Robust & Monotonic Neural Networks for Particle Physics & Beyond

Mike Williams

Department of Physics
NSF AI Institute for Artificial Intelligence and Fundamental Interactions
Laboratory for Nuclear Science
Statistics & Data Science Center
MIT

May 10, 2023

Institute for Artificial Intelligence and Fundamental Interactions *(IAIFI /ai fai/ https://iaifi.org)*

Power of AI/ML to process large, rich datasets

First principles and best practices from physics

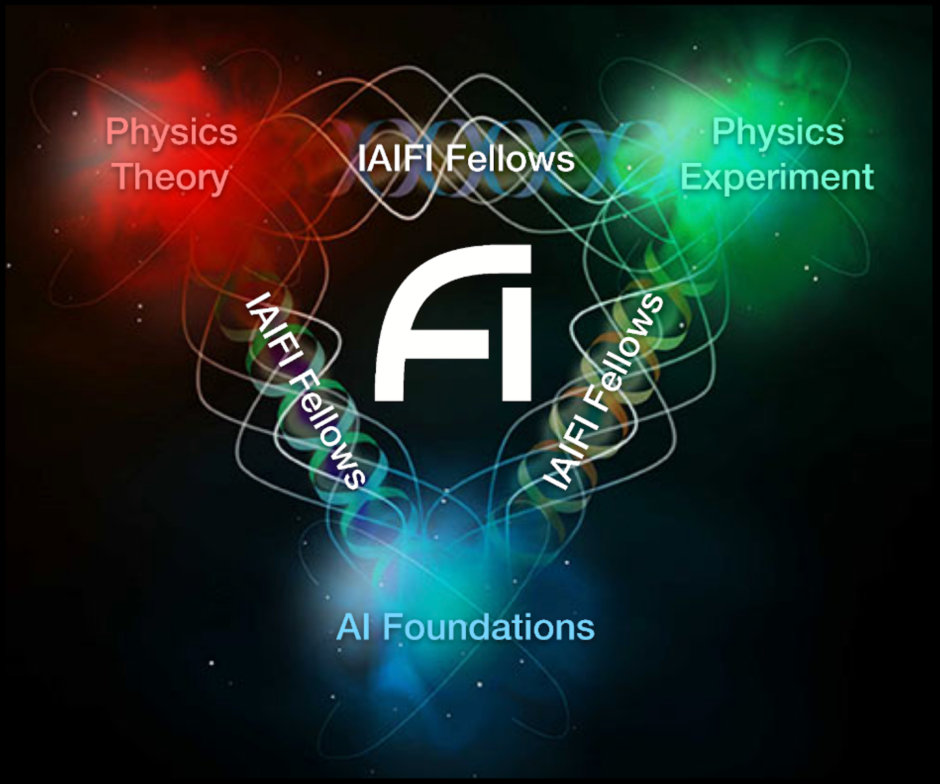
Enable physics discoveries by developing and deploying the next generation of AI technologies
Galvanize AI research innovation by incorporating physics intelligence into artificial intelligence

Shameless Advertisements

IAIFI Postdoctoral Fellowships

Each year we hold a competition to find the best early-career researchers at the intersection of AI+physics.

Fellows are given complete freedom to choose their research direction (as long as it is in the broad area of AI+physics).



This year's competition will open this summer with applications due in the fall. Come join us!

Current Fellows: https://iaifi.org/current-fellows.html

PhD in Physics, Statistics, & Data Science

Created in Fall 2020, this partnership between the Physics Department and Statistics & Data Science Center provides a formal education in statistics and data science in addition to the traditional physics PhD.

Co-chairs: Jesse Thaler & MW

MIT STATISTICS + DATA SCIENCE CENTER

MIT | Physics

https://physics.mit.edu/academic-programs/graduate-students/psds-phd/

Basis for this Talk

This talk is based on work with my post-doc Niklas Nolte and my PhD student Ouail Kitouni:

- Kitouni, Nolte, MW, *Robust and provably monotonic networks*, NeurIPS 2021 Physical Sciences. [2112.00038]

- Kitouni, Nolte, MW, *Expressive monotonic neural networks*, ICLR 2023. https://openreview.net/pdf?id=w2P7fMy_RH

- Kitouni, Nolte, MW, *Finding NEEMo: Geometric fitting using neural estimation of the energy mover's distance*, NeurIPS 2022 Physical Sciences [2209.15624]

- See also: Liu, Kitouni, Nolte, Michaud, Tegmark, MW, *Towards Understanding Grokking: An Effective Theory of Representation Learning*, Oral Highlight at NeurIPS 2022. [2205.10343] (sadly, no time to talk about this today)

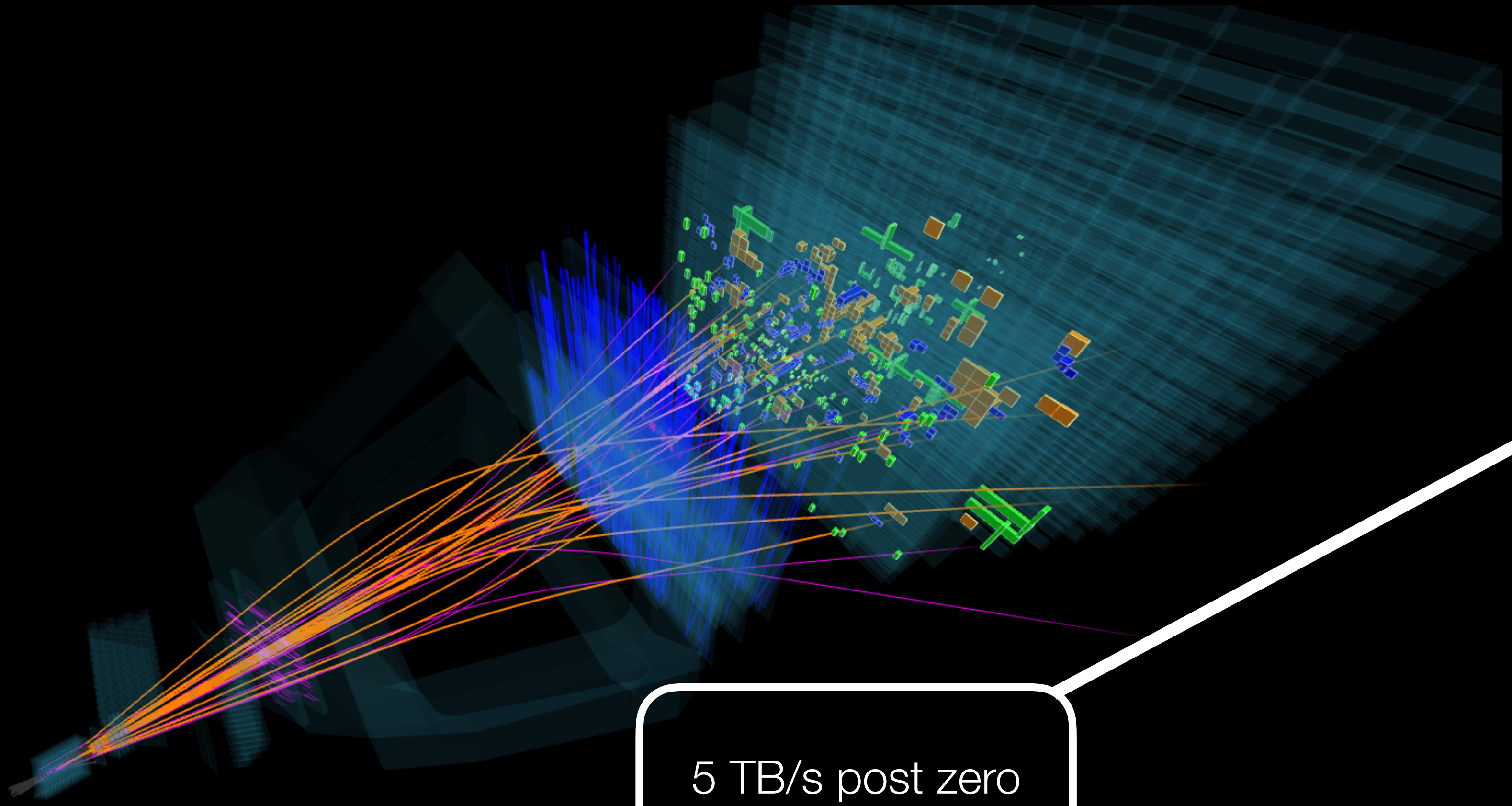
Niklas Nolte
Post-doc
IAIFI & LHCb

Starting at Meta AI Research in June

Ouail Kitouni
Candidate for PhD in Physics, Statistics, and Data Science
IAIFI

Interning this summer at Microsoft Research

Making Decisions @ 40 MHz
(and living with the consequences)



5 TB/s post zero suppression (30 EB / year)

All collisions processed in real time on GPUs to infer what particles were produced and what their properties were. Mixture of traditional and AI algorithms used.

Vast majority of data must be discarded. AI used to make most of these decisions.

Data analyzed later by physicists. Mixture of AI and traditional methods used to produce published results.

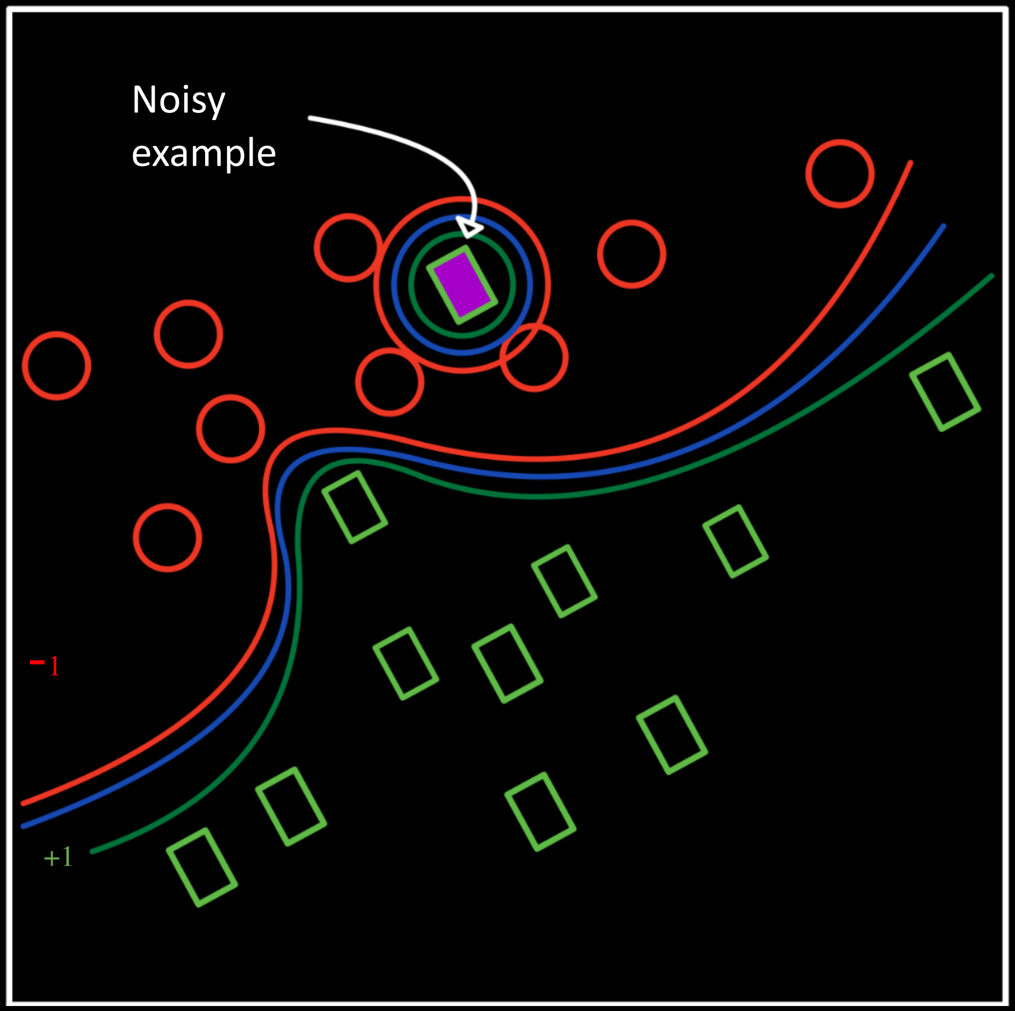
Algorithms used in the real-time environment must be **robust** and **interpretable** — they must account for detector resolution, instability, known unknowns, and must provide formal behavioral/performance **guarantees** to convince us that they are fit for purpose.

In short, we must be able to **trust** them to make important irreversible decisions.

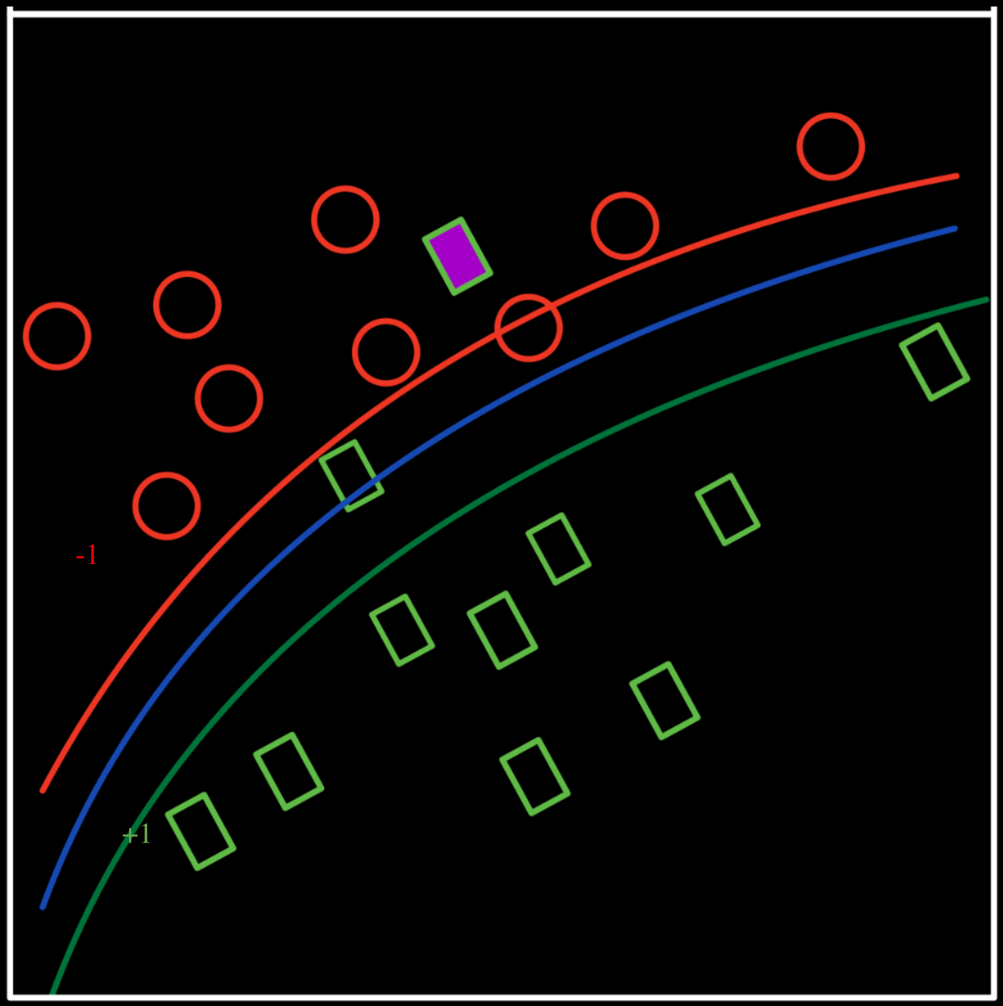
Robust AI

Neural networks can be universal function approximators even in high dimensions, which allows them to solve some incredibly hard problems — but in the real world our ideal solution is NOT found in the set of all functions, but a restricted set of robust ones.

Deep NN overfits on training noise

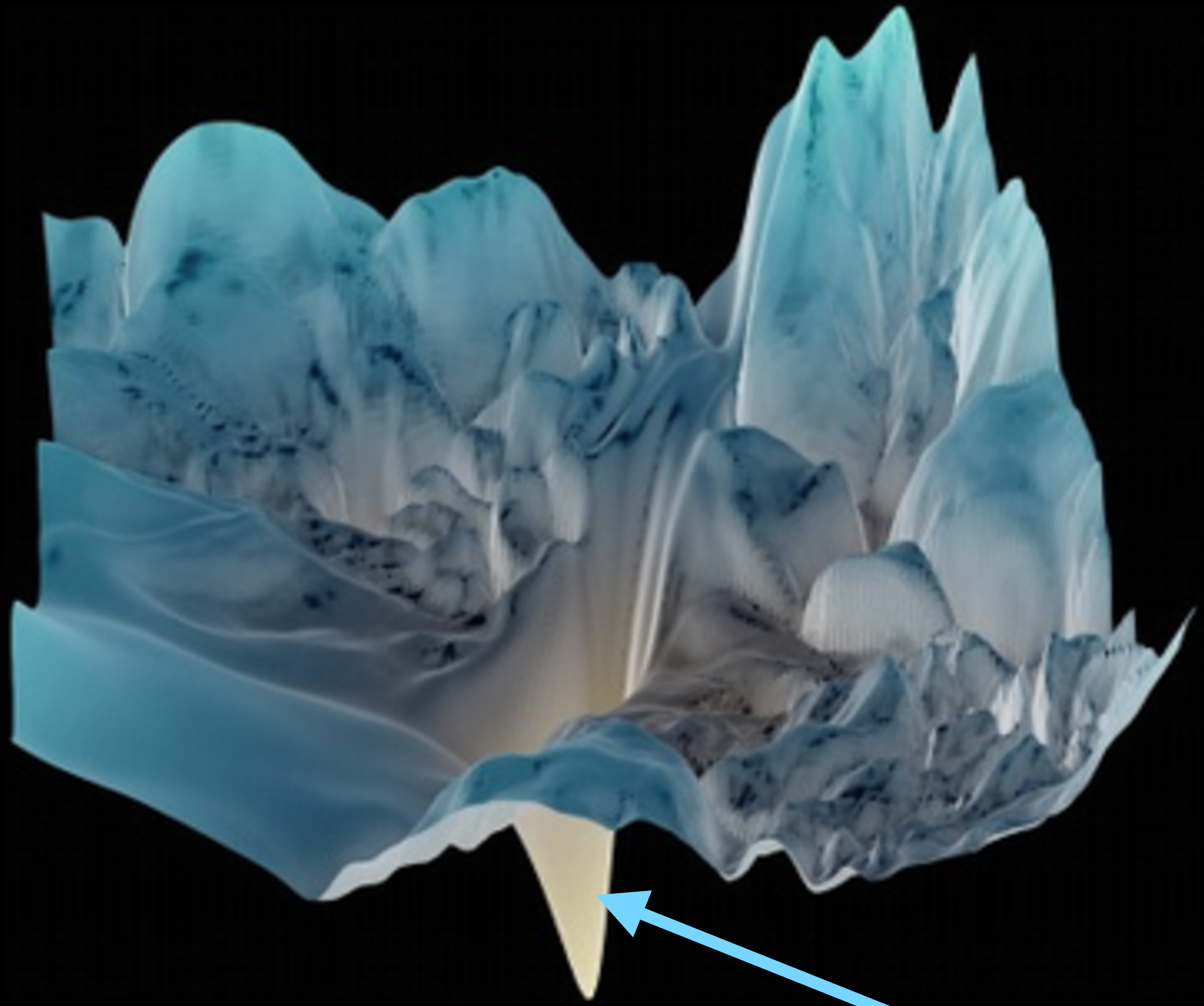


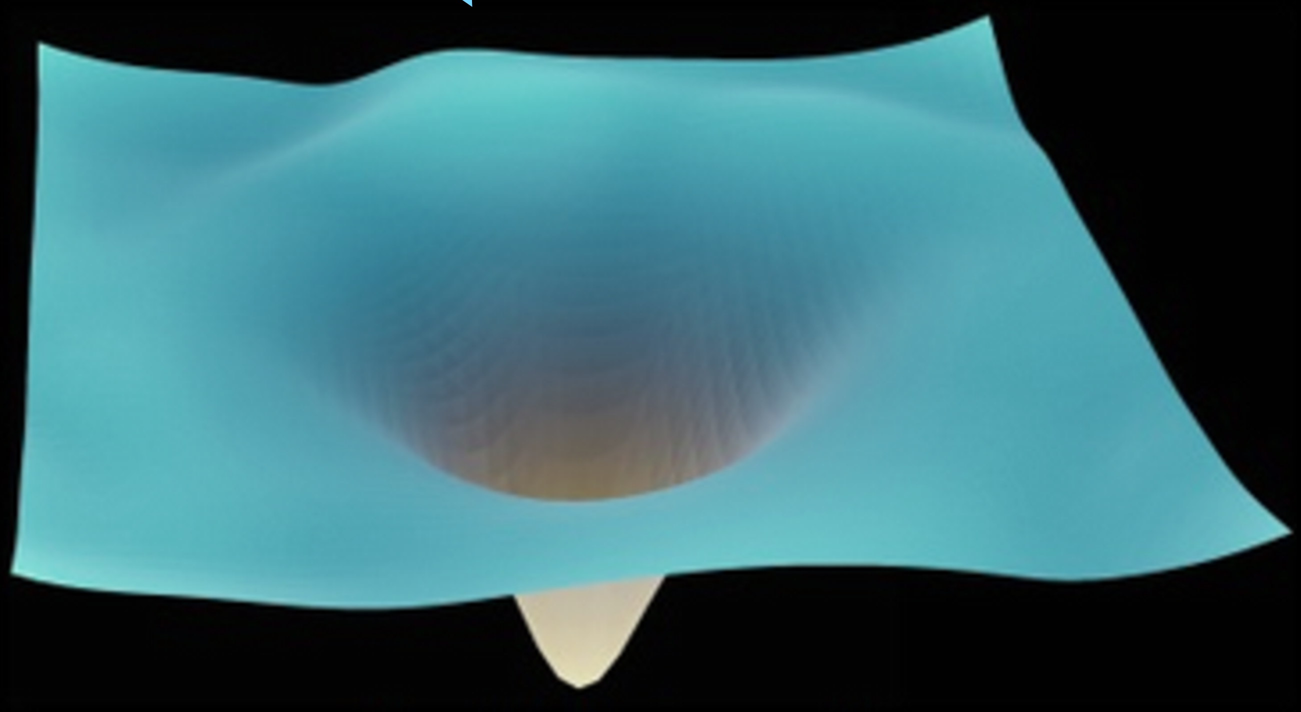
Robust NN respects resolution scale, etc.



Domain experts have a priori knowledge about what scales in each feature direction could contain meaningful information — goal is to input this a priori in a way that provides formal guarantees on the learned model.

want this





don’t want that

Robust AI

One solution to the problem on the previous slides is to create a neural network architecture that guarantees a **bound on the gradient** of the learned function in each direction in feature space.

The following can be proved without any approximations, but it's easier to see quickly by doing a Taylor expansion of the learned function (NN response):

$$|f(\overrightarrow{x} + \overrightarrow{\epsilon}) - f(\overrightarrow{x})| \approx \left| \sum_i \frac{\partial f}{\partial x_i} \epsilon_i \right|$$

$$\leq \max \left[\left| \frac{\partial f}{\partial x_i} \right| \right] \sum_i |\epsilon_i| \equiv ||\overrightarrow{\nabla} f||_\infty ||\overrightarrow{\epsilon}||_1$$

Therefore, how much the function can change is bounded by the maximum absolute gradient value and the 1-norm of the feature-space displacement.

Kitouni, Nolte, MW [NeurIPS 2021, 2112.00038]

Robust AI

Restricting to the set of functions with a bounded Lipschitz constant then also bounds the gradient:

Lipschitz constant, true for all x, epsilon

$$|f(\vec{x}+\vec{\epsilon})-f(\vec{x})| \leq \lambda \|\vec{\epsilon}\|_1 \to \left|\frac{\partial f}{\partial x_i}\right| \leq \lambda$$

Dependence of the desired Lipschitz constant on the location in feature space can easily be accounted for by rescaling the features, where we can also without loss of generality set lambda to unity. Such functions are said to be *1-Lipschitz continuous*.

Domain experts (us for LHCb) specify a priori **inductive bias** on feature scales, i.e. experts define the Lipschitz constants in each feature direction prior to training.

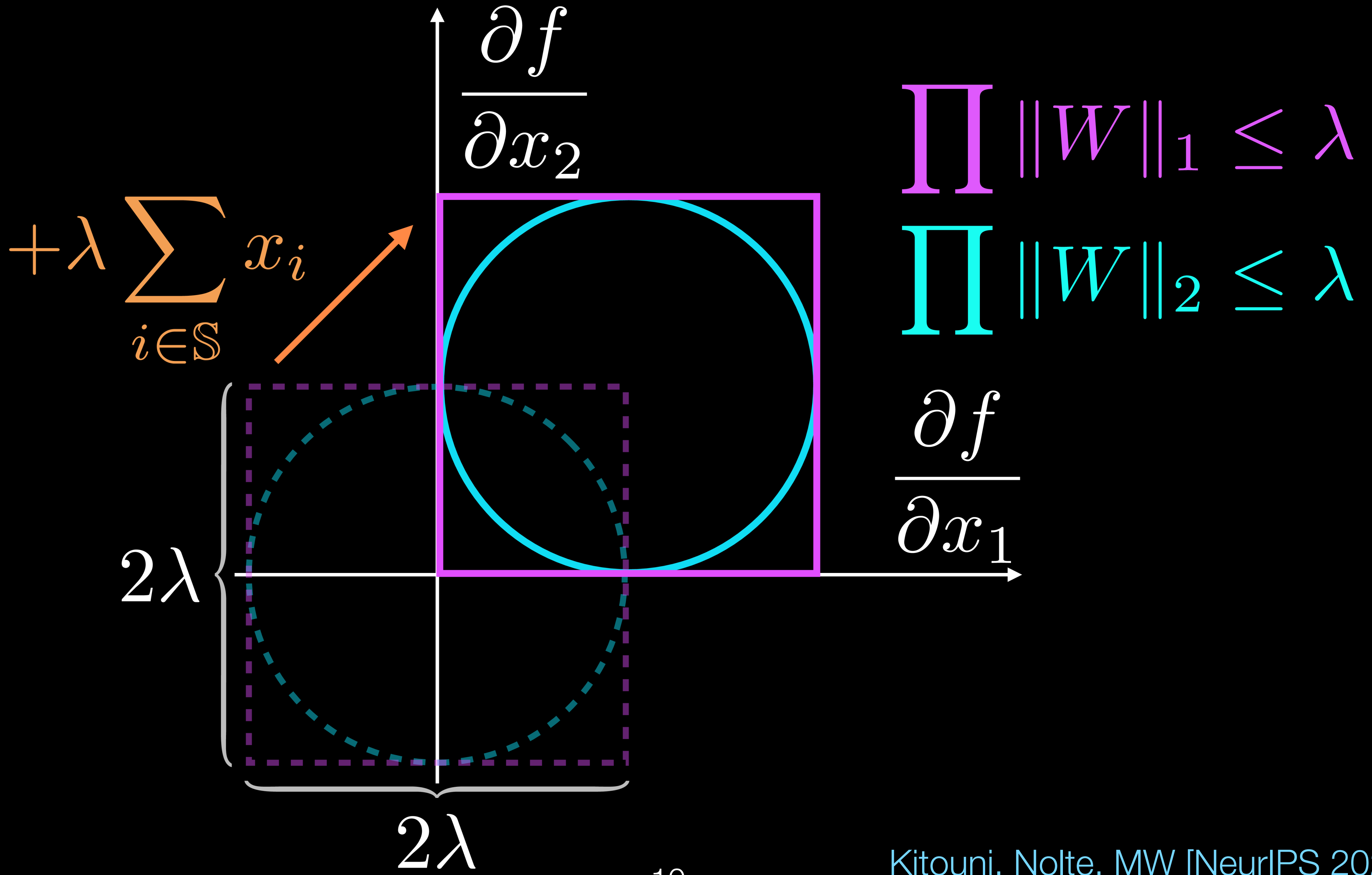
N.b., we can instead bound the changes in the function due to displacement in any direction independently. This is done by replacing the 1 norm with the infinity norm here and on the subsequent slides (which places a bound on the 1-norm of the function gradient, rather than its infinity norm).

Kitouni, Nolte, MW [NeurIPS 2021, 2112.00038]

Monotonic NNs

Furthermore, we can also make the learned function **monotonic** in any feature direction by simply adding a linear function in that direction! For the trigger this lets us guarantee ***outliers are better***.

$$g(\vec{x}) = f(\vec{x}) + \lambda \sum_{i\in\mathcal{M}} x_i \rightarrow \frac{\partial g}{\partial x_i} = \frac{\partial f}{\partial x_i} + \lambda \geq 0 \,\forall\, i \in \mathcal{M}$$



$\frac{\partial f}{\partial x_2}$
$\prod \|W\|_1 \leq \lambda$
$\prod \|W\|_2 \leq \lambda$
$+\lambda \sum_{i\in\mathbb{S}} x_i$
$\frac{\partial f}{\partial x_1}$
2λ
2λ

Kitouni, Nolte, MW [NeurIPS 2021, 2112.00038]

Lipschitz Bounding a Neural Network

To see how we can put a bound on the Lipschitz constant of a NN, consider this toy model NN:

this is normally
another activation

these are normally
more weights

$$f(x,y) = 1 \cdot (1,1) \cdot \overrightarrow{\sigma}\left[\begin{pmatrix} w_{xx} & w_{xy} \\ w_{yx} & w_{yy} \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}\right]$$

$$= \sigma(w_{xx}x + w_{xy}y) + \sigma(w_{yx}x + w_{yy}y)$$

The partial derivatives of this function are then simply: $$\frac{\partial f}{\partial x} = w_{xx}\sigma' + w_{yx}\sigma'$$

Kitouni, Nolte, MW [NeurIPS 2021, 2112.00038]

Lipschitz Bounding a Neural Network

The gradient elements are thus bounded as follows (assuming the activation function has a Lipschitz constant itself of at most unity, which most common ones do):

$$\left|\frac{\partial f}{\partial x}\right| = \left|w_{xx}\sigma' + w_{yx}\sigma'\right| \leq |w_{xx} + w_{yx}| \leq |w_{xx}| + |w_{yx}| \leq \|w\|_1$$

where the 1-norm of a matrix is defined as $$\|w\|_1 \equiv \max_{\text{cols}} \left[\sum_{\text{rows}} |w_{rc}|\right]$$

Thus, the standard approach (for lambda=1) is to rescale the weights of each layer as follows:

$$w^i \rightarrow \frac{w^i}{\max[1, \|w^i\|_1]}$$

However, as can easily be seen above, rescaling all elements is not required, we can get away with column-wise rescaling, ***which trains better***

$$w^i \rightarrow w^i \mathrm{diag}\left(\frac{1}{\max[1, \sum_r |w^i_{rc}|]}\right)$$

Kitouni, Nolte, MW [NeurIPS 2021, 2112.00038]

Lipschitz Bounding a Neural Network

To summarize, we can easily formally enforce a bound on the partial derivatives of the learned function (the neural network response) by norming the weight matrices using

$$w^i \rightarrow \frac{w^i}{\max[1, \|w^i\|_1]} \quad \text{—OR—} \quad w^i \rightarrow w^i \mathrm{diag}\left(\frac{1}{\max[1, \sum_r |w^i_{rc}|} \right)$$

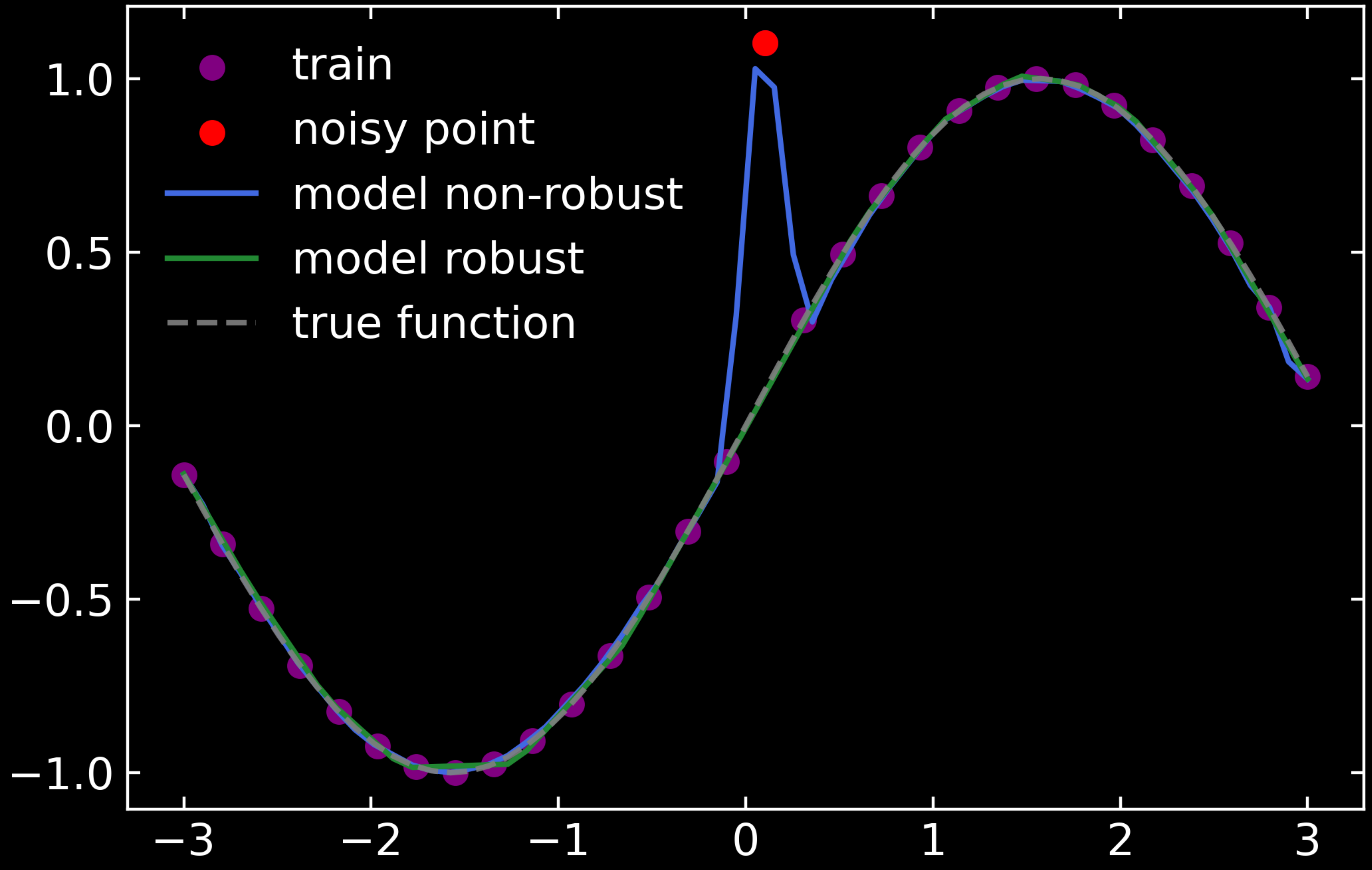
For a CNN, only the left option should be used (the right one breaks translational equivariance); however, for most use cases the right option trains better (since it touches the weights less often).

This only needs to be done during training, as the learned model can be exported with the weight matrices properly normed — hence, on the inference side, the fact that this is a Lipschitz function does not even need to be known (nothing special is required to run these networks).

Kitouni, Nolte, MW [NeurIPS 2021, 2112.00038]

Toy Example

A simple demonstration for 1-d regression with one extremely noisy outlier. The unconstrained NN will go through all points if given enough capacity, whereas our robust NN is smooth (to the degree specified a priori by the Lipschitz constant).



https://github.com/niklasnolte/MonotoneNorm

Saturating the Bound & Expressiveness

Enforcing the gradient bound is not sufficient for monotonic classifiers, which by necessity will need to take on a constant value in some one-class dominated regions. This requires:

$$g(\vec{x}) = f(\vec{x}) + \lambda \sum_{i \in \mathscr{M}} x_i \rightarrow \frac{\partial g}{\partial x_i} = \frac{\partial f}{\partial x_i} + \lambda \geq 0 \, \forall \, i \in \mathscr{M}$$

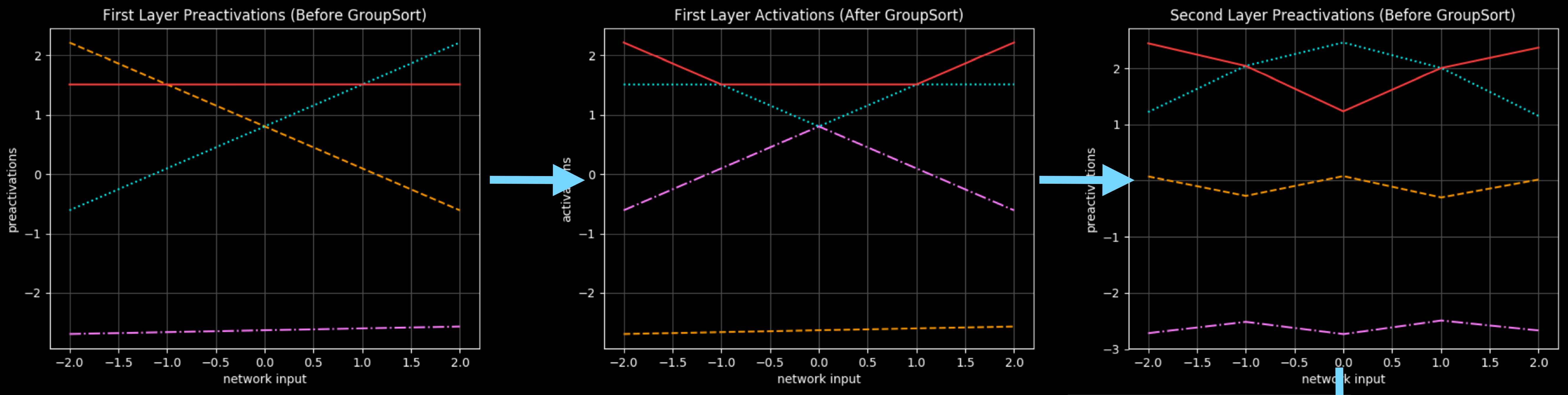
$$g(\vec{x}) = \text{constant} \rightarrow \frac{\partial f}{\partial x_i} = -\lambda$$

More generally, for our NN to be a universal approximator of all Lipschitz functions, we must be able to ***saturate the gradient bound*** at all x.

If you go back to the toy-model NN derivation, you can easily see that we need an activation function whose gradient is unity for all x — but of course any activation function must be non-linear, so we need a non-linear function whose derivative is always 1???

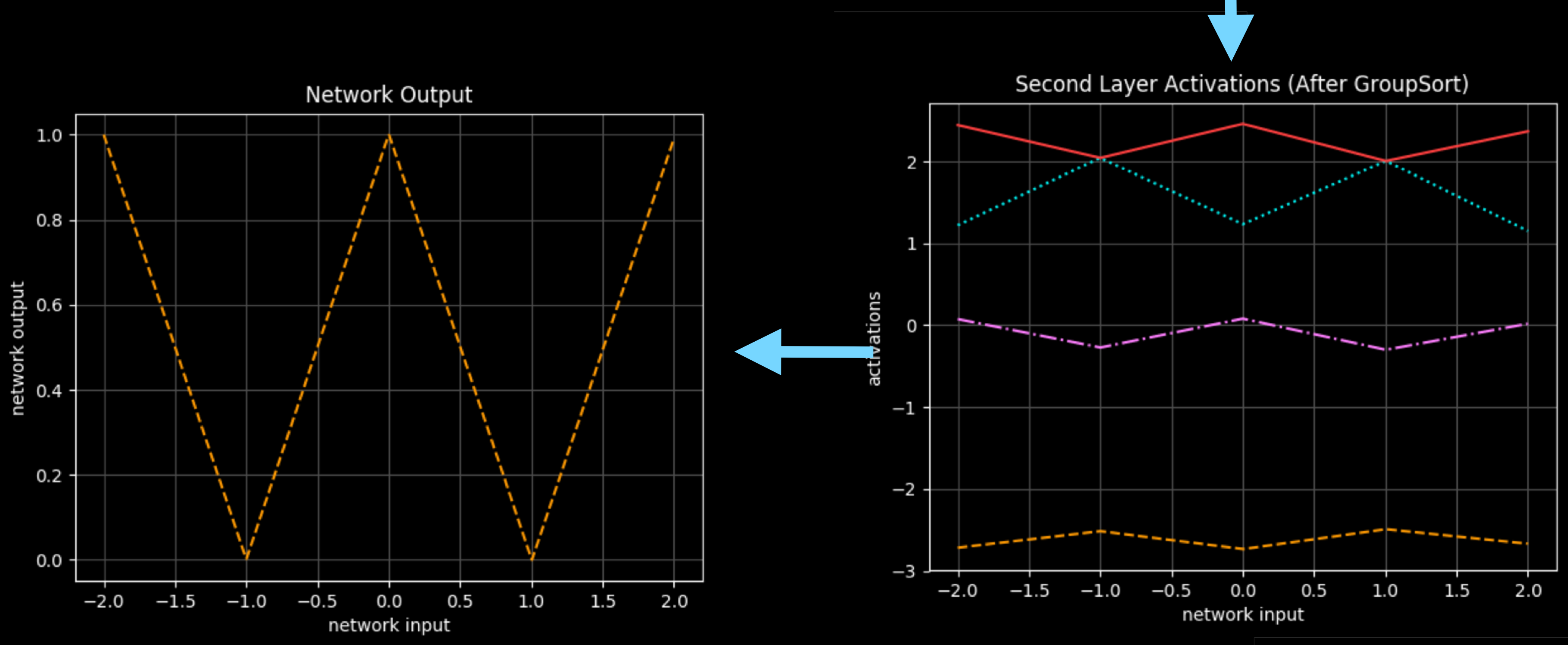
Saturating the Bound & Expressiveness

We can define a non-linear activation function with gradient one everywhere by using a ***non-element-wise function***! In this case, GroupSort (sorting chunks of the element vector), whose vector elements are just the original ones rearranged — activations do NOT need to be scalars!

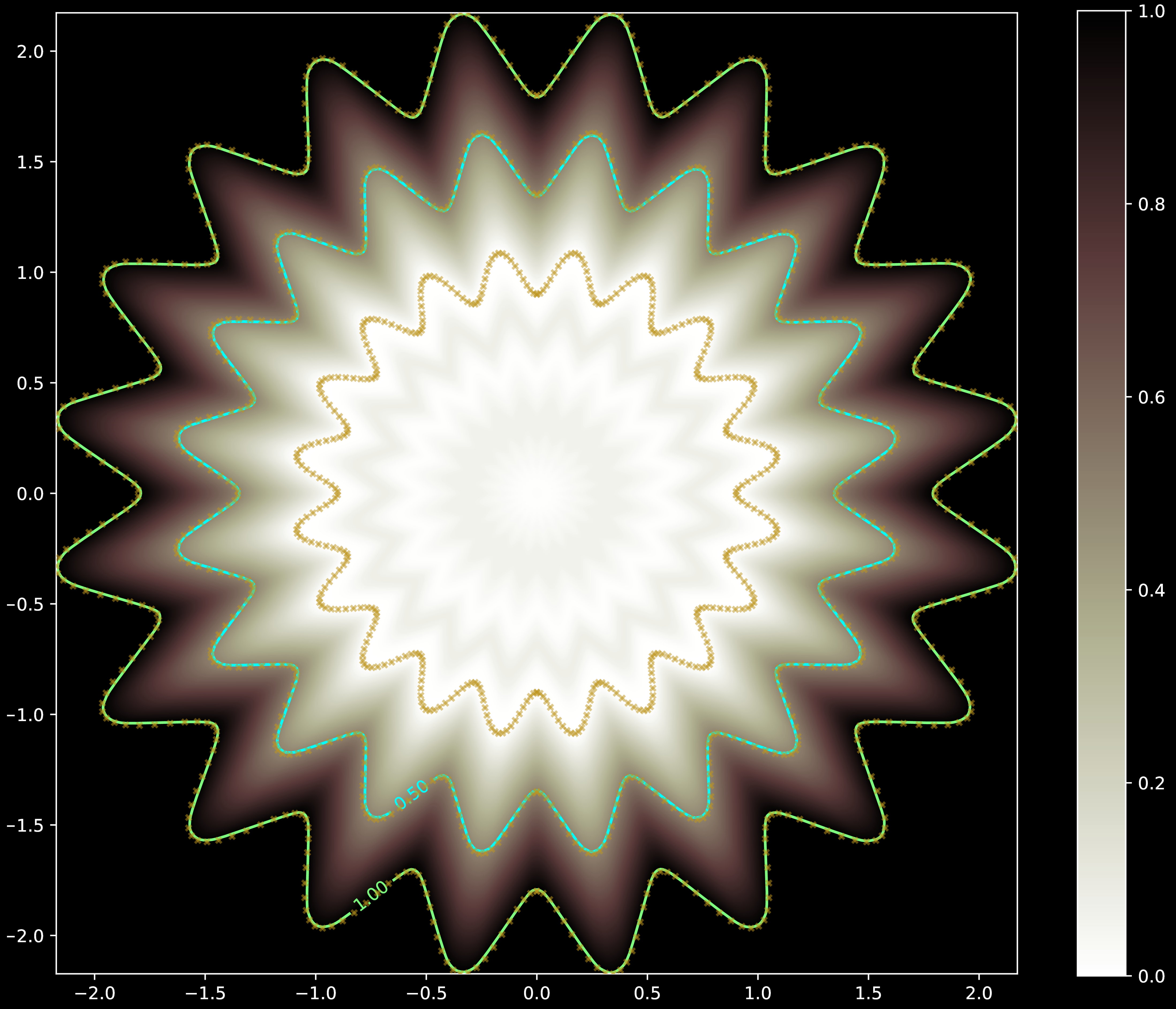


Anil+ prove that GroupSort is a universal approx.

Added bonus: builds up complicated shapes with few elements resulting in high expressivity for tiny networks!



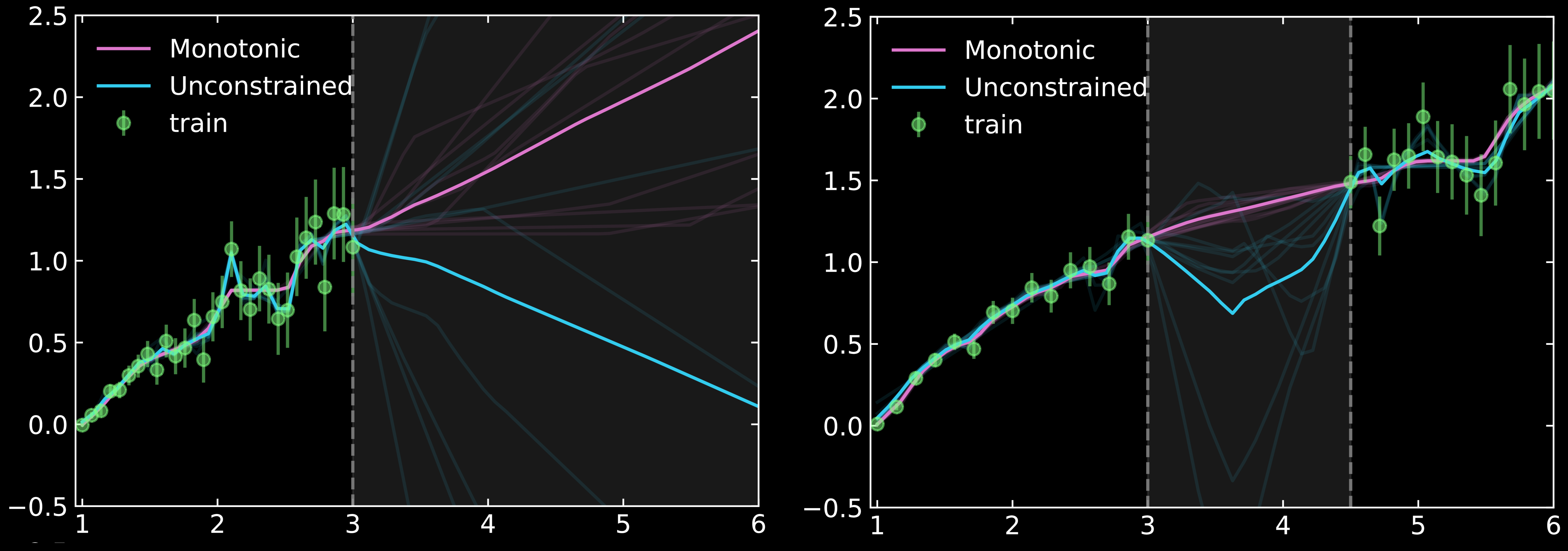
Anil, Lucas, Grosse [1811.05381]



Kitouni, Nolte, MW [NeurIPS 2021, 2112.00038]

Toy Monotonic Example

Learned function is guaranteed to be monotonic even where there is no training data, for both extrapolation beyond the domain of the training data and interpolation through an empty region.



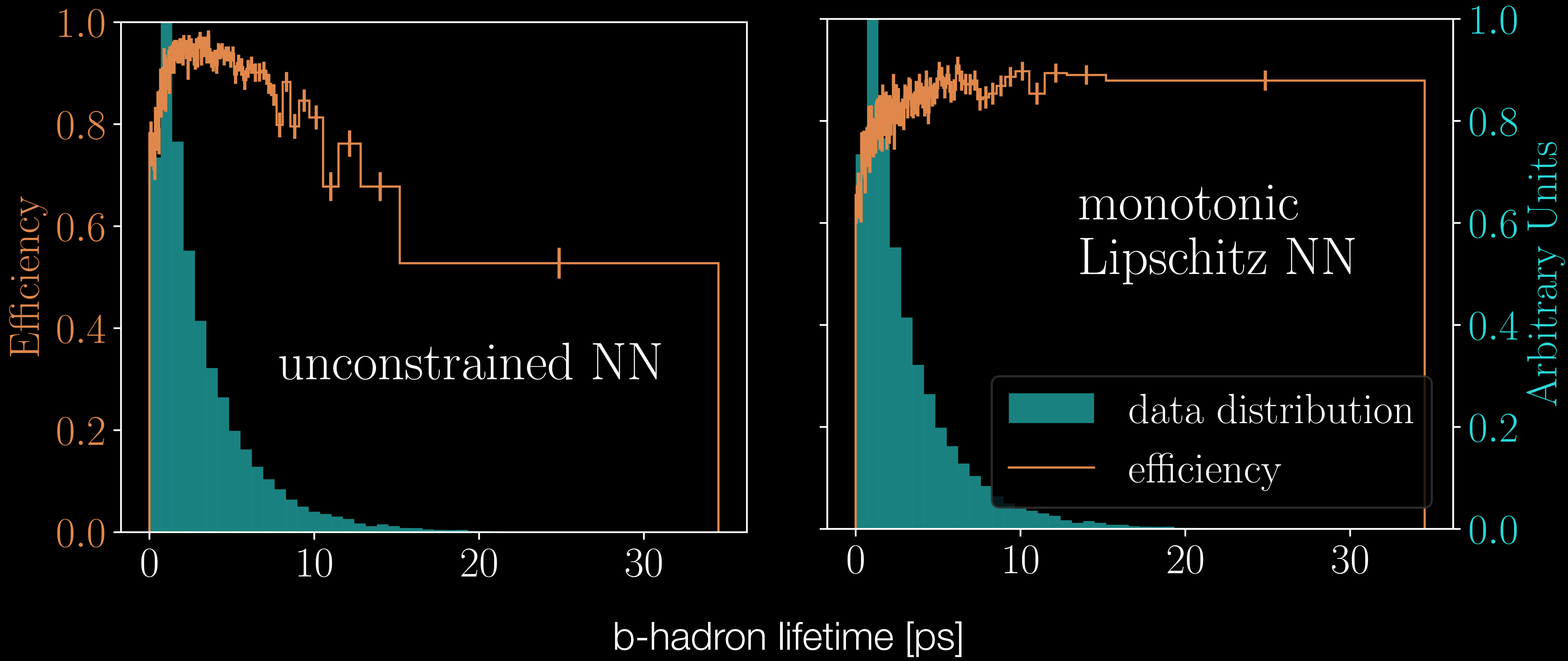
The light lines are regressions done with different seeds, the dark lines are the averages over the seeds. The gray regions do not contain any training data.

Kitouni, Nolte, MW [NeurIPS 2021, 2112.00038]

LHCb Inclusive Heavy Flavor

Our NNs have been adopted for the primary trigger selections at LHCb in Run 3. These inclusively look for secondary vertices consistent with heavy-flavor decays.

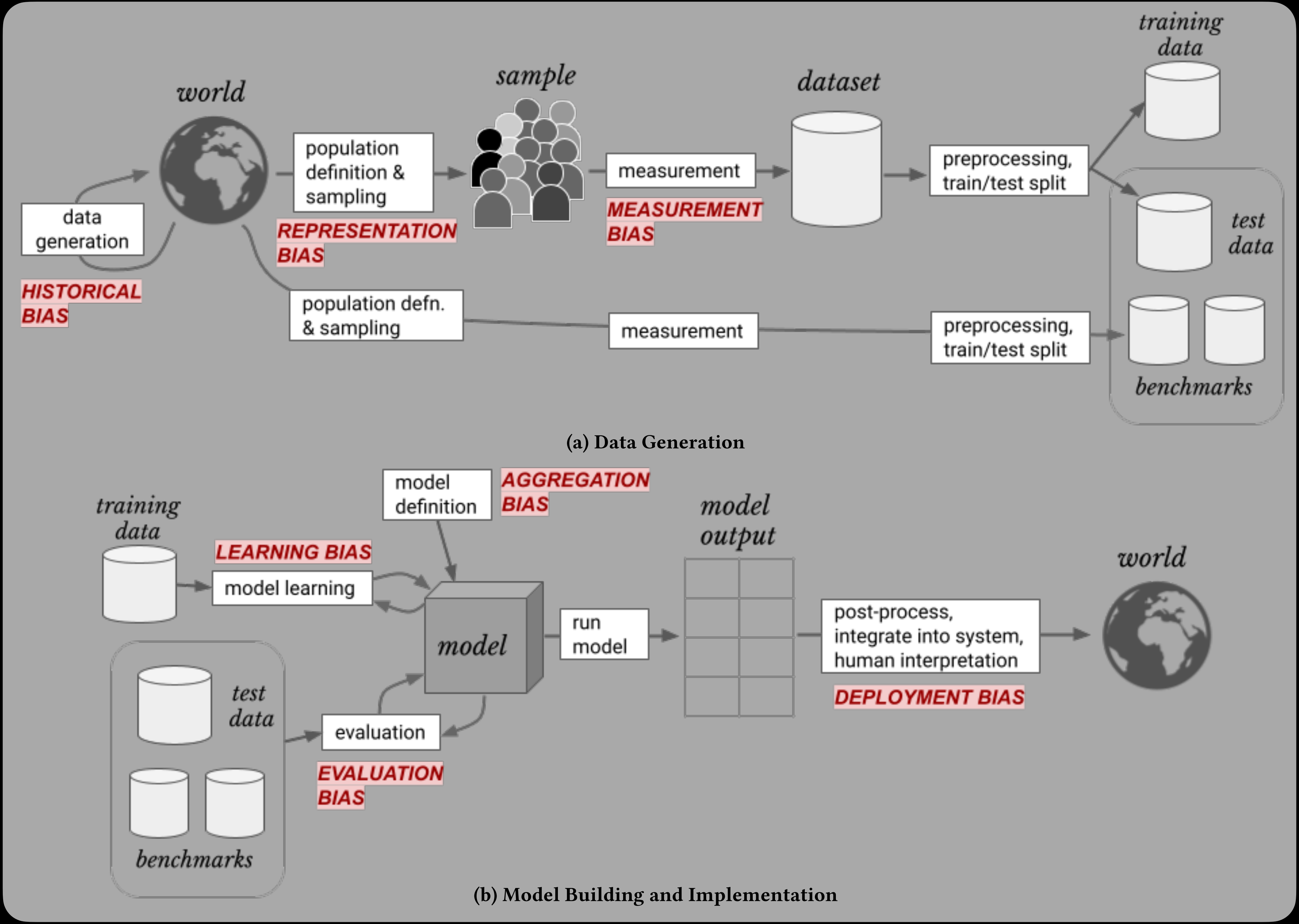
There are many input features, all of which are Lipschitz bounded based on domain knowledge. In addition, features related to pT and lifetime are required to be monotonic (increasing for signal).



Kitouni, Nolte, MW [NeurIPS 2021, 2112.00038]

AI Ethics/Fairness

Lipschitz bounds and monotonicity can more generally alleviate biases in AI/ML solutions, which has direct application in the areas of AI ethics and AI fairness.



(a) Data Generation

(b) Model Building and Implementation

Suresh, Guttag [1901.10002]

Robust & Monotonic AI Applications

We applied our LHC technology out of the box to various benchmark problems where some features are desired to be monotonic, and we beat state-of-the-art models everywhere — with tiny networks!

Kitouni, Nolte, MW [ICLR 2023]

COMPAS

Method	Parameters	$\upuparrows$ Test Acc
Certified	23112	$(68.8 \pm 0.2)\%$
LMN	**37**	$(\mathbf{69.3} \pm \mathbf{0.1})\%$

BlogFeedback

Method	Parameters	$\downdownarrows$ RMSE
Certified	8492	$.158 \pm .001$
LMN	**2225**	$\mathbf{.160} \pm \mathbf{.001}$
LMN mini	**177**	$\mathbf{.155} \pm \mathbf{.001}$

LoanDefaulter

Method	Parameters	$\upuparrows$ Test Acc
Certified	8502	$(65.2 \pm 0.1)\%$
LMN	**753**	$(\mathbf{65.44} \pm \mathbf{0.03})\%$
LMN mini	**69**	$(\mathbf{65.28} \pm \mathbf{0.01})\%$

ChestXRay

Method	Parameters	$\upuparrows$ Test Acc
Certified	12792	$(62.3 \pm 0.2)\%$
Certified E-E	12792	$(66.3 \pm 1.0)\%$
LMN	**1043**	$(\mathbf{67.6} \pm \mathbf{0.6})\%$
LMN E-E	**1043**	$(\mathbf{70.0} \pm \mathbf{1.4})\%$

Heart Disease

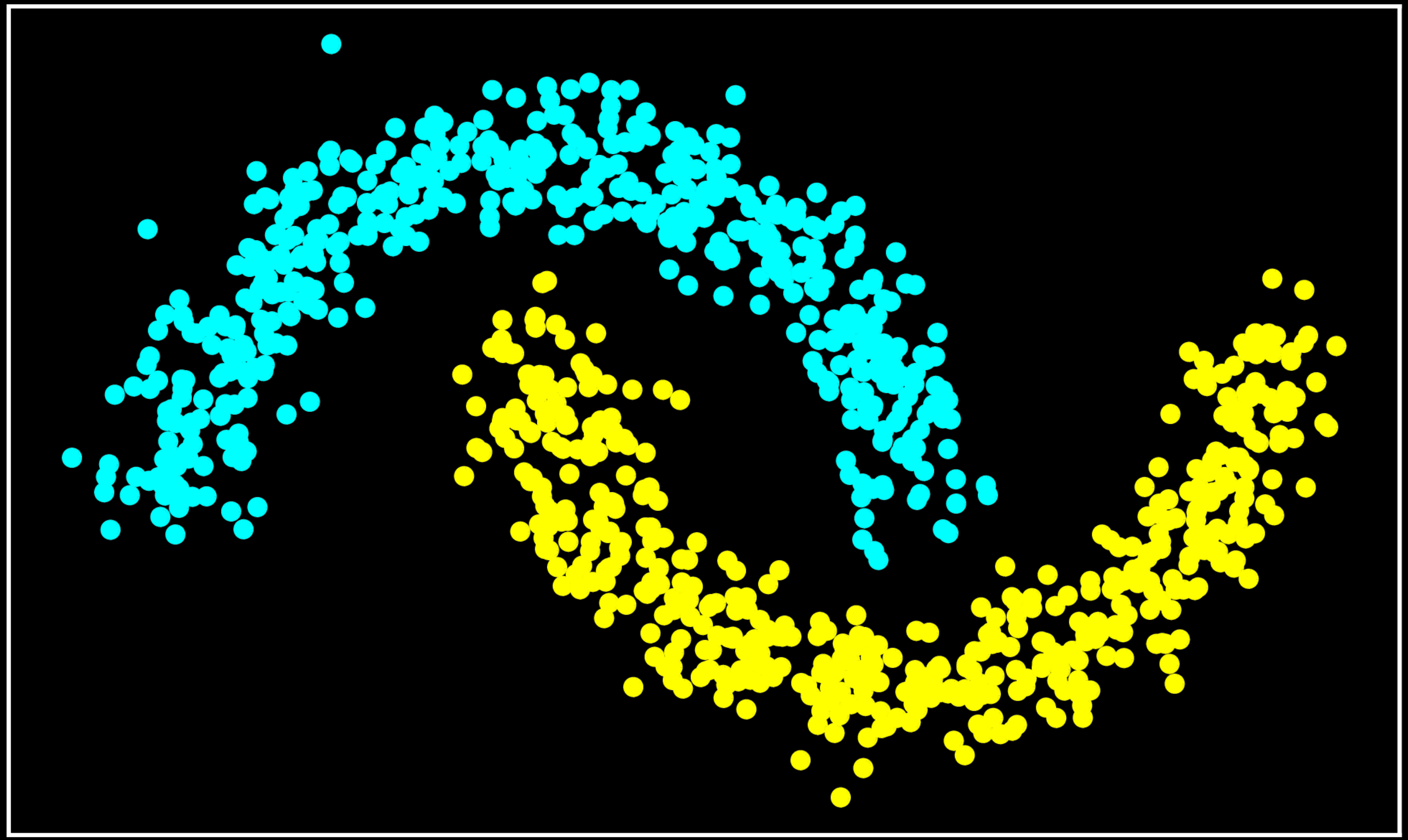
Method	$\upuparrows$ Test Acc
COMET	$(86 \pm 3)\%$
LMN	$(\mathbf{89.6} \pm \mathbf{1.9})\%$

Auto MPG

Method	$\downdownarrows$ MSE
COMET	$(8.81 \pm 1.81)\%$
LMN	$(\mathbf{7.58} \pm \mathbf{1.2})\%$

Optimal Loss Function?

Below is an example of a 2-class problem (2 moons) where the classes are separated. Clearly there exists a classifier F that will give 100% accuracy for this problem.



Since F exists, there must also exist a 1-Lipschitz classifier that also gives 100% accuracy, namely F divided by its Lipschitz constant (whatever that is).

Kitouni, Nolte, MW [NeurIPS 2021, 2112.00038]

Optimal Loss Function?

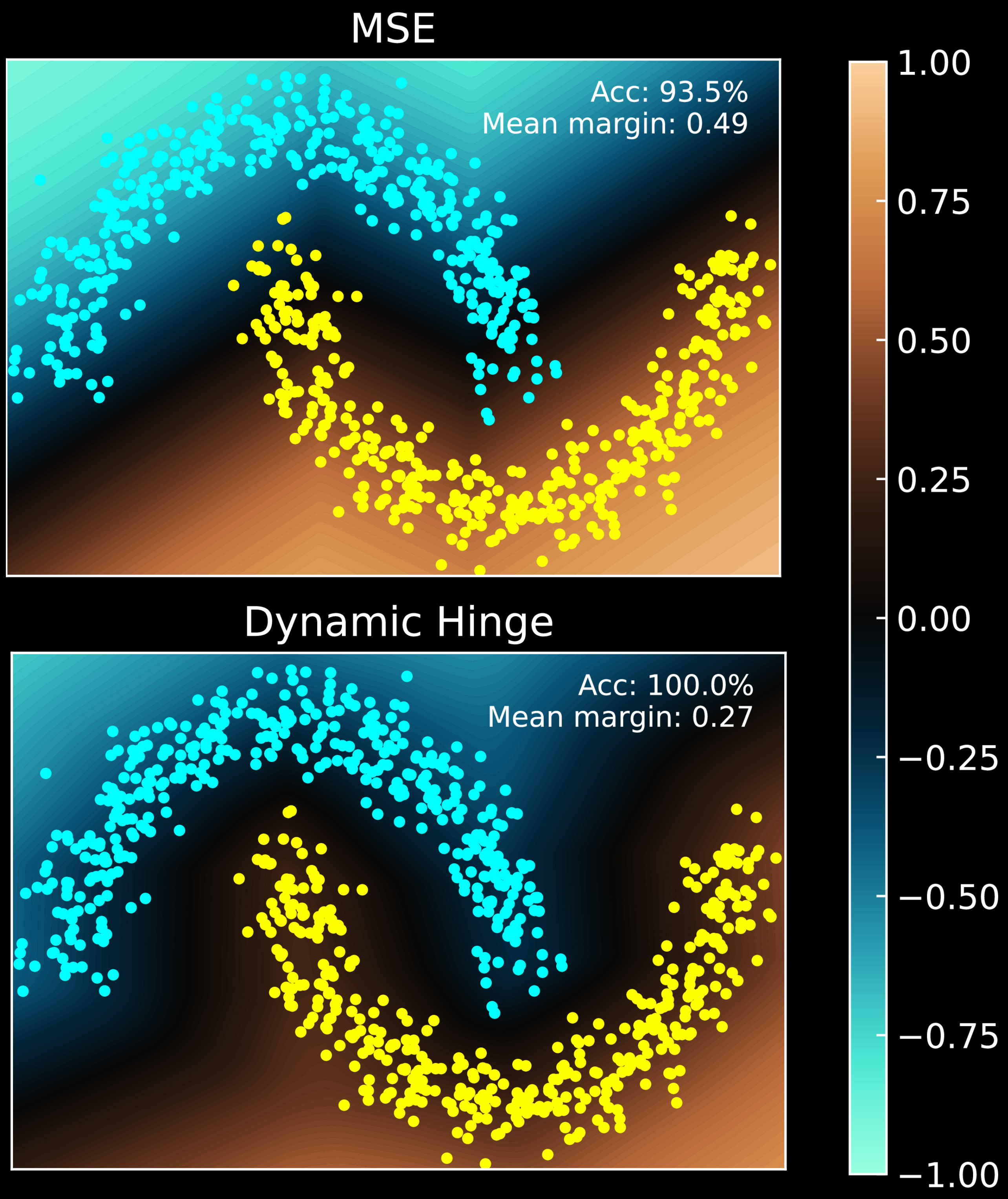
Even though a 100% accurate 1-Lipschitz F must exist, we cannot find it using common loss functions like MSE or BCE.

The reason is that MSE/BCE try to push responses as close to +-1 as possible; however, if the Lipschitz bound is too tight, such that the LR is inaccessible, then minimizing these loss functions is NOT the same as optimizing classification performance.

For this case, we can simply modify the loss function to minimize max[0,d/2 - y*yhat], where d is the distance between the 2 samples at each point and y(hat) is the predicted(true) score.

There is likely a more generic loss that works for any Lipschitz problem (people are studying this).

This seems to be an academic problem though given that you get to choose the Lipschitz constant — and this problem should really only be noticeable for a non-optimal choice.



Kitouni, Nolte, MW [NeurIPS 2021, 2112.00038]

Energy Mover's Distance

The Energy Mover's Distance (EMD), modeled after the earth mover's distance or Wasserstein metric, is the minimum work required to rearrange one event (or jet) into another.

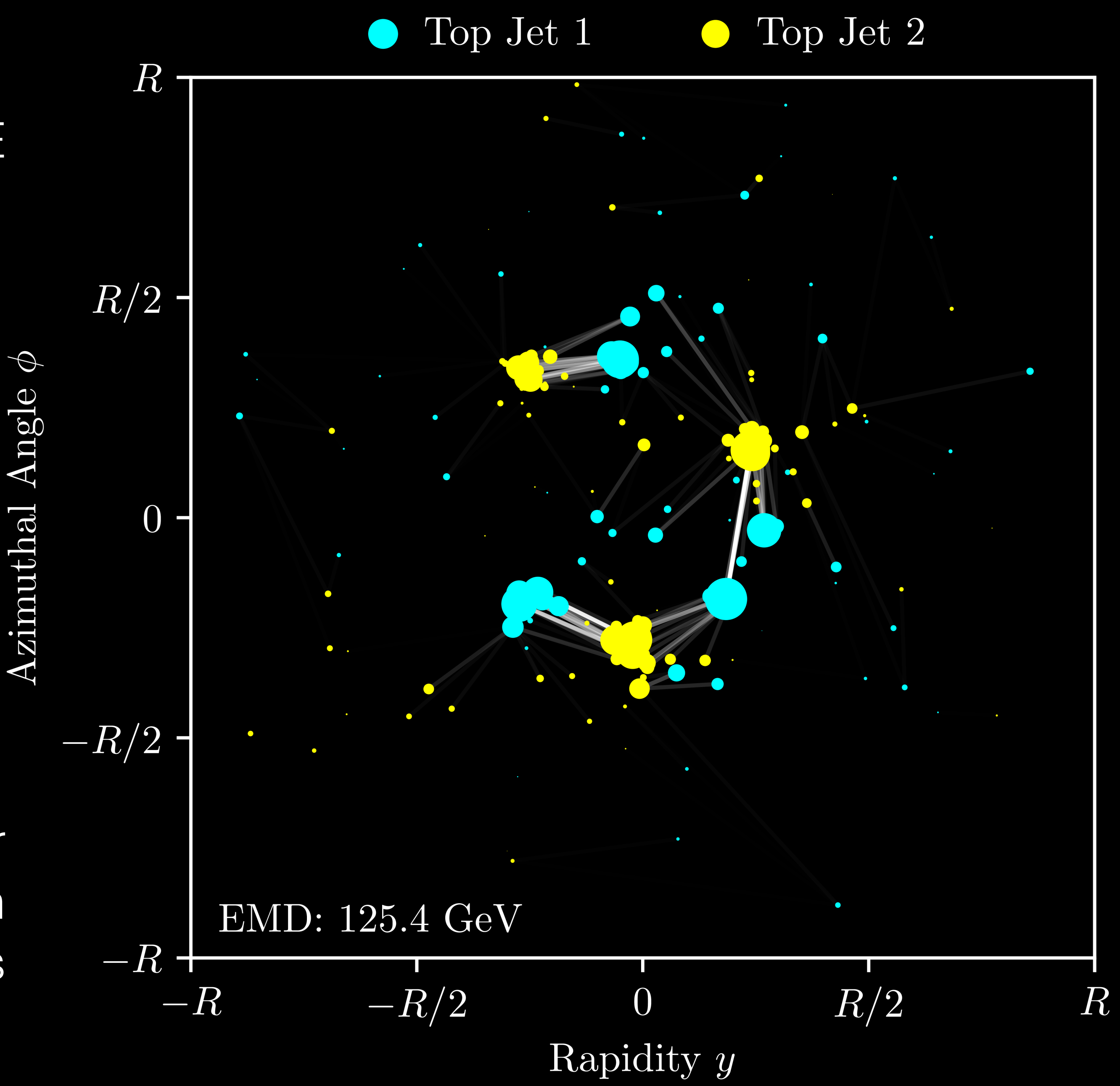
For the case where the 2 jets have equal energy E (just to simplify the expressions):

$$\text{EMD}(\{E_i\}, \{E'_j\}) = \min_{f_{ij}} \sum_{ij} f_{ij}\theta_{ij}$$

$$f_{ij} \geq 0, \sum_j f_{ij} \leq E_i, \sum_i f_{ij} \leq E'_j, \sum_{ij} f_{ij} = E$$

See *The hidden geometry of particle collisions* for detailed discussion on the relationship between the EMD metric and many fundamental concepts in QFT and collider physics.

Komiske, Metodiev, Thaler [2004.04159]



Komiske, Metodiev, Thaler [1902.02346]

KR Duality

The Kantorovich-Rubinstein duality allows us to recast the EMD calculation as a problem of finding the 1-Lipschitz function f that maximizes the RHS of

$$\mathrm{EMD}(\{E_i\}, \{E'_j\}) = \max_f \left[\sum_i E_i f(y_i, \phi_i) - \sum_j E'_j f(y'_j, \phi'_j) \right]$$

Since it's now possible to obtain highly expressive Lipschitz functions, we can determine the EMD by maximizing this expression using gradient descent and a sufficiently large 1-Lipschitz NN.

More interestingly, Gambhir, Thaler+ [2302.12266] propose rather than comparing 2 jets, replace one with a parametrized distribution to both define and quantify **shape-based observables** using EMD.

In their **SHAPER** algorithm, they use the well-known Sinkhorn approximation technique to make the EMD calculation differentiable, at the cost of sacrificing exactness (they use an iterative procedure to minimize the impact of the use of the approximate method).

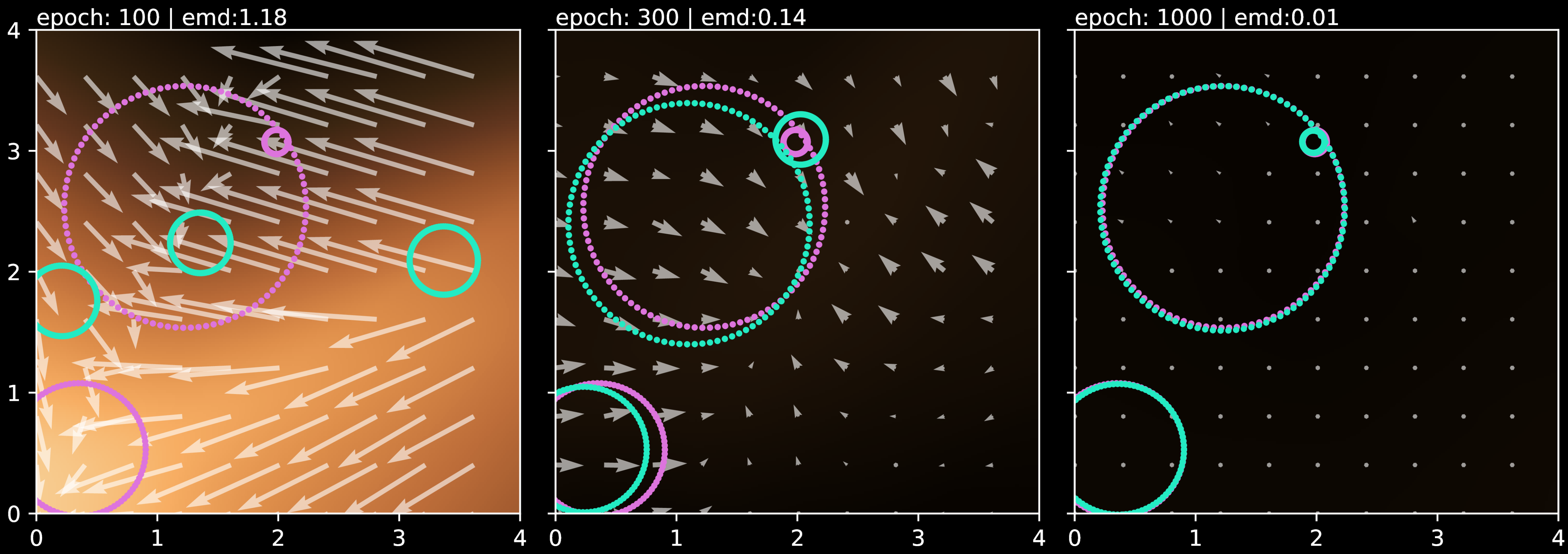
We can achieve this instead using Lipschitz networks, which provide an exact differentiable method for determining the EMD for any given shape-based distributions (n.b. this is a minimax problem).

Kitouni, Nolte, MW [NeurIPS 2022, 2209.15624]

NEEMo

Neural **E**stimation of the **E**nergy **Mo**ver’s distance

Repurposing our Lipschitz NN code quickly allowed us to enable parametric regression using the Wasserstein metric in an exact and differentiable formulation.



Toy example of fitting 3 circles of unknown location, radius (cyan points) to a fixed data set (magenta points). The goal is to minimize the EMD by solving the minimax problem from the previous slide.

The color map is the Kantorovich potential (Lipschitz NN), whose gradients (arrows) exert pseudo-forces on the cyan points during the gradience descent. In this toy example, the minimum EMD is zero because the data is a realization of the parametric shapes; in general, this will not be the case.

Kitouni, Nolte, MW [NeurIPS 2022, 2209.15624]

Summary

TLDR: Lipschitz networks are great https://github.com/niklasnolte/MonotoneNorm

- Lipschitz networks are NNs where the gradient of the learned model is bounded with respect to some chosen norm by a chosen Lipschitz constant. Bounding the gradient of the function reduces overfitting and makes the learning more robust.

- Choosing the 1 norm leads to bounds on each element of the gradient independently. We have derived a new way to do this that results in better learning dynamics.

- By adding a simple linear function to the NN we can force the learned function to be monotonic in any direction(s) we want. For a trigger, this lets us enforce outliers are better.

- Using the non-element-wise (vector) activation function GroupSort allows us to universally approximate any Lipschitz function — using tiny networks.

- This architecture likely has many applications in physics due to the inevitable appearance of scales due to resolution, stability, simulation quality, known unknowns, etc — and we showed that it also works well in other domains such as criminal justice, medicine, finance, etc.

- A perhaps academic concern: classification is not regression, which can result in standard loss functions not leading to the optimal Lipschitz-bounded functions, though it's not obvious this will happen in any real-world examples.

- Due to the KR duality, we can use Lipschitz NNs to determine the Energy Mover's Distance in an exact and differentiable way, which enables performing parametric regression without the need for the Sinkhorn approximation.