POLAR BACKGROUND PREDICTION

DOCUMENTATION: ACADEMIC REPORT

Stéphane Liem NGUYEN

Nicolas PRODUIT

Computer Science Department

DPNC: Nuclear and Corpuscular Physics Department

University of Geneva

October 3, 2023

Contents

List of Figures

List of Tables

Table 1: Non-Exhaustive List of Notation

		Single letter variables
i	:	index of an example
j	:	index of an input dimension
b	:	index of an output dimension or energy bin
k	:	selection threshold to select examples of interest
μ	:	mean
σ	:	standard deviation but could be the modified standard deviation obtained via our Modified Gaussian Fit[1]
		Other variables
$\mathbf{x}_i \in \mathbb{R}^{18}$	:	*un-normalized* example i
$\tilde{\mathbf{x}}_i \in \mathbb{R}^{18}$	:	*normalized*[2] example i (what is used as input to the model)
$\mathbf{y}_i \in \mathbb{R}^6$	:	target vector of example i
$\hat{\mathbf{y}}_i = f_\theta(\tilde{\mathbf{x}}_i) \in \mathbb{R}^6$	:	predicted target vector of example i using our model
$\mathbf{e}_i(b) \in \mathbb{R}$	:	error rate of example i for the photon rate in energy bin b
$\mathbf{r}_i \in \mathbb{R}^6$	:	residual=target-prediction of example i
$\mathbf{p}_i \in \mathbb{R}^6$	:	pull=(target-prediction)/error rate of example i
$\tilde{\mathbf{p}}_i \in \mathbb{R}^6$	:	normalized pull=pull/std of example i where std is the modified standard deviation obtained from the Modified Gaussian Fit applied to the pulls
f_θ	:	our model parameterized by θ
$(\cdot)^+ = \max(0, \cdot)$	:	Rectified Linear Unit (ReLU).
POLAR data	:	$\{(\mathbf{x}_i, \mathbf{y}_i)\}_i$
$\mathcal{E}^+(b) \subset$ POLAR data	:	set of positive examples of interest
$\mathcal{E}^-(b) \subset$ POLAR data	:	set of negative examples of interest
$\mathcal{E}^\pm(b) \subset$ POLAR data	:	either $\mathcal{E}^+(b)$ or $\mathcal{E}^-(b)$
$\mathcal{C}^+(b) \subset \mathcal{E}^+(b)$	:	positive cluster: subset of the positive examples of interest
$\mathcal{C}^-(b) \subset \mathcal{E}^-(b)$	:	negative cluster: subset of the negative examples of interest
$\mathcal{C}^\pm(b) \subset \mathcal{E}^\pm(b)$	:	either $\mathcal{C}^+(b)$ or $\mathcal{C}^-(b)$
$\mathcal{C}^\pm(bs), bs \in \mathcal{P}(\{0, 1, \ldots, 5\})$	:	*pure* cluster intersection between energy bins specified in bs
t_s	:	starting time of a cluster
t_e	:	ending time of a cluster

[1] See Appendix E

[2] See Appendix B.3.3

ABSTRACT

Machine Learning (ML) methods are increasingly used in variety of domains such as physics, in spite of the lack of model interpretability. Although the automatic real-time detection and localization of astronomical events like Gamma-Ray Bursts (GRBs) can be done efficiently with low miss-detection and play a key role in subsequent manual analysis, sequential models trained purely on photon rates and energy information cannot explicitly model the correlations between diverse un-unused measurements. Motivated by model interpretability, we introduce a methodology explicitly modelling photon rates from diverse measurements that were previously un-unused. The methodology uses an MLP to model the background and residual analysis to extract time intervals containing some GRBs or other events like solar flares. The MLP, instead of forecasting the occurrence of a GRB from past photon rates and energy information tries to predict the current photon rate from diverse measurements. With 3 hidden layers and 200 epochs, we extracted 3178 clusters based on cluster intersections of more than 3 energy bins. To foster future analysis on these clusters, we release our code at https://github.com/Zenchiyu/POLAR-background-prediction.

Keywords POLAR data · Regression · Anomaly Detection · Gamma Ray Bursts

1 Introduction

Many astroparticle physicists are interested in Gamma-Ray Bursts (GRBs), which are intense, energetic, and extra-galactic explosions. They are bursts of high-energy photons, more precisely gamma rays, with an undetermined cause.

As physicists want to answer questions about a phenomenon such as GRBs, they may build models at a finer scale to their questions to solve them. However, a model is only an abstraction of reality, and manually crafting useful ones can require deep expertise. For instance, the Lattice Boltzmann Method tackles fluid simulation by modelling densities of particles modified by collision and streaming operators.

In the current era of Machine Learning (ML) and especially Deep Learning (DL), one can solve or assist a range of tasks. As Machine Learning aims at automatically extracting refined information from data, it becomes appealing for physicists to train ML models on their measurements to gain insight into the underlying structure of the universe.

Although automatic extraction from data is attractive and can work very well, Machine Learning, especially Deep Learning models, are not currently suited for interpretability. In contrast, handcrafted models can thrive at the latter.

As GRBs lead to spikes in light curves, curves of photon counts or rates for specific energy bins/bands, physicists can use ML models for automatic GRB detection. In particular, Koziol (2023) designed HAGRID (High Accuracy GRB Rapid Inference with Deep Learning in Space), an ML-based method for fast real-time GRB detection and localization based on photon counts or rates with energy information.

Koziol's (2023) GRB detection was obtained by training a sequential model on temporal series, namely a Long-Short Term Memory (LSTM) model on labeled simulated data with simulated GRBs. His LSTM attempts to predict the occurrence of a GRB at time t based on the past 60 seconds of photon rates and energy information. Although there could be a data distribution mismatch between the training set and the data from the POLAR detector attached to the Tiangong-2 Spacelab, the LSTM predicted all GRBs within a 10-day window of POLAR data.

The training data consisted of time snippets with a simulated background that could include simulated GRBs.

Although GRBs lead to spikes in light curves, the converse is incorrect since astronomical events like solar flares[1] can cause bumps in the photon count as they emit electromagnetic

[1] See Solar flare on Wikipedia.

radiation from radio waves to gamma rays. Moreover, GRBs may take different forms. Therefore, Koziol (2023) mentioned that a more precise model of the background, requiring a finer classification of events, is preferable to reduce the false detection of GRBs.

As Koziol (2023) did not account for quantities/measurements other than photon rates and energy information, the relationships between diverse un-tracked measurements cannot be automatically inferred. Although adding the un-tracked measurements to the temporal series used as input to a sequential model is possible, we opted for a basic Multi-Layer Perceptron (MLP) to reduce the complexity barrier to interpretability.

In our work, we present a methodology for GRB detection using an MLP to model the background trained on POLAR data to predict photon rates from other quantities. We then extract thousands of time intervals based on their residuals. These time intervals require further analysis to categorize them since some include solar flares and known GRBs. Although our methodology lacks model interpretability, it is a step towards a better understanding of GRBs as we now explicitly try to model the relationships between diverse measurements.

We first present the details of our methodology, starting with a high-level overview (Section 2) and then display our results (Section 3). We finally discuss our results (Section 4) and conclude with some extensions (Section 5). We also report information on our dataset, technical specifications and so on in the different Appendices.

2 Methodology

In a high-level overview, we proceed as follows:

- Train the background model by solving a multi-output regression problem on the POLAR data **without** 25 known GRBs[2] (Xiong et al., 2017).
- Apply the background model to the POLAR data **with** the 25 known GRBs (Xiong et al., 2017).
- Extract time intervals based on the residuals (create clusters and cluster intersections)
- Brief peek at model interpretability using Jacobian matrices (containing the partial derivatives of the model output variables w.r.t. the input variables).

We first trained our model on an initial seed of 42 to have reproducible results. However since everything depends on this single initial seed, we later train our model on 10 randomly generated seeds[3] and report the two first statistical moments of some results, for instance, properties of examples of interest/clusters or cluster intersections.

2.1 Background model

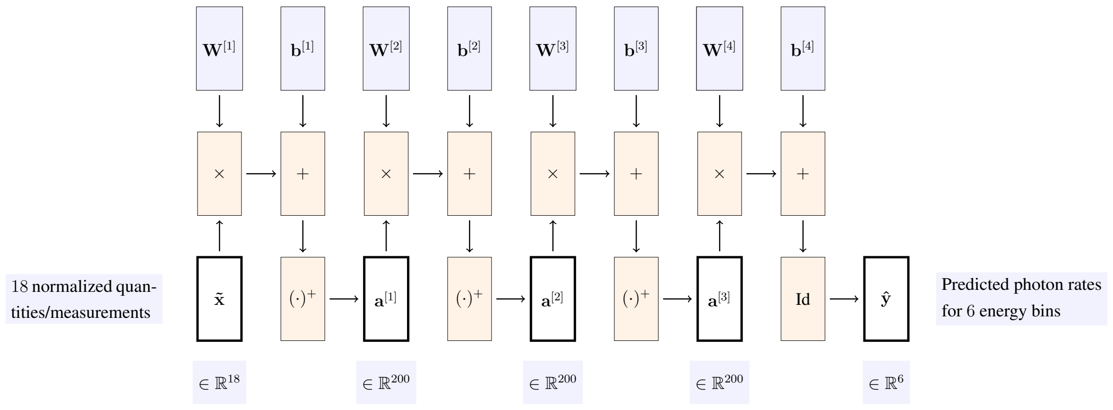


Figure 1: Representation of our model f_θ: a 4-layer Multi-Layer Perceptron with Rectified Linear Unit activation functions.
We derived this figure from section 3.1 Perceptron and section 3.4. Multi-Layer Perceptrons of Prof. François Fleuret Deep Learning course handouts/slides (Fleuret, 2023).

[2]See Appendix B for more information on what "POLAR data without 25 known GRBs" means, how it is split into train/val/test sets and how we normalize the data before feeding it to the model.

[3]See Appendix C for the exhaustive list of the 10 different seeds generated by the RANDOM shell variable with initial seed 42.

$\boxed{\text{Train the background model by solving a multi-output regression problem on the POLAR data without 25 known GRBs.}}$

Parametric model: The background model shown in Figure 1 is a 4-layer Multi-Layer Perceptron (MLP)[4] with an input dimension of 18 and an output dimension[5] of 6 (see Table 9 for an exhaustive list of their meanings). For simplicity, we used for each hidden layer, Rectified Linear Unit (ReLU) activation functions and 200 hidden units.

Our model takes a normalized example $\tilde{\mathbf{x}} \in \mathbb{R}^{18}$ as input[6] and spits out its predicted photon rates $\hat{\mathbf{y}} = f_\theta(\tilde{\mathbf{x}}) \in \mathbb{R}^6$.

We provide more information on the model architecture, hyper-parameters and optimization procedure in Appendix C.

Training/validation/test sets: We used our *periodical split*[7] method to split the POLAR data without 25 known GRBs into train/val/test sets. We then tune our model parameters using the training set.

Loss: We used the weighted mean squared error (WMSE) loss with the following form

$$\mathcal{L}(\theta) = \frac{1}{\sum_{i,b} w_{i,b}} \sum_{i,b} w_{i,b} \cdot (\hat{\mathbf{y}}_i(b) - \mathbf{y}_i(b))^2 \tag{1}$$

where $w_{i,b} = \frac{1}{\mathbf{e}_i(b)^2}$[8] is the inverse squared error rate of example i for the photon rate in energy bin b. Note that the index i only goes through the training set or a subset of the training set[9]. This specific loss introduces a bias in the penalization, aiming to penalize a greater number of bad predictions when error rates are low while being more tolerant towards bad predictions when error rates are high.

2.2 Predictions over the POLAR data with 25 known GRBs

$\boxed{\text{Apply the background model to the POLAR data } \textbf{with} \text{ the 25 known GRBs.}}$

To obtain our predictions over the POLAR data, we just pass the whole POLAR data to our model after using the same normalization procedure[10] as in the training phase.

[4]See Appendix A.1

[5]Multi-output regression: regression but with an output dimension > 1

[6]The data is fed to the model only after normalization as explained in Appendix B.3.3.

[7]See Appendix B.3.2

[8]To not confuse $w_{i,b}$ with the model parameters. Without using the loss function, altering the model parameters θ causes the predictions $\hat{\mathbf{y}}_i$'s to change.

[9]For instance when we work with mini-batches

[10]See Appendix B.3.3

2.3 Clusters and cluster intersections

Extract time intervals based on the residuals (create clusters and cluster intersections).

2.3.1 Clusters and cluster intersections based on residuals

Examples of interest Based on the residuals of $N = 3'954'837$ examples:

$$\mathbf{r}_i \doteq (\mathbf{y}_i - \hat{\mathbf{y}}_i) \in \mathbb{R}^6, \forall i = 0, \dots, N-1 \tag{2}$$

we can, for each energy bin $b \in \{0, \dots, 5\}$, retrieve examples i for which $\mathbf{r}_i(b) > k \cdot \sigma(b)$ where $\sigma(b)$ is the modified standard deviation obtained from applying the Modified Gaussian Fit[11] to $\{\mathbf{r}_i(b)\}_i$.

Therefore, the retrieved set of examples is:

$$\mathcal{E}^+(b) \doteq \{(\mathbf{x}_i, \mathbf{y}_i) \in \text{POLAR data} \,|\mathbf{r}_i(b) > k \cdot \sigma(b)\} \tag{3}$$

where each element will be called a *positive example of interest* because the target photon rate is significantly higher than the predicted photon rate.

Similarly, we can obtain the *negative examples of interest* by taking examples for which the target photon rate is significantly lower than the predicted photon rate.

$$\mathcal{E}^-(b) \doteq \{(\mathbf{x}_i, \mathbf{y}_i) \in \text{POLAR data} \,|\mathbf{r}_i(b) < -k \cdot \sigma(b)\} \tag{4}$$

Clusters This leads to the idea of *positive/negative* clusters of a particular energy bin b which are subsets of $\mathcal{E}^+(b)$ or $\mathcal{E}^-(b)$ respectively.

Informally, a positive/negative cluster is a set of time-consecutive[12] positive/negative examples of interest.

Therefore, a cluster $\mathcal{C}^{\pm}(b) \subset \mathcal{E}^{\pm}(b)$ can alternatively be described by its start time t_s [s] and end time t_e [s] (or cluster length $t_e - t_s$ [s]).

Using two properties of a cluster:

- Cluster length $t_e - t_s$ [s]
- Integral $\sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{C}^{\pm}(b)} \mathbf{y}_i(b)$ (sum of rates for energy bin b for all examples in the cluster)

We can visualize our clusters in 2 dimensions using a scatter plot.

[11] See Appendix E

[12] The notion of "time-consecutive examples" is time binning-dependent. For example, if the binning is 1 second between examples, then two examples are considered time-consecutive if they are at [more or less] 1 second apart. Non-time-consecutive examples may exist due to missing data.

Intersection of clusters Since GRBs are supposed to be visible in different energy bins as our energy bins overlap[13], we are likewise interested in the intersection of clusters across/between different energy bins.

In simple words, interesting examples should appear in multiple energy bins.

We denote a *pure* cluster intersection by $\mathcal{C}^{\pm}(bs), bs \in \mathcal{P}(\{0, 1, \ldots, 5\})$ where $|bs|$ indicates how many energy bins contributed to our cluster. We call it *pure* because all its examples of interest come from the same energy bins bs.

However, we can also create *mixed* cluster intersections based on examples for which their energy bins bs are mixed[14]. In particular, we can define cluster intersections via the number of energy bins of individual examples of interest. Therefore, when talking about a cluster intersection with 3 energy bins, each of its examples has a $|bs| = 3$. However, due to the mixed nature of the cluster intersection, the union of the sets of energy bins does not necessarily have a cardinality of 3.

To compute the intersections of $\mathcal{C}^{\pm}(b)$'s across energy bins, we add labels to each example of interest. The labels would then indicate the energy bins in which the example is an example of interest. After labeling each example of interest, we can compute the start and end times of clusters after masking examples of interest based on their labels or the number of labels.

For example, if we want the set of cluster intersections with > 3 energy bins, we take all the examples of interest with more than 3 labels, i.e. $|bs| > 3$ and recompute, based on them, the start and end times of each possible cluster.

Otherwise, if we were interested in a particular set of labels bs, we could recompute based on all the examples of interest with the same bs, the start and end times.

2.3.2 Clusters and cluster intersections based on pulls

We can do everything above but with the *pulls* instead of the *residuals*[15]. This gives us clusters and cluster intersections based on the pulls.

We define the *pulls* as the residuals divided by their corresponding photon error rates[16]. Therefore, discrepancies between the targets and the predictions are less worrying for large error rates.

[13]See comments in Table 9

[14]In other words, the bs of these examples are not necessarily all the same

[15]The modified standard deviation using the Modified Gaussian fit is also recomputed based on the *pulls*

[16]See Appendix B.1.1

Mathematically, we define the *pulls* as follows:

$$\mathbf{p}_i(b) \doteq \frac{\mathbf{r}_i(b)}{\mathbf{e}_i(b)}, \forall b = 0, \ldots, 5, \forall i = 0, \ldots, 3954837 - 1 \tag{5}$$

where $\mathbf{e}_i(b)$ is the error rate of example i for the photon rate in energy bin b[17].

Normalized pulls After obtaining the examples of interest using the pulls, normalized pulls are interesting to look at as they simplify interpretation.

The *normalized pulls* are defined as follows:

$$\tilde{\mathbf{p}}(b) = \frac{\mathbf{p}(b)}{\sigma(b)}, \forall b = 0, \ldots, 5 \tag{6}$$

where the modified standard deviation $\sigma(b) \in \mathbb{R}_{\geq 0}$ is obtained by applying the Modified Gaussian Fit[18] to the pulls $\{\mathbf{p}_i(b)\}_i$.

Normalized pulls simplify interpretation as

$$\mathcal{E}^+(b) = \{(\mathbf{x}_i, \mathbf{y}_i) \in \text{POLAR data } | \mathbf{p}_i(b) > k \cdot \sigma(b)\} \tag{7}$$

is equivalent to

$$\mathcal{E}^+(b) = \{(\mathbf{x}_i, \mathbf{y}_i) \in \text{POLAR data } | \tilde{\mathbf{p}}_i(b) > k\} \tag{8}$$

Therefore, *positive* examples of interest lie above a horizontal line at $y = k$ in the normalized pull plot. Similarly, *negative* examples of interest lie below a horizontal line at $y = -k$ in the normalized pull plot.

2.3.3 Discarding clusters temporally close to missing data

We may sometimes choose to drop clusters temporally close to missing data for analysis purposes and due to the behavior of our predictions around them. For instance, we may discard a cluster if its start or end time is within some time window size (in seconds) after or before missing data respectively.

Generally, sensors are switched off in the South Atlantic Anomaly (SAA), causing missing data. We will come back in the discussion section on why we might want to discard some clusters.

[17] In general, since we don't necessarily try to predict all the photon rates, the photon rate in energy bin b is not the same as the photon rate b; `rate[b]`. Similarly, in general, the error rate of energy bin b is not the same as the error rate of photon rate b; `rate_err[b]`.

[18] See Appendix E

2.4 Model interpretability with partial derivatives

Brief peek at model interpretability using Jacobian matrices (containing the partial derivatives of the model output variables w.r.t. the input variables).

Effect of a normalized feature on a predicted photon rate One way to interpret how one of the model output variables changes according to one of its input variables is to slightly change the input and see what we get as output.

This notion is captured by the partial derivatives of the model output variables w.r.t. the [normalized] input variables, which can be arranged in a matrix.

Therefore, we just need to compute the Jacobian matrix $\mathbf{J}(\tilde{\mathbf{x}}_i)$ for each normalized example $\tilde{\mathbf{x}}_i$[19]:

$$\mathbf{J}(\tilde{\mathbf{x}}_i) = \left[\frac{\partial \hat{\mathbf{y}}_i(b)}{\partial \tilde{\mathbf{x}}_i(j)}\right]_{b=0,\dots,5, j=0,\dots17} = \begin{bmatrix} \frac{\partial \hat{\mathbf{y}}_i(0)}{\partial \tilde{\mathbf{x}}_i(0)} & \frac{\partial \hat{\mathbf{y}}_i(0)}{\partial \tilde{\mathbf{x}}_i(1)} & \cdots & \frac{\partial \hat{\mathbf{y}}_i(0)}{\partial \tilde{\mathbf{x}}_i(17)} \\ \frac{\partial \hat{\mathbf{y}}_i(1)}{\partial \tilde{\mathbf{x}}_i(0)} & \frac{\partial \hat{\mathbf{y}}_i(1)}{\partial \tilde{\mathbf{x}}_i(1)} & \cdots & \frac{\partial \hat{\mathbf{y}}_i(1)}{\partial \tilde{\mathbf{x}}_i(17)} \\ \vdots & \vdots & & \\ \frac{\partial \hat{\mathbf{y}}_i(5)}{\partial \tilde{\mathbf{x}}_i(0)} & \frac{\partial \hat{\mathbf{y}}_i(5)}{\partial \tilde{\mathbf{x}}_i(1)} & \cdots & \frac{\partial \hat{\mathbf{y}}_i(5)}{\partial \tilde{\mathbf{x}}_i(17)} \end{bmatrix} \in \mathbb{R}^{6\times 18} \tag{9}$$

Effect of an un-normalized feature on a predicted photon rate However, physicists are more interested in the measurements themselves such as the un-normalized crabarf[20] rather than the normalized measurements.

This leads to the partial derivatives of the model output variables w.r.t. the *un-normalized* input variables, which can again be arranged in a Jacobian matrix for each un-normalized example $\mathbf{x}_i$:

$$\mathbf{J}(\mathbf{x}_i) = \left[\frac{\partial \hat{\mathbf{y}}_i(b)}{\partial \mathbf{x}_i(j)}\right]_{b=0,\dots,5, j=0,\dots17} = \begin{bmatrix} \frac{\partial \hat{\mathbf{y}}_i(0)}{\partial \mathbf{x}_i(0)} & \frac{\partial \hat{\mathbf{y}}_i(0)}{\partial \mathbf{x}_i(1)} & \cdots & \frac{\partial \hat{\mathbf{y}}_i(0)}{\partial \mathbf{x}_i(17)} \\ \frac{\partial \hat{\mathbf{y}}_i(1)}{\partial \mathbf{x}_i(0)} & \frac{\partial \hat{\mathbf{y}}_i(1)}{\partial \mathbf{x}_i(1)} & \cdots & \frac{\partial \hat{\mathbf{y}}_i(1)}{\partial \mathbf{x}_i(17)} \\ \vdots & \vdots & & \\ \frac{\partial \hat{\mathbf{y}}_i(5)}{\partial \mathbf{x}_i(0)} & \frac{\partial \hat{\mathbf{y}}_i(5)}{\partial \mathbf{x}_i(1)} & \cdots & \frac{\partial \hat{\mathbf{y}}_i(5)}{\partial \mathbf{x}_i(17)} \end{bmatrix} \in \mathbb{R}^{6\times 18} \tag{10}$$

Relation between $\mathbf{J}(\mathbf{x}_i)$ and $\mathbf{J}(\tilde{\mathbf{x}}_i)$ Since PyTorch autograd provides us with the $\mathbf{J}(\tilde{\mathbf{x}}_i)$'s after some tweaks[21], all that remains is to apply the chain rule:

$$\frac{\partial \hat{\mathbf{y}}_i(b)}{\partial \mathbf{x}_i(j)} = \frac{\partial \hat{\mathbf{y}}_i(b)}{\partial \tilde{\mathbf{x}}_i(j)} \frac{\partial \tilde{\mathbf{x}}_i(j)}{\partial \mathbf{x}_i(j)} == \frac{\partial \hat{\mathbf{y}}_i(b)}{\partial \tilde{\mathbf{x}}_i(j)} \frac{\partial \left(\frac{\mathbf{x}_i(j) - \boldsymbol{\mu}_{\text{train}}(j)}{\boldsymbol{\sigma}_{\text{train}}(j)}\right)}{\partial \mathbf{x}_i(j)} = \frac{1}{\boldsymbol{\sigma}_{\text{train}}(j)} \frac{\partial \hat{\mathbf{y}}_i(b)}{\partial \tilde{\mathbf{x}}_i(j)} \tag{11}$$

[19] $\tilde{\mathbf{x}}_i$ is what we get after applying input feature normalization to $\mathbf{x}_i$

[20] one of the un-normalized input dimension

[21] This is thanks to autograd tracking all tensor operations and automatically computing partial derivatives

which leads to this relation:

$$\mathbf{J}(\mathbf{x}_i) = \mathbf{J}(\tilde{\mathbf{x}}_i)\text{diag}\left(\left[\frac{1}{\boldsymbol{\sigma}_{\text{train}}(j)}\right]_{j=0,\dots,17}\right) \tag{12}$$

where $\text{diag}\left(\left[\frac{1}{\boldsymbol{\sigma}_{\text{train}}(j)}\right]_{j=0,\dots,17}\right)$ is a matrix whose diagonals are inverses of feature standard deviations[22].

Computing $\mathbf{J}(\mathbf{x}_i)$ in PyTorch Computing $\mathbf{J}(\mathbf{x}_i)$, as shown previously, is doable using PyTorch broadcasting, which removes the need to compute the diagonal matrix.

However, it is also possible to directly obtain $\mathbf{J}(\mathbf{x}_i)$'s with PyTorch autograd by specifying that we want the partial derivatives w.r.t. the un-normalized examples and not w.r.t the model input.

Both methods should lead to the same $\mathbf{J}(\mathbf{x}_i)$.

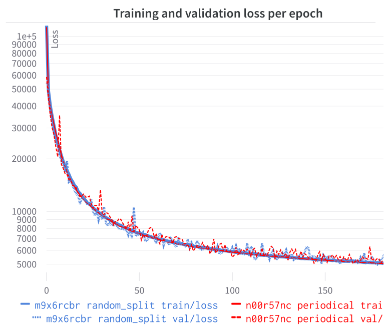
Histograms of partial derivatives For each (target, feature) pair, we create histograms of their partial derivatives[23] and on top, we add a Modified Gaussian Fit[24].

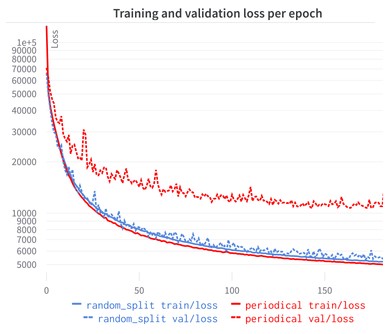
[22]Not from the Modified Gaussian Fit

[23]Either using $\mathbf{J}(\mathbf{x}_i)(b, j)$ or $\mathbf{J}(\tilde{\mathbf{x}}_i)(b, j)$

[24]See Appendix E

3 Results


(a) Periodical vs. random, 98/1/1


(b) Periodical vs. random, 80/10/10

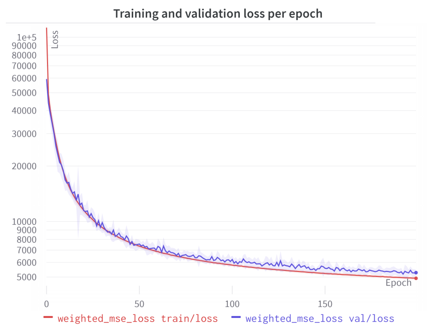
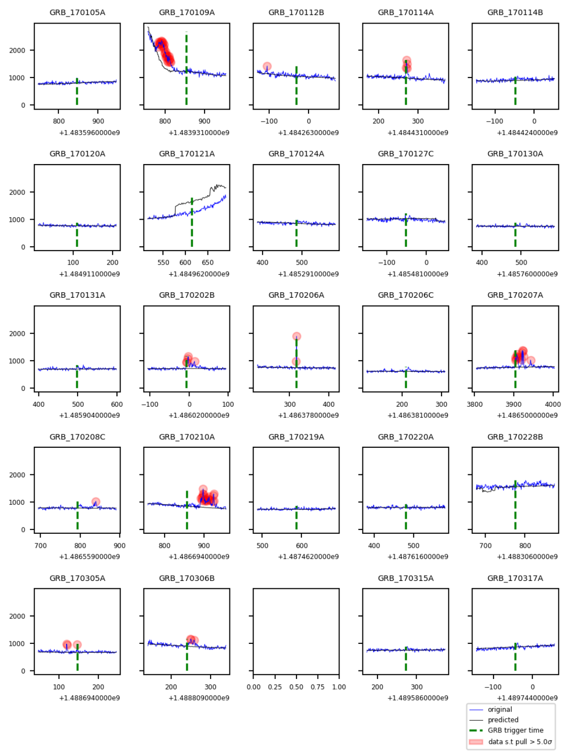

(c) Avg ± std over 10 different seeds

Figure 2: Training & validation WMSE loss per epoch using seed 42 for different train/val/test splitting methods/ratios (a & b) or 10 different seeds[25] with periodical split and $98/1/1\%$ splitting ratios (c).


(a) Predicted vs. target photon rates

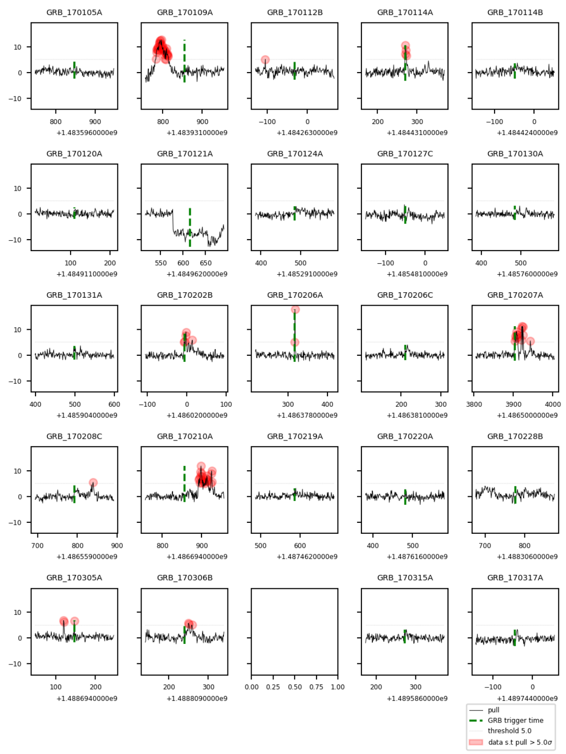

(b) Normalized pull plot

Figure 3: At inference time, our model is applied to the POLAR data and these figures only show what happens for energy bin 0 and time windows of ± 100 [s] around 25 known GRB trigger times (Xiong et al., 2017). Based on pulls with a selection threshold of $k = 5$, positive examples happening within these time intervals are shown in red.

3.1 Multi-output regression

For reproducibility, we trained our model on a fixed seed of 42 and this achieves, after 200 epochs, a WMSE of approximately $4.946 \cdot 10^3$ and $5.062 \cdot 10^3$ on the training and validation set respectively.

We show in Figure 2a and 2b the training and validation loss per epoch for two different data splitting methods: `random_split` and `periodical`. These two graphs also show what happen when we change the train/val/test splitting ratios.

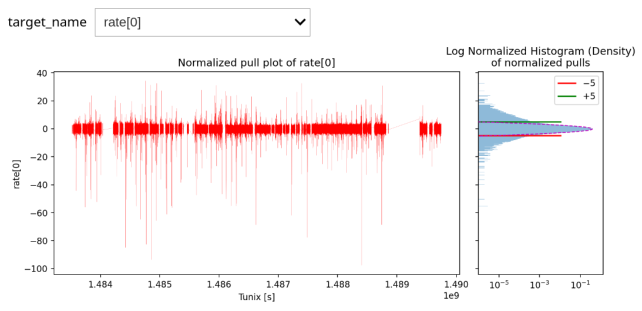
Because the model parameters obtained after training depend on the initial seed, we also train our model on 10 different seeds and report, in Figure 2c, the average and standard deviation WMSE loss per epoch for both the training and validation set. In particular we obtain after 200 epochs, an average WSME of approximately $4.920 \cdot 10^3$ and $5.280 \cdot 10^3$ on the training and validation set respectively (See Table 2 which includes the standard deviations).

Table 2: Training and validation WSME after 200 epochs

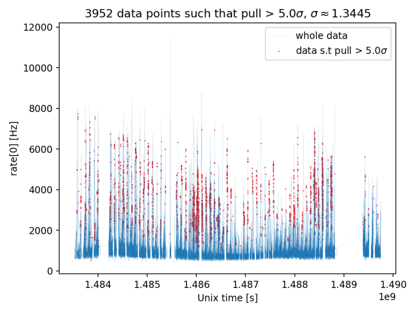
seed(s)	training	validation
seed 42	4945.775	5062.002
$\mu \pm \sigma$ over 10 other seeds	4919.711 ± 87.771	5279.669 ± 196.458

[25]See Appendix C for the exhaustive list of the 10 different seeds generated by the RANDOM shell variable with initial seed 42.

3.2 Examples of interest, clusters and cluster intersections



(a) Normalized Pull plot



(b) Positive examples (red) from the light curve

Figure 4: Positive examples of interest in energy bin 0, i.e. from $\mathcal{E}^+(0)$. Although similar plots exist for the 5 other energy bins, we preferred to summarize them differently using Table 3.

Once our model is trained on a subset of the POLAR data without 25 known GRBs and with a seed of 42, we apply it to the data with the 25 known GRBs. Figure 4a shows the normalized pulls from the first energy bin and its histogram in logarithmic scale. Figure 4b shows informally how positive examples (from the first energy bin) are spread across the mission and Figure 3a with Figure 3b both show zoomed-in time intervals of 25 known GRBs (Xiong et al., 2017). Unless stated otherwise, we will proceed with seed 42 and a selection threshold[26] $k = 5$.

We report in Table 3 some properties of our set of positive/negative examples or clusters for each energy bin b where:

- $\sigma(b)$ is the modified standard deviation[27]
- $|\mathcal{E}^{\pm}(b)|$ is the number of positive or negative examples
- $|\mathcal{C}^{\pm}(b)|$ is the number of positive or negative clusters

Note that no cluster temporally close to missing data are discarded here ("no discard cluster").

[26]Reminder: the selection threshold k was used to select positive examples. See Section 2.3.2.

[27]Reminder: the modified standard deviation is the one obtained from the Modified Gaussian Fit. See Appendix E

Table 3: Properties of our set of positive/negative examples $\mathcal{E}^{\pm}(b)$ or clusters $\mathcal{C}^{\pm}(b)$ for each energy bin b (no "discard cluster")

b	photon rate of energy bin b	$\sigma(b)$	$\lvert\mathcal{E}^{+}(b)\rvert$	$\lvert\mathcal{E}^{-}(b)\rvert$	$\lvert\mathcal{C}^{+}(b)\rvert$	$\lvert\mathcal{C}^{-}(b)\rvert$
0	rate[0]	1.344533	3952	2576	1507	1570
1	rate[1]	1.199649	5172	2834	1969	1682
2	rate[5]	1.322103	18408	15227	5500	6666
3	rate[6]	1.256458	20015	15480	5893	6807
4	rate[10]	1.506859	13121	10317	3967	4700
5	rate[12]	1.501043	13227	10423	4004	4745

We also report in Table 4 the number of positive/negative cluster intersections for all sets of energy bins for which examples of interest appear in and for different conditions (e.g. cluster intersections for which examples of interest individually appear in more than 3 energy bins).

Table 4: Number of positive/negative cluster intersections for different sets of energy bins bs or conditions[28]. This time, there's a discard window of 30 seconds. We underlined a few elements as they may reappear in different figures.

bs or condition	negative	positive
# inter > 0	6798	5408
# inter > 1	5550	4552
# inter > 2	4029	3332
# inter > 3	$\underline{3744}$	$\underline{3178}$
# inter > 4	1261	1422
# inter > 5	833	1028
0	54	40
0,1	26	42
0,1,2	5	12
0,1,2,3	15	31
0,1,2,3,4	–	1
0,1,2,3,4,5	$\underline{833}$	$\underline{1028}$
0,1,2,4,5	2	17
0,1,3,4,5	–	2
0,1,4,5	5	7
0,2	12	5
0,2,3	9	2
0,2,3,4,5	263	150
0,2,4,5	17	5
0,2,5	1	–
0,3,4,5	–	1
0,4,5	2	10
1	50	63
1,2	8	8
1,2,3	62	90
1,2,3,4	–	2
1,2,3,4,5	426	718
1,2,3,5	1	5
1,2,4,5	1	6
1,3	28	47
1,3,4,5	–	1
1,4,5	3	6
2	926	662
2,3	2874	2815
2,3,4	14	10
2,3,4,5	3703	3474
2,3,5	78	89
2,4	2	–
2,4,5	301	157
2,5	10	8
3	1452	1573
3,4	2	1
3,4,5	20	15
3,5	3	–
4	2	–
4,5	151	122
5	13	9

Since the model also depends on the initial seed we track the same properties but for the 10 different seeds (see Table 5 and 6 for statistics over the 10 different seeds.).

Note that these statistics may implicitly be computed over less than 10 different seeds depending on whether the bs or conditions appeared in them.

The asterisk denotes the case for which there's a single run/seed/model.

[28]See our definitions of pure and mixed clusters in Section 2.3.2

Table 5: Mean ± standard deviation (over 10 different seeds) properties of our set of positive/negative examples $\mathcal{E}^{\pm}(b)$ or clusters $\mathcal{C}^{\pm}(b)$ for each energy bin b (no "discard cluster")

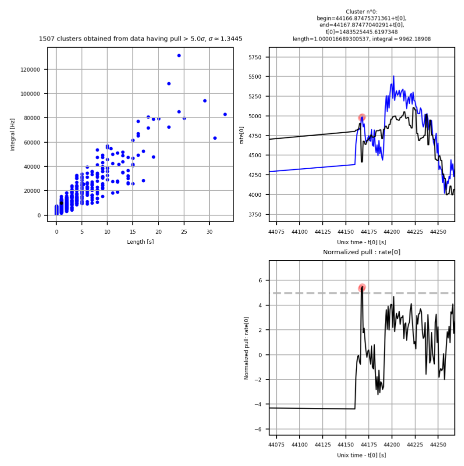
b	photon rate of energy bin b	$\sigma(b)$	$\lvert\mathcal{E}^{+}(b)\rvert$	$\lvert\mathcal{E}^{-}(b)\rvert$	$\lvert\mathcal{C}^{+}(b)\rvert$	$\lvert\mathcal{C}^{-}(b)\rvert$
0	rate[0]	1.346 ± 0.004	4580.5 ± 942.655	2480.7 ± 543.045	1790.6 ± 260.226	1416.2 ± 205.895
1	rate[1]	1.202 ± 0.004	5640.9 ± 1439.2	2811.8 ± 765.776	2194.9 ± 356.798	1587.8 ± 306.404
2	rate[5]	1.326 ± 0.007	21036.0 ± 3428.799	13540.9 ± 2770.79	6427.9 ± 881.338	5828.4 ± 959.143
3	rate[6]	1.257 ± 0.007	21968.4 ± 3357.42	13874.4 ± 2903.117	6731.1 ± 885.362	5957.6 ± 1012.683
4	rate[10]	1.515 ± 0.007	14732.1 ± 2519.334	9365.5 ± 1974.629	4645.4 ± 659.937	4225.7 ± 701.068
5	rate[12]	1.509 ± 0.007	15253.3 ± 2929.28	9264.9 ± 2122.366	4790.1 ± 696.887	4179.2 ± 746.54

Table 6: Mean ± standard deviation (over 10 different seeds) number of positive/negative cluster intersections for different sets of energy bins bs or conditions[29]. This time, there's a discard window of 30 seconds.

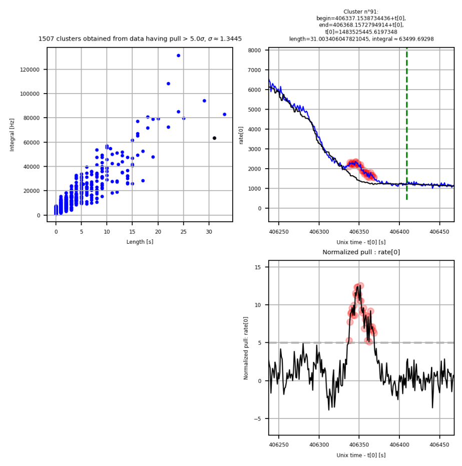
bs or condition	negative	positive
# inter > 0	5762.1 ± 965.708	6487.6 ± 826.131
# inter > 1	4802.1 ± 822.34	5450.6 ± 752.954
# inter > 2	3493.8 ± 610.247	4083.5 ± 564.85
# inter > 3	3238.4 ± 602.769	3821.5 ± 538.957
# inter > 4	1150.2 ± 232.379	1595.0 ± 202.499
# inter > 5	791.3 ± 165.14	1135.2 ± 149.199
0	23.8 ± 8.23	71.6 ± 18.368
0,1	16.6 ± 6.059	62.3 ± 45.631
0,1,2	4.0 ± 2.0	15.2 ± 9.739
0,1,2,3	12.4 ± 3.806	30.5 ± 17.219
0,1,2,3,4	23.0^{*}	2.5 ± 0.707
0,1,2,3,4,5	791.3 ± 165.14	1135.2 ± 149.199
0,1,2,3,5	1.0 ± 0.0	3.667 ± 4.899
0,1,2,4	1.0^{*}	3.0^{*}
0,1,2,4,5	2.8 ± 1.751	11.2 ± 7.315
0,1,2,5	1.0^{*}	2.667 ± 2.082
0,1,3	1.25 ± 0.5	4.875 ± 7.2
0,1,3,4,5	2.5 ± 2.811	2.25 ± 0.5
0,1,3,5	–	1.0^{*}
0,1,4	1.0^{*}	6.0^{*}
0,1,4,5	2.875 ± 1.959	5.9 ± 4.202
0,1,5	2.0^{*}	2.0^{*}
0,2	6.9 ± 2.726	10.7 ± 4.111
0,2,3	6.0 ± 3.432	7.1 ± 1.37
0,2,3,4	8.0^{*}	–
0,2,3,4,5	212.2 ± 42.158	242.7 ± 65.109
0,2,3,5	1.0 ± 0.0	3.5 ± 3.536
0,2,4	1.5 ± 0.707	1.0^{*}
0,2,4,5	5.8 ± 4.367	15.8 ± 5.095
0,2,5	1.0 ± 0.0	2.5 ± 2.38
0,3	1.0^{*}	3.0 ± 1.414
0,3,4,5	1.667 ± 1.155	1.75 ± 0.957
0,4	2.0^{*}	–
0,4,5	2.25 ± 1.753	11.6 ± 3.062
0,5	1.0^{*}	1.75 ± 1.5
1	52.6 ± 15.16	104.0 ± 118.135
1,2	3.444 ± 1.424	8.5 ± 4.143
1,2,3	56.1 ± 19.221	115.1 ± 41.391
1,2,3,4	14.75 ± 27.5	1.4 ± 0.548
1,2,3,4,5	413.5 ± 141.535	714.4 ± 132.802
1,2,3,5	2.167 ± 0.753	9.667 ± 19.268
1,2,4	–	4.0^{*}
1,2,4,5	3.143 ± 2.673	4.889 ± 3.333
1,2,5	–	3.0^{*}
1,3	15.6 ± 6.398	51.2 ± 42.695
1,3,4	1.0^{*}	–
1,3,4,5	1.333 ± 0.577	1.5 ± 0.577
1,3,5	–	5.5 ± 6.364
1,4	1.0^{*}	3.0^{*}
1,4,5	2.125 ± 1.356	5.5 ± 3.742
1,5	–	2.0 ± 0.0
2	734.8 ± 169.59	939.8 ± 235.236
2,3	2459.9 ± 493.897	3265.5 ± 779.499
2,3,4	114.8 ± 335.669	11.75 ± 12.395
2,3,4,5	3156.9 ± 657.884	4049.1 ± 676.033
2,3,5	60.444 ± 11.479	225.0 ± 392.74
2,4	18.333 ± 38.563	2.333 ± 1.751
2,4,5	212.3 ± 59.913	244.9 ± 73.942
2,5	9.8 ± 6.356	55.5 ± 127.411
3	1206.3 ± 282.876	1633.0 ± 282.032
3,4	6.25 ± 9.215	1.0 ± 0.0
3,4,5	19.7 ± 8.166	16.5 ± 4.503
3,5	1.889 ± 0.782	21.0 ± 58.877
4	9.333 ± 20.174	3.833 ± 4.167
4,5	123.5 ± 30.395	124.0 ± 31.383
5	7.3 ± 3.302	88.1 ± 234.779

We display in Figure 5 for seed 42, a few positive or negative clusters in the first energy bin to get an idea of how clusters look like and why we may want to discard clusters temporally close to missing data[30].

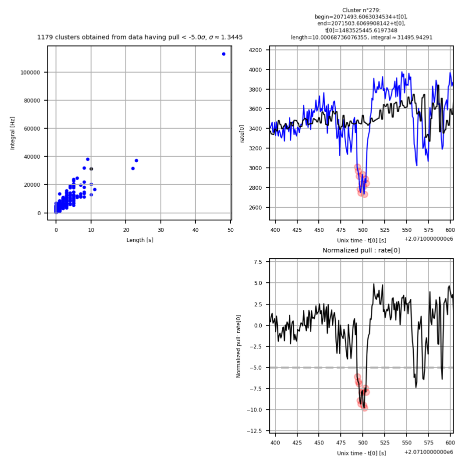
[30]We discarded for our cluster intersections



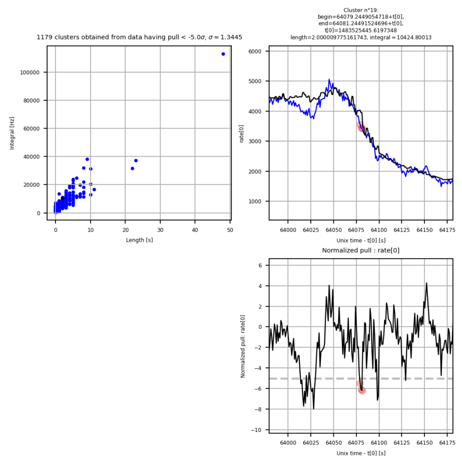
(a) Positive cluster close to missing data



(b) Solar flare confirmed by Prof. Nicolas Produit



(c) Undetermined negative cluster



(d) Another undetermined negative cluster

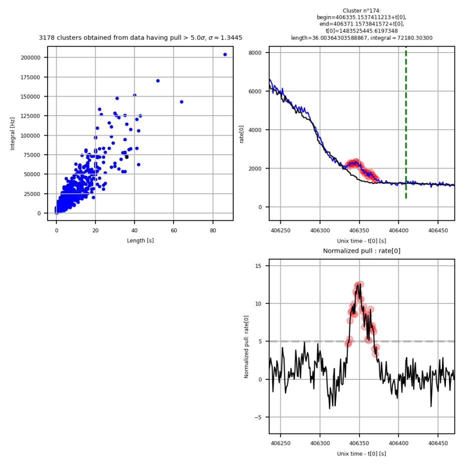
Figure 5: Two positive and negative clusters in energy bin 0, i.e. $\mathcal{C}^{+}(0)$ (no "discard cluster"). The prediction curve is black and the light curve for energy bin 0 is blue. Although similar plots exist for the 5 other energy bins, we omitted to report them.

Remarks:

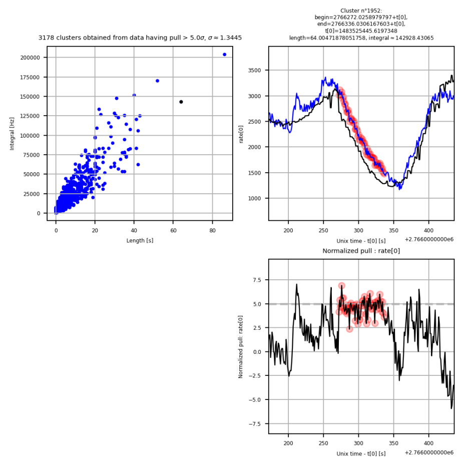
- Examples of interest from the selected cluster are highlighted in red (see Figure 5d where examples of interest outside of the cluster are not highlighted.)
- Y-axis scales can differ
- We defined the cluster length as the ending time minus the starting time.
- Therefore, the single-datapoint cluster length is 0, not 1. However, since the sample spacing is 1 second, one could interpret the cluster length as the ending time minus the starting time plus one.

Similarly, we display in Figure 6 and 7 for seed 42, a few positive cluster intersections to get an idea of how cluster intersections look like.

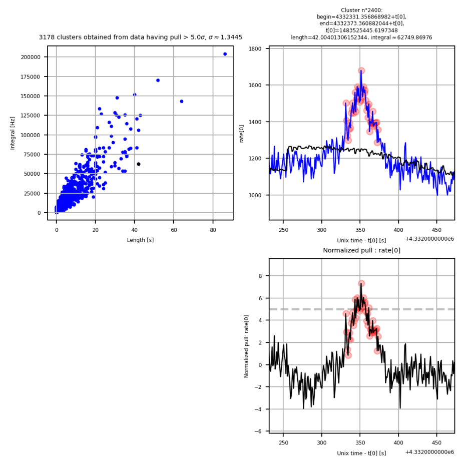
Although we only display the light curve and the normalized pull using the first energy bin, the cluster intersections may span different energy bins. Therefore, if the set of energy bins used excludes the first energy bin, the normalized pull plot and light curve would lose parts of their meanings.



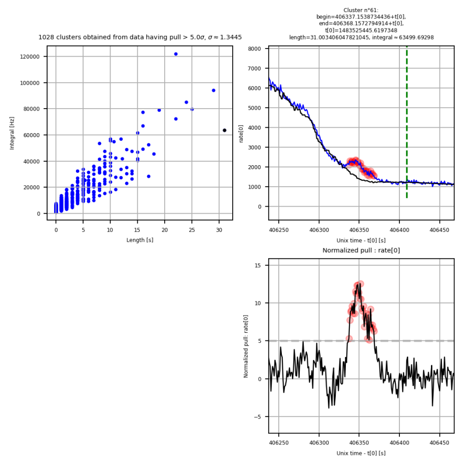
(a) Solar flare confirmed by Prof. Nicolas Produit



(b) Undetermined positive cluster intersection



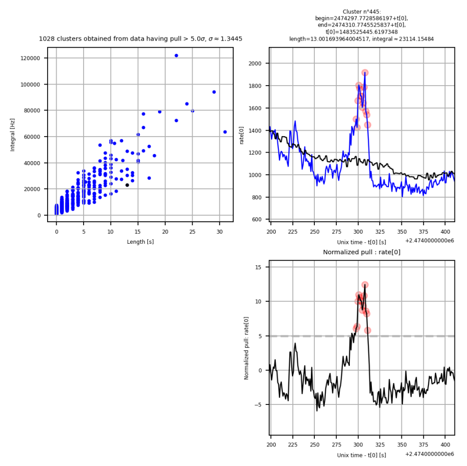
(c) Undetermined positive cluster intersection



(d) Solar flare confirmed by Prof. Nicolas Produit

Figure 6: Positive cluster intersections with more than 3 energy bins[31] (a, b & c) or exactly 6 energy bins (d) (each case with a 30-second discard window). The prediction curve is black and the light curve for energy bin 0 is blue.

[31]Note that mixed cluster intersections are possible, which clarifies why all have highlighted examples above and below the $y = 5$ line. Above the line, the examples have 0 in their labels, while below it, they do not.


(a) Undetermined positive cluster intersection

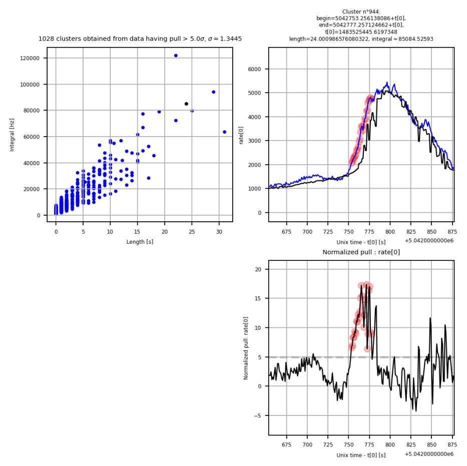
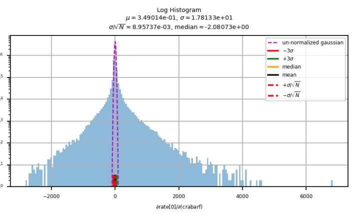

(b) Undetermined positive cluster intersection

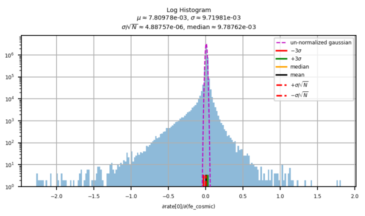
Figure 7: Positive cluster intersections with all energy bins and with a 30-second discard window. The prediction curve is black and the light curve for energy bin 0 is blue.

3.3 Model interpretability

Since our model tries to predict photon rates from diverse measurements, analyzing the relationship between the latter and the former becomes possible. However, due to the black-box nature of neural networks, it is hard to know what happens under the hood and what relationships the model found.

We only try to explore the average slopes, whether the predicted photon rates generally increase or decrease by varying an input variable. Although we only show in Figure 8, partial derivatives histograms of photon rate in energy bin 0 w.r.t. 3 un-normalized features[32], similar plots[33] exist for the 5 other energy bins and 15 other input variables.


(a) crabarf


(b) fe_cosmic

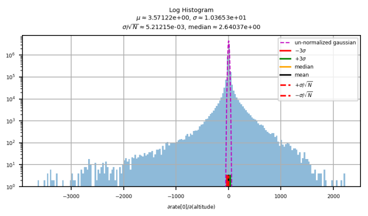

(c) altitude

Figure 8: Partial derivatives histograms of photon rate in energy bin 0 w.r.t 3 un-normalized features.

[32]Although we arbitrarily chose the cosmic rates and the altitude, Prof. Nicolas Produit was particularly interested in crabarf.

[33]In total, we have $18 \cdot 6 = 108$ partial derivatives histograms. You can find them in our Jupyter Notebook.

We summarize the 108 partial derivatives histograms with the following:

- A table containing the average partial derivative for each (un-normalized input, output) pair of variables.
- Another table reporting estimates of the standard deviations of average partial derivatives (see standard error) for each (un-normalized input, output) pair of variables. More specifically, the table reports partial derivatives' modified standard deviation divided by the square root of the sample size[34] for each (un-normalized input, output) pair of variables: $\frac{\sigma}{\sqrt{N}}$.
- A figure comparing the average slopes to their standard error.

Table 7: Average partial derivative for each (un-normalized input, output) pair of variables

	(unix_time-1474004181.5460)//5535.4+10	glat	glon	altitude	temperature	fe_cosmic	raz	decz	rax	decx	is_orbit_up	time_since_saa	crabarf	sun	sun_spot	B_r	B_theta	B_phi
rate[0]	-0.006348	-3.386663	0.330786	3.571223	-33.282600	0.007810	-0.047000	-0.641413	0.030088	-0.098069	12.639869	0.001951	0.349014	1.295663	-0.264237	-0.003815	0.053873	0.003346
rate[1]	-0.010823	-2.452013	0.265785	2.924349	-17.132353	0.000875	-0.036983	-0.460314	0.045000	0.008619	5.853580	0.001382	0.053972	1.044451	-0.104708	-0.002897	0.037117	0.003317
rate[5]	-0.053386	-10.668039	1.213200	15.969975	-57.475315	-0.001699	-0.164603	-2.503866	0.234378	-0.045545	62.795231	0.003153	-1.327233	5.108298	-2.024002	-0.014491	0.178131	0.019369
rate[6]	-0.052165	-9.160622	1.098845	14.209127	-39.907715	-0.007354	-0.152853	-2.144137	0.227230	0.061300	56.010948	0.002400	-1.196382	4.478731	-1.815495	-0.012730	0.151589	0.018343
rate[10]	-0.021446	-11.731500	1.283699	15.576557	-87.980400	0.018327	-0.166831	-2.739606	0.165998	-0.444852	79.014359	0.003677	0.121556	5.035269	-2.237882	-0.015298	0.200744	0.017750
rate[12]	-0.021986	-11.747561	1.283299	15.543121	-87.998154	0.018294	-0.166303	-2.735729	0.166432	-0.441618	78.930946	0.003694	0.085271	5.036674	-2.252446	-0.015306	0.200876	0.017737

Table 8: Partial derivatives' modified standard deviation σ divided by the square root of the sample size N for each (un-normalized input, output) pair of variables: $\frac{\sigma}{\sqrt{N}}$

	(unix_time-1474004181.5460)//5535.4+10	glat	glon	altitude	temperature	fe_cosmic	raz	decz	rax	decx	is_orbit_up	time_since_saa	crabarf	sun	sun_spot	B_r	B_theta	B_phi
rate[0]	0.000075	0.001859	0.000348	0.005212	0.011950	0.000005	0.000243	0.000861	0.000213	0.000575	0.234473	0.000002	0.008957	0.001062	0.004869	0.000004	0.000014	0.000005
rate[1]	0.000048	0.001171	0.000209	0.003411	0.006818	0.000003	0.000150	0.000558	0.000136	0.000363	0.154695	0.000001	0.005837	0.000695	0.003162	0.000002	0.000009	0.000004
rate[5]	0.000243	0.005862	0.001037	0.017287	0.028890	0.000014	0.000765	0.003023	0.000696	0.001871	0.904378	0.000007	0.029830	0.003432	0.015886	0.000010	0.000041	0.000019
rate[6]	0.000206	0.004875	0.000847	0.014772	0.023359	0.000012	0.000638	0.002596	0.000586	0.001583	0.772325	0.000006	0.025531	0.002923	0.013480	0.000008	0.000034	0.000017
rate[10]	0.000282	0.006962	0.001291	0.019615	0.036508	0.000016	0.000914	0.003465	0.000814	0.002198	1.028384	0.000008	0.034103	0.003911	0.018295	0.000013	0.000050	0.000022
rate[12]	0.000282	0.006966	0.001291	0.019625	0.036574	0.000016	0.000914	0.003466	0.000814	0.002199	1.027362	0.000008	0.034117	0.003913	0.018300	0.000014	0.000050	0.000022

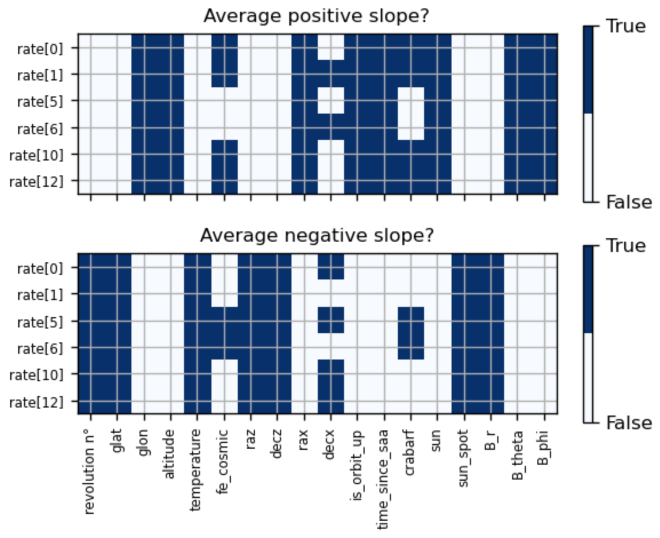


Figure 9: Is the average slope above (or below) its (negative) standard error[35]?

[34]The sample size is the number of partial derivatives and examples in the POLAR data: $N = 3'954'837$

4 Discussion

4.1 Multi-output regression

The weighted MSE loss per epoch for both the validation and training set do not appear, when using 98/1/1% splitting ratios, to widely differ between our data splitting methods for the 98/1/1% splitting ratios (see Figure 2a).

However, this no longer holds for 80/10/10% splitting ratios as we can observe a disparity between the training and validation set for `periodical` split, a gap which does not exist for `random_split` (see Figure 2b). Therefore, we decided to continue with the periodical split as it better showed over-fitting for smaller training sets.

Although everything depends on the initial seed, its impact on the loss seems, from Figure 2c, to be minimal.

4.2 Examples of interest, clusters and cluster intersections

We can also observe from Figure 3a that our methodology misses some GRBs. After some manual inspection, both via the ROOT-CERN tool and in Python, we hypothesize that the time-binning is the general cause. Short GRBs (e.g. fraction of a second) barely appear in our 1 second-binned light curve (photon rate curve). In other words, some GRBs would look like background.

While poor predictions are interesting, they might happen for events other than GRBs. This happens as our model wasn't told whether some examples were part of a GRB or not. For instance, Prof. Nicolas Produit confirmed that the detected red positive cluster from GRB170109A is a solar flare (see Figure 5b, 6a, 6d and 3a).

Since there are positive/negative clusters temporally close to missing data, like the one in Figure 5a, where predictions behave strangely, we may discard clusters or cluster intersections temporally close to missing data[36].

Although many cluster or cluster intersections detect spikes in the light curves, some give questionable time intervals (see Figure 5d, 6b and 7b).

Our methodology leads us to many positive and negative examples of interest, clusters and cluster intersections (see Tables 3, 5, 4 and 6). By comparing the means, we can see a tendency to have more positive elements than negative ones. However, that tendency is

[35]Not using the usual sample standard deviation. We're using the modified standard deviation obtained from the Modified Gaussian Fit on the partial derivatives. See Appendix E.

[36]This is what we did for cluster intersections

weaker between positive and negative clusters or cluster intersections than for examples of interest. Moreover, we can see that our methodology is not really stable to a change of seed[37]. Changing the latter makes the numbers widely vary.

However, these numbers are not everything because clusters obtained for different seeds can overlap. Future research assessing the quality and stability of our extracted time intervals would be interesting. And by quality, we also include categorizing clusters into subgroups, for instance, solar flares, turned-off sensors, GRBs and so on.

In short, further analysis of our extracted time intervals is required.

4.3 Model interpretability

Since our model tries to predict photon rates from diverse measurements, analyzing the relationship between the latter and the former becomes possible. However, due to the black-box nature of neural networks, it is hard to know what happens under the hood and what relationships the model found.

We briefly compare the sample mean of partial derivatives and its standard error[38] for each (un-normalized input, output) pair and observe some trends. However, because we only started looking into model interpretability at the very end of the project, the analysis of our partial derivatives histograms are very shallow[39].

Our method found that increasing the crabarf and the cosmic rates do not always increase the photon rates on average. The behavior depends on the energy bin (see Figure 8a, 8b and both parts of Figure 9). However, our method found that increasing the longitude and altitude increased the photon rates on average for all our energy bins.

In contrast, it found that incrementing, for instance, the revolution number decreases the photon rates on average (see bottom-left of Figure 9). This finding matches the trend we can informally see in Figure 4b. However, as we can see from the figure, the data varies so widely that a more detailed analysis of the local behaviors, going beyond average slopes, is required.

[37]Our methodology is likewise not stable when going from GPU to CPU. We can get different results with CPU vs. GPU due to the operation execution order.

[38]Instead of the sample standard deviation, we use the modified standard deviation obtained from the Modified Gaussian Fit on the partial derivatives, as explained in Appendix E.

[39]We did not provide a formal statistical description of our "trends". It would include hypothesis testing involving null and alternative hypotheses related to the average partial derivatives.

5 Conclusion

We introduced a methodology to extract time intervals containing some GRBs, solar flares and many undetermined events. It consists of building a model of the background and using poor predictions to extract time intervals.

The model is a 4-layer MLP trained on a subset of the POLAR data without 25 known GRBs. It is then applied to the POLAR data with the known GRBs. The resulting residuals, or the resulting pulls, are used to retrieve examples of interest. Aggregating the latter leads to clusters or cluster intersections.

After 200 epochs, we extracted 11 sets of cluster intersections of more than 3 energy bins. The first set based on seed 42 contains 3178 cluster intersections. The rest[40] have 3821.5 ± 538.957 cluster intersections.

Despite the sensitivity in the count of clusters, cluster intersections, or examples of interest to the initial seed, we did not confirm whether there are overlaps across different seeds.

Therefore, future analysis of our extracted time intervals is required, for instance, to categorize them into solar events, GRBs and so on or to assess the clusters' stabilities due to initial seeds. Manual inspection of our clusters by experts can give crucial information telling us what to do next.

A few possible directions are to use something other than the Modified Gaussian Fit, to use different regularization techniques such as dropout, to try batch normalization, to purify the training set of anomalies in a feedback loop to get a better representative of normal behavior and so on. One can also try to use sequential models as our data is a time series. Implementation-wise, future work can also focus on the technological details. For instance, one can improve the GPU utilization and code quality, decrease the memory footprint and even add unit tests.

We also barely scratched the surface of model interpretability since we only started looking into it at the very end of the project. Therefore, the analysis of our partial derivatives histograms is very shallow.

However, our methodology could be a starting point for a future project strongly focused on model interpretability since our model is built based on the relationship between photon rates and diverse measurements.

[40]Based on 10 other seeds, see Table 11

Acknowledgements

Firstly, I'd like to thank Prof. Nicolas Produit for his constant involvement with this project. Hancheng Li and Johannes Hulsman also provided me with insightful suggestions and comments.

Secondly, I'm grateful to Eloi Alonso for his PyTorch technical tips and, together with Vincent Micheli, for their inspiring Deep Reinforcement Learning project (Micheli et al., 2022).

Thirdly, I would like to thank Angel Gaspard-Fauvelle for his participation in the discovery, research, and optimization of errors, as well as in the correction of the introductory text and, alongside Francesca Zucchelli, for proofreading my report.

Finally, I acknowledge the rest of the POLAR research group for their warm welcome and active lunch discussions.

References

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

François Fleuret. 2023. Deep Learning Course. URL `https://fleuret.org/dlc/`. Accessed: 2023-09-22.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. ArXiv:1412.6980 [cs].

Gilles Koziol. 2023. HAGRID in Space. URL `https://cernbox.cern.ch/s/X1OvZ4iY19vewcV`. Master's thesis, Accessed: 2023-09-22.

Vincent Micheli, Eloi Alonso, and François Fleuret. 2022. Transformers are Sample-Efficient World Models.

Solar flare. Solar flare — Wikipedia, the free encyclopedia. URL `https://en.wikipedia.org/wiki/Solar_flare`. Accessed: 2023-10-03.

T. Tieleman and G. Hinton. Lecture 6.5 - RMSProp, COURSERA: Neural networks for machine learning. Technical report, 2012.

Shaolin Xiong, Yuanhao Wang, Zhengheng Li, Jianchao Sun, Yi Zhao, Hancheng Li, and Yue Huang. 2017. Overview of the GRB observation by POLAR. page 640.

A Machine Learning

A.1 Multi-Layer Perceptron

A Multi-Layer Perceptron, also called Fully Connected Neural Network (FCNN), is one of the first deep learning models we may encounter. In short, the model is composed of a successive application of linear and non-linear transformations.

The MLP takes as input a vector $\mathbf{x}$ and spits out a scalar $\hat{y}$ or vector $\hat{\mathbf{y}}$. How can we obtain good predictions close to the target y or $\mathbf{y}$, even for an unseen $\mathbf{x}$?

A.1.1 Forward pass

We can describe the forward propagation of an L-layer MLP as follows:

$$\begin{cases} \mathbf{a}^{[0]} &= \mathbf{x} \\ \mathbf{a}^{[l]} &= \sigma^{[l]}\left(\mathbf{W}^{[l]}\mathbf{a}^{[l-1]} + \mathbf{b}^{[l]}\right), \quad \forall l = 1, \ldots, L \\ \hat{\mathbf{y}} &= \mathbf{a}^{[L]} \end{cases} \tag{13}$$

where:

- $\mathbf{W}^{[l]} \in \mathbb{R}^{n^{[l]} \times n^{[l-1]}}$, $\mathbf{b}^{[l]} \in \mathbb{R}^{n^{[l]}}$ are respectively called the weights and biases of the lth layer.
- $n^{[l]}$ is the number of units in layer l and $n^{[0]}$ is the number of input features.
- $\sigma^{[l]}$ is the [non-linear] activation function of layer l. It can be a component-wise operation. Rectified Linear Unit is a famous activation function.
- For an L-layer MLP, there are $L - 1$ hidden layers.

The weights and biases are the model parameters that we want to tune based on training data such that, given an input vector $\mathbf{x}$, the MLP spits out a good output vector $\hat{\mathbf{y}}$. We measure the quality of the prediction by a loss function $\mathcal{L}$.

A.1.2 Backward pass

The forward pass alone doesn't take us to a local/global minimum of the loss function by taking steps (based on the training data) in the parameter space.

Gradient Descent[41] is an optimization method that allows us to move in the parameter space, hoping to reach the global minimum of the loss function when there's no analytical solution.

[41] It's also called Batch Gradient Descent since we're computing the gradient of the loss where the latter operates over all the training examples

We can briefly summarize the algorithm with:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_\theta \mathcal{L}(\theta^{(t)}) \tag{14}$$

where $\theta^{(t)}$ represents the parameters at iteration t, η is the learning rate (or step size), and $\nabla_\theta \mathcal{L}(\theta_t)$ is the gradient of the loss w.r.t. the parameters.

However, a question remains. How can we compute the gradient $\mathcal{L}(\theta^{(t)})$?

Notation Before diving into the back-propagation, we rewrite our forward pass as follows:

$$\begin{cases} \mathbf{a}^{[0]} &= \mathbf{x} \\ \mathbf{a}^{[l]} &= \sigma^{[l]}\left(\mathbf{s}^{[l]}\right), \quad \mathbf{s}^{[l]} = \mathbf{W}^{[l]}\mathbf{a}^{[l-1]} + \mathbf{b}^{[l]}, \quad \forall l = 1, \dots, L \\ \hat{\mathbf{y}} &= \mathbf{a}^{[L]} \end{cases} \tag{15}$$

Moreover, let's denote by E the per-sample loss.

Back-propagation for MLPs Using the chain rule, we obtain the following:

$$\frac{\partial E}{\partial \mathbf{W}^{[l]}_{i,j}} = \frac{\partial \mathbf{s}^{[l]}_i}{\partial \mathbf{W}^{[l]}_{i,j}} \frac{\partial \mathbf{a}^{[l]}_i}{\partial \mathbf{s}^{[l]}_i} \frac{\partial E}{\partial \mathbf{a}^{[l]}_i} = \frac{\partial \mathbf{s}^{[l]}_i}{\partial \mathbf{W}^{[l]}_{i,j}} \frac{\partial E}{\partial \mathbf{s}^{[l]}_i} = \mathbf{a}^{[l-1]}_j \frac{\partial E}{\partial \mathbf{s}^{[l]}_i} \tag{16}$$

$$\frac{\partial E}{\partial \mathbf{b}^{[l]}_i} = \frac{\partial \mathbf{s}^{[l]}_i}{\partial \mathbf{b}^{[l]}_i} \frac{\partial \mathbf{a}^{[l]}_i}{\partial \mathbf{s}^{[l]}_i} \frac{\partial E}{\partial \mathbf{a}^{[l]}_i} = \frac{\partial \mathbf{s}^{[l]}_i}{\partial \mathbf{b}^{[l]}_i} \frac{\partial E}{\partial \mathbf{s}^{[l]}_i} = \frac{\partial E}{\partial \mathbf{s}^{[l]}_i} \tag{17}$$

with

$$\begin{aligned} &\frac{\partial E}{\partial \mathbf{a}^{[l]}_i} = \sum_k \frac{\partial \mathbf{s}^{[l+1]}_k}{\partial \mathbf{a}^{[l]}_i} \frac{\partial E}{\partial \mathbf{s}^{[l+1]}_k} = \sum_k \mathbf{W}^{[l+1]}_{k,i} \frac{\partial E}{\partial \mathbf{s}^{[l+1]}_k} = \left(\mathbf{W}^{[l+1]}_{\cdot,i}\right)^T \frac{\partial E}{\partial \mathbf{s}^{[l+1]}}, \forall l < L \\ &\frac{\partial E}{\partial \mathbf{a}^{[L]}} = \frac{\partial E}{\partial \hat{\mathbf{y}}}, \quad \text{from the def. of our loss} \\ &\frac{\partial E}{\partial \mathbf{s}^{[l]}_i} = \frac{\partial \mathbf{a}^{[l]}_i}{\partial \mathbf{s}^{[l]}_i} \frac{\partial E}{\partial \mathbf{a}^{[l]}_i} = \sigma^{[l]\prime}(\mathbf{s}^{[l]}_i) \frac{\partial E}{\partial \mathbf{a}^{[l]}_i}, \quad \text{if } \sigma^{[l]} \text{ is component-wise} \end{aligned} \tag{18}$$

We can rewrite our four equations into matrix form as follows:

$$\begin{cases} \frac{\partial E}{\partial \mathbf{W}^{[l]}} &= \frac{\partial E}{\partial \mathbf{s}^{[l]}} \left(\mathbf{a}^{[l-1]}\right)^T, \quad \text{outer product} \\ \frac{\partial E}{\partial \mathbf{b}^{[l]}} &= \frac{\partial E}{\partial \mathbf{s}^{[l]}} \\ \frac{\partial E}{\partial \mathbf{a}^{[l]}} &= \left(\mathbf{W}^{[l+1]}\right)^T \frac{\partial E}{\partial \mathbf{s}^{[l+1]}}, \quad \forall l < L \\ \frac{\partial E}{\partial \mathbf{s}^{[l]}_i} &= \sigma^{[l]\prime}(\mathbf{s}^{[l]}) \circ \frac{\partial E}{\partial \mathbf{a}^{[l]}} \end{cases} \tag{19}$$

where $\frac{\partial E}{\partial \mathbf{a}^{[L]}} = \frac{\partial E}{\partial \hat{\mathbf{y}}}$ comes from the definition of our loss. Back-propagation as its name suggests, propagates backward gradients by applying the chain rule many times.

Finally, by the linearity of partial derivatives:

$$\nabla_\theta \mathcal{L}(\theta^{(t)}) = \nabla_\theta \left(\frac{1}{N} \sum_{i=1}^{N} E(\theta^{(t)}) \right) = \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta E(\theta^{(t)}) \tag{20}$$

which is what we seek.

A.1.3 Two loss functions for regression

Mean Squared Error Let N be the number of training examples, the Mean Squared Error (MSE) of a single-output regression task is the average squared residuals:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2 \tag{21}$$

For a multi-output regression task, it becomes:

$$\mathcal{L}(\theta) = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} (\hat{\mathbf{y}}_{i,j} - \mathbf{y}_{i,j})^2 = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} (\hat{\mathbf{y}}_i(j) - \mathbf{y}_i(j))^2 \tag{22}$$

where j goes over the output dimensions.

Weighted Mean Squared Error We can also weigh residuals differently based on weights[42] $w_{i,j} \in \mathbb{R}_{\geq 0}$ such that $\sum_{i,j} w_{i,j} = 1$.

The weighted MSE loss (WMSE) is:

$$\mathcal{L}(\theta) = \frac{1}{\sum_{i,j} w_{i,j}} \sum_{i,j} w_{i,j} \cdot (\hat{\mathbf{y}}_{i,j} - \mathbf{y}_{i,j})^2 = \frac{1}{\sum_{i,j} w_{i,j}} \sum_{i,j} w_{i,j} \cdot (\hat{\mathbf{y}}_i(j) - \mathbf{y}_i(j))^2 \tag{23}$$

where j goes over the output dimensions.

[42]To not confuse with the model parameters.

A.1.4 Weight decay regularization

Although increasing the capacity of a model by adding more parameters leads to a better training loss, it does not necessarily mean that it would perform well on data it has not seen. A high-capacity model can over-fit the noise in the target values, thus leading to low training errors but high validation errors.

It may be counter-intuitive to think that a model with more parameters would perform worse since setting some parameters to zero would give us our smaller model.

In particular, for polynomial curve fitting, Bishop (2006) explains the following:

"This may seem paradoxical because a polynomial of given order contains all lower order polynomials as special cases. The M = 9 polynomial is therefore capable of generating results at least as good as the M = 3 polynomial. Furthermore, we might suppose that the best predictor of new data would be the function $sin(2\pi x)$ from which the data was generated (and we shall see later that this is indeed the case). We know that a power series expansion of the function $sin(2\pi x)$ contains terms of all orders, so we might expect that results should improve monotonically as we increase M."– on page 8 of 1.1. Example: Polynomial Curve Fitting in Bishop (2006).

Bishop (2006) also explains a way to gain insight into the problem by looking at the magnitude of the parameters. The parameters typically get larger as the model capacity increases (refer to his table of the coefficients 1.1).

Therefore, penalizing the magnitude of the coefficients/weights/parameters may make sense. We can do it by adding additional terms to our loss functions. That makes the selected functional less dependent on data (Fleuret, 2023).

L_2 regularization, weight decay Let N be the number of training examples, the Mean Squared Error (MSE) with L_2 penalty/regularization of a single-output regression task is the average squared residuals plus an additional term:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 + \frac{\lambda}{2}\|\theta\|_2 \tag{24}$$

We wrote $\frac{\lambda}{2}\|\theta\|_2$ instead of $\lambda\|\theta\|_2$ because PyTorch Adam optimizer uses $\lambda \cdot \theta$ in its algorithm.

L_1 regularization The Mean Squared Error (MSE) with L_1 penalty/regularization of a single-output regression task is the average squared residuals plus an additional term:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 + \lambda\|\theta\|_1 \tag{25}$$

Probabilistic view Under the probabilistic interpretation, we want to find the most likely parameters θ given our training data $\mathcal{D}$:

$$\underset{\theta}{\text{argmax}}\, p(\theta|\mathcal{D}) = \underset{\theta}{\text{argmax}}\, \frac{p(\mathcal{D}|\theta)\cdot p(\theta)}{p(\mathcal{D})} = \underset{\theta}{\text{argmax}}\, p(\mathcal{D}|\theta)\cdot p(\theta) \tag{26}$$

where $p(\theta|\mathcal{D})$ is called the posterior, $p(\theta)$ the prior and $p(\mathcal{D})$ the evidence.

The L_2 regularization corresponds to maximizing the posterior distribution $p(\theta|\mathcal{D})$ with a Gaussian prior $p(\theta)$ and the L_1 regularization corresponds to a Laplacian prior $p(\theta)$.

We can derive[43] our MSE Loss with L_2 regularization by assuming the following:

- Examples are i.i.d.
- Gaussian prior $p(\theta) \sim \mathcal{N}(\mathbf{0}, \xi^{-1}\cdot\mathbf{I})$
- Normal distribution of the residuals $\forall i, f_\theta(x_i) - y_i \sim \mathcal{N}(0,\sigma)$. Or equivalently, $\forall i, y_i \sim \mathcal{N}(f_\theta(x_i),\sigma) = p(y_i|\theta, x_i)$

We can then maximize $\log(p(\theta|\mathcal{D}))$ that we can rewrite as follows:

$$\begin{aligned}
\log(p(\theta|\mathcal{D})) &= \log(p(\mathcal{D}|\theta)) + \log(p(\theta)) \\
&= \log(p(\mathcal{D}|\theta)) + \log\left(\frac{1}{Z}e^{-\frac{\xi}{2}\theta^T\theta}\right) \\
&= \log(p(\mathcal{D}|\theta)) - \frac{\xi}{2}\theta^T\theta + Z' \\
&= \sum_{i=1}^{N}\log(p(x_i, y_i|\theta)) - \frac{\xi}{2}\theta^T\theta + Z', \quad \text{i.i.d examples} \\
&= \sum_{i=1}^{N}\log(p(y_i|\theta, x_i)) + \log(p(x_i|\theta)) - \frac{\xi}{2}\theta^T\theta + Z' \\
&= \sum_{i=1}^{N}\log(p(y_i|\theta, x_i)) + \log(p(x_i)) - \frac{\xi}{2}\theta^T\theta + Z' \\
&= \sum_{i=1}^{N}\log(p(y_i|\theta, x_i)) - \frac{\xi}{2}\theta^T\theta + Z''
\end{aligned} \tag{27}$$

[43]We took the derivation from section 2.3 Bias-variance dilemma of Prof. François Fleuret's Deep Learning course handouts/slides (Fleuret, 2023) and changed some notation.

$$\begin{aligned}\log(p(\theta|\mathcal{D})) &= \sum_{i=1}^{N} \log(p(y_i|\theta, x_i)) - \frac{\xi}{2}\theta^T\theta + Z'' \\ &= -\frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - f_\theta(x_i))^2 - \frac{\xi}{2}\theta^T\theta + Z'''\end{aligned} \tag{28}$$

and because maximizing $\log(p(\theta|\mathcal{D}))$ is the same as maximizing $\frac{2\sigma^2}{N}\log(p(\theta|\mathcal{D}))$, we get

$$\begin{aligned}\underset{\theta}{\text{argmax}}\, p(\theta|\mathcal{D}) &= \underset{\theta}{\text{argmin}}\, \frac{1}{N}\sum_{i=1}^{N}(y_i - f_\theta(x_i))^2 + \frac{2\xi\sigma^2}{2N}\theta^T\theta \\ &= \underset{\theta}{\text{argmin}}\, \frac{1}{N}\sum_{i=1}^{N}(y_i - f_\theta(x_i))^2 + \frac{\lambda}{2}\theta^T\theta, \quad \text{where } \lambda = \frac{2\xi\sigma^2}{N}\end{aligned} \tag{29}$$

which is exactly our MSE loss plus the regularization term.

We can observe that strong prior beliefs that our parameters are close to 0 due to high ξ or low standard deviations lead to a stronger L_2 penalty.

A.2 Optimization Algorithms

In mathematical optimization, we seek at least one solution which maximizes or minimizes an objective function. However, the exact formula describing the solution(s) does not always exist, i.e. not always an analytical solution.

In this subsection, we present some gradient-based optimization methods from the plain batch gradient descent to the Adam optimization method to find the best parameters minimizing a given loss function.

These methods update the parameters by using how much the loss varies with respect to the different parameters. Mathematically speaking, they use the partial derivative of the loss with respect to each parameter.

A.2.1 Batch Gradient Descent

Batch Gradient Descent is the same as Gradient Descent and is named as such since it uses the gradient of the loss where the latter operates over all the training examples. It is an optimization method allowing us to move in the parameter space, hoping to reach the global minimum of the loss function.

We can describe the algorithm as the application of the following rule until some stopping criterion (e.g. stagnation/convergence or a number of epochs/iterations[44]):

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_\theta \mathcal{L}(\theta^{(t)}) \tag{30}$$

where $\theta^{(t)}$ represents the parameters at iteration t, η is the learning rate (or step size), and $\nabla_\theta \mathcal{L}(\theta_t)$ is the gradient of the loss with respect to the parameters.

Intuitively, at each iteration, the method will move us in the parameter space a little bit in the direction of the steepest descent.

Note that nothing prevents us from getting stuck in a local optima, going past one (may overshoot the global one[45]) or even performing tiny steps in the parameter space due to flat regions in which the gradients are close to 0. There are alternatives for which the learning rate is decayed with time (adaptive learning rate) and makes overshooting less likely.

How can we obtain such a rule? To derive our Gradient Descent rule, we consider a local approximation of our objective function inspired by the Taylor expansion[46]:

$$\mathcal{L}(\theta^{(t)} + \theta) = \mathcal{L}(\theta^{(t)}) + \nabla \mathcal{L}_\theta(\theta^{(t)})^T(\theta - \theta^{(t)}) + \mathcal{O}\left(\left\|\theta - \theta^{(t)}\right\|^2\right) \tag{31}$$

where we define our local approximation[47] at $\theta^{(t)}$ as:

$$g(\theta) = \hat{\mathcal{L}}(\theta^{(t)} + \theta) = \mathcal{L}(\theta^{(t)}) + \nabla \mathcal{L}_\theta(\theta^{(t)})^T(\theta - \theta^{(t)}) + \frac{1}{2\eta}\left\|\theta - \theta^{(t)}\right\|^2 \tag{32}$$

Minimizing our local approximation $g(\theta)$ w.r.t. θ leads to $\theta^{(t+1)}$:

$$\begin{gathered} \nabla g(\theta) = \nabla \mathcal{L}_\theta(\theta^{(t)}) + \frac{1}{\eta}(\theta^* - \theta^{(t)}) = 0 \\ \iff \\ \theta^{(t+1)} = \theta^* = \theta^{(t)} - \eta \nabla \mathcal{L}(\theta^{(t)}) \end{gathered} \tag{33}$$

A.2.2 Stochastic Gradient Descent

Batch gradient descent exploits vectorization, which helps speed up execution time as operations are optimized under the hood[48]. However, it needs to see the whole training set, which is time-consuming, before taking a little step in the parameter space.

[44] An epoch is an iteration in which we go through the whole training set once

[45] This happens for big learning rates/step sizes

[46] You can find more details at this URL: `https://www.cs.cornell.edu/courses/cs4780/2022sp/notes/Notes11.pdf`

[47] Taken from section 3.5. Gradient-descent of Fleuret (2023)'s Deep Learning course handouts/slides

[48] For instance, Python for loops is known to be extremely slow. Therefore, using them to do matrix multiplications is not a good idea. It is better to exploit NumPy or PyTorch operations on tensors

Stochastic Gradient Descent (SGD) takes the opposite approach by using a single example to update the parameters. However, SGD is very noisy[49] and loses the vectorization advantages.

We can describe the algorithm as the application of the following instructions until some stopping criterion (e.g. number of epochs):

- Randomly shuffle the training set
- For each $(\mathbf{x}_i, \mathbf{y}_i)$ in the shuffled training set:

$$\theta^{(t)} = \theta^{(t)} - \eta \nabla_\theta l(f(\mathbf{x}_i; \theta^{(t)}), \mathbf{y}_i) \tag{34}$$

- $t \leftarrow t + 1$ (move to the next epoch)

where l is the per-sample loss that is related to the loss over all the training examples via a sample mean[50]; $\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} l(f(\mathbf{x}_i; \theta), \mathbf{y}_i) = \frac{1}{N}\sum_{i=1}^{N} l_i(\theta)$.

As before $\theta^{(t)}$ represents the parameters at epoch/iteration t, η is the learning rate (or step size), and $\nabla_\theta l_i(\theta^{(t)})$ is the gradient of the per-sample loss of example i with respect to the parameters.

A.2.3 Mini-Batch Stochastic Gradient Descent

To get some advantages of both Batch Gradient Descent and SGD, we may choose a number of examples in between. This number is the *mini-batch size*.

Mini-batch SGD takes advantage of batch processing where model parameters are not moved again into cache memory for each example:

"Memory transfers are slower than computation. Batch processing cuts down to one copy of the parameters to the cache per batch." – section 4.3 Batch Processing of Fleuret (2023) Deep Learning course handouts/slides.

Moreover, the model is updated more frequently using partial sample means that cut computation a lot while keeping similar performances[51] (see section 5.2 Stochastic Gradient Descent Fleuret (2023)).

And just like SGD, mini-batch SGD can escape local optima due to its stochasticity.

[49] SGD can escape local optima due to its stochastic nature. Although $l_i(\theta^{(t)})$ is an unbiased estimate of the expected risk, it has greater variance than $\mathcal{L}(\theta)$.

[50] We can extend this sample mean to a weighted average leading to the weighted MSE.

[51] Same performances if the partial sample mean is the same as the expected risk (a.k.a. loss)

We can describe the algorithm as the application of the following instructions until some stopping criterion (e.g. number of epochs):

- Randomly shuffle the training set and split it into mini-batches of *mini-batch size* examples.
- For each mini-batch $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i \in I}$:

$$\theta^{(t)} = \theta^{(t)} - \eta \nabla_\theta \mathcal{L}_{\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i \in I}}(\theta^{(t)}) \tag{35}$$

- $t \leftarrow t + 1$ (move to the next epoch)

where $\mathcal{L}_\mathcal{D}$ represents the loss function as if the complete training set was $\mathcal{D}$. So if $|\mathcal{D}| = 1$, we get SGD and if $|\mathcal{D}| = N$, we get Batch Gradient Descent.

A.2.4 Momentum

The aforementioned methods assume an isotropic loss function since they use the same learning rate for each parameter. However, the loss function can be steeper in one dimension than another leading to oscillations in the trajectory of our mini-batch SGD (see section 5.2 Stochastic Gradient Descent Fleuret (2023)).

Momentum is an optimization method that doesn't follow the exact gradient. Instead, the parameters are attracted to the previous direction and the steepest slope direction.

We can describe the algorithm as the application of the following instructions until some stopping criterion (e.g. number of epochs):

- Randomly shuffle the training set and split it into mini-batches of *mini-batch size* examples.
- For each mini-batch $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i \in I}$:

$$\begin{aligned} v^{(t)} &= \gamma v^{(t-1)} + \eta g^{(t)}, \quad g^{(t)} = \nabla_\theta \mathcal{L}_{\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i \in I}}(\theta^{(t)}) \\ \theta^{(t)} &= \theta^{(t)} - v^{(t)} \end{aligned} \tag{36}$$

- $t \leftarrow t + 1$ (move to the next epoch)

where $\mathcal{L}_\mathcal{D}$ represents the loss function as if the complete training set was $\mathcal{D}$. So if $|\mathcal{D}| = 1$, we get SGD with momentum and if $|\mathcal{D}| = N$, we get Batch Gradient Descent with momentum. It's noteworthy to remember that if $\gamma = 1 - \eta$, then we get an *Exponentially Weighted Moving Average* (EWA) of gradients.

The update accelerates if $\eta g^{(t)}$ and $\gamma v^{(t-1)}$ are more or less in the same direction[52]. In other words, the update accelerates when the gradient doesn't change much.

In narrow valleys, vector addition between $\eta g^{(t)}$ and $\gamma v^{(t-1)}$ cancels each other in the narrow directions and accelerates in the others.

We can relate Momentum to Particle Swarm Optimization (PSO) where particles (parameters) continue in the same direction due to inertia but are also attracted to the best global solution so far and their local best solution so far.

Although we can apply PSO to find parameters of a neural network, it is computationally heavy since each PSO particle represents all parameters of a neural network and we have no convergence nor quality guarantees.

A.2.5 RMSProp

Similarly, RMSProp (Tieleman and Hinton) is an optimization method that doesn't follow the exact gradient. Instead, the gradient is rescaled differently for each dimension/coordinate based on statistics over their magnitude. Therefore, the effective learning rate is no longer the same for all parameters.

We can describe the algorithm as the application of the following instructions until some stopping criterion (e.g. number of epochs):

- Randomly shuffle the training set and split it into mini-batches of *mini-batch size* examples.
- For each mini-batch $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i \in I}$ and for each component j:

$$\begin{aligned} v_j^{(t)} &= \gamma v_j^{(t-1)} + (1-\gamma) g_j^{(t)2}, \quad g_j^{(t)} = \left(\nabla_\theta \mathcal{L}_{\{(\mathbf{x}_i,\mathbf{y}_i)\}_{i\in I}}(\theta^{(t)})\right)_j \\ \theta_j^{(t)} &= \theta_j^{(t)} - \frac{\eta}{\sqrt{v_j^{(t)} + \epsilon}} g_j^{(t)} \end{aligned} \tag{37}$$

- $t \leftarrow t+1$ (move to the next epoch)

where $\mathcal{L}_\mathcal{D}$ represents the loss function as if the complete training set was $\mathcal{D}$. It's noteworthy to remember that there is an *Exponentially Weighted Moving Average* (EWA) of the squared gradient components.

The update accelerates in directions where the gradient does not change much and decelerates in the others.

[52]You can graphically think about vector addition where vectors are nearly colinear and point in the same direction

A.2.6 Adam

Combining ideas from Momentum, RMSProp, and bias correction leads to the Adam optimization method (Kingma and Ba, 2017). Therefore, Adam is an optimization method that doesn't follow the exact gradient.

We can describe the algorithm as the application of the following instructions until some stopping criterion (e.g. number of epochs):

- Randomly shuffle the training set and split it into mini-batches of *mini-batch size* examples.
- For each mini-batch $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i\in I}$ and for each component j:

$$\begin{aligned}
m_j^{(t)} &= \beta_1 v_j^{(t-1)} + (1-\beta_1) g_j^{(t)}, \quad g_j^{(t)} = \left(\nabla_\theta \mathcal{L}_{\{(\mathbf{x}_i,\mathbf{y}_i)\}_{i\in I}}(\theta^{(t)})\right)_j \\
v_j^{(t)} &= \beta_2 v_j^{(t-1)} + (1-\beta_2) g_j^{(t)2} \\
\hat{m}_j^{(t)} &= \frac{m_j^{(t)}}{1-\beta_1^t}, \quad \text{bias correction} \\
\hat{v}_j^{(t)} &= \frac{v_j^{(t)}}{1-\beta_2^t}, \quad \text{bias correction} \\
\theta_j^{(t)} &= \theta_j^{(t)} - \frac{\eta}{\sqrt{\hat{v}_j^{(t)}} + \epsilon} \hat{m}_j^{(t)}
\end{aligned} \tag{38}$$

- $t \leftarrow t+1$ (move to the next epoch)

where $\mathcal{L}_\mathcal{D}$ represents the loss function as if the complete training set was $\mathcal{D}$. It's noteworthy to remember that there are two *Exponentially Weighted Moving Averages* (EWAs), one of the gradients and the other of the squared gradient components.

Bias correction is used against poor initial estimates in the Exponentially Weighted Moving Averages. The latter suffer at the beginning when there's not enough data. As the number of epochs increases, the effect of bias correction decreases.

B Data

B.1 POLAR data

The POLAR data originally consisted of $3'956'646$ examples. However, since we always filter out some examples based on two quantities we'll specify later, we will refer to the filtered POLAR data as just POLAR data. Therefore, the POLAR data consists of $N = 3'954'837$ examples, slightly less than before.

B.1.1 Features and targets

We want a model able to predict from different quantities at time t, photon rates for different energy bins at the same time t (Table 9).

Therefore, the POLAR data is restructured with tuples of the form $(\mathbf{x}(t), \mathbf{y}(t)) \in \mathbb{R}^{18} \times \mathbb{R}^{6}$ where they represent at time t, 18 features and 6 target rates (6 energy bins).

Table 9: Feature and target names with some additional comments

Index	Feature name: different quantities/measurements	Comment
0	`(unix_time-1474004181.5460)//5535.4+10`	number of revolutions
1	`glat`	latitude
2	`glon`	longitude
3	`altitude`	
4	`temperature`	
5	`fe_cosmic`	front-end cosmic rates
6	`raz`	
7	`decz`	
8	`rax`	
9	`decx`	
10	`is_orbit_up`	
11	`time_since_saa`	
12	`crabarf`	
13	`sun`	
14	`sun_spot`	
15	`B_r`	
16	`B_theta`	
17	`B_phi`	

Index	Target name: Photon counts/sec [Hz] for different energy bins	Comment
0	`rate[0]`	single bar > 15 keV
1	`rate[1]`	single bar > 20 keV
2	`rate[5]`	any bar > 15 keV
3	`rate[6]`	any bar > 20 keV
4	`rate[10]`	any
5	`rate[12]`	any no post cosmics

The *un-normalized* features $\mathbf{x}(t)$ are first **centered and normalized** before going through the model. We will talk more about it when describing the train/val/test split.

Other quantities not used as input to the model but may be used in the loss function or when filtering examples are the *error rates* `rate_err`. For each photon rate, there's an error rate specifying an error in the measurement. A higher error rate would indicate a higher imprecision.

B.1.2 Filtering

To obtain the POLAR data from the original data, we filter out examples for which $\frac{\texttt{rate[0]}}{\texttt{rate\_err[0]}} \leq 20$ (not using the other (error) rates).

B.2 POLAR data without 25 known GRBs

Because we want to build a model of the background, we must work with a representative filtered dataset. However, the *background* word is not clearly defined. Moreover, due to the lack of labels stating whether an example is normal (i.e. from the background) or abnormal (e.g. from a GRB), the model we construct based on a subset of the POLAR data is therefore not a good model of normal behavior. Anomalies, outliers or abnormal behaviors such as unknown GRBs, solar flares and so on pollute the constructed model.

However, we can still remove the 25 known GRBs from the POLAR data. Although the resulting dataset would be slightly more representative of the background, it stays polluted.

B.2.1 The 25 known GRBs

Researchers from the POLAR collaboration provided us with a list of 55 known GRB trigger times (Xiong et al., 2017) in UTC format, 25 of which happen within the time range of our POLAR data (after converting UTC to Unix time).

Based on these 25 GRB trigger times, we filtered out (removed) data within ± 100 seconds from the GRB trigger times.

The new data will be referred to as the *POLAR data without* 25 *known GRBs*.

B.3 Training, validation and test set for the multi-output regression problem

We build our model based on a subset of the POLAR data without 25 known GRBs. This subset is the training set and appears after splitting our data into a 98% training, 1% validation and 1% test set. Although we later show two different ways for data splitting illustrated in Figure 10, we carried over with the periodical split as it better showed over-fitting via a large gap between its train and validation loss per epoch for 80/10/10% split ratios[53]. We chose it over PyTorch random split since the latter gave abnormally close train and validation loss per epoch for the same 80/10/10% split ratios.

Although our data contains multiple time series, we treat each example (data point) as independent and identically distributed[54].

[53]See Figure 2b from our results.

[54]The i.i.d. assumption does not necessarily hold

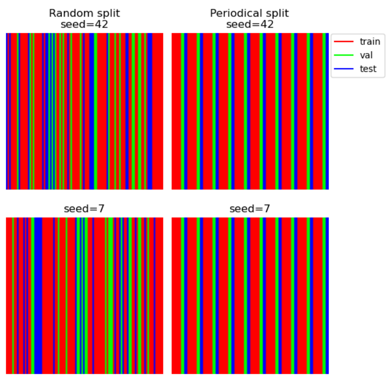


Figure 10: Random split vs. periodical split using $60/20/20\%$ split ratios and two initial seeds on a toy dataset with 100 examples.

B.3.1 Random split

After random shuffling, the dataset is split 98% into training, 1% into validation and 1% into test set. Since this splitting method is seed-dependent, changing the seed would completely change the training, validation, test sets and all subsequent operations.

B.3.2 Periodical split

The dataset is periodically partitioned into train/val/test. In particular, each non-overlapping block of 256 examples[55] is contiguously split, without shuffling, into train/val/test subsets with the 98/1/1% ratios.

Remark: Although periodical split doesn't shuffle the data, the model at training time eats mini-batches of 256 [training] examples[56] randomly shuffled from the whole training set.

B.3.3 Input feature normalization

The features $\mathbf{x}$ are *un-normalized* features as our model doesn't directly take them as input. Our model takes the *normalized* features (the inputs) defined as follows:

$$\tilde{\mathbf{x}}(j) \doteq \frac{\mathbf{x}(j) - \boldsymbol{\mu}_{\text{train}}(j)}{\boldsymbol{\sigma}_{\text{train}}(j)}, \quad \forall j = 0, \dots, 17 \tag{39}$$

where $\boldsymbol{\mu}_{\text{train}}(j), \boldsymbol{\sigma}_{\text{train}}(j) \in \mathbb{R}$ are the sample mean and sample standard deviation of the jth feature, only based on the training set.

[55] $\neq$ mini-batch size

[56] The 256 here is not related to the periodical split

C Training/methodology details

We show in Table 10 our hyperparameters. The asterisk indicates the default hyper-parameters used in PyTorch.

Table 10: Hyper-parameters

Hyper-parameter	Description
Epoch	200
Loss	WMSE
Split type	periodical
Periodicity	256
Split ratios	$98/1/1\%$
Mini-batch size	256
Model type	MLP
n° layers	4
n° hidden units	200
activation function	ReLU

Hyper-parameter	Description
Learning rate* $\eta = \gamma$	10^{-3}
Adam* β_1	0.9
Adam* β_2	0.999
Adam* ϵ	10^{-8}
Adam* λ (weight decay, L_2 reg.)	0
Selection threshold[57] k	5

We also show in Table 11 our run IDs and seeds of 11 trained models with the same hyperparameters. All the runs except the first one have their seeds generated by the RANDOM shell variable with initial seed 42.

Table 11: Run IDs and seeds of 11 trained models with the same hyperparameters

Run n°	0	1	2	3	4	5
Run id	n00r57nc	0307eblk	162pncmd	45dljtwk	5fs1oqli	6wsxi1hf
Run seed	42	17772	26794	1435	24388	11074
Run n°	6	7	8	9	10	
Run id	7sdm9qvf	h444g60r	ntlx23x5	tziecklt	xetzm1dg	
Run seed	32198	5016	25179	767	5153	

The previous table can be used to load/analyze a specific trained model (via PyTorch or Weights & Biases).

[57] Reminder: the definition of positive and negative examples of interest used k.

D Technical specifications

Table 12: System specifications

Class	Description
CPU	AMD Ryzen 5 5600 6-Core Processor
CPU Count	6
GPU	Quadro RTX 4000
GPU Count	1
VRAM	8 GiB
RAM	66 GiB
SSD	Samsung SSD 950 PRO 256GB
OS	Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-75-generic x86_64)

We used Python 3.10.6, PyTorch 2.0.1, Torchvision 0.15.2 and ROOT 6.26/06. We specify the other libraries in our `requirements.txt` from our GitHub repository.

E Modified Gaussian Fit

Instead of computing sample standard deviations by Maximum Likelihood Estimation, we use another technique described by Prof. Nicolas Produit (See Alg 1) motivated by the fact that outliers might heavily influence the usual Gaussian fit by increasing the sample standard deviation.

We call that method *Modified Gaussian Fit* ($\neq$ MGF as in Moment Generating Function).

We used that method to fit different histograms, for instance, histograms of partial derivatives, residuals or pulls. We also used the Modified Gaussian Fit when specifying which examples are interesting.

Another method for threshold selection, with a solid theory in terms of convergence and other properties might be more appropriate.

Algorithm 1 Modified Gaussian Fit on $\mathcal{D} = \{x_i\}_i \subset \mathbb{R}$ with $\epsilon \geq 0$

```
$a \leftarrow -\infty$
$b \leftarrow +\infty$
$\sigma_{\text{old}} \leftarrow \infty$
$\mu \leftarrow \frac{1}{N} \sum_{i=1}^{N} x_i$
$\sigma \leftarrow \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (x_i - \mu)^2}$                ▷ with Bessel correction
while $|\sigma_{\text{old}} - \sigma| > \epsilon$ and $|a - b| > \epsilon$ do        ▷ We used np.isclose
    $a \leftarrow \mu - 3\sigma$
    $b \leftarrow \mu + 3\sigma$
    $\mathcal{D} \leftarrow \{x_i | x_i \in [a, b]\}$
    $\sigma_{\text{old}} \leftarrow \sigma$
    $\sigma \leftarrow \sqrt{\frac{1}{|D|-1} \sum_{x \in \mathcal{D}} (x - \mu)^2}$    ▷ Compute std on the restricted set
end while
if $|a - b| > \epsilon$ then                      ▷ Degenerate case where $x_i$'s are all the same
    return $\mu, 0$
end if
return $\mu, \sigma$                              ▷ $\mu$ never changes and is still the sample mean.
```

Informally, we can verify that if a and b are monotonically increasing/decreasing respectively, then $|\mathcal{D}|$ and σ are both monotonically decreasing. And if σ is monotonically decreasing, then a and b are monotonically increasing/decreasing respectively.

We also notice that σ is lower bounded by 0 and upper-bounded by the initial sample standard deviation[58].

[58]It is impossible to have a larger standard deviation for the same initial data