Advanced applications for generative models

25 March 2021, YSDA

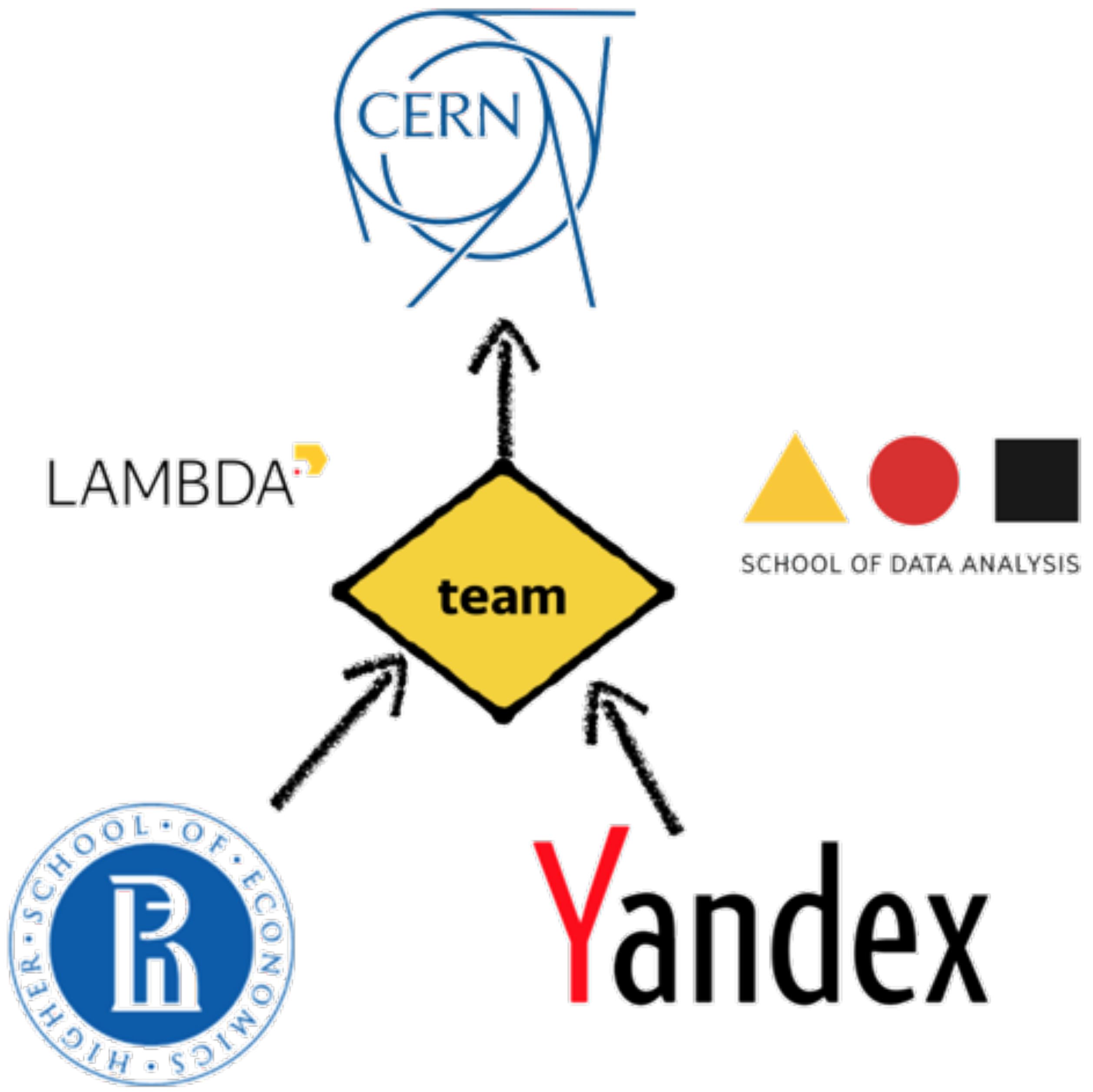
A. Ustyuzhanin

NRU HSE

YSDA

Quick self-intro

- Collaborates with LHCb, SHiP, OPERA, CRAYFIS experiments
- Development and application of Machine Learning methods for solving tough scientific challenges;
- Research Project examples:
  - Storage/speed optimization for LHCb triggers;
  - Particle identification algorithms;
  - Optimization of detector devices;
  - Fast and meaningful physical process simulation.
- Co-organization of ML challenges: Flavours of Physics, TrackML
- 6 Summer schools on Machine Learning for High-Energy Physics
- Open for interns, graduate students and post doc researchers!



Overview

- Fast physics simulation
- Surrogate models for optimization
- Data augmentation for anomaly detection

Fast Physics Simulation

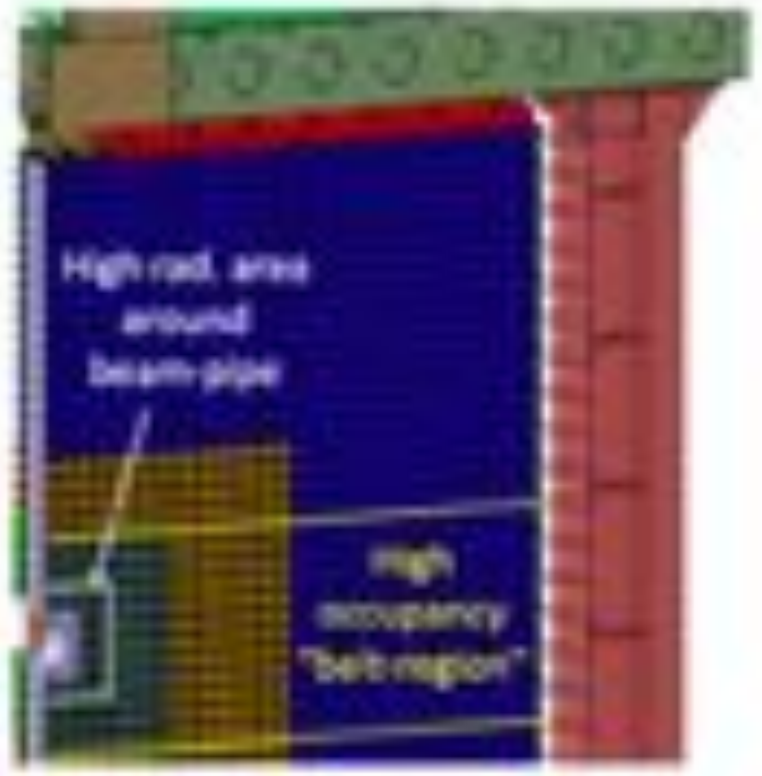
LHCb Calorimeter

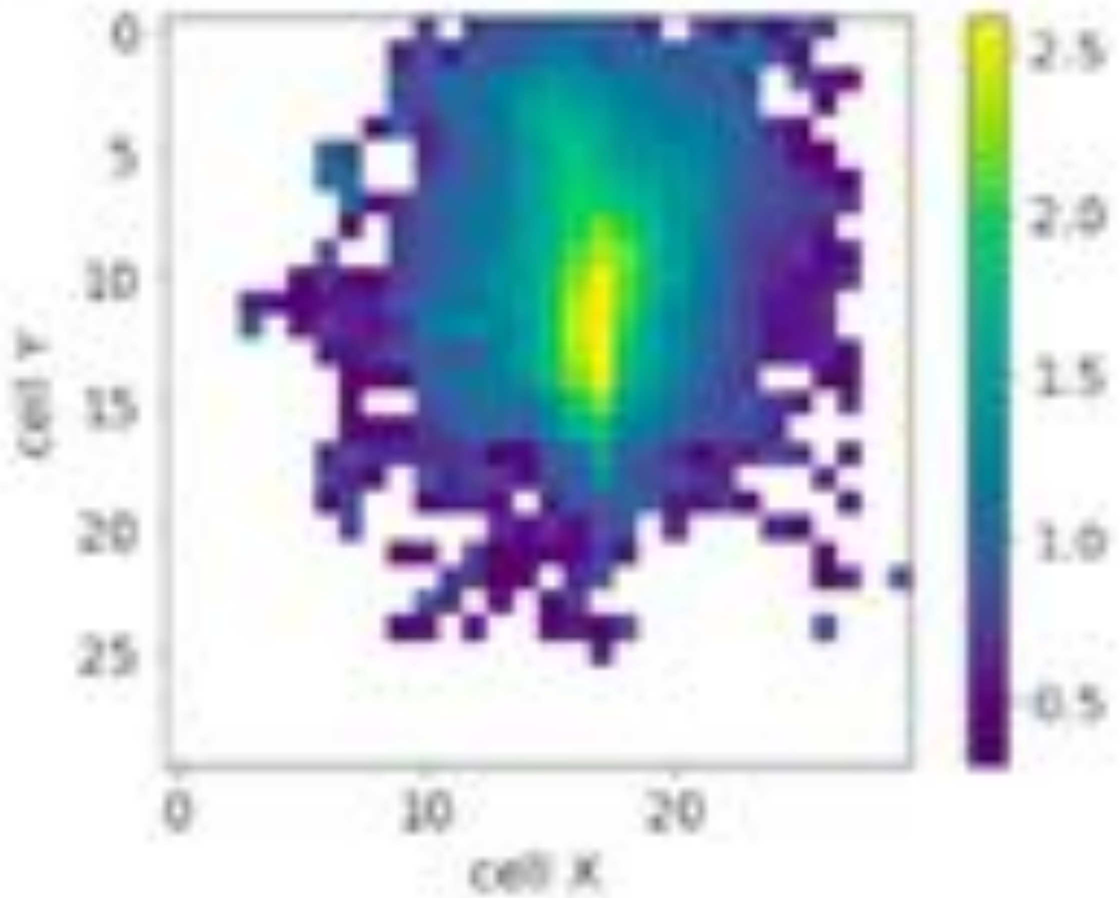
The calorimeter consists of many cells that reads out the energy deposit of a single particle.

The LHCb calorimeter construction motivated by the need to have better performance in the most populated regions.

A single particle deposits energy to several cells. An event is a sum of all particles and some noise.

We are normally in some reconstructed parameters of the event.





Calorimeter Simulation

- Since we know all processes in the sub-detector, we can fully simulate an event using precise physics-motivated rules.
- For calorimeters this means considering the structure of response that consists of many secondary particles.
- This is done using Geant V toolkit.
- **Pro: controlled simulation physics**
- **Cons:**
  - **slow,**
  - **needs fine tuning.**

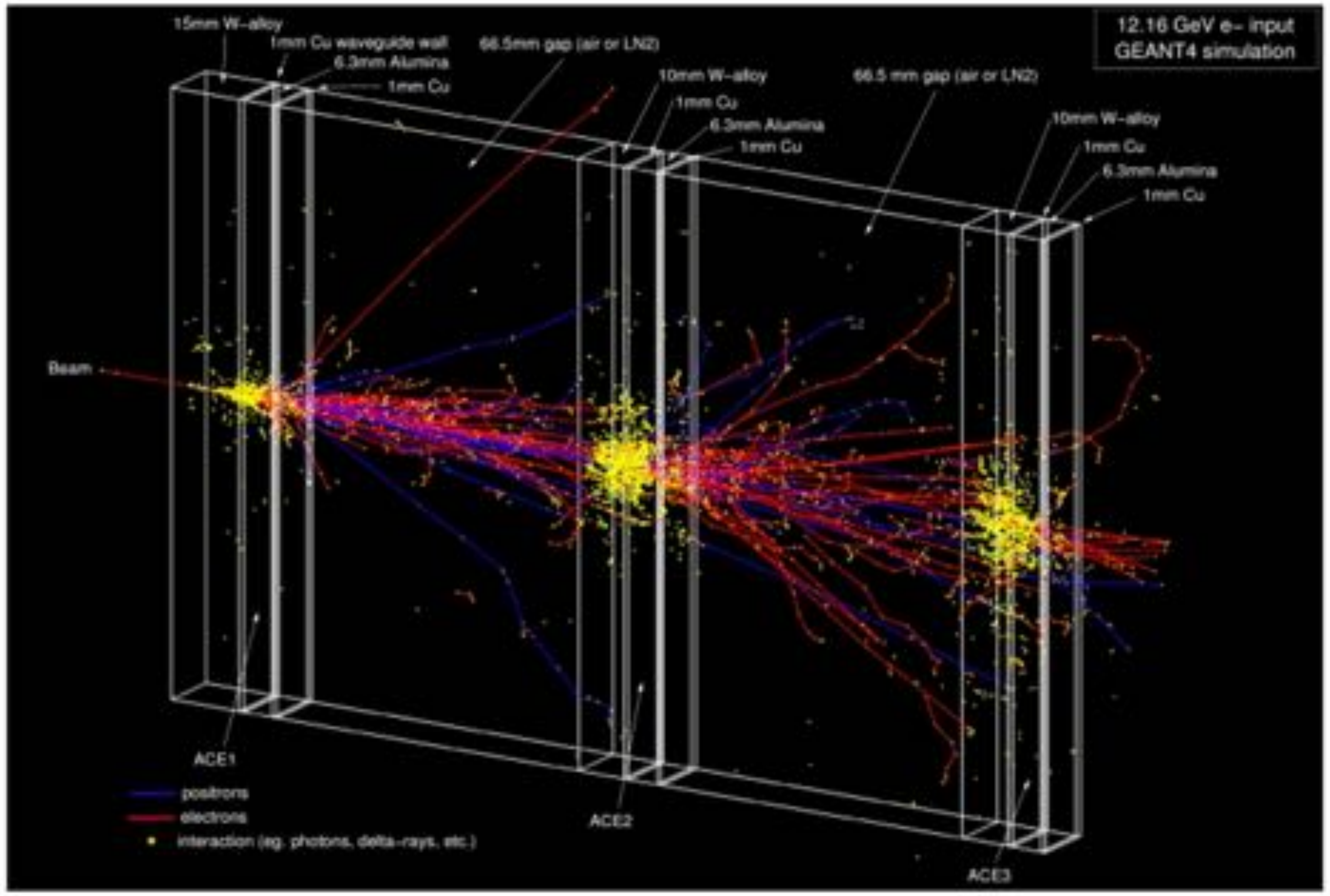
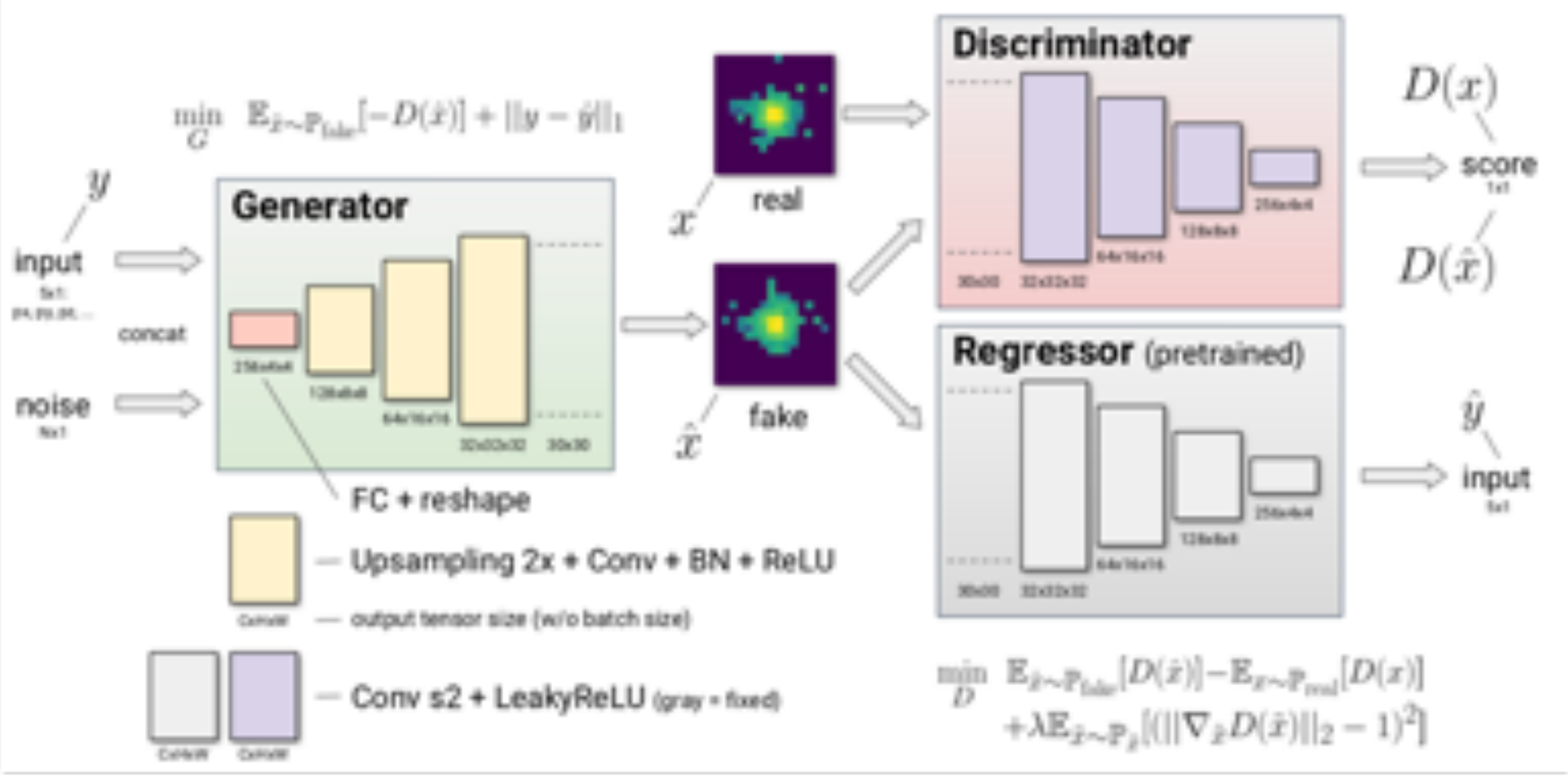
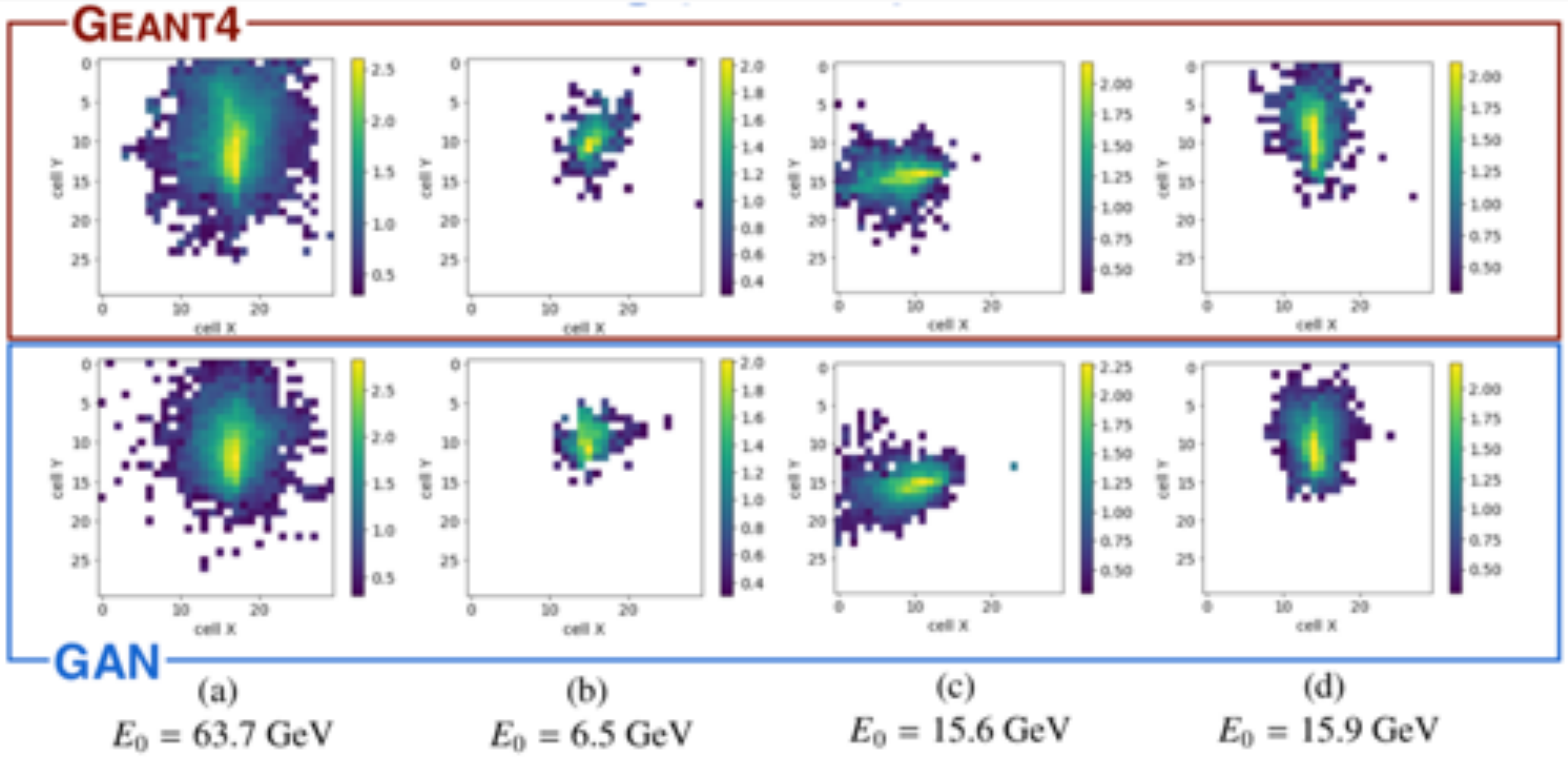


FIG. 2: Layout diagram, and GEANT4 simulation of a single 12.16 GeV electron event in our ACE detector system; in this case liquid nitrogen occupies the interelement spaces.

Example 1. Fast calorimeter simulation

- LHCb-like calorimeter 30x30
- 5 conditional parameters per particle (3D momentum, 2D coordinate)
- Electrons from particle gun shot at 1x1 cm square at the center of the calorimeter face
- Approach: use GANs
- 10^5 x speed-up!





Chekalina V, Orlova E, Ratnikov F, Ulyanov D, AU, Zakharov E, https://doi.org/10.1051/epjconf/201921402034

Black-Box Optimization with Local Generative Surrogates (L-GSO)

Example 2: SHiP Detector Shield Optimization

$$\text{background}(\theta) = \mathop{\mathbb{E}}_{\text{event}} \mathbb{I}\left[\text{muons} > 0 \mid \text{event}, \theta\right] \to \min$$

- How can we optimize new experiment shield hardware design with respect to smallest amount of muons and budget limits?
- Computationally expensive!
- **<See below>**

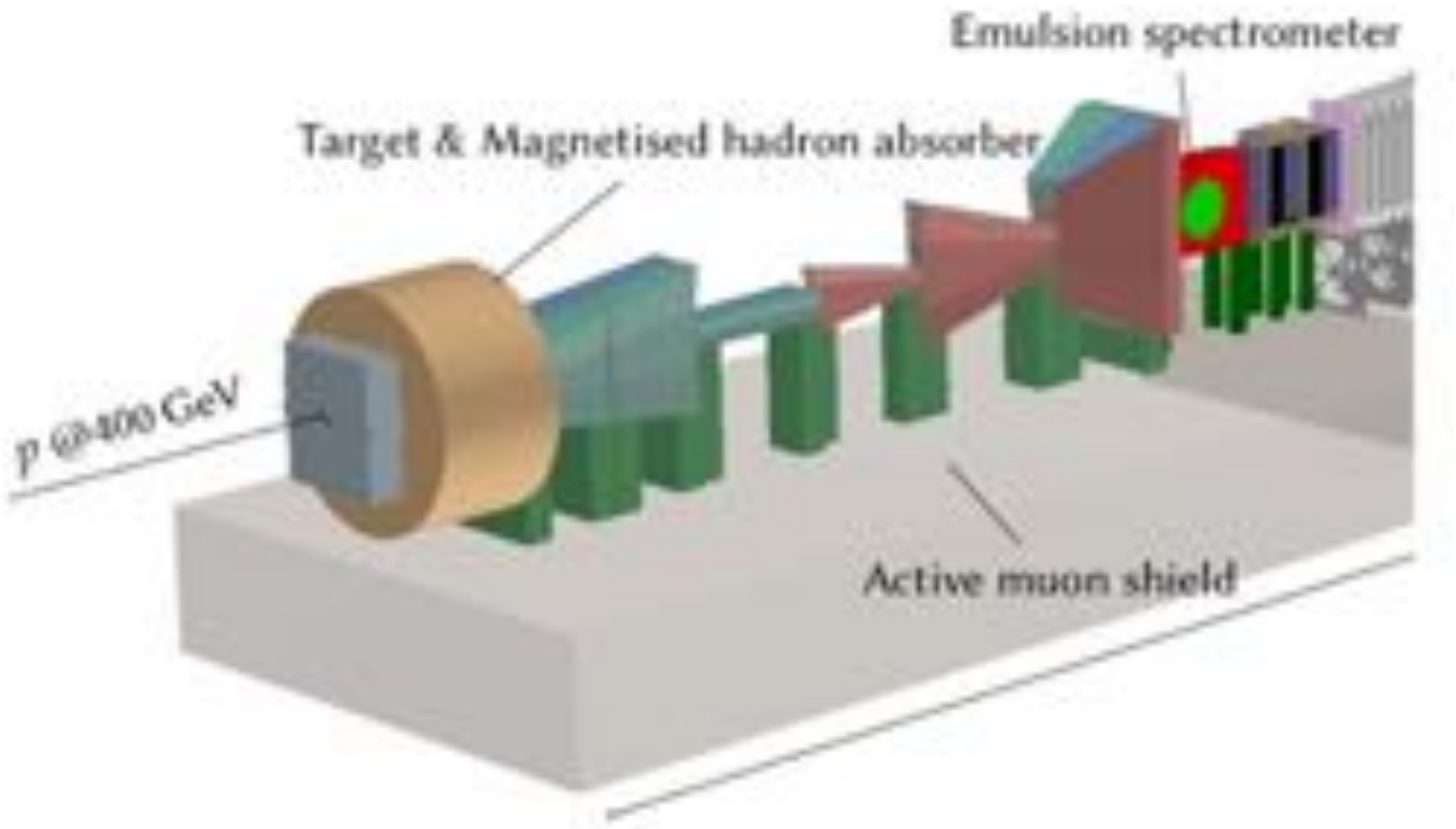


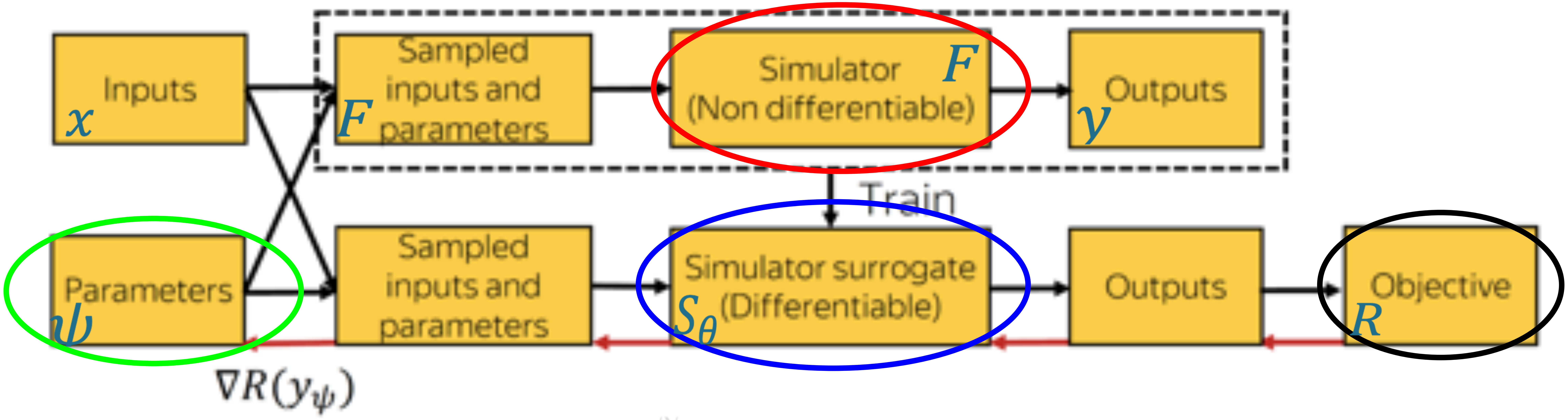
Image source: Oliver Lantwin, Bayesian optimisation of the SHiP muon shield.

TL;DR:
Let's approximate a stochastic black-box with a local generative surrogate.

This allows computing gradients of the **objective** w.r.t. parameters of the black-box.

$$\mathbb{E}[\mathcal{R}(\boldsymbol{y})] = \int \mathcal{R}(\boldsymbol{y}) \boxed{p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\psi})} q(\boldsymbol{x}) d\boldsymbol{x} d\boldsymbol{y} \approx \frac{1}{N}\sum_{i=1}^{N} \mathcal{R}\boxed{(F(\boldsymbol{x}_i;\boldsymbol{\psi}))}$$

$\boldsymbol{y}_i = F(\boldsymbol{x}_i;\boldsymbol{\psi}) \sim p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\psi}),$
$x_i \sim q(\boldsymbol{x})$



$$\nabla_{\psi}\mathbb{E}[\mathcal{R}(\boldsymbol{y})] \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_{\psi}\mathcal{R}(S_\theta(\boldsymbol{z}_i,\boldsymbol{x}_i;\boldsymbol{\psi}))$$

$$\nabla_{\psi}\,\mathbb{E}[\mathcal{R}(\boldsymbol{y})] = \int \nabla_{\psi}\mathcal{R}(\boldsymbol{y})\boxed{p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\psi})}q(\boldsymbol{x})d\boldsymbol{x}d\boldsymbol{y} \approx \frac{1}{N}\sum_{i=1}^{N}\nabla_{\psi}\mathcal{R}(\boxed{F(\boldsymbol{x}_i;\boldsymbol{\psi})})$$

From intractable gradient estimation of the black-box.

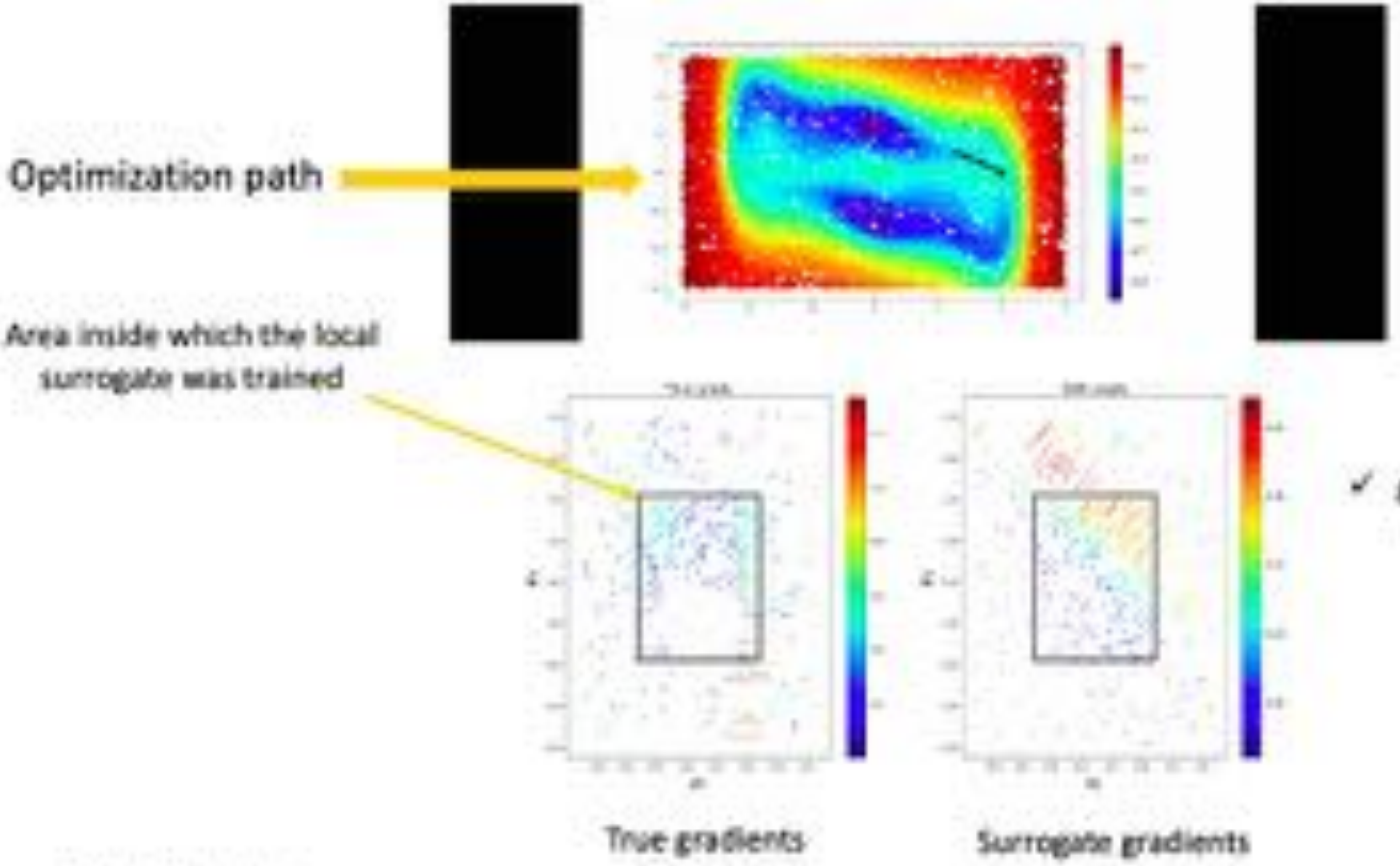
$$\nabla_{\psi}\,\mathbb{E}[\mathcal{R}(\boldsymbol{y})] \approx \frac{1}{N}\sum_{i=1}^{N}\nabla_{\psi}\mathcal{R}(S_{\theta}(\boldsymbol{z}_i,\boldsymbol{x}_i;\boldsymbol{\psi}))$$

To gradient estimation with learnable generative surrogate(GAN, NF, etc).

$$\boldsymbol{\psi} = \boldsymbol{\psi} - \mu\frac{1}{N}\sum_{i=1}^{N}\nabla_{\psi}\mathcal{R}(S_{\theta}(\boldsymbol{z}_i,\boldsymbol{x}_i;\boldsymbol{\psi}))$$

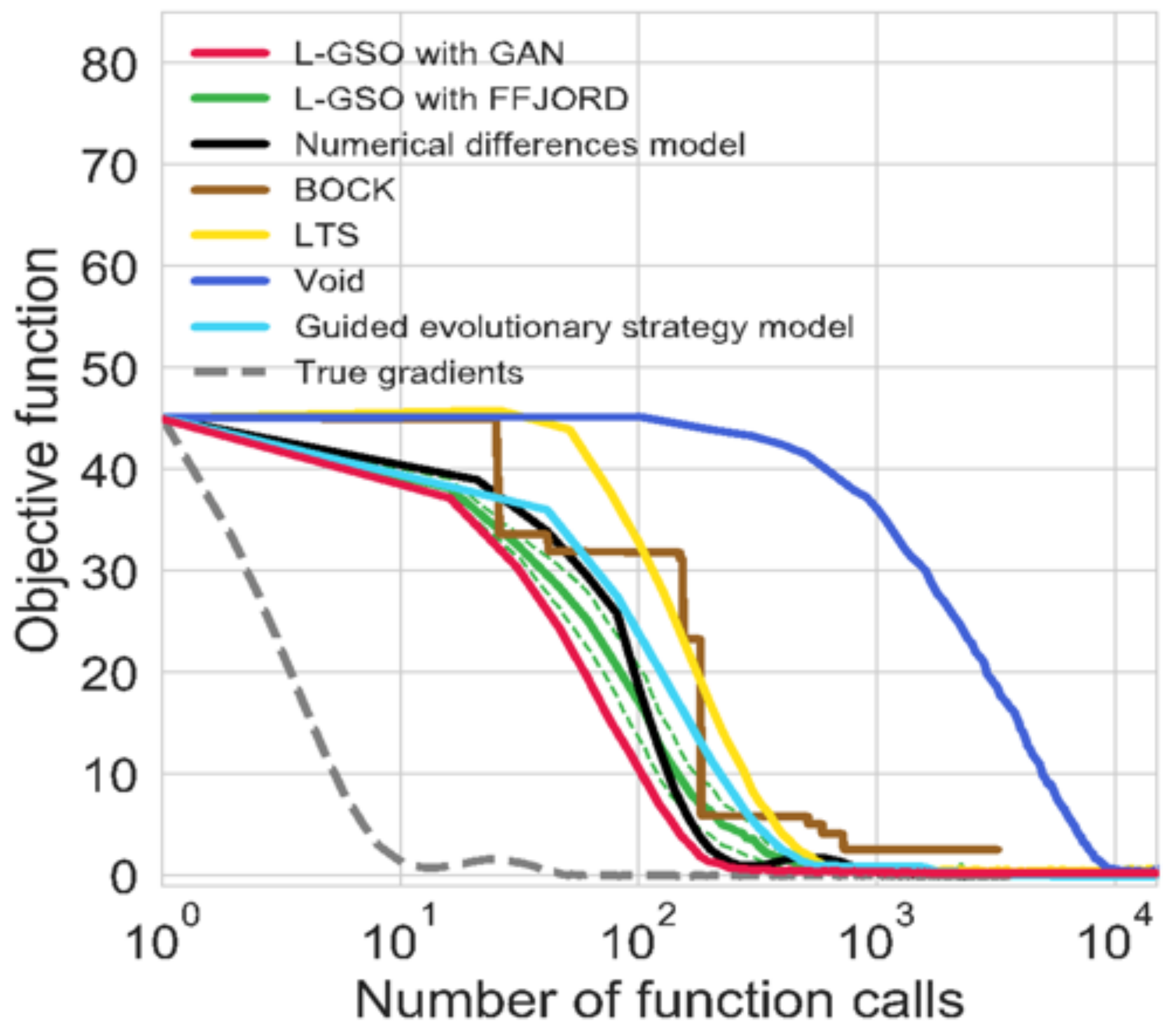
And successive gradient based optimization of the parameters.

Key point: training **local** generative surrogate

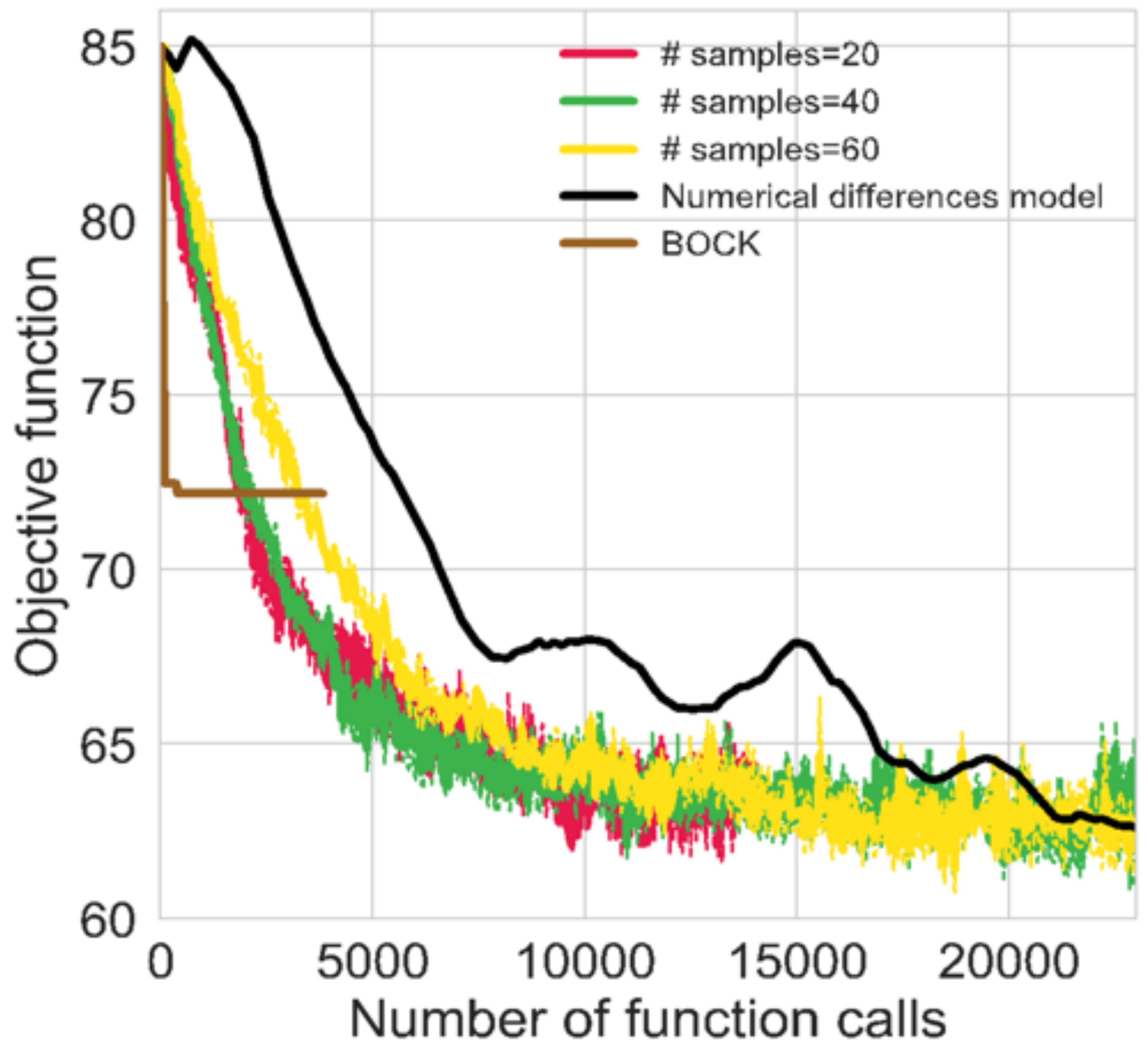


- ✓ gradients of the non-linear surface are well estimated inside the local area.

Results on high-dimensional problems with low-dimensional manifold



Nonlinear Three Hump problem, 40dim



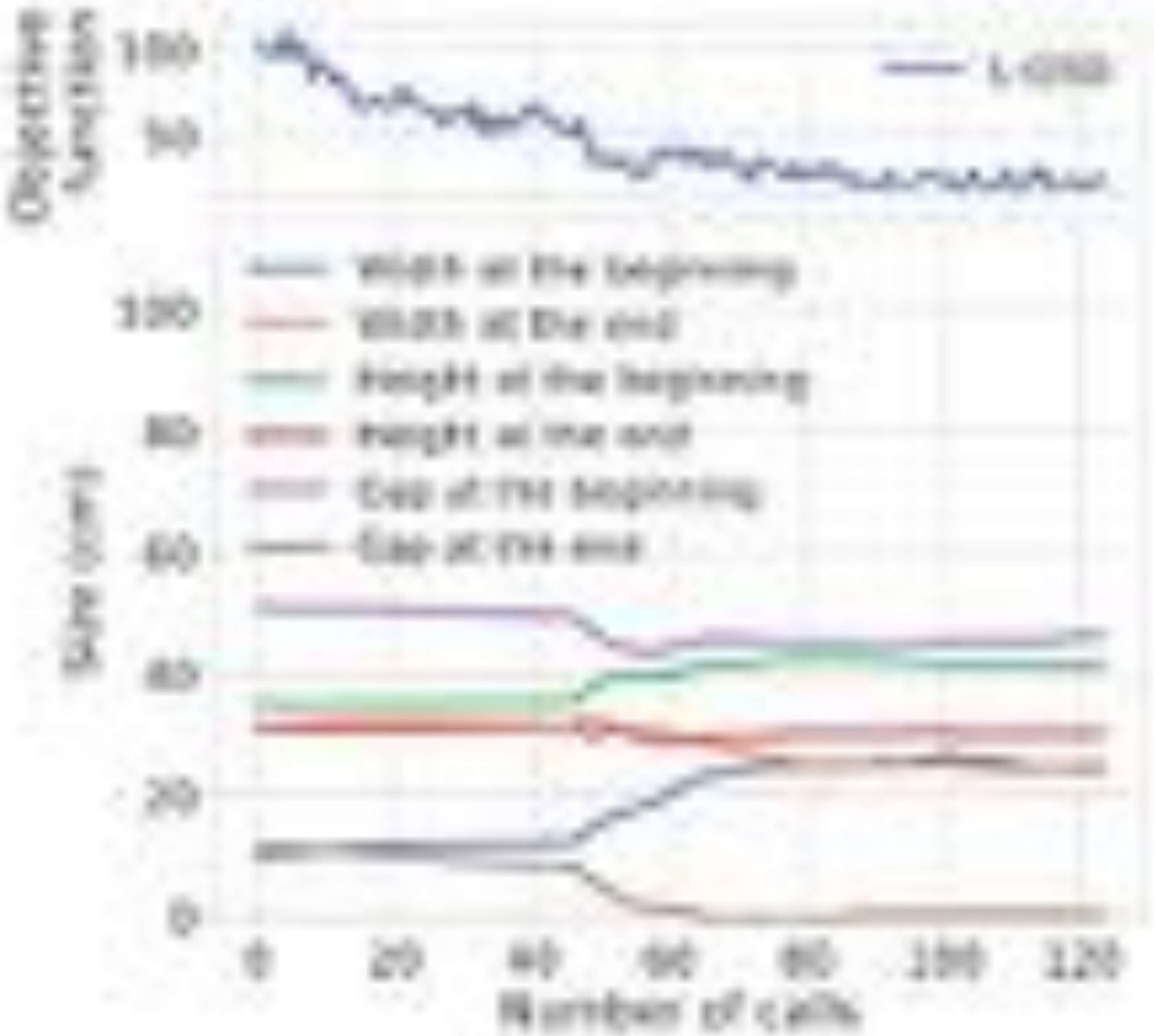
Neural network weights optimization, 91dim

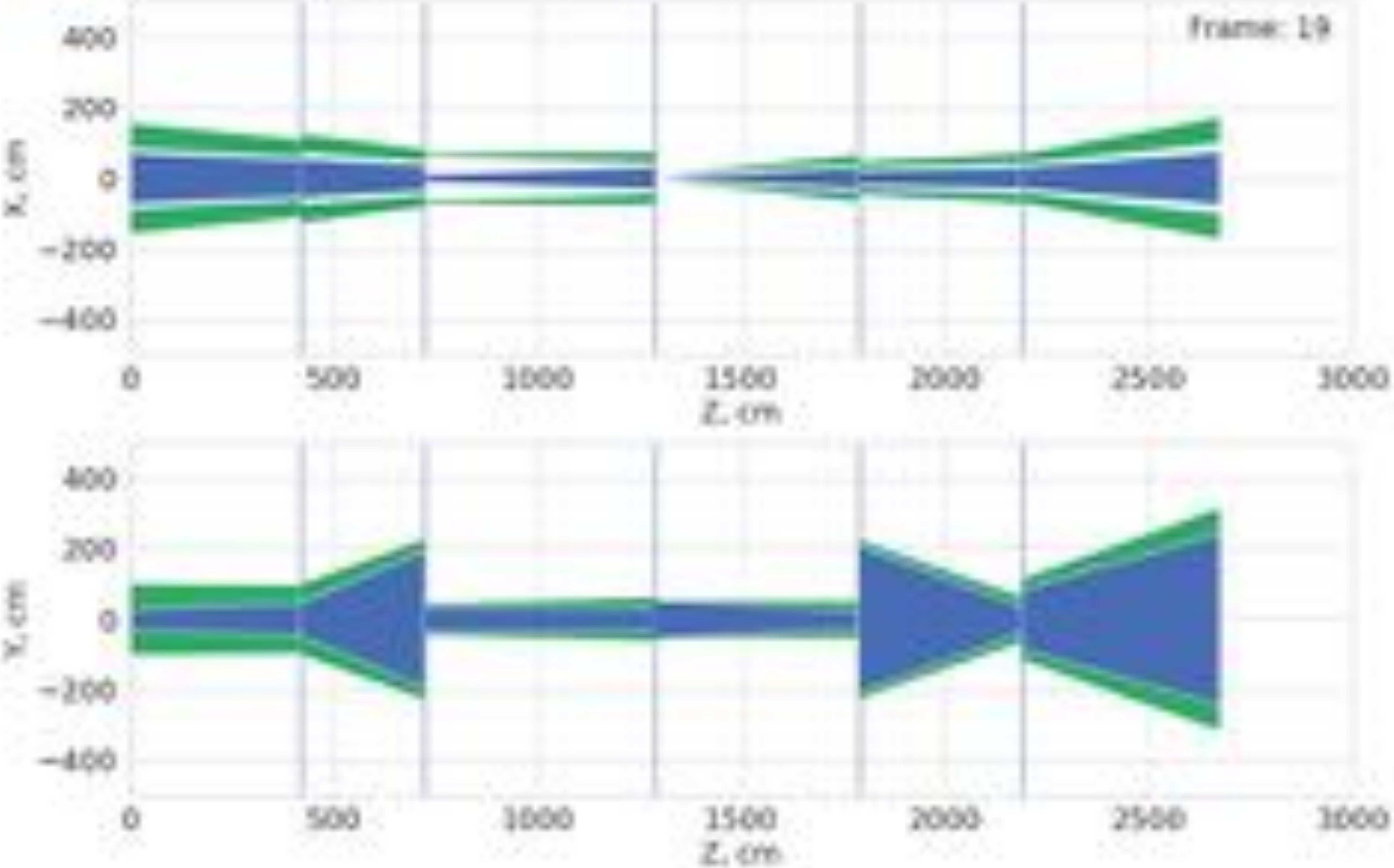
L-GSO outperforms all algorithms in a high-dimensional setting when parameters lie on a lower dimension manifold.

1. Liu, Shuang, and Kamalika Chaudhuri. "The inductive bias of restricted f-gans." *arXiv preprint arXiv:1809.04542* (2018).

2. Uppal, Ananya, Shashank Singh, and Barnabás Póczos. "Nonparametric density estimation & convergence rates for gans under besov ipm losses." *Advances in Neural Information Processing Systems*. 2019.

Design optimisation in 42 dimensional space of physics simulator



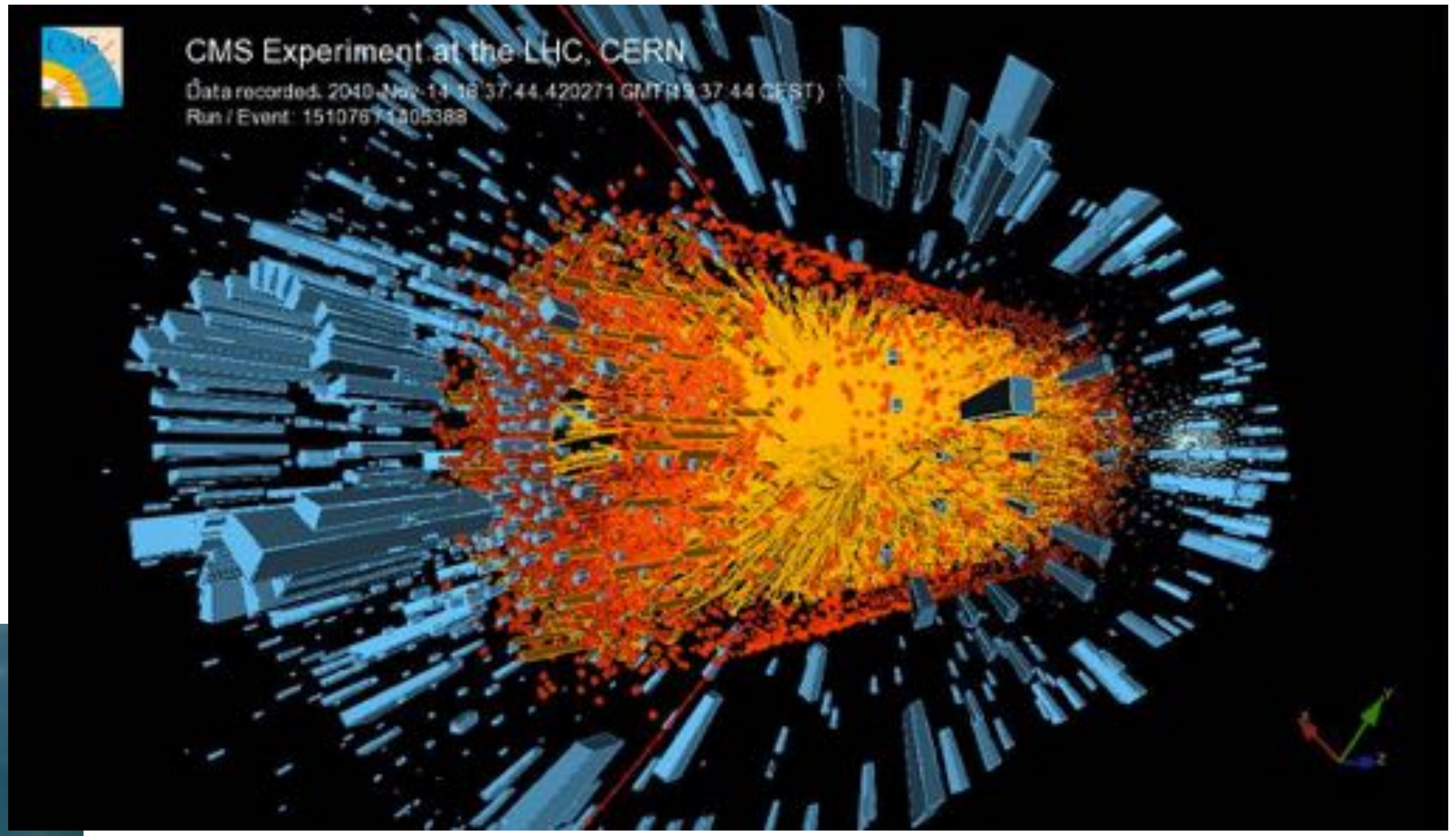


- L-GSO improves previous results obtained with BO with the same computational budget.
- New design is 25% more efficient.

Shirobokov S., Belavin V., Kagan M, AU, Baydin A., NeurIPS'20 paper
https://arxiv.org/abs/2002.04632

Anomaly Detection

Example 3. Anomaly



Looking for unexpected correlations between features

One-class approaches

Assume we have C^{+} (normal examples) represented by whole X

Given information about normal instances, the algorithm looks for meaningful boundary

Examples: Isolation forest, One-class SVM, Support Vector Data Description (SVDD), Local Outlier Factor, One-class NN,

Two-class approaches

In some cases information about anomalies (C^{-}) is available but limited, so regular two-class approaches can be useful

$$\mathcal{L}_2(f) = - \mathop{\mathbb{E}}_{x \sim \mathcal{C}^+} \log f(x) - \mathop{\mathbb{E}}_{x \sim \mathcal{C}^-} \log(1 - f(x))$$

Examples: Gradient Boosting, ANN, AutoEncoders + Classifier
Optimal decision function is given by:

$$f^*(x) = P(\mathcal{C}^+ \mid x) = \frac{P(x \mid \mathcal{C}^+)}{P(x \mid \mathcal{C}^+) + P(x \mid \mathcal{C}^-)}$$

Why worry?

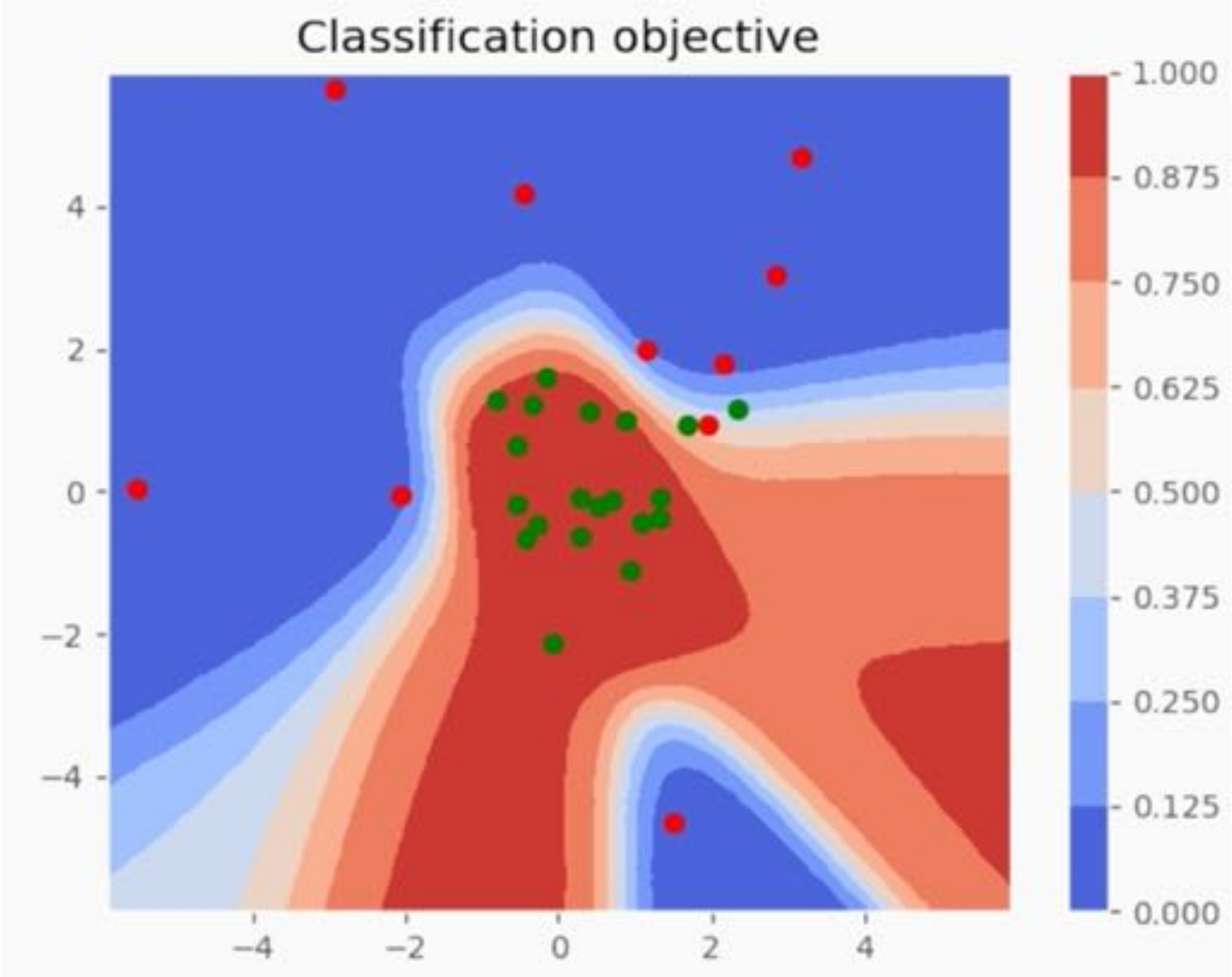
Plenty of methods are already there

- Traditional: One-class SVM, Isolation forest, DeepSVDD, OneClassNN
- Two-class methods (Xgboost, ANNs, AutoEncoder + classifier [4, 5])

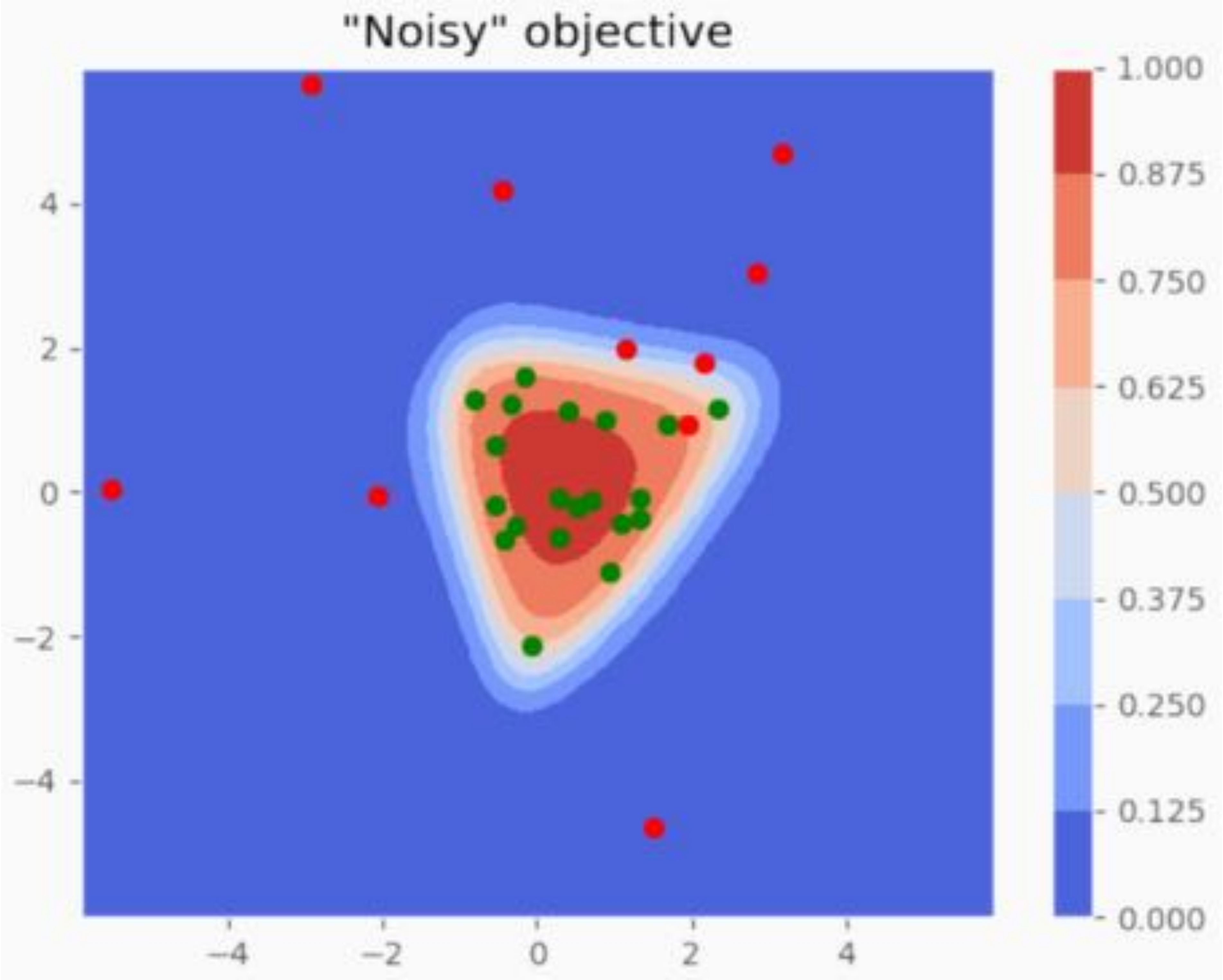
However,

- One-class methods perform poorly when anomaly examples looks similar to normal ones
- Two-class methods perform poorly in the unpopulated regions

Illustration



Binary classification



Unitary classification

Color map represent posterior distribution density

Fundamental idea

Let's combine strong sides of both approaches by enhancing binary classification loss function with unitary classification part L_0. Where L_0 would represent loss of pseudo-anomalies augmenti C^-.

Generative augmentation models for anomalies:

- Uniform distribution
- 'Ambiguous' distribution sampling (MCMC, energy-based, adversarially-trained, ...)

Method 1. Uniform sampling

Regular 2-class loss function

$$\mathcal{L}_2(f) = - \mathop{\mathbb{E}}_{x \sim \mathcal{C}^+} \log f(x) - \mathop{\mathbb{E}}_{x \sim \mathcal{C}^-} \log(1 - f(x))$$

leads to the solution in asymptotic limit

$$f^*(x) = P(\mathcal{C}^+ \mid x) = \frac{P(x \mid \mathcal{C}^+)}{P(x \mid \mathcal{C}^+) + P(x \mid \mathcal{C}^-)}$$

However, if statistics $P(x \mid C^-)$ is limited, it leads to unstable or high variance (overfitted) solutions.

Method 1. Uniform sampling

Let's consider C^0 (surrogate anomaly, or *noise*) to be sampled from U on that includes $\mathrm{supp}(C^+)$, then 2-class loss function

$$\mathcal{L}_2(f) = -\mathop{\mathbb{E}}_{x\sim\mathcal{C}^+} \log f(x) - \mathop{\mathbb{E}}_{x\sim\mathcal{C}^-} \log(1 - f(x))$$

Turns into

$$\mathcal{L}_1(f) = -\mathop{\mathbb{E}}_{x\sim\mathcal{C}^+} \log f(x) - \mathop{\mathbb{E}}_{x\sim U} \log(1 - f(x))$$

Thus, solution f can be found as ($P(\mathcal{C}^+) = P(\mathcal{C}^0) = \frac{1}{2}$):

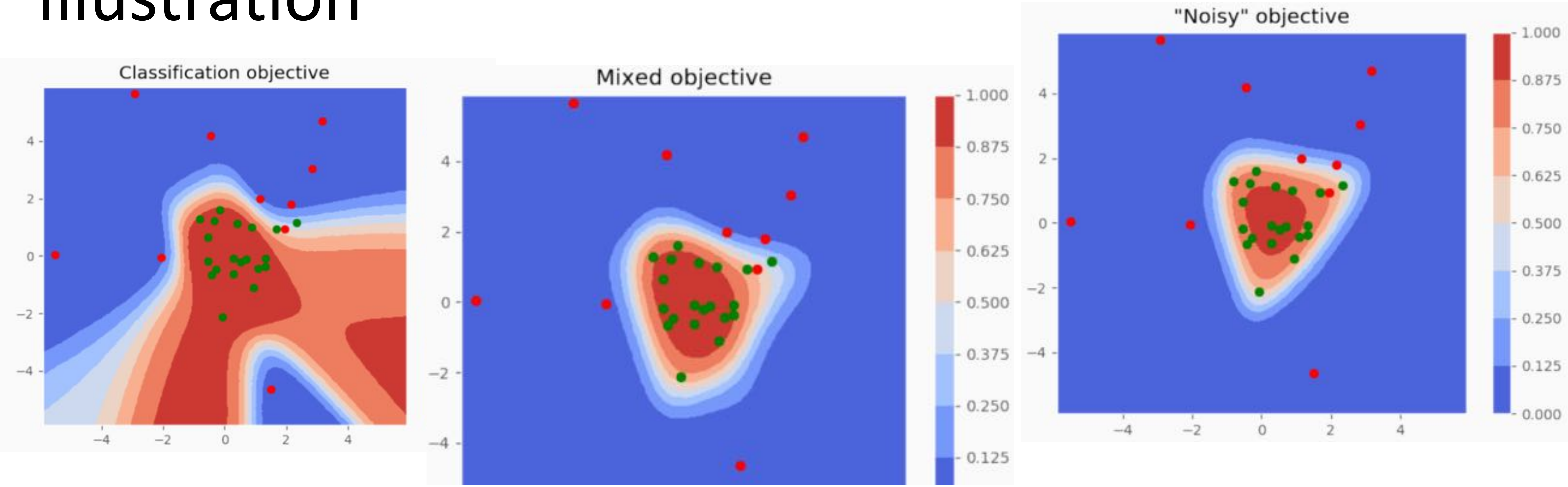
$$f_1^*(x) = \arg\min_f \mathcal{L}_1(f) = \frac{P(x \mid \mathcal{C}^+)}{P(x \mid \mathcal{C}^+) + \mathrm{const}} = h(P(X \mid \mathcal{C}^+))$$

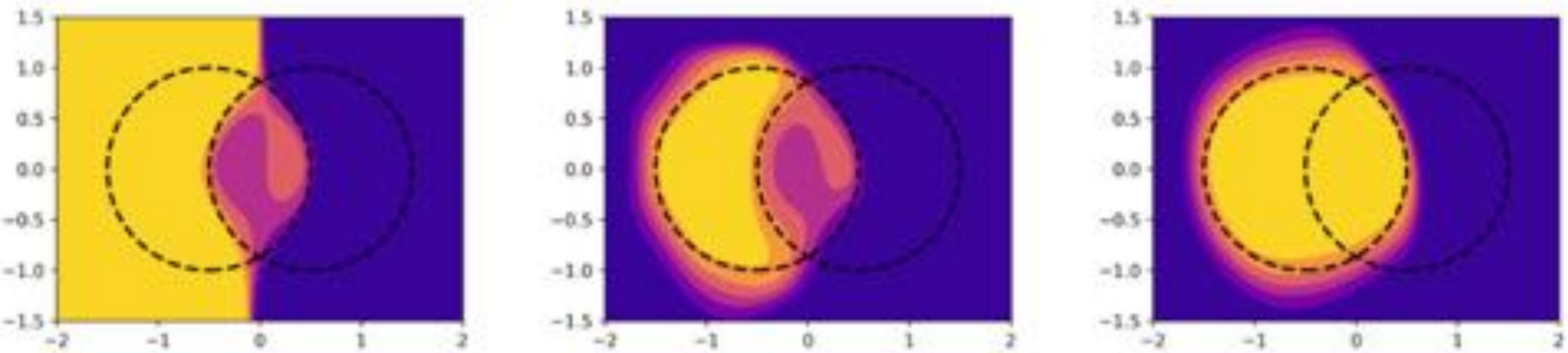
Adding labeled anomalies

$$
\begin{aligned}
\mathcal{L}_{1+\varepsilon}(f) &= \frac{1}{2}\left(L^{+}(f) + (1-\varepsilon)\, L^{0}(f) + \varepsilon\, L^{-}(f)\right); \\
L^{+}(f) &= -\mathop{\mathbb{E}}_{x\sim\mathcal{C}^{+}} \log f(x); \\
L^{-}(f) &= -\mathop{\mathbb{E}}_{x\sim\mathcal{C}^{-}} \log(1 - f(x)); \\
L^{0}(f) &= -\mathop{\mathbb{E}}_{x\sim U} \log(1 - f(x)),
\end{aligned}
$$

ε is a hyper-parameter that allows for gradually switching between one-class ($\varepsilon = 0$) and two-class ($\varepsilon = 1$) classification solutions

Illustration





(a) two-class classification (b) OPE classification (c) one-class classification.

Method 2. MCMC sampling

Method 1 is nice and easy, however in high-dimension case becomes intractable

assume the most important surrogates are placed around normal instances, thus we have to find such distribution Q induced by previous epoch of f:

$$P_f(x) = \frac{1}{Z}\frac{f(x)}{1 - f(x)};$$
$$P_{f^*}(x) = P(x \mid \mathcal{C}^+)$$

where Z– normalization constant, $f(x)$ – output of a neural network corresponding to the normal class, hence we can compute $p(x \mid C^+)$ up to normalization constant. Which may give practical way to do MCMC, but is computationally expensive.

Method 3. Energy-based sampling

Assume, f can be written as $f(x) = \sigma(g(x))$, where σ – sigmoid function

$$P_f(x) = \frac{1}{Z}\frac{f(x)}{1 - f(x)} =$$

$$\frac{1}{Z}\frac{1}{1 + \exp(-g(x))}\frac{1 + \exp(-g(x))}{\exp(-g(x))} = \frac{1}{Z}\exp(g(x))$$

where:

$$Z = \int_{\Omega} \exp(g(x))dx$$

NB: Energy of x is a scalar function g that leads to non-normalized distribution through $p \sim \exp(-g(x))$

EOPE loss function

$$L^0(f) \;=\; -\mathop{\mathbb{E}}_{x\sim U} \log(1 - f(x)).$$

$$L^0(f) = Z \cdot \mathop{\mathbb{E}}_{x\sim P_f} \frac{1 - f(x)}{f(x)} \log(1 - f(x)).$$

Using Jensen inequality:

$$L^0 = \mathop{\mathbb{E}}_{x\sim U} \log(1 + \exp(g(x))) \le \log\left[1 + \mathop{\mathbb{E}}_{x\sim U} \exp(g(x))\right] = \log(1 + Z)$$

If we assume, $\log(1+Z) \sim \log(Z)$ then (as shown in [3]):

$$\nabla \log Z = \frac{1}{Z} \cdot \nabla Z = \int_\Omega \frac{1}{Z} \exp(g(x)) \nabla g(x) dx = \mathop{\mathbb{E}}_{x\sim P_f} \nabla g(x)$$

$$\nabla \mathcal{L}_{1+\varepsilon} = -\mathop{\mathbb{E}}_{x\sim \mathcal{C}^+} \sigma(-g(x))\nabla g(x) + \varepsilon \mathop{\mathbb{E}}_{x\sim \mathcal{C}^-} \sigma(g(x))\nabla g(x) + (1-\varepsilon) \mathop{\mathbb{E}}_{x\sim P_f} \nabla g(x)$$

Energy-based sampling algorithm

Algorithm 2: Energy OPE

Input: normal data, anomalous data — samples from $\mathcal{C}^+$, $\mathcal{C}^-$, the latter might be absent; g_θ — a classifier with parameters θ.

Hyper-parameters: γ — ratio of class priors; ε — controls strength of regularization; MCMC — Monte-Carlo sampling procedure.

while *not converged* **do**

  sample normal data $\{x_i^+ \sim \text{normal data}\}_{i=1}^m$;

  sample known anomalies $\{x_i^- \sim \text{anomalous data}\}_{i=1}^m$;

  sample negative examples $\{x_i^0 \sim \text{MCMC}\left[x \mapsto \exp(g(x))\right]\}_{i=1}^m$;

  $\nabla L^+ \leftarrow \sum_i \nabla_\theta \log(1 + \exp(-g_\theta(x_i^+)))$;

  $\nabla L^- \leftarrow \sum_i \nabla_\theta \log(1 + \exp(g_\theta(x_i^-)))$;

  $\nabla L^E \leftarrow \sum_i \nabla_\theta g_\theta(x_i^0)$;

  $\theta \leftarrow \text{adam}\left(\nabla L^+ + \gamma \nabla L^- + (1 - \varepsilon)\nabla L^E\right)$

end

Observation

This method gives practical hint for MCMC sampling as $f(x) = \sigma(g(x))$ is essentially the output layer of a 2-class ANN

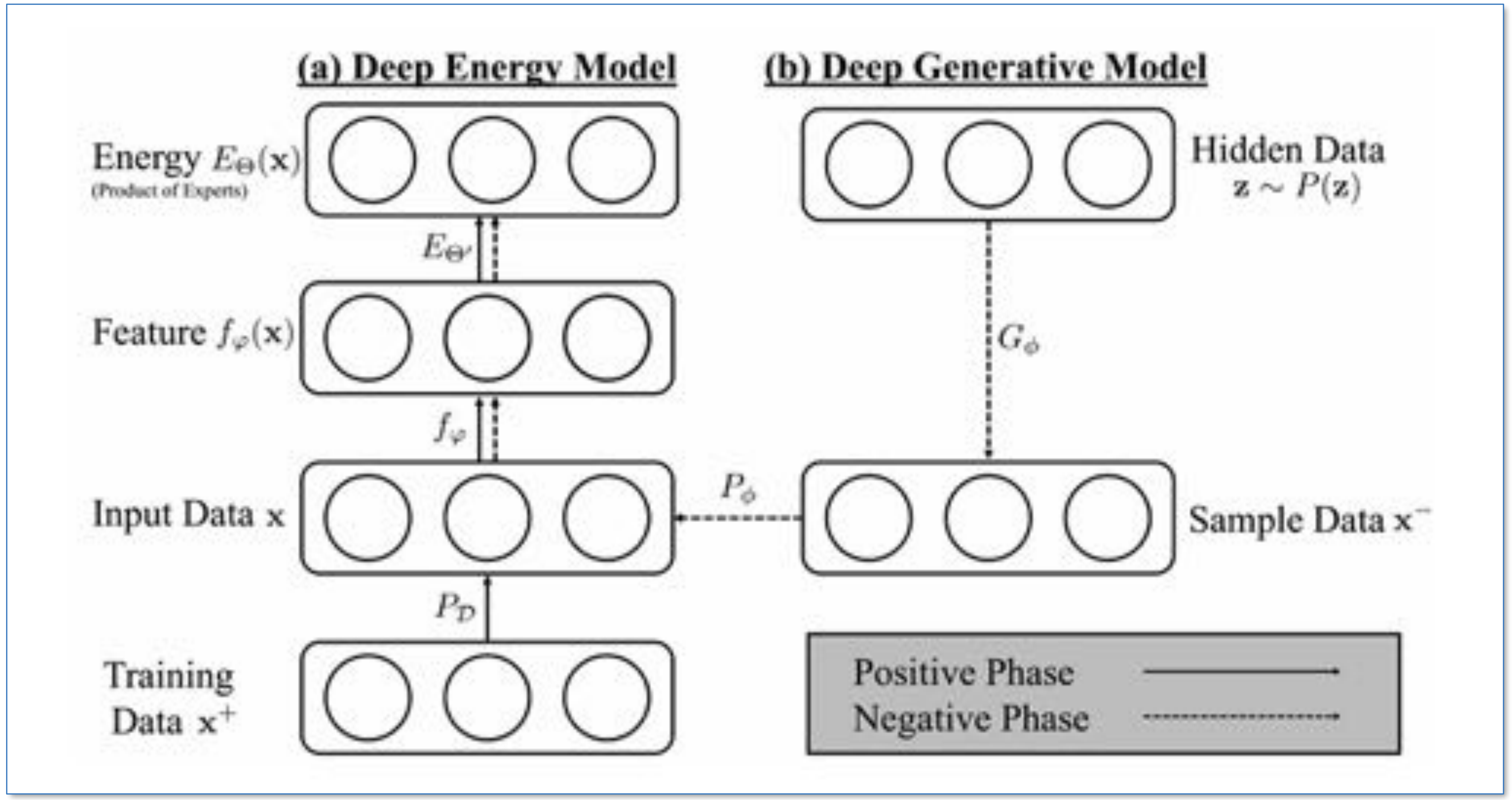
Both L^0 and $\log(Z)$ can be thought of as regularizing terms which result in

- Restricting f values
- Zeroing f in regions without positive samples

Still it requires MCMC which is quite computationally expensive

Method 4. Deep Energy Based sampling

$P_f(x) = \frac{1}{Z}\exp(g(x))$



Let's introduce generator *G(z)* :

$$\mathrm{KL}(Q\|P_f) = \mathop{\mathbb{E}}_{x\sim Q}[-\log P_f(x)] - H(Q);$$

$$\nabla \mathop{\mathbb{E}}_{x\sim Q}[-\log P_f(x)] = \nabla \mathop{\mathbb{E}}_{z\sim P(z)}[-\log P_f(G(z))] = - \mathop{\mathbb{E}}_{z\sim P(z)}[\nabla g(G(z))] \approx -\frac{1}{N}\sum_{i=1}^{N}\nabla g(G(z)).$$

Entropy *H(Q)* is estimated as in [3].

$$H(P_\phi(\mathbf{x})) \approx \sum_{a_i} H(\mathcal{N}(\mu_{a_i}, \sigma_{a_i})) = \sum_{a_i}\frac{1}{2}\log\left(2e\pi\sigma_{a_i}^2\right)$$

Then we follow the same approach as in Method 3.

Observation

Requires training of $G(z)$, that can be trained in parallel with $f(x)$

Works much faster than energy-based MCMC

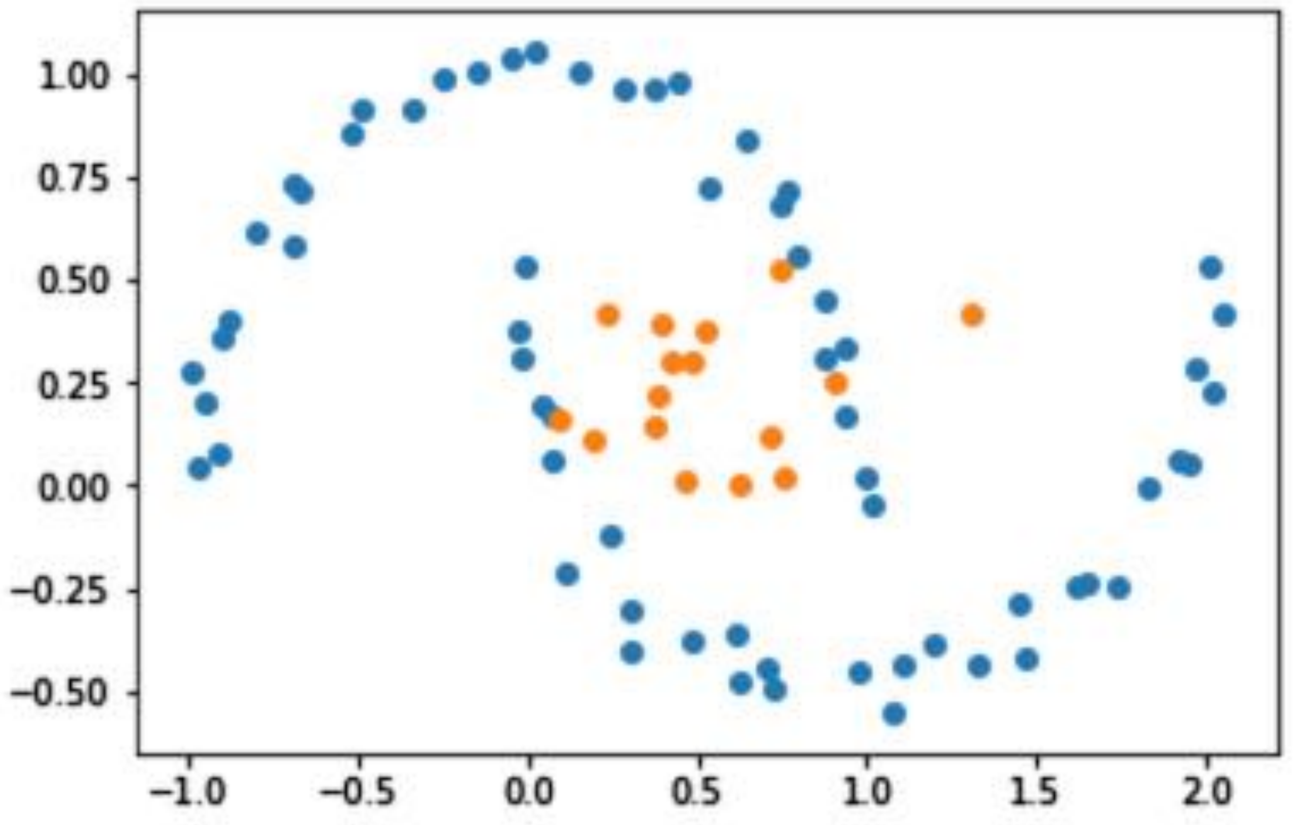
Experiments

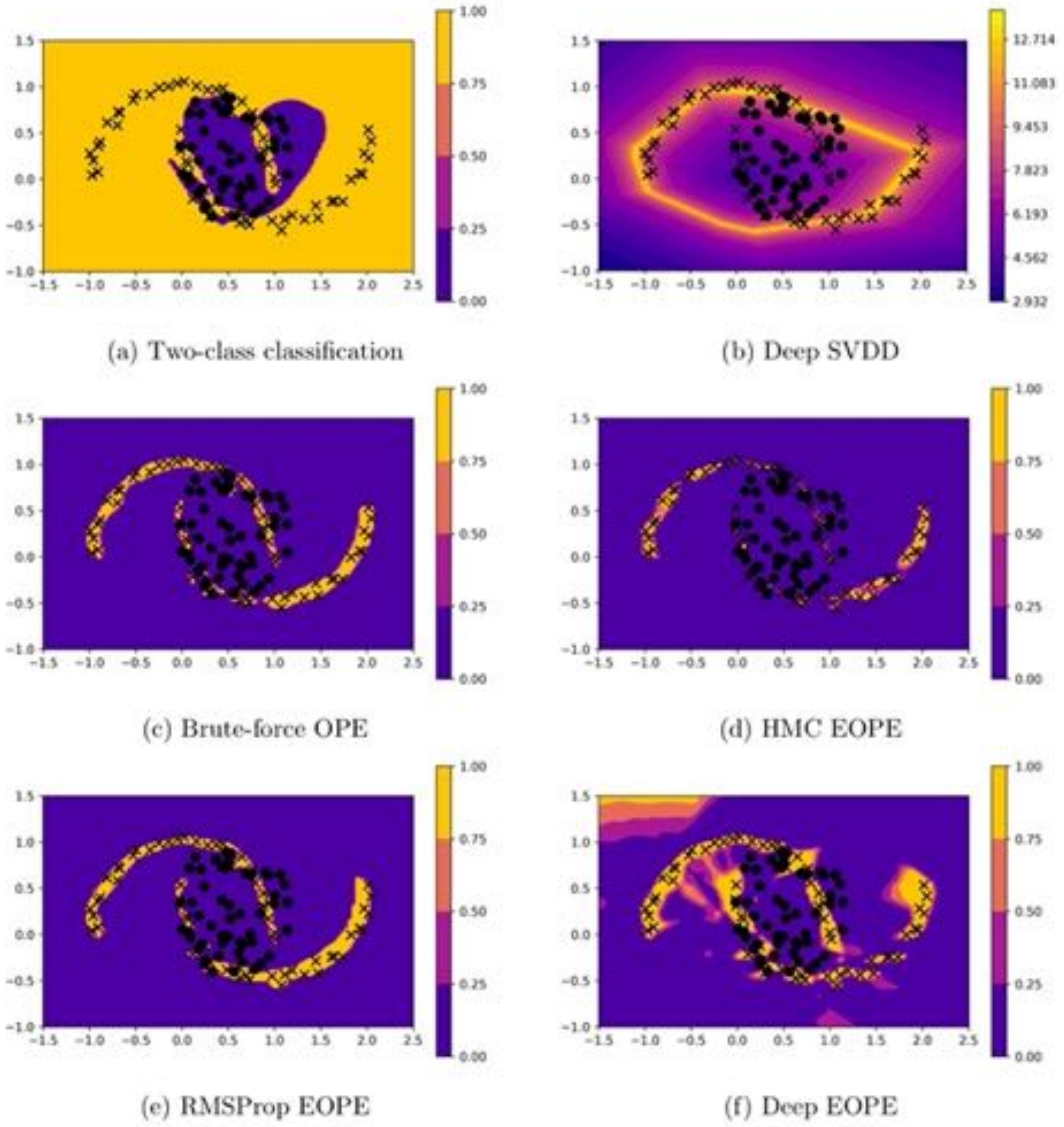
1. Synthetic data (moons)
2. Tabular datasets: HIGGS [1], SUSY [2]
3. Images: CIFAR-10, Omniglot

Comparison

Method	Comment
Robust AE	Robust AutoEncoder (Zhou and Paffenroth, 2017)
Deep SVDD	One-class Deep SVDD (Ruff et al., 2018)
cross-entropy	Baseline, regular 2-class NN
semi-supervised	dimensionality reduction by a deep AutoEncoder followed by a classifier with the cross-entropy loss
brute-force-OPE	**Uniform sampling**
HMC EOPE	**MCMC sampling with energy**
RMSProp EOPE	**RMSProp – based sampling**
Deep-EOPE	**Deep energy sampling**

Synthetic example





Andrey Ustyuzhanin

Synthetic example, dependence on ε

	one class	100	1000	10000	1000000
Robust AE	0.530 ± 0.002	0.530 ± 0.002	0.530 ± 0.002	0.530 ± 0.002	0.530 ± 0.002
Deep SVDD	0.497 ± 0.006	0.497 ± 0.006	0.497 ± 0.006	0.497 ± 0.006	0.497 ± 0.006
cross-entropy	-	0.496 ± 0.017	0.529 ± 0.007	0.566 ± 0.006	0.858 ± 0.002
semi-supervised	-	0.498 ± 0.003	0.522 ± 0.003	0.603 ± 0.002	0.745 ± 0.005
brute-force OPE	0.499 ± 0.009	0.500 ± 0.009	0.520 ± 0.003	0.572 ± 0.005	0.859 ± 0.001
HMC EOPE	0.491 ± 0.000	0.523 ± 0.005	$\mathbf{0.567} \pm 0.008$	$\mathbf{0.648} \pm 0.005$	0.848 ± 0.001
RMSProp EOPE	0.498 ± 0.002	0.494 ± 0.008	0.531 ± 0.008	0.593 ± 0.011	$\mathbf{0.861} \pm 0.000$
Deep EOPE	$\mathbf{0.531} \pm 0.000$	$\mathbf{0.537} \pm 0.011$	0.560 ± 0.008	0.628 ± 0.005	0.860 ± 0.001

Figure 4: Results on HIGGS data set. The first row indicates numbers of negative samples used in training.

1.1e7 instances described by 28 attributes
2-classes.
Well-balanced

	one class	100	1000	10000	1000000
Robust AE	0.394 ± 0.012	0.394 ± 0.012	0.394 ± 0.012	0.394 ± 0.012	0.394 ± 0.012
Deep SVDD	0.541 ± 0.022	0.541 ± 0.022	0.541 ± 0.022	0.541 ± 0.022	0.541 ± 0.022
cross-entropy	-	0.658 ± 0.033	0.736 ± 0.021	0.757 ± 0.036	0.871 ± 0.006
semi-supervised	-	0.715 ± 0.020	0.766 ± 0.009	$\mathbf{0.847} \pm 0.002$	0.876 ± 0.000
brute-force OPE	$\mathbf{0.648} \pm 0.035$	0.678 ± 0.025	0.729 ± 0.029	0.757 ± 0.036	0.871 ± 0.006
HMC EOPE	0.472 ± 0.000	$\mathbf{0.738} \pm 0.019$	$\mathbf{0.770} \pm 0.012$	0.816 ± 0.006	0.877 ± 0.000
RMSProp EOPE	0.443 ± 0.038	0.714 ± 0.019	0.760 ± 0.016	0.807 ± 0.004	0.877 ± 0.000
Deep EOPE	0.468 ± 0.118	0.670 ± 0.054	0.746 ± 0.024	0.813 ± 0.003	$\mathbf{0.878} \pm 0.000$

Figure 5: Results on SUSY data set. The first row indicates numbers of negative samples used in training.

	one class	1	2	4
Robust AE	**0.585** $\pm$ 0.126	0.585 $\pm$ 0.126	0.585 $\pm$ 0.126	0.585 $\pm$ 0.126
Deep SVDD	0.546 $\pm$ 0.058	0.546 $\pm$ 0.058	0.546 $\pm$ 0.058	0.546 $\pm$ 0.058
cross-entropy	-	0.659 $\pm$ 0.093	0.708 $\pm$ 0.086	0.748 $\pm$ 0.082
semi-supervised	-	0.587 $\pm$ 0.109	0.634 $\pm$ 0.109	0.671 $\pm$ 0.093
brute-force OPE	0.549 $\pm$ 0.098	**0.688** $\pm$ 0.087	**0.719** $\pm$ 0.079	**0.757** $\pm$ 0.073
HMC EOPE	0.547 $\pm$ 0.116	0.678 $\pm$ 0.091	0.709 $\pm$ 0.084	0.739 $\pm$ 0.074
RMSProp EOPE	0.565 $\pm$ 0.111	0.678 $\pm$ 0.081	0.715 $\pm$ 0.083	0.746 $\pm$ 0.069
Deep EOPE	0.564 $\pm$ 0.094	0.674 $\pm$ 0.100	0.690 $\pm$ 0.092	0.719 $\pm$ 0.099

Figure 8: Results on CIFAR-10 data set. The first row indicates numbers of original classes selected as negative class, 10 images are sampled from each original class.

	one class	1	2	4
Robust AE	**0.771** $\pm$ 0.221	0.771 $\pm$ 0.221	0.771 $\pm$ 0.221	0.771 $\pm$ 0.221
Deep SVDD	0.640 $\pm$ 0.153	0.640 $\pm$ 0.153	0.640 $\pm$ 0.153	0.640 $\pm$ 0.153
cross-entropy	-	0.799 $\pm$ 0.162	**0.862** $\pm$ 0.115	0.855 $\pm$ 0.125
semi-supervised	-	0.737 $\pm$ 0.134	0.821 $\pm$ 0.104	0.805 $\pm$ 0.121
brute-force OPE	0.591 $\pm$ 0.161	0.724 $\pm$ 0.222	0.765 $\pm$ 0.208	0.825 $\pm$ 0.126
HMC EOPE	0.710 $\pm$ 0.178	0.801 $\pm$ 0.139	0.842 $\pm$ 0.112	0.842 $\pm$ 0.115
RMSProp EOPE	0.678 $\pm$ 0.274	**0.821** $\pm$ 0.143	0.855 $\pm$ 0.112	**0.863** $\pm$ 0.111
Deep EOPE	0.696 $\pm$ 0.172	0.808 $\pm$ 0.140	0.851 $\pm$ 0.110	0.842 $\pm$ 0.122

Figure 9: Results on Omniglot data set. The first row indicates numbers of original classes selected as negative class, 10 images are sampled from each original class. Greek, Braille and Futurama alphabets are used as normal classes.

Discussion and Outlook

https://arxiv.org/abs/1906.06096

- Initial approach to the problem of combining **measure-based** heuristics with **discriminative ones** works!
- Requires tuning of hyperparameter ε, (thus the name of the method: 1+ ε)
  - MCMC is slow
  - Uniform works even for images

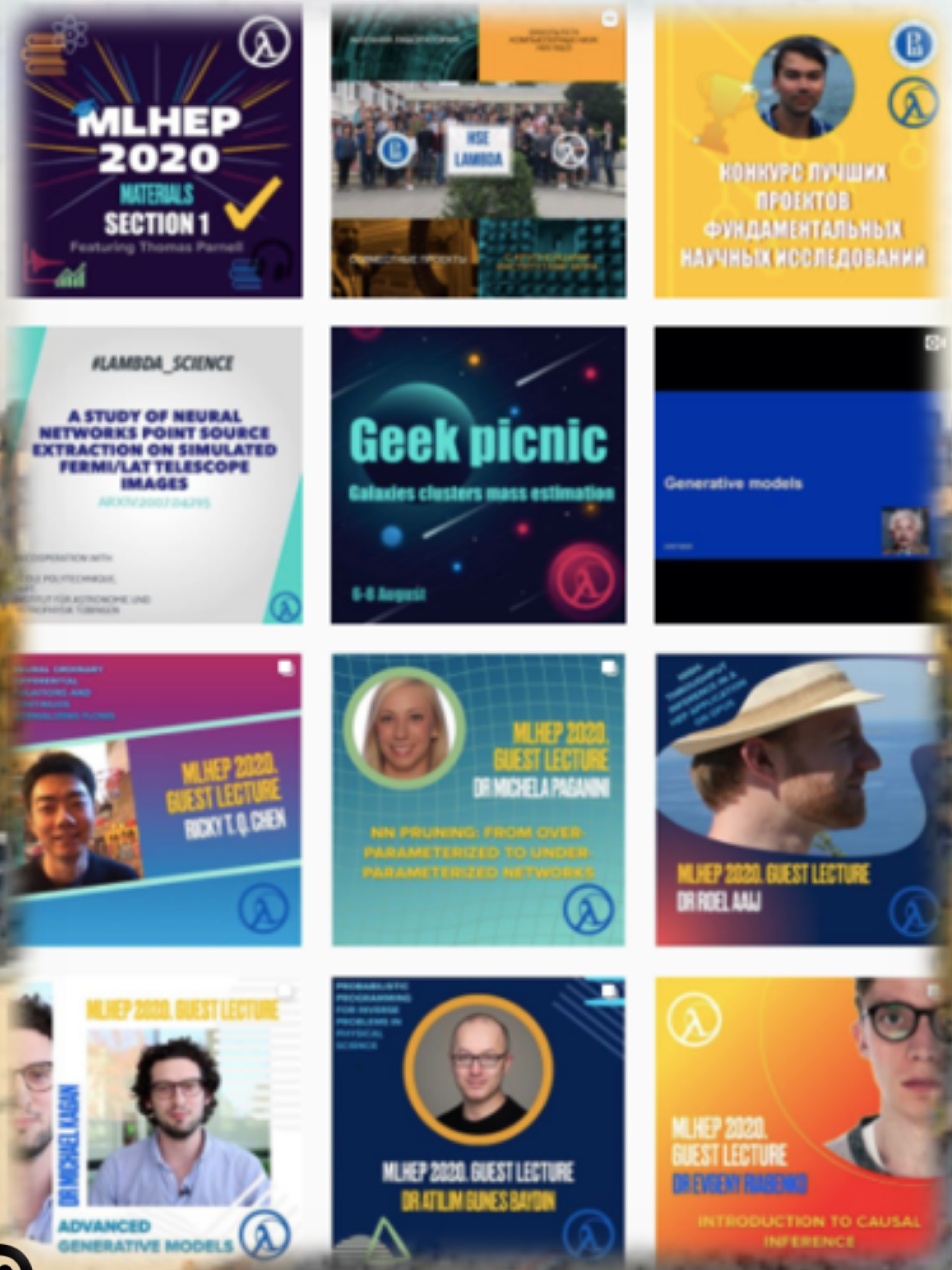
Next steps

- Use normalizing flows for sampling pseudo-anomalies
- What is the relation to adversarial attack robustness?

Conclusion

- Generative models can be applied to a variety practical problems:
  - Simulation speed-up, optimization, anomalies
- Open questions:
  - Uncertainty estimation (say, through Bayesian inference, interpretability)
  - Generate semantically rich structures
  - Meaningful representation learning
  - Efficient and interpretable generative model ensembling

Thank you for the attention!



austyuzhanin@hse.ru

anaderiRu

[hse_lambda]()

Andrey Ustyuzhanin

Backup

References

1. D. Whiteson, Higgs dataset. UCI machine learning repository, 2014
2. D. Whiteson, Susy dataset. UCI machine learning repository, 2014.
3. T.Kim, Y.Bengio, Deep Directed Generative Models with Energy-Based Probability Estimation, 2016, arXiv:1606.03439
4. Wu Haiyan ; Yang Haomin ; Li Xueming ; Ren Haijun, Semi-Supervised Autoencoder: A Joint Approach of Representation and Classification, 2015
5. D.Kingma, D.Rezende, S.Mohamed, M.Welling , Deep Generative Models, 2014, arXiv:1406.5298v2