Robust and Provably Monotonic Networks

Ouail Kitouni[*] **Niklas Nolte**[*] **Mike Williams**
NSF AI Institute for Artificial Intelligence and Fundamental Interactions
Laboratory for Nuclear Science, MIT

Abstract

The Lipschitz constant of the map between the input and output space represented by a neural network is a natural metric for assessing the robustness of the model. We present a new method to constrain the Lipschitz constant of dense deep learning models that can also be generalized to other architectures. The method relies on a simple weight normalization scheme during training that ensures the Lipschitz constant of every layer is below an upper limit specified by the analyst. A simple residual connection can then be used to make the model monotonic in any subset of its inputs, which is useful in scenarios where domain knowledge dictates such dependence. Examples can be found in algorithmic fairness requirements or, as presented here, in the classification of the decays of subatomic particles produced at the CERN Large Hadron Collider. Our normalization is minimally constraining and allows the underlying architecture to maintain higher expressiveness compared to other techniques which aim to either control the Lipschitz constant of the model or ensure its monotonicity. We show how the algorithm was used to train a powerful, robust, and interpretable discriminator for heavy-flavor decays in the LHCb real-time data-processing system.

1 Introduction

The sensor arrays of the LHC experiments produce over 100 TB/s of data, more than a zettabyte per year. After drastic data-reduction performed by custom-built read-out electronics, the annual data volumes are still $O(100)$ exabytes, which cannot be stored indefinitely. Therefore, each experiment processes the data in real-time and decides whether each proton-proton collision event should remain persistent or be discarded permanently, referred to as *triggering* in particle physics. Trigger classification algorithms must be designed to minimize the impact of effects like experimental instabilities that occur during data taking—and deficiencies in simulated training samples. (If we knew all of the physics required to produce perfect training samples, there would be no point in performing the experiment.) The need for increasingly complex discriminators for the LHCb trigger system [1, 2] calls for the use of expressive models which are both robust and interpretable. Here we present an architecture based on a novel weight normalization technique that achieves both of these requirements.

Robustness A natural way of ensuring the robustness of a model is to constrain the Lipschitz constant of the function it represents. To this end, we developed a new architecture whose Lipschitz constant is constrained by design using a novel layer-wise normalization which allows the architecture to be more expressive than the current state-of-the-art with more stable and faster training.

Interpretability An important inductive bias in particle detection at the LHC is the idea that particular collision events are more *interesting* if they are outliers, *e.g.*, possible evidence of a particle produced with a longer-than-expected (given known physics) lifetime would definitely warrant further

Fourth Workshop on Machine Learning and the Physical Sciences (NeurIPS 2021).

[*] Equal contribution

detailed study. The problem is that outliers are often caused by experimental artifacts or imperfections, which are included and labeled as background in training; whereas the set of all possible *interesting* outliers is not possible to construct *a priori*, thus not included in the training process. This problem is immediately solved if *outliers are better* is implemented directly using an expressive monotonic architecture. Some work was done in this regard [12, 15, 14] but most implementations are either not expressive enough or provide no guarantees. We present Monotonic Lipschitz Networks which overcome both of these problems by building an architecture that is monotonic in any subset of the inputs by design, while keeping the constraints minimal such that it still offers significantly better expressiveness compared to current methods.

2 Monotonic Lipschitz Networks

The goal is to develop a neural network architecture representing a scalar-valued function

$$f(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R} \tag{1}$$

that is provably monotonic in any subset of inputs and whose gradient (with respect to its inputs) has a constrained magnitude in any particular direction. In an experimental setting, this latter property is a measure of robustness to small changes in experimental conditions or to small deficiencies in the training samples.

Constraints with respect to a particular L_p metric will be denoted as Lip^p. We start with a model $g(\boldsymbol{x})$ that is Lip^1 with Lipschitz constant λ if $\forall\, \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ (rescaling x_i allows λ directional dependence)

$$|g(\boldsymbol{x}) - g(\boldsymbol{y})| \leq \lambda \|\boldsymbol{x} - \boldsymbol{y}\|_1 \,. \tag{2}$$

The choice of 1-norm is crucial because it allows a well defined maximum directional derivative for each input regardless of the gradient direction. This has the convenient side effect that we can tune the robustness requirement for each input individually.

We can then proceed to make an architecture with built-in monotonicity by adding a term that is linear (or has gradient λ) in each direction in which we want to be monotonic:

$$f(\boldsymbol{x}) = g(\boldsymbol{x}) + \lambda \sum_{i \in I} x_i, \tag{3}$$

where I denotes the set of indices of the input features for which we would like to be monotonic. This residual connection enforces monotonicity:

$$\frac{\partial f}{\partial x_i} = \frac{\partial g}{\partial x_i} + \lambda \geq 0 \quad \forall i \in I \tag{4}$$

Note that the construction presented here only works with Lip^1 constraints as Lip$^{p \neq 1}$ functions introduce dependencies between the partial derivatives. To the best of our knowledge, the only use of residual connections in the literature when trying to learn monotonic functions is in the context of invertible ResNets [7]. Instead, the state-of-the-art approach for learning monotonic functions involves penalizing negative gradients in the loss, then certifying the final model is monotonic, rather than enforcing it in the architecture (*e.g.* in [12].) To be able to represent all monotonic Lip^1 functions with 2λ Lipschitz constant, the construction $g(\boldsymbol{x})$ needs to be a universal approximator of Lip^1 functions. In the next section, we will discuss possible architectures for this task.

3 Lip^1 approximators

Lip^1 constrained models Fully connected networks can be Lipschitz bounded by constraining the matrix norm of all weights [9, 13]. Given the fully connected network with activation σ

$$g(\boldsymbol{x}) = W^m \sigma(W^{m-1} \sigma(...\sigma(W^1 \boldsymbol{x} + b^1)...) + b^{m-1}) + b^m, \tag{5}$$

where W^m is the weight matrix of layer m, $g(\boldsymbol{x})$ satisfies Eq. (2) if

$$\prod_{i=0}^{m} \|W^i\|_1 \leq \lambda \tag{6}$$

and σ has a Lipschitz constant less than or equal to 1. There are multiple ways to enforce Eq. (6). Two possibilities that involve scaling by the operator norm of the weight matrix [9] are:

$$W^i \to W'^i = \lambda^{1/m} \frac{W^i}{\max(1, \|W^i\|_1)} \qquad \text{or} \qquad W^i \to W'^i = \frac{W^i}{\max(1, \lambda^{-1/m} \cdot \|W^i\|_1)} \,. \tag{7}$$

In our studies thus far, the latter variant seems to train slightly better. However, in some cases it might be useful to use the former to avoid the scale imbalance between the neural network's output and the residual connection used to induce monotonicity. In order to satisfy Eq. (6), it is not necessary to divide the entire matrix by its 1-norm. It is sufficient to ensure that the absolute sum over each column is constrained:

$$W^i \to W'^i = W^i \text{diag}\left(\frac{1}{\max\left(1, \sum_j |W^i_{jk}|\right)}\right) \,. \tag{8}$$

This novel normalization scheme tends to give even better training results in practice. While Eq. (8) is not suitable as a general-purpose scheme, *e.g.* it would not work in convolutional networks, its performance in training in our analysis motivates further study of this approach in future work.

In addition, the constraints in Eqs. (7) and (8) can be applied in different ways. For example, one could normalize the weights directly before each call such that the induced gradients are propagated through the network like in [13]. While one could come up with toy examples for which propagating the gradients in this way hurts training, it appears that this approach is what usually is implemented for spectral norm [13] in PyTorch and TensorFlow. Alternatively, the constraint could be applied by projecting any infeasible parameter values back into the set of feasible matrices after each gradient update as in Algorithm 2 of [9].

Preserving expressive power Some Lipschitz network architectures (*e.g.* [13]) tend to over constrain the model in the sense that these architectures cannot fit all functions λ-Lip^1 due to *gradient attenuation*. For many problems this is a rather theoretical issue. However, it becomes a practical problem for the monotonic architecture since it often works on the edges of its constraints, for instance when partial derivatives close to zero are required. The authors of [10] showed that ReLU networks are unable to fit the function $f(x) = |x|$ if the layers are norm-constrained with $\lambda = 1$. The reason lies in fact that ReLU, and most other commonly used activations, do not have unit gradient with respect to the inputs over their entire domain. While element-wise activations like ReLU cannot have unit gradient over the whole domain without being exactly linear, the authors of [4] explore activations that introduce nonlinearities by reordering elements of the input vector. They propose the following activation function:

$$\sigma = \textbf{GroupSort}, \tag{9}$$

which sorts its inputs in chunks (groups) of a fixed size. This operation has gradient 1 with respect to every input and gives architectures constrained with Eq. (6) increased expressive power.

Limitations We are working on improving the architecture as follows. First, common initialization techniques are not optimal for weight normed networks [5]. Simple modifications to weight initialization could improve convergence significantly. Second, it appears from empirical investigation that the networks described in Eq. (6) with GroupSort activation could be universal approximators, as we have yet to find a function that could not be approximated well enough with a deep enough network. A proof for universality is still required and could be developed in the future. Neither of these limitations has any impact on the example application discussed in the following section.

4 Experiment: The LHCb inclusive heavy-flavor Run 3 trigger

The architecture presented here has been developed with a specific purpose in mind: The classification of the decays of heavy-flavor particles produced at the Large Hadron Collider, which are bound states that contain a beauty or charm quark that live long enough to travel an observable distance $\mathcal{O}(1\,\text{cm})$ before decaying. The data set used here is built from simulated proton-proton (pp) collisions in the LHCb detector. Charged particles that survive long enough to traverse the entire detector before decaying are reconstructed and combined pairwise into decay-vertex (DV) candidates.

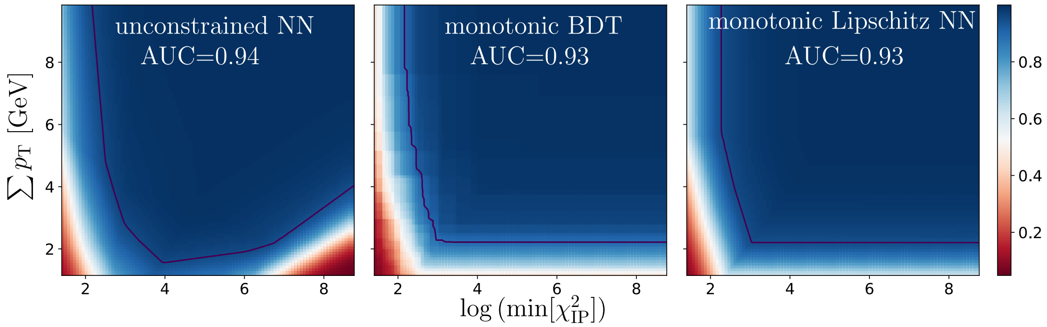

Figure 1: Simplified version of the LHCb inclusive heavy-flavor trigger problem using only 2 inputs, which permits displaying the response everywhere in the feature space; shown here as a heat map with more signal-like (background-like) regions colored blue (red). The dark solid line shows the decision boundary predicted to give the required output bandwidth in Run 3.

The task concerns discriminating between DV candidates corresponding to the decays of heavy-flavor particles versus all other sources of DVs. The signatures of a heavy-flavor DV are substantial separation from the pp collision point, due to the relatively long heavy-flavor particle lifetimes, and sizable transverse momenta, p_{T}, of the component particles, due to the large heavy-flavor particle masses. There are three main sources of background DVs. The first involves DVs formed from particles that originated directly from the pp collision, but where the location of the DV is measured to have non-zero displacement due to resolution effects. These DVs will typically have small displacements and small p_{T}. The second source of background DVs arises due to particles produced in the pp collision interacting with the LHCb detector material, creating new particles at a point in space far from the pp collision point. Such DVs will have even larger displacement than the signal, but again have smaller p_{T}. The third source involves at least one *fake* particle, *i.e.* a particle inferred from detector information that did not actually exist in the event. Since the simplest path through the detector (a straight line) corresponds to the highest possible momentum, DVs involving fake particles can have large p_{T} values.

In the first decision-making stage of the LHCb trigger, a pre-selection is applied to reject most background DVs, followed by a classifier based on the following four DV features: $\sum p_{\mathrm{T}}$, the scalar sum of the p_{T} of the two particles that formed the DV; $\min[\chi^2_{\mathrm{IP}}]$, the smaller of the two increases observed when attempting to instead include each component particle into the pp-collision vertex fit, which is large when the DV is from the pp collision point; the quality of the DV vertex fit; and the spatial distance between the DV and pp-collision locations, relative to their resolutions. The threshold required on the classifier response is fixed by the maximum output bandwidth allowed from the first trigger stage.

Unfortunately, extremely large values of both displacement and momentum are more common for backgrounds than for heavy-flavor signals. For the former, this is easily visualized by considering a simplified problem using only the two most-powerful inputs, $\sum p_{\mathrm{T}}$ and χ^2_{IP}. Figure 1 (left) shows that an unconstrained neural network (NN) learns to reject DVs with increasing larger displacements, corresponding to the lower right corner in the figure. Figure 2 (left) shows that this leads to a dependence of the signal efficiency on the lifetime of the decaying heavy-flavor particle. Larger lifetimes are disfavored since few heavy-flavor particles live more than $\mathcal{O}(10\,\mathrm{ps})$.

The LHCb community is generally interested in studying highly displaced DVs for many physics reasons. Therefore, we want to ensure that a larger displacement corresponds to a more signal-like response. The same goes for DVs with higher $\sum p_{\mathrm{T}}$. Enforcing a monotonic response in both features is thus a desirable property, especially because it also ensures the desired behaviour for data points that are outside the boundaries of the training data. Multiple methods to enforce monotonic behavior in BDTs already exist [6], and Figs. 1 (middle) and 2 (middle) show that this works here. However, the jagged decision boundary can cause problems, *e.g.*, when measuring the heavy-flavor p_{T} spectrum. Figure 1 (right) shows that our novel approach, outlined above, successfully produces a smooth and monotonic response, and Fig. 2 (right) shows that this provides the monotonic lifetime dependence we wanted in the efficiency.

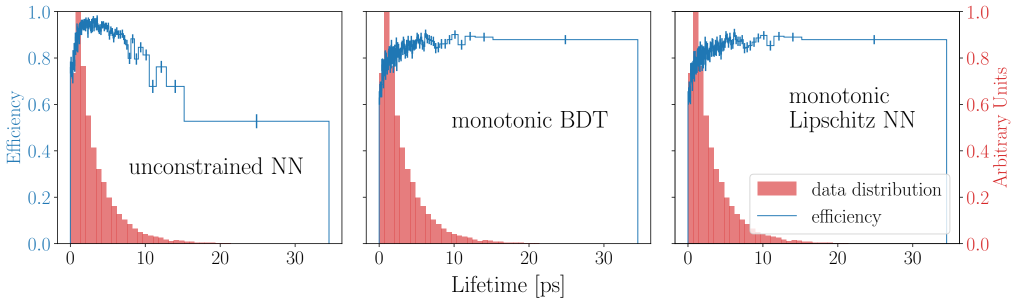

Figure 2: Efficiency of each model shown in Fig. 1 at the expected Run 3 working point versus the proper lifetime of the decaying heavy-flavor particle selected. The monotonic models produce a nearly uniform efficiency above a few ps at the expense of a few percent lifetime-integrated efficiency. Such a trade off is desirable as explained in the text.

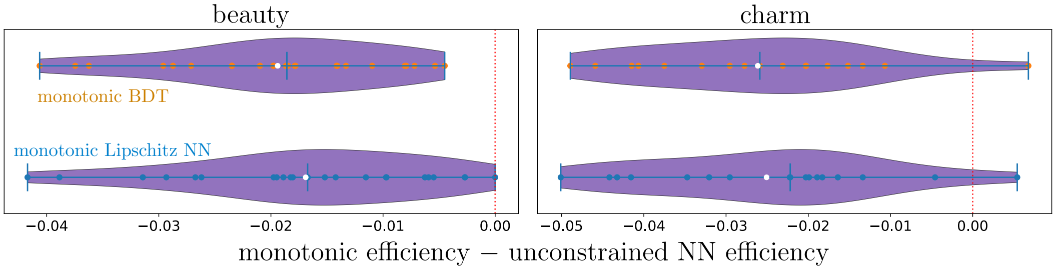

Figure 3: Performance as quantified by the difference in signal efficiency (true positive rate) relative to the unconstrained NN at the expected Run 3 working point for the (left) 24 beauty and (right) 17 charm decays currently being used to benchmark this trigger. Each colored data point shows the change in efficiency for a given decay, while the shaded bands represent the local density of points. The white points show the median values for each set of points.

Not only does our architecture guarantee a monotonic response in whatever features the analyst wants, it is guaranteed to be robust with respect to small changes to the inputs as governed by the constrained Lipschitz constant. Because calibration and resolution effects play a role in obtaining the features during detector operation, robustness is a necessary requirement for any classification performed online. Downstream analyses of these data depend on their stability. Figure 3 shows that the cost in terms of signal efficiency loss of enforcing monotonicity and robustness is small, even under the unrealistic assumption that the training data were, in fact, perfect. Therefore, the actual cost is likely negligible, while the benefits of the guarantees provided is hard to quantify but immediately obvious to the LHCb collaboration. Our algorithm runs in the LHCb trigger software stack and is under consideration to replace Refs. [8, 11] as the primary trigger-selection algorithm used by LHCb in Run 3.

Experiment details The default LHCb model shown here is a 4-input, 3-layer (width 20) network with GroupSort activation (here, all outputs are sorted), $\lambda = 2$, constrained using Eq. (8). Inference times in the fully GPU-based LHCb trigger application [3] are 4 times faster than the default Run 3 trigger BDT, which is based on the model used during data taking in Run 2 [8, 11]. $\mathcal{O}(1000)$ runs with different seeds were performed but the differences were negligible ($\mathcal{O}(0.1\%)$). Due to its guaranteed robustness—and excellent expressiveness even for small networks—similar models are being explored for other uses within the LHCb trigger system for Run 3. Code for the monotonic network implementation can be found at `https://github.com/niklasnolte/MonotOneNorm`.

5 Broader impact

Our implementation of Lipschitz constrained networks is minimally constraining compared to other weight-normed models. This allows the underlying architecture to be more expressive and easier to train while maintaining explicit robustness guarantees. Thanks to the expressive capacity of the architecture, we were able to shrink the number of model parameters to meet the memory and latency requirements of the LHCb trigger system which allows for faster event selection. This translates to higher sensitivity to the elusive physics phenomena we aim to observe. This architecture could also be used in various applications in which robustness is required such as safety-critical environments and those which need protection against adversarial attacks. Monotonicity is a desirable property in various applications where fairness and safety are a concern. There are many scenarios in which models which are not monotonic are unacceptable. For example, in admissions decisions, it might be undesirable for a student to be admitted over another who has a higher grade (ceteris paribus). A potential misuse of this work would be the introduction of monotonicity in problems where it is unethical to do so; however, we believe this would either need to be intentional (nefarious) or an easily preventable blunder.

6 Acknowledgement

The authors would like to thank the LHCb computing and simulation teams for their support and for producing the simulated LHCb samples used to benchmark the performance of RTA software. This work was supported by NSF grants PHY-2019786 (The NSF AI Institute for Artificial Intelligence and Fundamental Interactions, http://iaifi.org/) and OAC-2004645.

References

[1] R. Aaij et al. The LHCb trigger and its performance in 2011. *JINST*, 8:P04022, 2013.

[2] Roel Aaij et al. Performance of the LHCb trigger and full real-time reconstruction in Run 2 of the LHC. *JINST*, 14(LHCb-DP-2019-001):P04013, 2019.

[3] Roel Aaij et al. Allen: A high level trigger on GPUs for LHCb. *Comput. Softw. Big Sci.*, 4(1):7, 2020.

[4] Cem Anil, James Lucas, and Roger Grosse. Sorting out Lipschitz function approximation. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 291–301. PMLR, 09–15 Jun 2019.

[5] Devansh Arpit, Victor Campos, and Yoshua Bengio. How to initialize your network? robust initialization for weightnorm & resnets, 2019.

[6] Charles Auguste, Sean Malory, and Ivan Smirnov. A better method to enforce monotonic constraints in regression and classification trees, 2020.

[7] Jens Behrmann, Will Grathwohl, Ricky T. Q. Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks, 2019.

[8] V. V. Gligorov and M. Williams. Efficient, reliable and fast high-level triggering using a bonsai boosted decision tree. *JINST*, 8:P02013, 2013.

[9] Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J. Cree. Regularisation of neural networks by enforcing lipschitz continuity, 2020.

[10] Todd Huster, Cho-Yu Jason Chiang, and Ritu Chadha. Limitations of the lipschitz constant as a defense against adversarial examples, 2018.

[11] T. Likhomanenko et al. LHCb topological trigger reoptimization. *J. Phys. Conf. Ser.*, 664:082025, Oct 2015.

[12] Xingchao Liu, Xing Han, Na Zhang, and Qiang Liu. Certified monotonic neural networks, 2020.

[13] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral Normalization for Generative Adversarial Networks. *arXiv e-prints*, page arXiv:1802.05957, February 2018.

[14] Joseph Sill. Monotonic networks. In M. Jordan, M. Kearns, and S. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. MIT Press, 1998.

[15] Seungil You, David Ding, Kevin Canini, Jan Pfeifer, and Maya Gupta. Deep lattice networks and partial monotonic functions, 2017.