Studying the parton content of the proton with deep learning models

Juan M Cruz-Martinez[∗], **Stefano Carrazza, Roy Stegeman**

TIF Lab, Dipartimento di Fisica, Università degli Studi di Milano and INFN Sezione di Milano
Via Celoria 16, 20133, Milano, Italy
E-mail: juan.cruz@mi.infn.it, stefano.carraza@mi.infn.it,
roy.stegeman@mi.infn.it

Parton Distribution Functions (PDFs) model the parton content of the proton. Among the many collaborations which focus on PDF determination, NNPDF pioneered the use of Neural Networks to model the probability of finding partons (quarks and gluons) inside the proton with a given energy and momentum. In this proceedings we make use of state of the art techniques to modernize the NNPDF methodology and study different models and optimizers in order to improve the quality of the PDF: improving both the quality and efficiency of the fits. We also present the evolutionary_keras library, a Keras implementation of the Evolutionary Algorithms used by NNPDF.

[∗]Speaker.

1. Introduction

The determination of parton distribution functions (PDFs) is a particular topic which strongly relies on three dynamic and time dependent factors: new experimental data, higher order theoretical predictions and fitting methodology. There are two main tasks for PDF fitters such as the NNPDF collaboration [1, 2], MMHT [3] or CTEQ [4]. The first of which consists in maintaining and organizing a workflow which incrementally implements new features proposed by the respective experimental and theoretical communities. The second task of a PDF fitter is the investigation of new numerical and efficient approaches to PDF fitting methodology. While the former relies almost exclusively on external groups and communities, the latter is under full control of the PDF fitting collaborations, and in most cases it reflects the differences between them.

In their most generic form, PDFs are functions which depend on an energy scale (usually labelled Q) and an energy fraction (x) which represents the fraction of the energy of the proton carried by the parton. Detailed reviews on the topic can be found in Refs. [5–7].

Historically PDFs have been fitted via fixed functional forms which were made more complex as the amount of available data increased [8–11]. The work presented in this proceedings is based on the framework developed by the NNPDF collaboration which introduces for the first time in the PDF determination context the employment of unbiased functional forms in the form of neural networks to model the parton content of the proton [12, 13].

2. Neural Networks PDF: NNPDF

The target goal in PDF determination is the optimization of the χ^2 of the fit. This is a problem well suited for machine learning techniques, as the functional form is unknown while the input and output are well defined. The details for the current implementation of the NNPDF methodology can be found in [2, 14]. An exhaustive overview is beyond the scope of this proceedings, in what follows we outline some of the most relevant aspects of the NNPDF methodology used in the last main releases (3.0 and 3.1) as a baseline for our current discussion.

In the NNPDF methodology fits are parameterized at a reference scale Q_0 and expressed in terms of a neural network whose outputs correspond to a set of basis functions. The neural network consists of a fixed-size feedforward multi-layer perceptron. The input node (x) is split by the first layer on the pair $(x, \log(x))$. The hidden layers use the sigmoid activation function while the output node is linear. The functional form of this model is:

$$f_i(x, Q_0) = A_i x^{-\alpha_i} (1-x)^{\beta_i} \mathrm{NN}_i(x), \tag{2.1}$$

where NN_i is the output of the neural network corresponding to a given flavour i, usually expressed in terms of the PDF evolution basis $\{g, \Sigma, V, V_3, V_8, T_3, T_8, c^+\}$. A_i is an overall normalization constant which enforces sum rules and $x^{-\alpha_i}(1-x)^{\beta_i}$ is a preprocessing factor which controls the PDF behaviour at small and large x. Both the α_i and β_i parameters are randomly selected within a defined range for each replica at the beginning of the fit and kept constant thereafter.

Unlike usual regression problems, where during the optimization procedure the model is compared directly to the training input data, in PDF fits the theoretical predictions are constructed through the convolution operation per data point between a fastkernel table (FK) [15, 16] and the
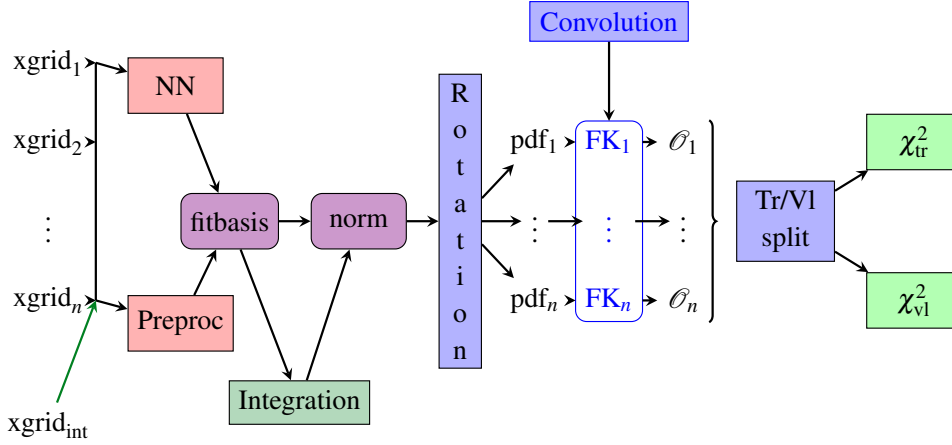
Figure 1: Diagrammatic view of the NNPDF fitting model as implemented in the `n3fit` code. Each different block is set as a different Keras layer. The red squared blocks correspond to blocks with potentially fittable parameters while other blocks represent parameters or transformations which are fixed during the fit.

PDF model. For DIS-like processes the convolution is performed once, while for hadron collision-like processes PDFs are convoluted twice (once per hadron).

The optimization procedure consists in minimizing the loss function

$$\text{Loss} = \chi^2 = \sum_{i,j}^{N_{\text{dat}}} (D - P)_i \sigma_{ij}^{-1} (D - P)_j, \tag{2.2}$$

where D_i is the i-th artificial data point from the training set, P_i is the convolution product between the fastkernel tables for point i and the PDF model, and σ_{ij} is the covariance matrix between data points i and j (following the t_0 prescription defined in appendix of Ref. [17]). The covariance matrix can include both experimental and theoretical components as presented in Ref. [18]. As a summary, in Fig. 1 a schematic view of the code is represented.

Up to the current NNPDF release (3.1 at the time of writing of these proceedings, Ref. [2]) only genetic algorithms had been used as the optimizer for the weights and biases of the neural network. The optimization procedure makes use of the Nodal Genetic Optimizer defined in Ref. [14] and can be summarized as follows: the weights of the neural network are initialized using a random gaussian distribution and checking that sum rules are satisfied. From that first network 80 mutant copies are created based on a given mutation probability and update size. The training procedure is fixed to $3 \cdot 10^4$ iterations and the best mutant is determined using a look-back algorithm which stores the best weights for the lowest validation loss value.

The experimental data used in the PDF fit is preprocessed according to a cross-validation strategy based on randomly splitting the data for each replica into a training set and a validation set. The optimization is then performed on the training set while the validation loss is monitored to reduce overlearning.

The codebase driving this methodology has up to now been a fully in-house C++ program. Such an in-house codebase allowed for a more fine grained control of the details of the methodology

but, as a consequence, the implementation of new features and studies were hindered by the extra amount of effort for both developers and maintainers.

3. A new methodology: `n3fit`

The advances in the field of machine learning have worked their way into the field of PDFs in the form of a new methodology which inherits directly from NNPDF and has been named `n3fit`[1]. This new methodology has been presented for the first time in [19]. The codebase for `n3fit` has been written in Python, with Keras [20] and Tensorflow [21] as the engine behind the training and construction of the neural network. Other engines can be implemented as most of the codebase is library agnostic.

This new methodology has successfully introduced several changes to the NNPDF methodology and allowed for an improvement of the research results and workflow [19, 22]. Here we focus on the substitution of the genetic algorithm for an optimization based on gradient descent and on a comparison with previous results.

3.1 Optimization algorithm: from genetic algorithms to gradient descent

The default minimizer of the C++ version of the NNPDF fitting code (`nnfit`) is the aforementioned Nodal Genetic Algorithm (or NGA). The NGA has been used, for instance, in NNPDF3.0 [14] and NNPDF3.1 [2]. Other evolutionary optimizers have also been implemented in `nnfit`, such as Covariance Matrix Adaptation Evolution Strategy (CMA) which was used in the release of hadronic fragmentation functions [23, 24].

Newer releases by the NNPDF collaboration explore the usage of deterministic minimizers. The nuclear PDF set nNNPDF1.0 [25] make use of the ADAM gradient descent optimizer [26] which is closely related to RMSProp [27] and Adagrad [28]. The full NNPDF fitting mechanism has also been upgraded to support gradient descent optimizers and it has been implemented in `n3fit` [19]. The n3fit code supports the usage of the Adagrad, RMSProp and ADAM minimizers as well as any optimizer compatible with the Keras library.

We do a thorough exploration of algorithms previously used by the NNPDF collaboration and gradient descent based algorithms. In Fig. 2a we observe that the CMA algorithm produces smoother results than the NGA and thus convergence is improved: the number of outliers is reduced and postfit selections are less aggressive. We find, however, longer running times, which makes it unsuitable for our purposes.

For this particular problem we have access to the gradient of the model, as such Gradient Descend (GD) based algorithms can be expected to outperform evolutionary strategies. In Fig. 2b we present a comparison between a GD based algorithm and the NGA used in previous releases of NNPDF. We find that using SGD results in features similar to those we observe when using CMA, that is, smoother replicas and better convergence. At the same time, the performance is also improved: computing times are greatly reduced. As a consequence, new lines of investigation appear which would not have been possible before, such as hyperparameter scans [19, 30].

---

[1]We follow the well known trend in High Energy Physics which relates the quality of a calculation to the multiplier of the letter `n` in their name.
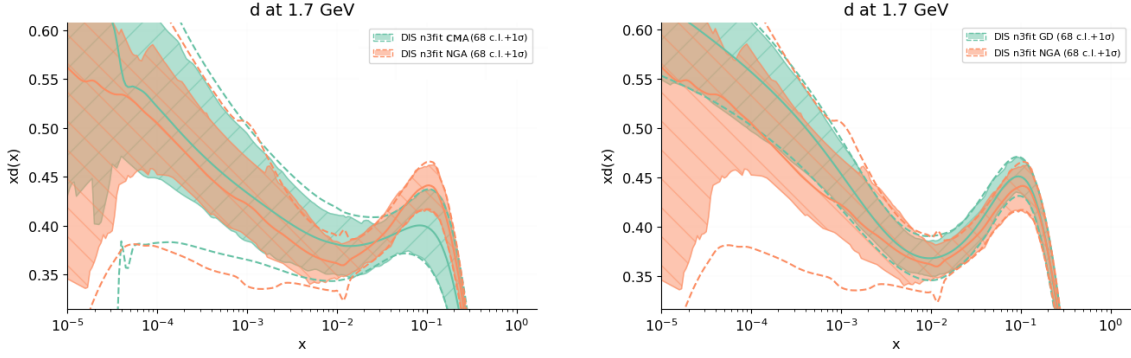
(a) Comparison of the NGA algorithm (orange) and the CMA algorithm (green). The CMA produce a smoother curve at the cost of significantly longer running times (up to a factor of two in our tests).

(b) Comparison of the NGA algorithm (orange) and a gradient descent algorithm (green) for the d quark inside the proton. The better stability of the deterministic optimizer produces a smoother error band and central value all across the considered ranges.

Figure 2: Comparison of different algorithms in `n3fit` in DIS only fit. Note that all of them produce compatible result within their error bands in the considered ranges. Plots are generated with `reportengine` [29].

The gradient descent implementation can comfortably be run in a consumer-grade desktop computer while the NGA instead has been run on a cluster with the aid of `pyHepGrid` [31].

The Keras library does not include evolutionary strategies which has motivated us to develop a new library: `evolutionary_keras` [32]. This library implements evolutionary strategies used by NNPDF retaining full compatibility with any Keras model. The fit shown in Fig. 2b based on the DIS dataset from NNPDF3.1 and labelled GA use the NGA algorithm described in [14] as implemented in `evolutionary_keras` and it is compared against the Adadelta [33] optimizer implemented in the standard Keras library. The engine behind both fits is `n3fit`.

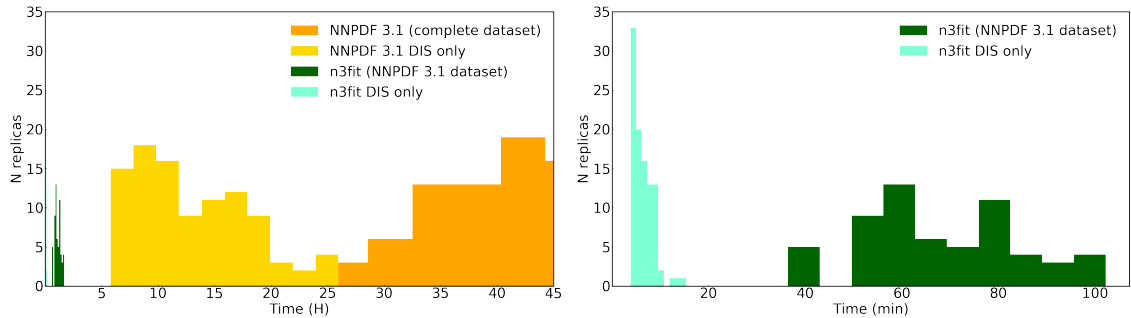## 3.2 Comparison with old methodology

Figure 3: Distribution of computing times for a full DIS and global fit for the NNPDF 3.1 methodology (yellow) and `n3fit` (green). While for the old methodology the computing time for one replica could take days, the new code achieves similar fit goodness in a scale of hours.

From a technical perspective, one of the most relevant advantages introduces by `n3fit` is the reduction of the computational time required to obtain PDF sets based on the NNPDF3.1 dataset [2]. These improvements are in part due to a combination of the new minimizer based on gradient descent and the usage of multi-threading CPU calculations when executing the TensorFlow graph model. In Fig. 3 we show the distribution of computing time in a per-replica basis (each fit is composed of up to 100 replicas). Note that while a global NNPDF fit with the 3.1 methodology could take up to two days per replica, the same NNPDF fit with the new methodology only takes a few hours. Further attempts to reduce these figures are currently underway with the implementation of custom TensorFlow operators and the usage of Graphical Processing Units (GPUs) for the FK convolution [22].
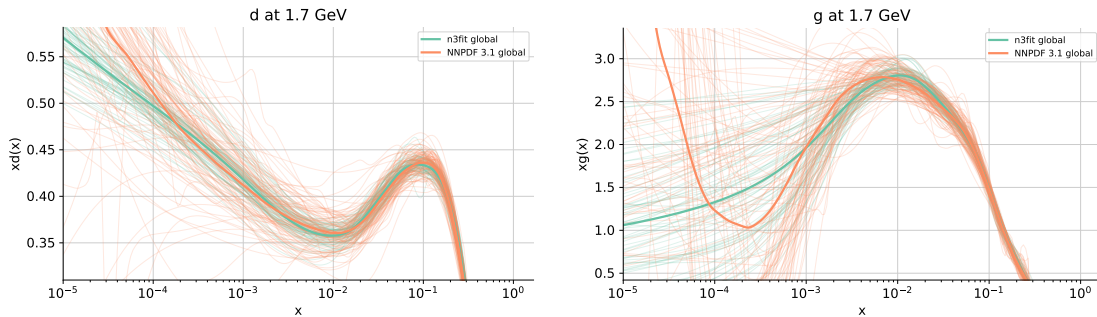

Figure 4: Comparison between the results obtained using the old methodology (orange) and the new `n3fit` methodology (green) for two of the partons found inside the proton, the d quark (left) and the gluon (right). Note that the results are compatible across the entirety of the studied range with overlapping central values in the region where most of the data is concentrated.

The number of PDF replicas that are necessary for a full fit is also reduced due to the better replica-by-replica stability that gradient descent methods produce for this problem with respect to genetic algorithms. This stability is also apparent when we plot the PDF replicas themselves as seen in Fig. 4 where in green we show the results of the new methodology while in orange those of the methodology used up to NNPDF 3.1. Nevertheless, the central value only shows a different behaviour in the extrapolation error (where the lack of data and the size of the errorbar make the difference irrelevant). In the data region `n3fit` shows much smoother lines.

4. A modern Genetic Algorithm library: `evolutionary_keras`

As part of the current research a new library has been developed which aims to implement the evolutionary strategies used by the NNPDF collaboration. The library has been named `evolutionary_keras` [32] and the code is publicly available in the N3PDF repository at https://github.com/N3PDF/evolutionary_keras. It can easily be installed with the `pip` package manager.

```
~$ pip install evolutionary-keras
```

It is also included in the `conda-forge` repository so it can be installed for Anaconda projects.

```
~$ conda install -c conda-forge evolutionary_keras
```

Documentation for the library is found at https://evolutionary-keras.readthedocs.io where all available optimizer are listed and some examples of usage can be found.

The library consists of a module which extends the Keras Model class and implements optimizers based on Genetic Algorithms, currently includes the aforementioned NGA and CMA evolutionary strategies. Compatibility with Keras is retained and it can be used as any other Keras optimizer.

In order to use the capabilities of `evolutionary_keras` in a project, it is necessary to use the model-classes provided. These classes inherit from the Keras model class and transparently defer to them whenever a Gradient Descent algorithm is used.

Below we present one example in which a Keras model is created with some input and output tensors previously defined. Those familiar with Keras will immediately see the similarities.

```python
1    from evolutionary_keras.models import EvolModel
2    my_model = EvolModel(input_layer, output_layer)
3    from evolutionary_keras.optimizers import NGA
4    my_nga = NGA(population_size = 42, mutation_rate = 0.2)
5    my_model.compile(optimizer = my_nga, loss = 'mean_squared_error')
6    my_model.fit(x = input_data, y = output_data, epochs = 10)
```

In summary, any Keras `Model` must become an `EvolModel`, but it can be instantiated with the same syntax as can be seen in line 2. From that point onwards the usage is unchanged with respect to a normal Keras project. In line 4 the genetic optimizer NGA is instantiated with some initial parameters and then compiled into the model together with one of the default loss functions included in Keras. For an up to date list of included algorithms, please consult the documentation.

Acknowledgments

References

[1] NNPDF collaboration, *Parton Distributions with Theory Uncertainties: General Formalism and First Phenomenological Studies*, *Eur. Phys. J.* **C79** (2019) 931 [1906.10698].

[2] NNPDF collaboration, *Parton distributions from high-precision collider data*, *Eur. Phys. J.* **C77** (2017) 663 [1706.00428].

[3] L. A. Harland-Lang, A. D. Martin, P. Motylinski and R. S. Thorne, *Parton distributions in the LHC era: MMHT 2014 PDFs*, *Eur. Phys. J.* **C75** (2015) 204 [1412.3989].

[4] T.-J. Hou et al., *New CTEQ Global Analysis with High Precision Data from the LHC*, 1908.11238.

[5] S. Forte and G. Watt, *Progress in the Determination of the Partonic Structure of the Proton*, *Ann. Rev. Nucl. Part. Sci.* **63** (2013) 291 [1301.6754].

[6] J. Rojo, *The Partonic Content of Nucleons and Nuclei*, `1910.03408`.

[7] J. J. Ethier and E. R. Nocera, *Parton Distributions in Nucleons and Nuclei*, `2001.07722`.

[8] D. W. Duke and J. F. Owens, *Q**2 Dependent Parametrizations of Parton Distribution Functions*, *Phys. Rev.* **D30** (1984) 49.

[9] E. Eichten, I. Hinchliffe, K. D. Lane and C. Quigg, *Super Collider Physics*, *Rev. Mod. Phys.* **56** (1984) 579.

[10] A. D. Martin, R. G. Roberts and W. J. Stirling, *Structure Function Analysis and psi, Jet, W, Z Production: Pinning Down the Gluon*, *Phys. Rev.* **D37** (1988) 1161.

[11] M. Diemoz, F. Ferroni, E. Longo and G. Martinelli, *Parton Densities from Deep Inelastic Scattering to Hadronic Processes at Super Collider Energies*, *Z. Phys.* **C39** (1988) 21.

[12] S. Forte, L. Garrido, J. I. Latorre and A. Piccione, *Neural network parametrization of deep inelastic structure functions*, *JHEP* **05** (2002) 062 [`hep-ph/0204232`].

[13] NNPDF collaboration, *A Determination of parton distributions with faithful uncertainty estimation*, *Nucl. Phys.* **B809** (2009) 1 [`0808.1231`].

[14] NNPDF collaboration, *Parton distributions for the LHC Run II*, *JHEP* **04** (2015) 040 [`1410.8849`].

[15] R. D. Ball, L. Del Debbio, S. Forte, A. Guffanti, J. I. Latorre, J. Rojo et al., *A first unbiased global NLO determination of parton distributions and their uncertainties*, *Nucl. Phys.* **B838** (2010) 136 [`1002.4407`].

[16] V. Bertone, S. Carrazza and N. P. Hartland, *APFELgrid: a high performance tool for parton density determinations*, *Comput. Phys. Commun.* **212** (2017) 205 [`1605.02070`].

[17] R. D. Ball et al., *Parton Distribution Benchmarking with LHC Data*, *JHEP* **04** (2013) 125 [`1211.5142`].

[18] R. Abdul Khalek et al., *A First Determination of Parton Distributions with Theoretical Uncertainties*, `1905.04311`.

[19] S. Carrazza and J. Cruz-Martinez, *Towards a new generation of parton densities with deep learning models*, *Eur. Phys. J.* **C79** (2019) 676 [`1907.05075`].

[20] F. Chollet et al., "Keras." `https://keras.io`, 2015.

[21] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro et al., *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015.

[22] S. Carrazza, J. Cruz-Martinez, J. Urtasun-Elizari and E. Villa, *Towards hardware acceleration for parton densities estimation*, in *International Conference on the Structure and the Interactions of the Photon (Photon 2019) Frascati, Italy, June 3-7, 2019*, 2019, `1909.10547`.

[23] NNPDF collaboration, *A determination of the fragmentation functions of pions, kaons, and protons with faithful uncertainties*, *Eur. Phys. J.* **C77** (2017) 516 [`1706.07049`].

[24] S. Carrazza and N. P. Hartland, *Minimisation strategies for the determination of parton density functions*, *J. Phys. Conf. Ser.* **1085** (2018) 052007 [`1711.09991`].

[25] NNPDF collaboration, *Nuclear parton distributions from lepton-nucleus scattering and the impact of an electron-ion collider*, *Eur. Phys. J.* **C79** (2019) 471 [`1904.00018`].

[26] D. Kingma and J. Ba, *Adam: A method for stochastic optimization*, *International Conference on Learning Representations* (2014) .

[27] T. Tieleman and G. Hinton, "Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude." COURSERA: Neural Networks for Machine Learning, 2012.

[28] J. C. Duchi, E. Hazan and Y. Singer, *Adaptive subgradient methods for online learning and stochastic optimization.*, *J. Mach. Learn. Res.* **12** (2011) 2121.

[29] Z. Kassabov, *Reportengine: A framework for declarative data analysis*, Feb., 2019. 10.5281/zenodo.2571601.

[30] J. Bergstra, D. Yamins and D. D. Cox, *Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures*, in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pp. I–115–I–123, JMLR.org, 2013, http://dl.acm.org/citation.cfm?id=3042817.3042832.

[31] J. Cruz-Martinez, D. Walker and J. Whitehead, *pyhepgrid: distributed computing made easy*, May, 2019. 10.5281/zenodo.3233861.

[32] J. Cruz-Martinez, R. Stegeman and S. Carrazza, *evolutionary_keras: a genetic algorithm library*, Jan., 2020. 10.5281/zenodo.3630339.

[33] M. D. Zeiler, *ADADELTA: an adaptive learning rate method*, *CoRR* **abs/1212.5701** (2012) [1212.5701].

PoS(AISIS2019)008