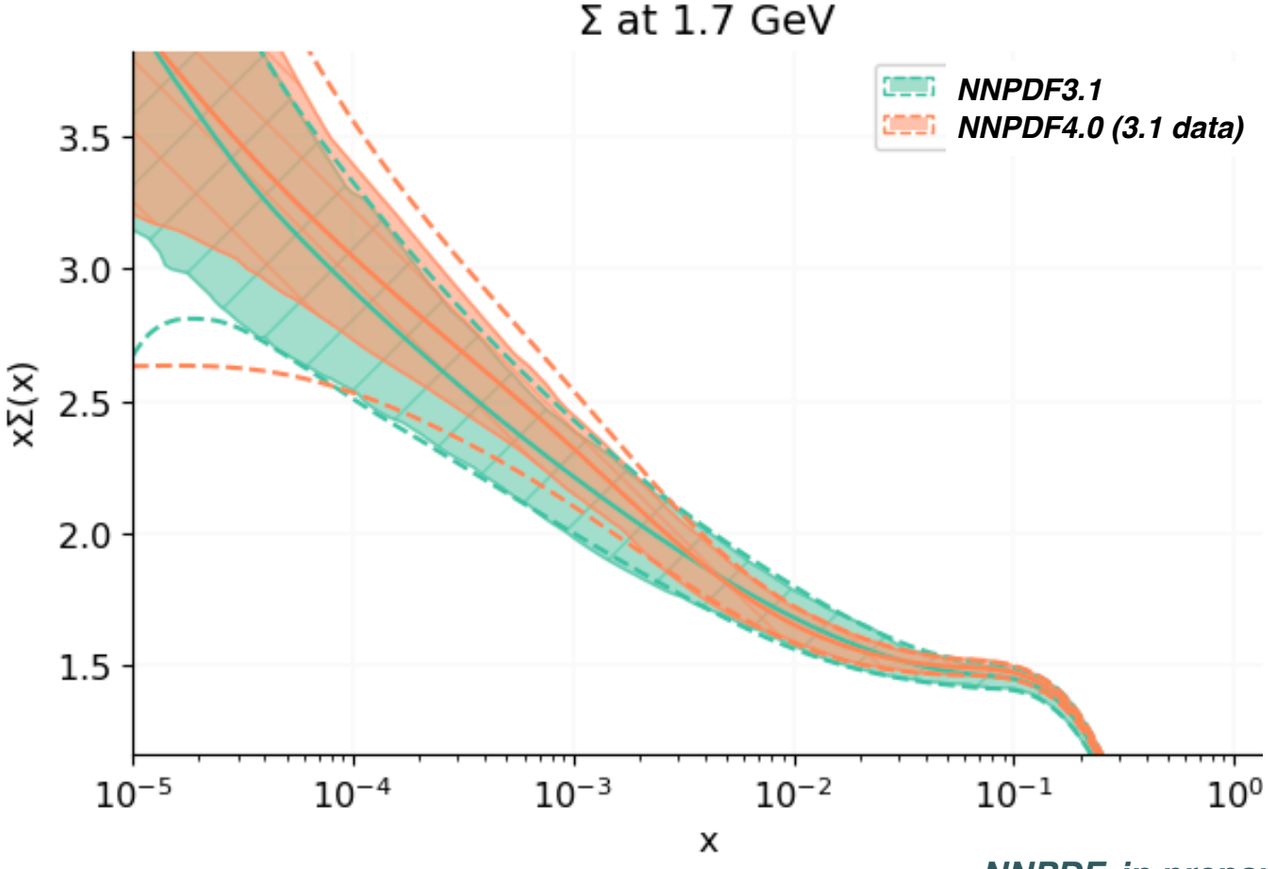


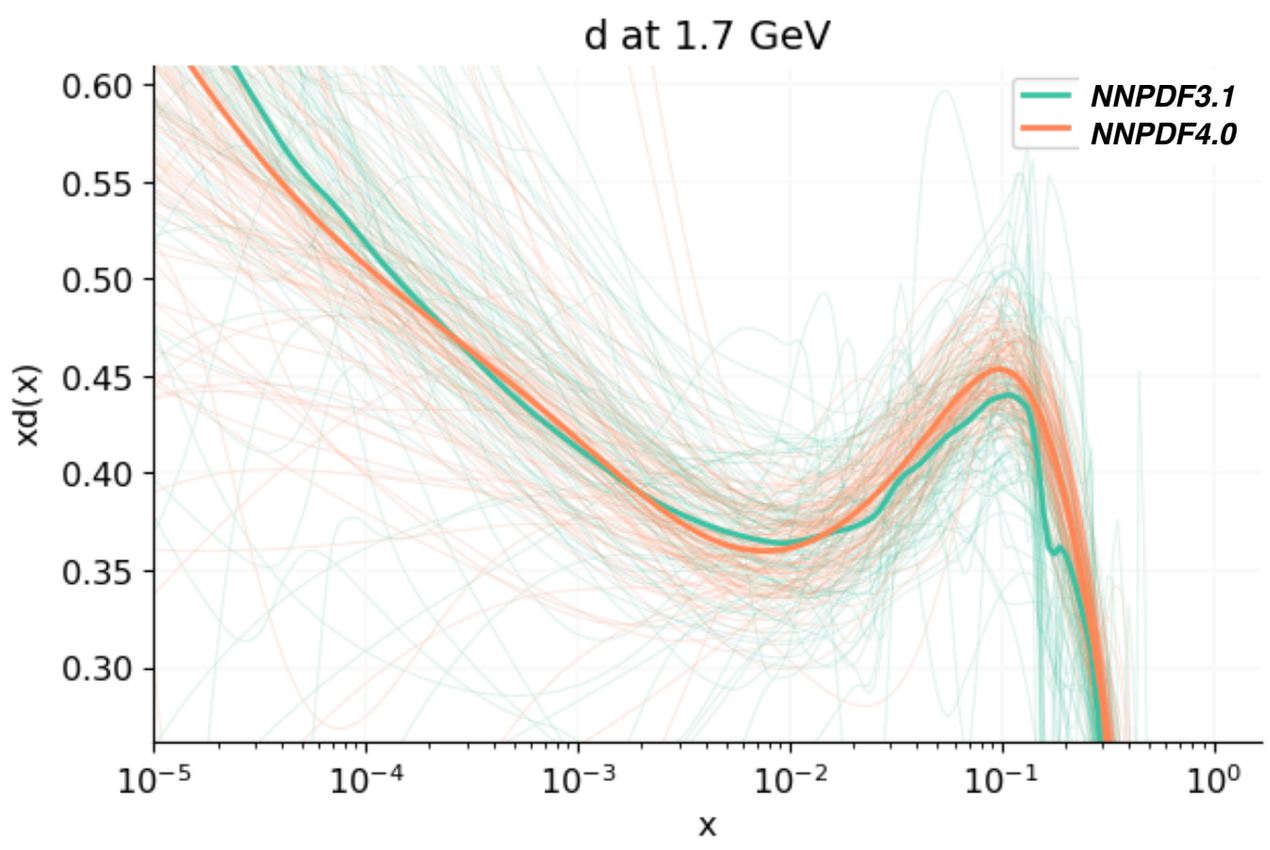
Juan Rojo

Towards NNPDF4.0

- Per-replica training time reduced by **factor O(30)** thanks to **SGD minimisers in TensorFlow**
- **Smoother** individual replicas, higher fraction of replicas satisfying quality requirements
- **Reduction in PDF uncertainties** in data region (for same dataset) thanks to improved methodology

	`n3fit`	NNPDF 3.1
χ^2	1.149	1.158
Avg time	70 minutes	35 hours
Memory	16 Gb	5 Gb
Good replicas	95%	70%



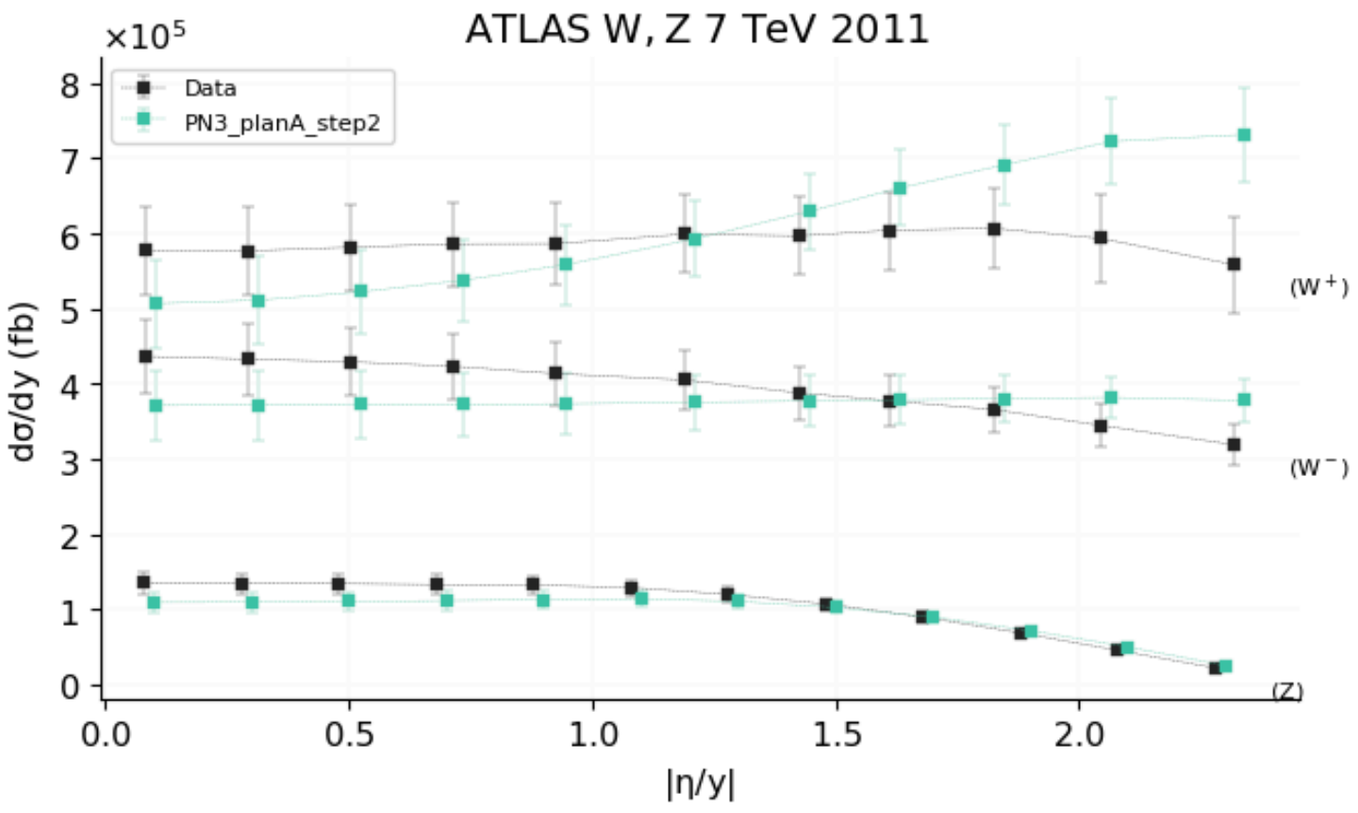


NNPDF, in preparation

AI & forecasting tests

- Crucial aspect of ML methods, beyond describing existing data, is to **generalise to future data**
- Train PDFs on **pre-HERA** and **pre-LHC data**, and then **forecast** for all data available now
- Include in this exercise PDF errors in the **χ^2 definition**

		n3fit pre-hera		nnfit pre-hera	
		ndata	χ^2/*ndata*	ndata	χ^2/*ndata*
HERACOMB	**Total**	1145	1.135	1145	1.089
ATLAS	**Total**	360	0.9744	360	0.9443
CMS	**Total**	409	0.9699	409	0.9200
LHCb	**Total**	85	1.195	85	1.008
Total	**Total**	2215	1.055	2215	1.013

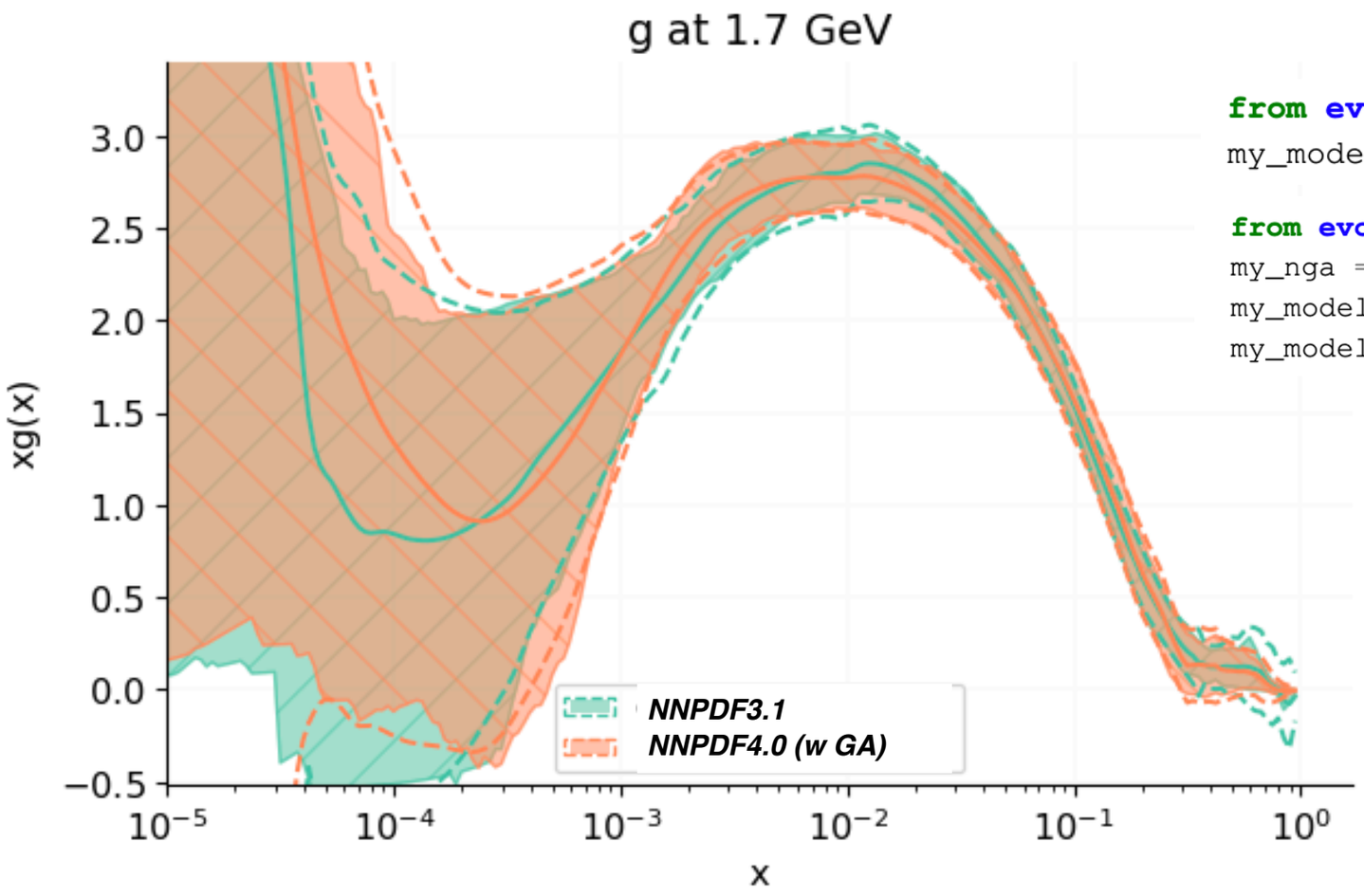


- Training PDFs on only old fixed-target DIS and DY datasets, the extrapolation to ``future'' data is fully satisfactory: **χ^2_{new} =1**
- Test succesful both with 3.1 and 4.0 methodologies: in both cases the PDF uncertainties are **faithfully estimated,** with 4.0 being **more accurate** than 3.1

NNPDF, in preparation

Evolutionary Keras

- NNPDF until 3.1 based on **evolutionary-type minimisers** (Genetic Algorithms, CMA-ES)
- From NNPDF4.0 (also in nuclear fits) **Gradient-Descent optimisers** adopted instead
- Since **Keras** library did not include evolutionary strategies, a new library, **evolutionary_keras,** was developed to run variants of NNPDF4.0 with evolutionary algorithms



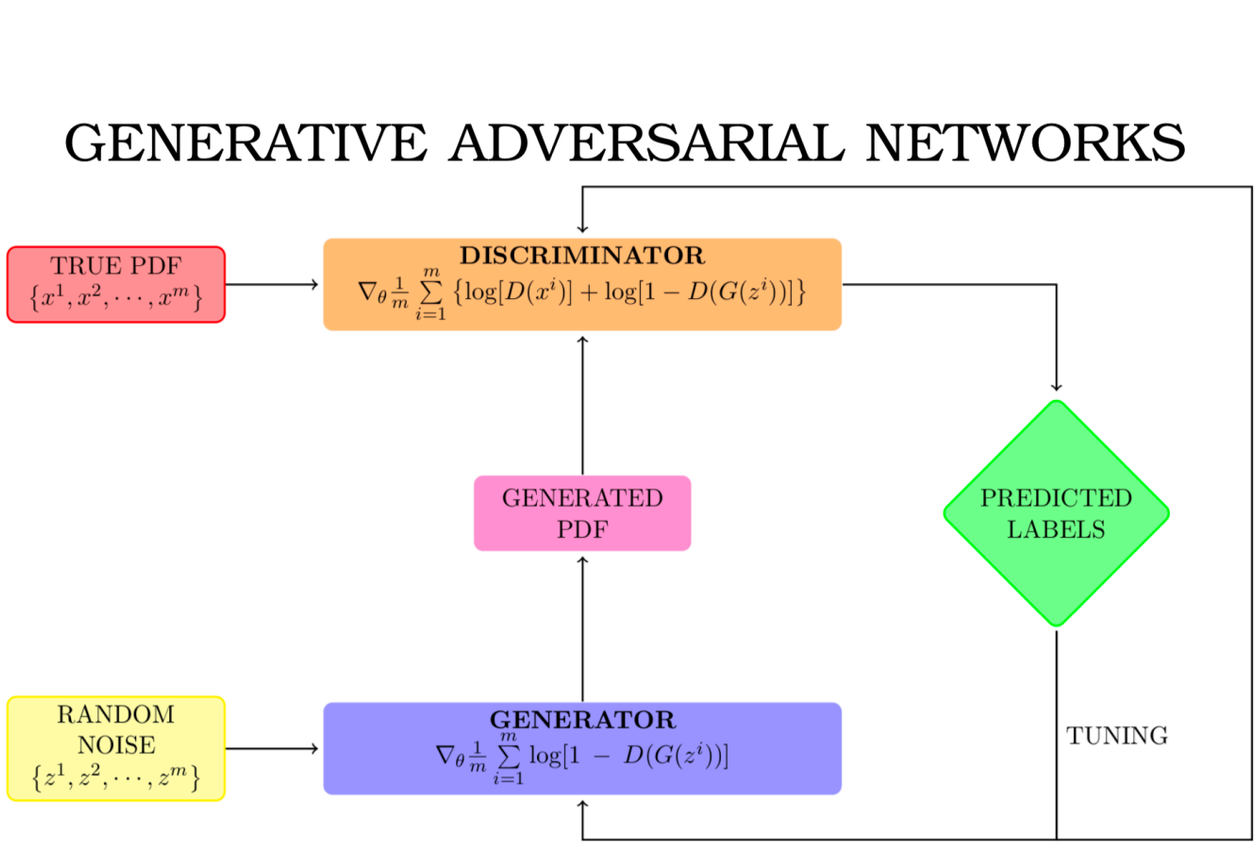
Cruz-Martinez, Carrazza, Stegeman, arXiv:2002.06587

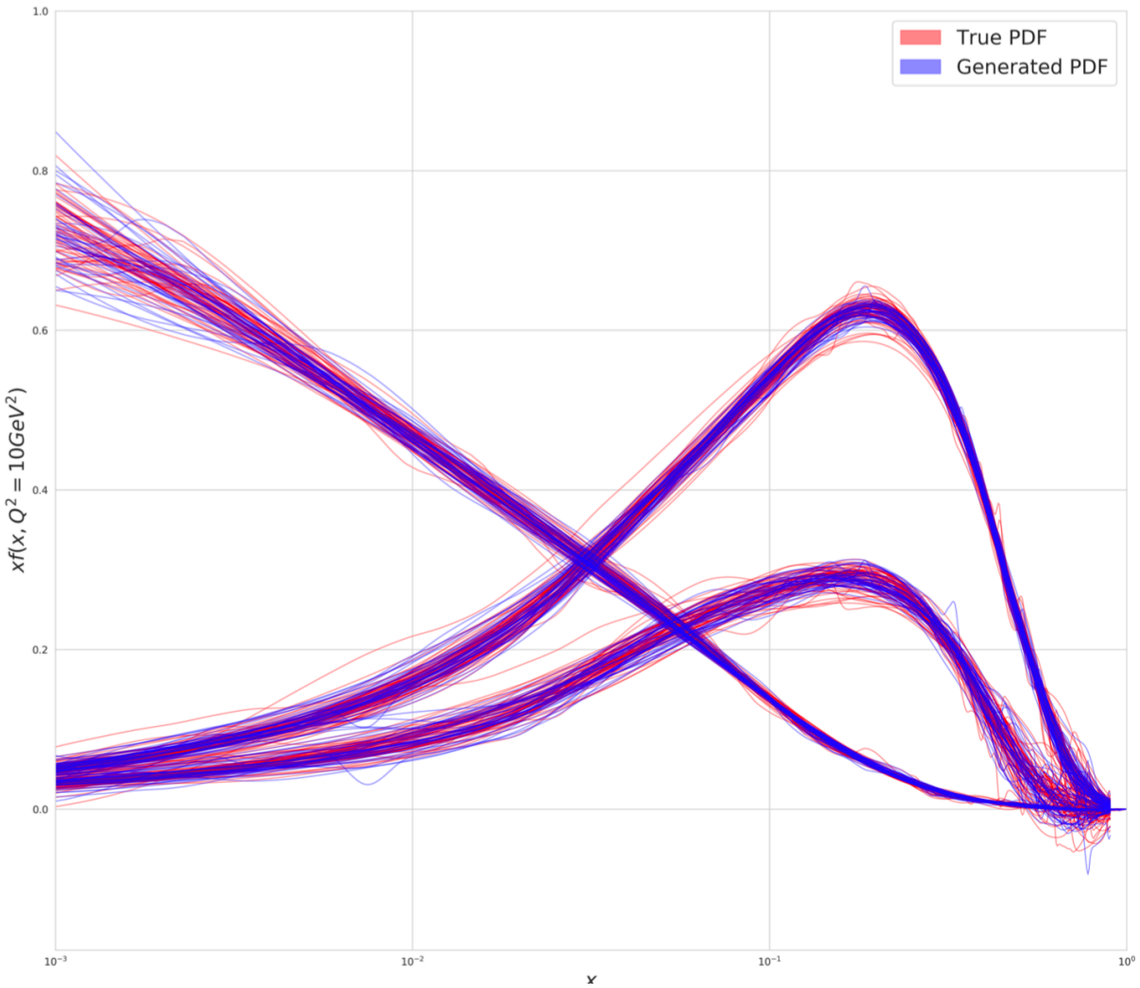
```
from evolutionary_keras.models import EvolModel
my_model = EvolModel(input_layer, output_layer)

from evolutionary_keras.optimizers import NGA
my_nga = NGA(population_size = 42, mutation_rate = 0.2)
my_model.compile(optimizer = my_nga, loss = 'mean_squared_error')
my_model.fit(x = input_data, y = output_data, epochs = 10)
```

- Demonstrates **extreme flexibility** of n3fit code: varying minimiser trivial
- NNPDF4.0 fits using **Evolutionary Keras library** in agreement with NNPDF3.1 for same input dataset

GANs for PDF fits

- Even with all the n3fit speedups, producing large samples of PDF replicas still time-consuming
- Solution: produce new PDF fit replicas using **Generative Adversarial Networks**
- While no additional information is being added, such method can be applied to many cases with a very large $\mathbf{N_{rep}}$ is beneficial, such as **Bayesian reweighting studies**





n3pdf group, in preparation

Summary and outlook

The accurate determination of the **quark and gluon structure of the proton** is an essential ingredient for **LHC phenomenology** and **beyond**

- Working towards a next major release, **NNPDF4.0**, with significant improvements from the theoretical, data, and methodological aspects

- Thanks to n3fit code, achieved c**omplete modularity of the NNPDF fitting framework**, allowing us to exploit **external ML libraries** such as TensorFlow & vary optimisers

- Methodology validated systematically on **closure test and historical (forecasting) test**: both 3.1 and 4.0 methodologies succesful, with 4.0 resulting in smaller PDF uncertainties

- The NNPDF methodology also used to produce **nuclear PDF fits,** using an independent fitting framework based on TensorFlow with SGD minimisation

Jake's talk tomorrow!