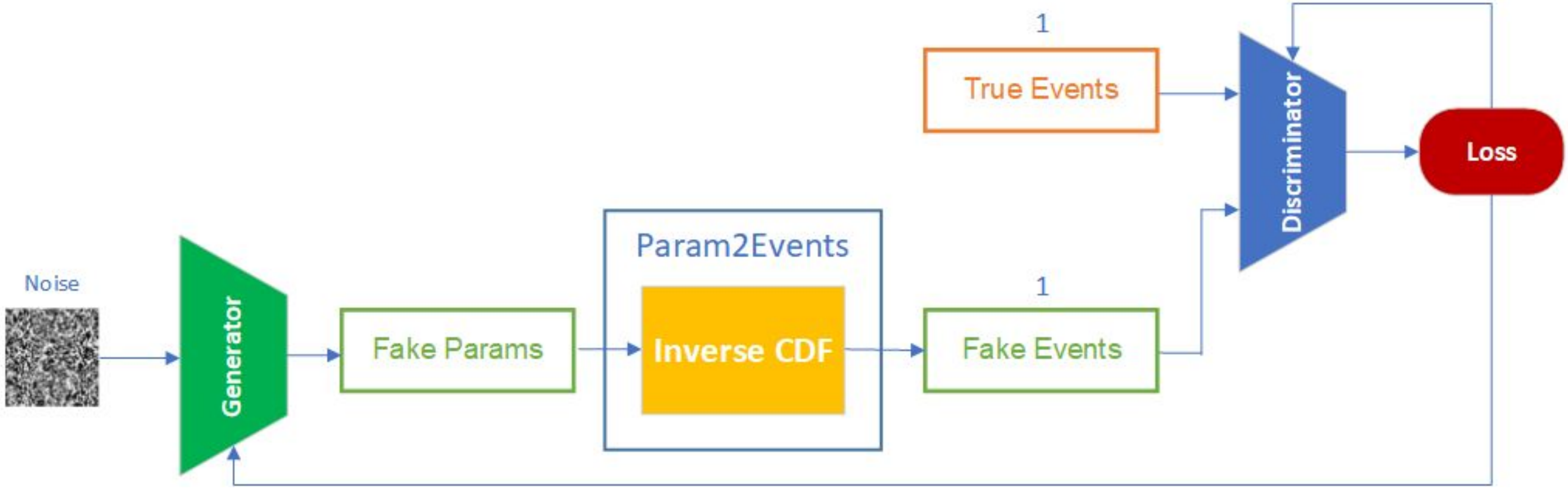
Collaboration Meeting

Oct. 20 - 22, 2022

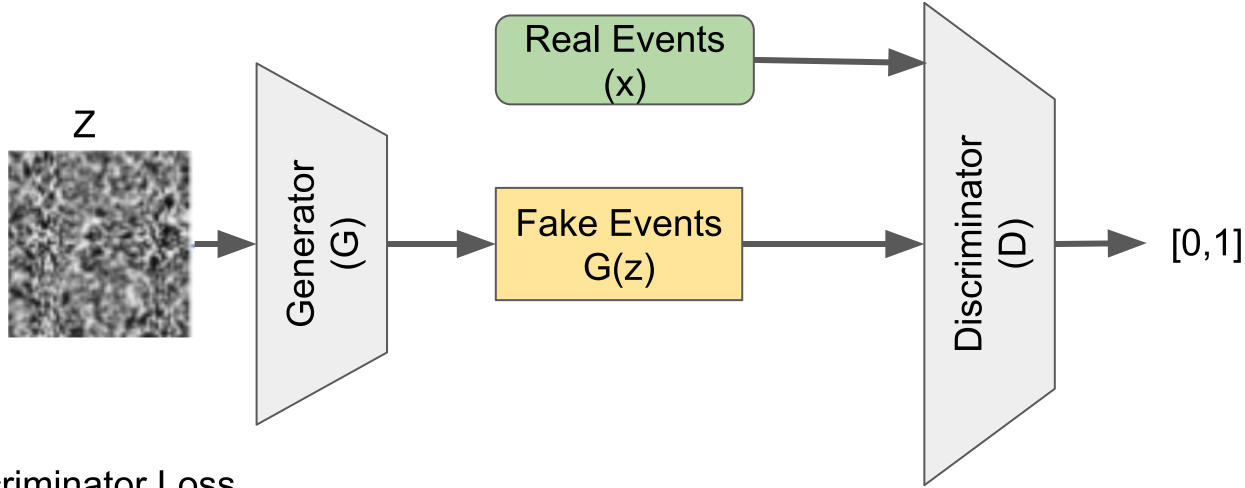
Outline

- GAN fundamentals (How it works…)
- Current workflow used in the proxy app and results
- Multiple discriminators workflow
- MCMC and batch-events training
- Surrogate model for module-2
- Deep Hyper - HPO
- Path forward

Architecture



How GAN works?



Discriminator Loss

$$L^{(D)} = \max[log(D(x)) + log(1 - D(G(z)))]$$

Generator Loss

$$L^{(G)} = \min[log(D(x)) + log(1 - D(G(z)))]$$

$$L = \min_G \max_D [log(D(x)) + log(1 - D(G(z)))]$$

$$\min_G \max_D V(D,G) = \min_G \max_D \left(E_{x \sim P_{data}(x)}[logD(x)] + E_{z \sim P_z(z)}[log(1 - D(G(z)))]\right)$$

https://arxiv.org/pdf/1406.2661.pdf

GAN Fundamentals

- **Generator G**
  - A Function: Input z, Output x
  - Given a prior distribution $P_{prior}(z)$, a probability distribution $P_G(x)$ is defined by function G
- **Discriminator D**
  - A Function: Input x, Output a scalar
  - Evaluate the difference between $P_G(x)$ and $P_{data}(x)$

Kullback–Leibler Divergence

- **Kullback–Leibler divergence (Relative Entropy)**
  - measures how one probability distribution is different from a reference probability distribution
  - Given probability distributions *P* and *Q*
    - Discrete version

$$D_{\mathrm{KL}}(P||Q) = -\sum_{x} P(x)\log\left(\frac{Q(x)}{P(x)}\right)$$

    - Continuous version

$$D_{\mathrm{KL}}(P||Q) = -\int P(x)\log\left(\frac{Q(x)}{P(x)}\right)dx$$

Properties of Kullback–Leibler Divergence

- **Explanation of KL divergence**

Cross Entropy of *P* and *Q*

$$D_{\mathrm{KL}}(P||Q) = -\sum_x P(x)\log\left(\frac{Q(x)}{P(x)}\right)$$

$$= -\sum_x P(x)\log Q(x) - \left(-\sum_x P(x)\log P(x)\right)$$

Entropy of *P*

- **Properties of KL divergence**
  - Non-symmetric
  - Non-negative

Jensen-Shannon Divergence

- **Jensen-Shannon Divergence**
  - Measures the similarity between two probability distributions
  - A symmetrized and smoothed version of the Kullback–Leibler divergence
  - Definition

$$JSD(P||Q) = \frac{1}{2} D_{\mathrm{KL}}(P||M) + \frac{1}{2} D_{\mathrm{KL}}(M||Q)$$

where

$$M = \frac{1}{2}(P + Q)$$

  - Bounds

$$0 \leq JSD(P||Q) \leq \log(2)$$

GAN Loss Function

- **An optimization problem**
  - Find an optimal generator G* such that

$$G^*=\arg\min_G \max_D V(G,D)$$

  - A minimax algorithm
- **Cross entropy loss**
  - $V = E_{x\sim P\_data}[\log D(x)] + E_{x\sim P\_G}[\log(1-D(x))]$

$\max_D V(G, D)$

- $\max_D V(G, D)$
  - Given a generator G
  - $\max_D V(G,D)$ evaluates the "difference" between P_G and P_{data}
- **What is the optimal D^* that maximize $V(G, D)$?**

$$V = E_{x \sim P_{data}}[\log D(x)] + E_{x \sim P_G}[\log(1 - D(x))]$$

$$= \sum_x P_{data}(x) \log D(x) + \sum_x P_G(x) \log(1 - D(x))$$

Then

$$D^* = P_{data}(x) / (P_{data}(x) + P_G(x))$$

$\min_G \max_D V(G,D)$

$\max_D V(G,D)$
$= V(G,D^*)$ **where** $D^* = P_{data}(x)/(P_{data}(x) + P_G(x))$

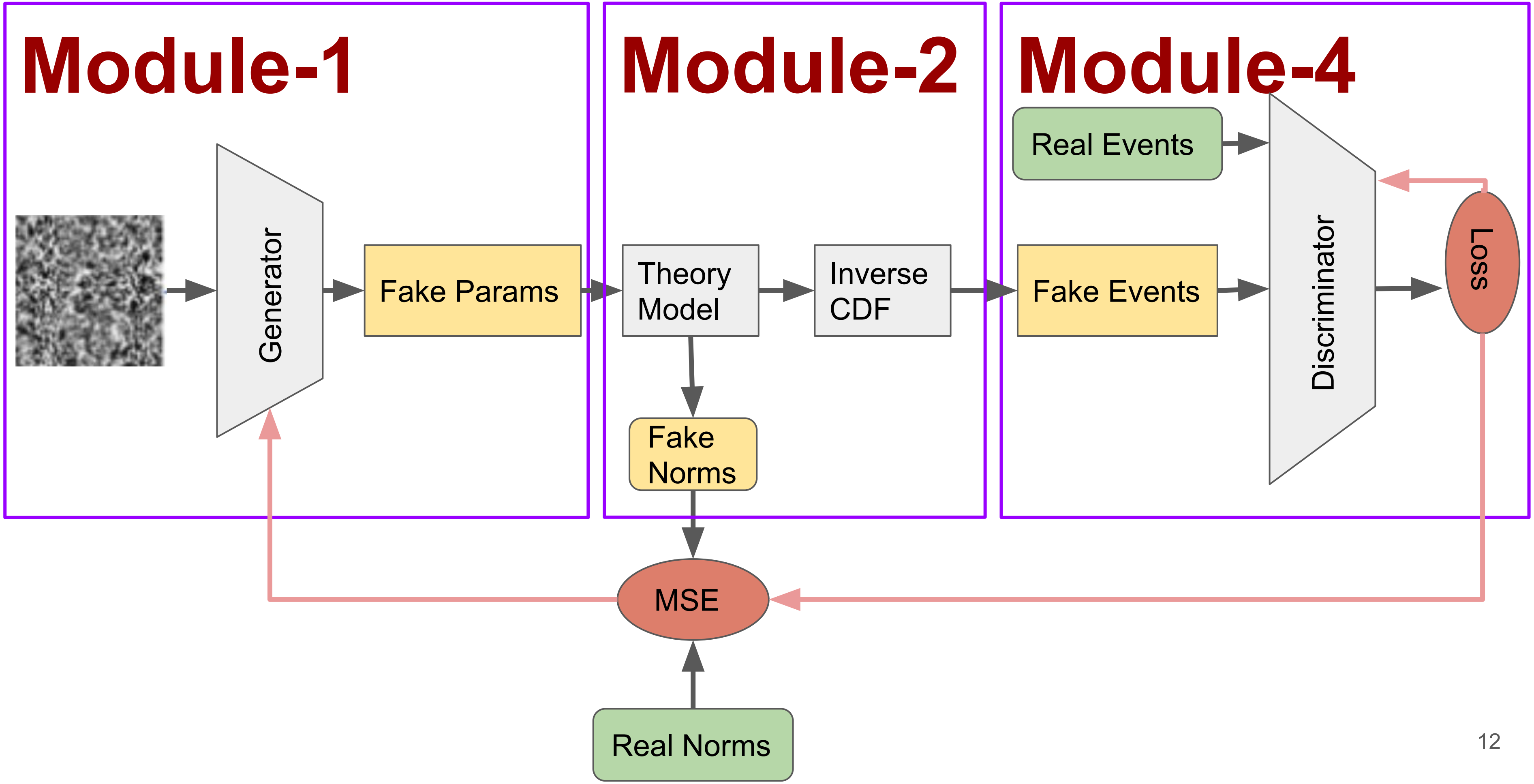
$$= E_{x\sim P_{data}}[\log D^*(x)] + E_{x\sim P_G}[\log(1 - D^*(x))]$$
$$= \sum_x P_{data}(x)\log D^*(x) + \sum_x P_G(x)\log(1 - D^*(x))$$
$$= -2\log 2 + 2JSD(P_{data}||P_G)$$

What is G^* with $\min_G \max_D V(G,D)$?

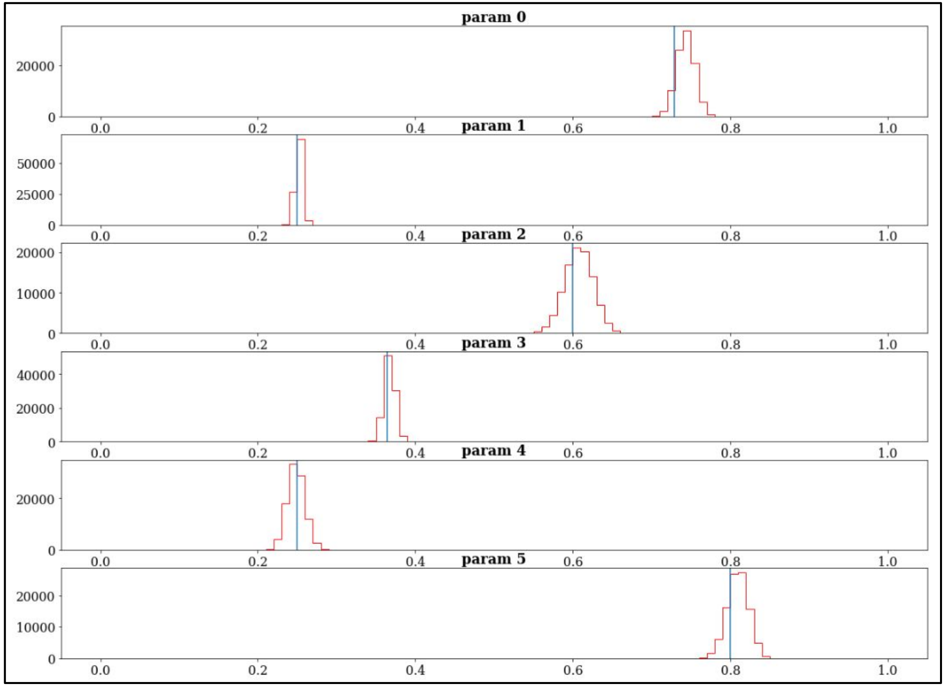
$$JSD(P_{data}||P_G) = 0$$

i.e., $P_{data} = P_G$

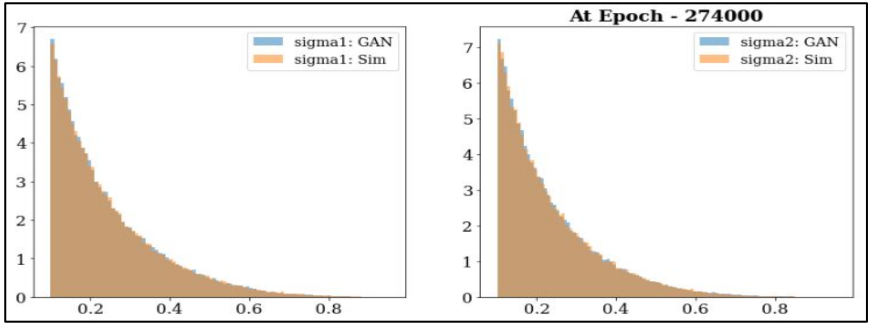
Current Workflow implementation



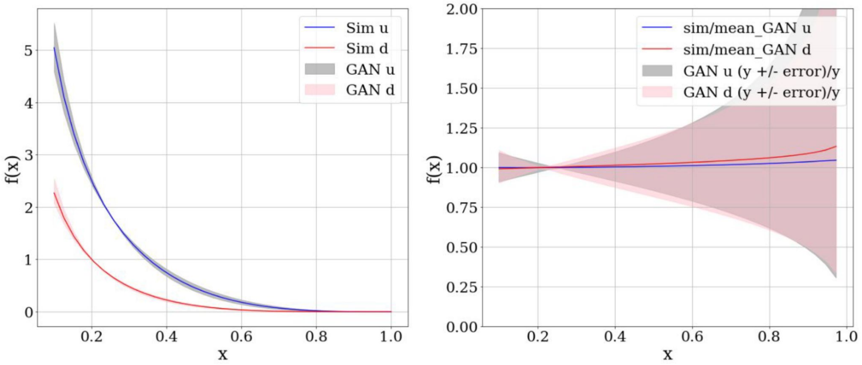
Preliminary Results on the proxy example



Events



PDF

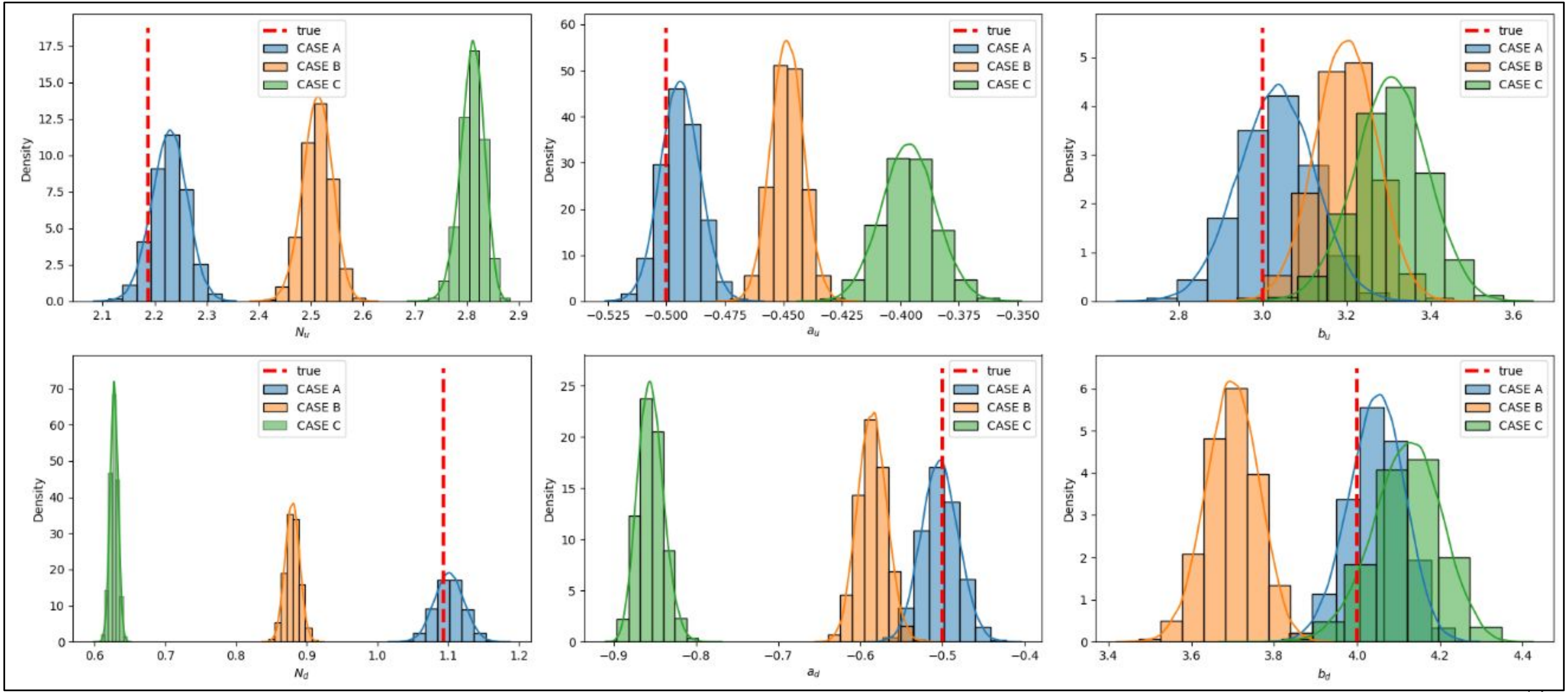


CASE A:
σ1=100,000
σ2=50,000

CASE B:
σ1=10,000
σ2=5,000

CASE C:
σ1=1000
σ2=500

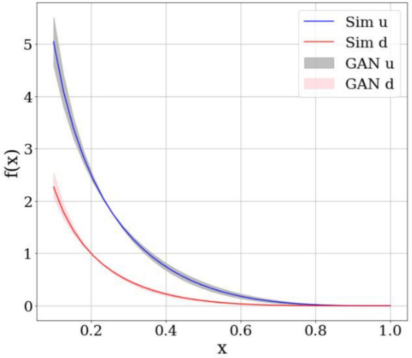
True parameters: [2.1875, -0.5, 3, 1.09375, -0.5, 4]

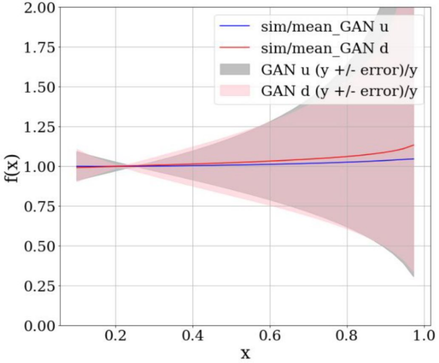


CASE A:	CASE B:	CASE C:
σ1=100,000	σ1=10,000	σ1=1000
σ2=50,000	σ2=5,000	σ2=500

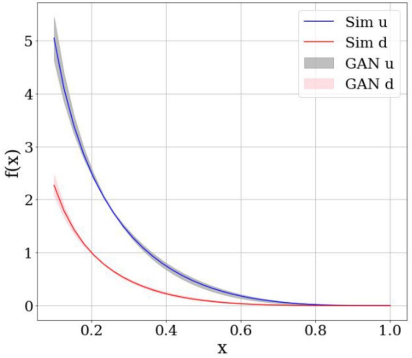
True parameters: [2.1875, -0.5, 3, 1.09375, -0.5, 4]

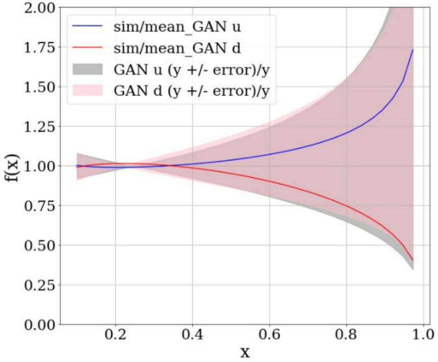
CASE A



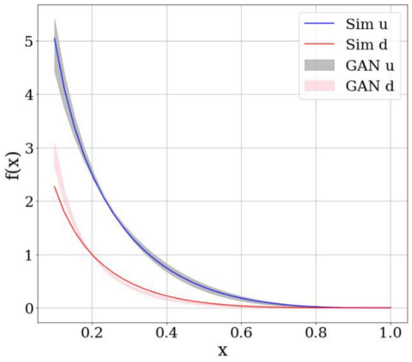


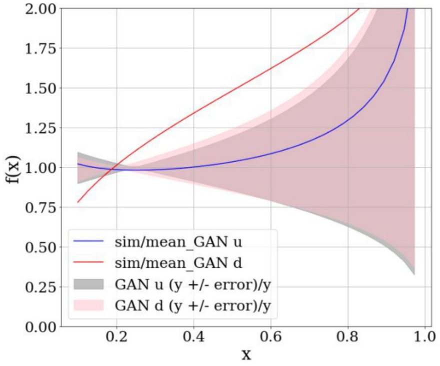
CASE B



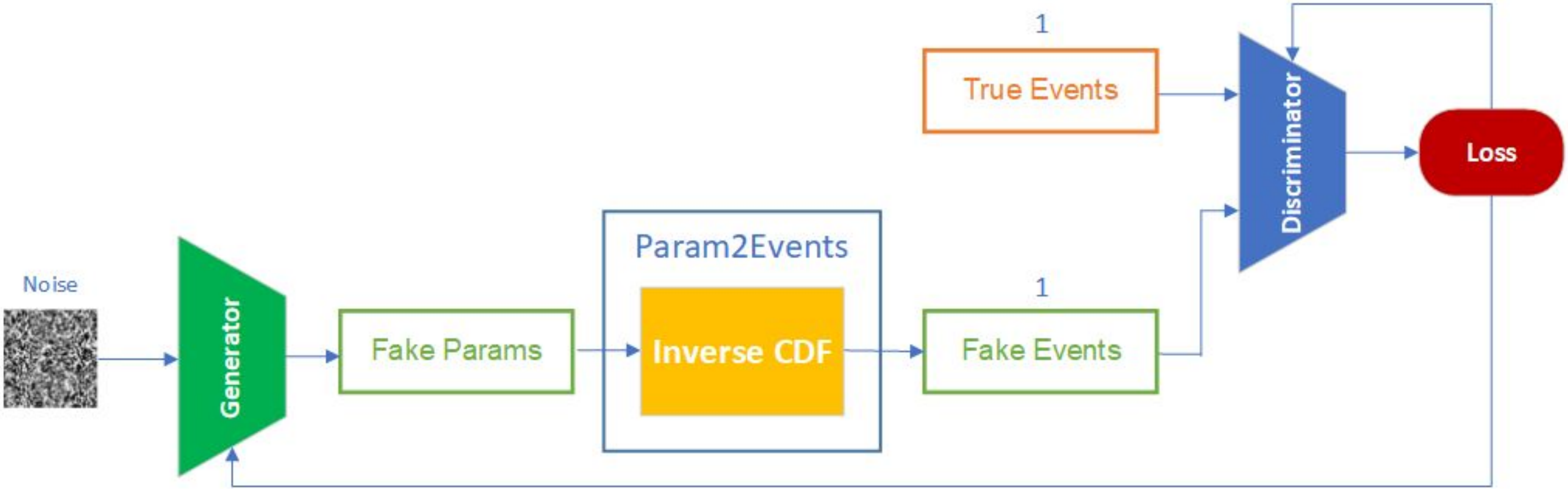


CASE C





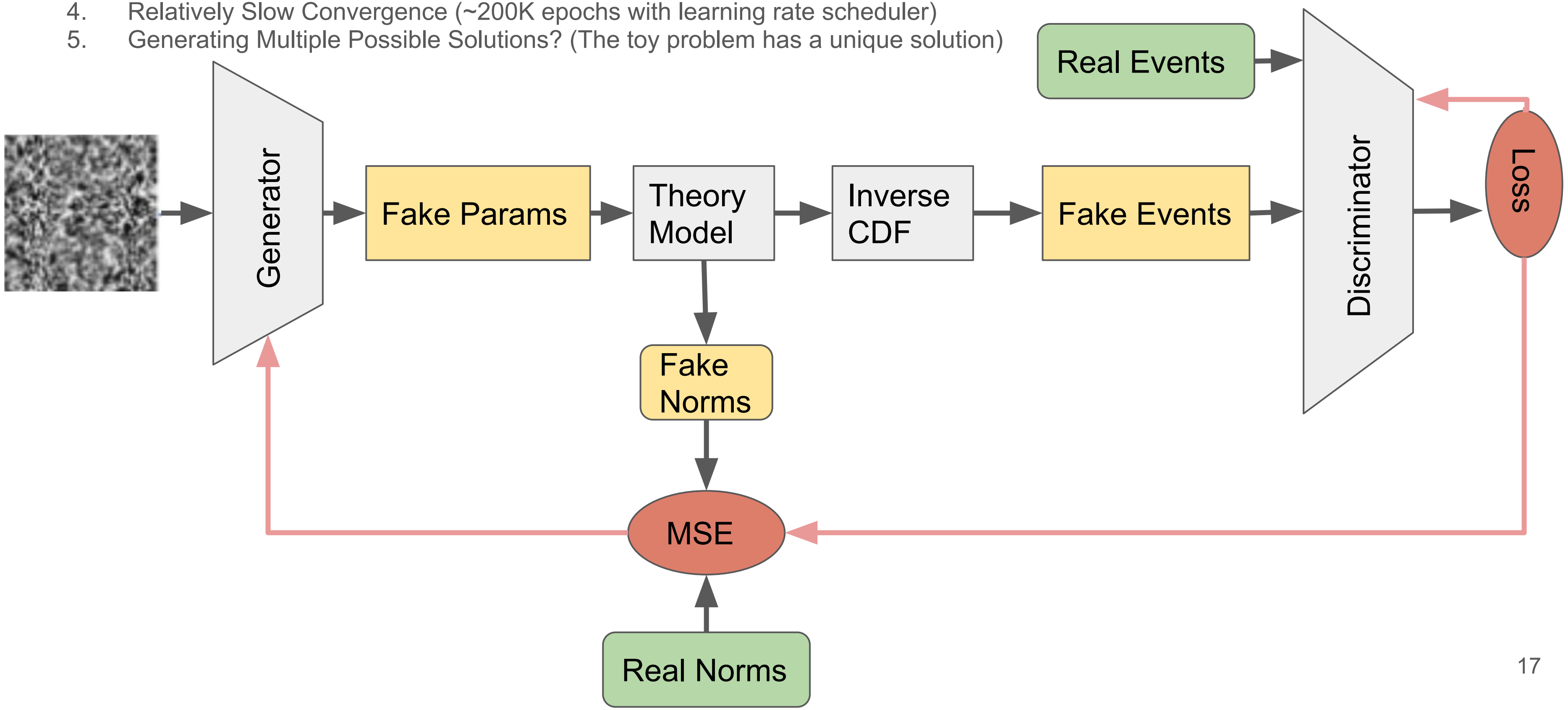
Roadblocks



1. Unable to Precisely Recover the Params (solved by regulating norms)
2. Insensitive Gradient (solved)
3. Divided by Zero in Inverse CDF (solved)
4. Relatively Slow Convergence (~200K epochs with learning rate scheduler)
5. Generating Multiple Possible Solutions? (The toy problem has a unique solution)

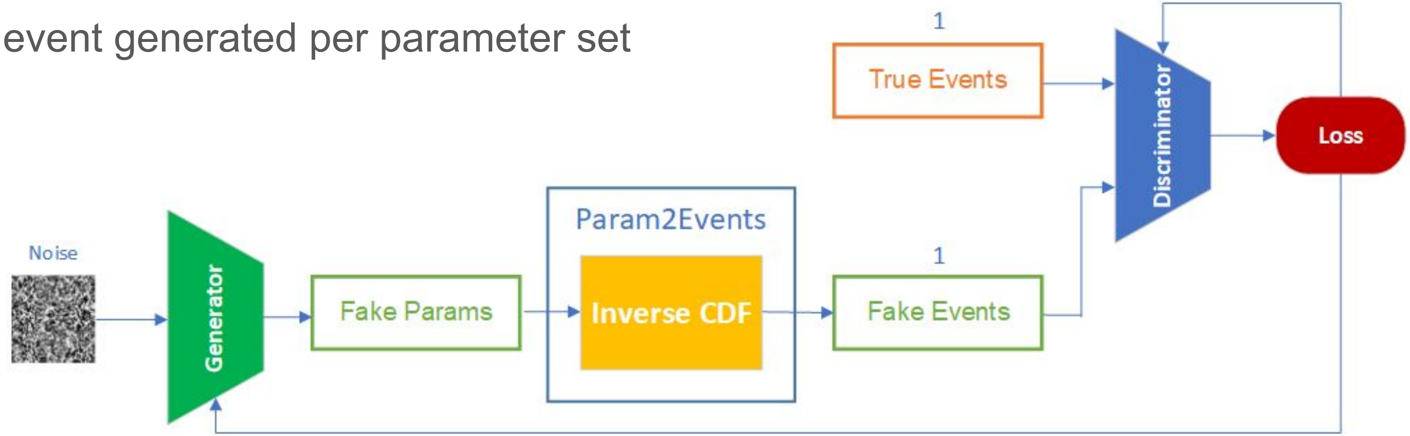
Roadblocks

1. Unable to Precisely Recover the Params (solved by regulating norms)
2. Insensitive Gradient (solved)
3. Divided by Zero in Inverse CDF (solved)
4. Relatively Slow Convergence (~200K epochs with learning rate scheduler)
5. Generating Multiple Possible Solutions? (The toy problem has a unique solution)

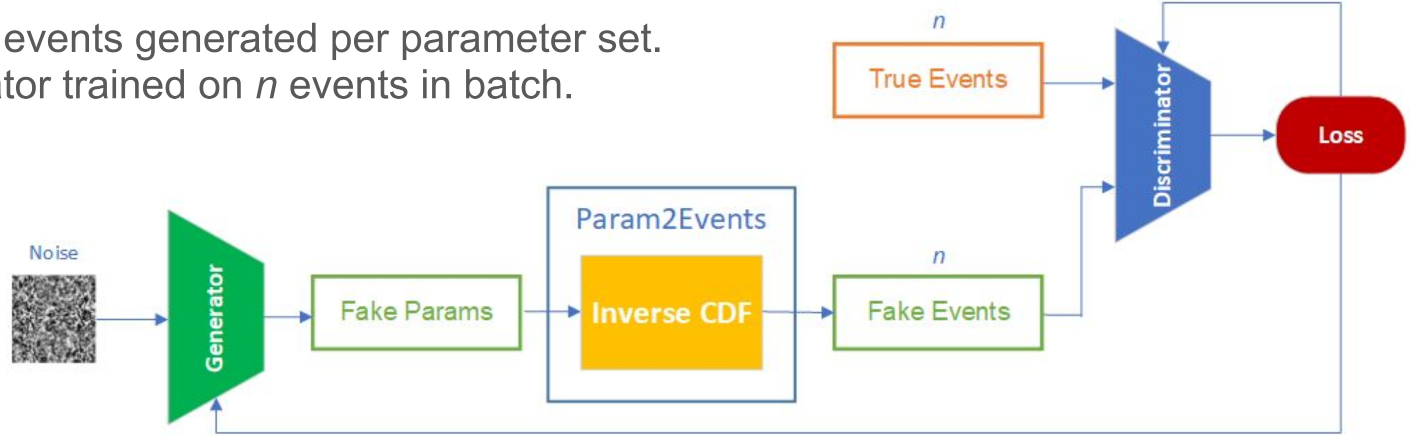


1 event vs. *n* events

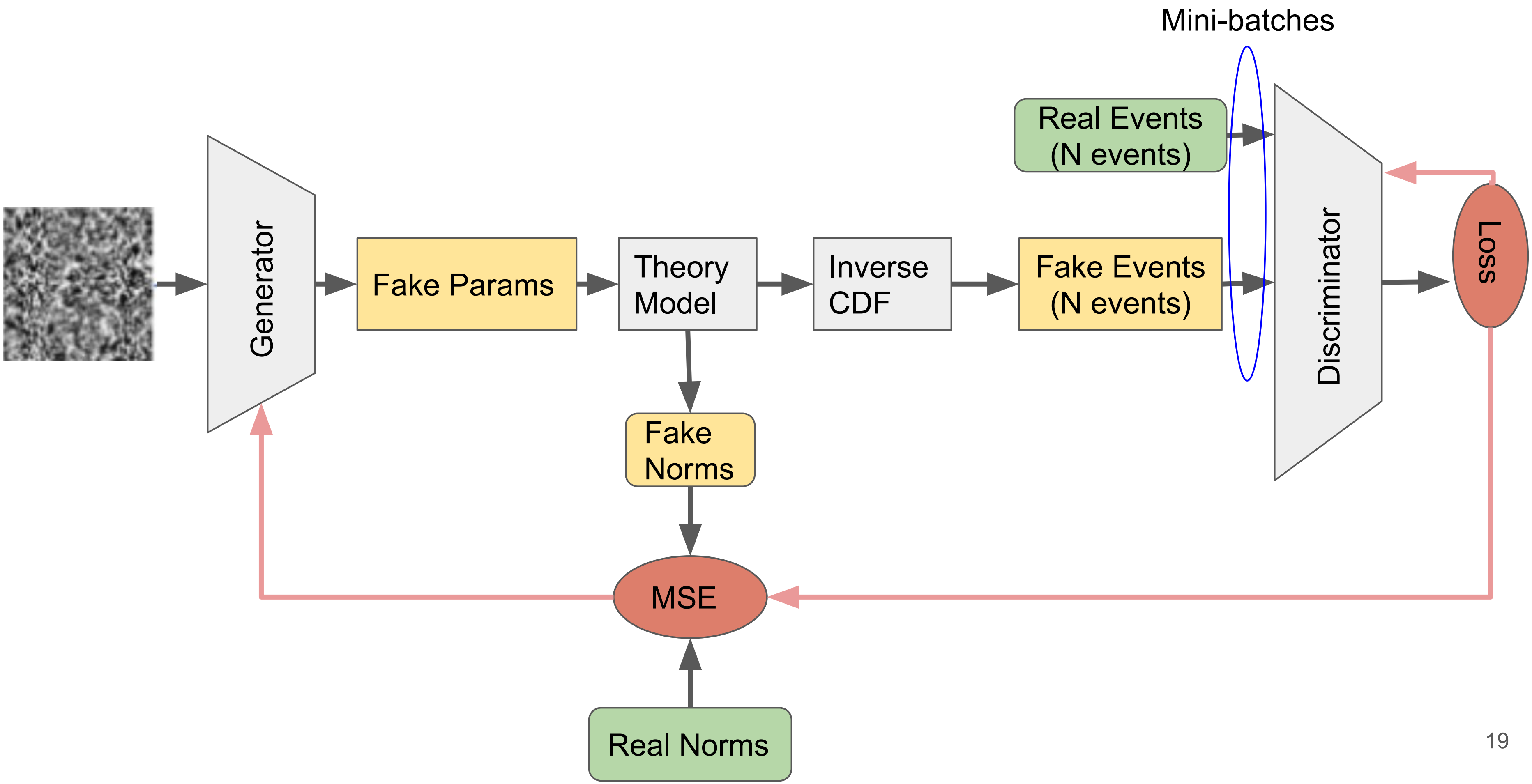
1 event: 1 event generated per parameter set



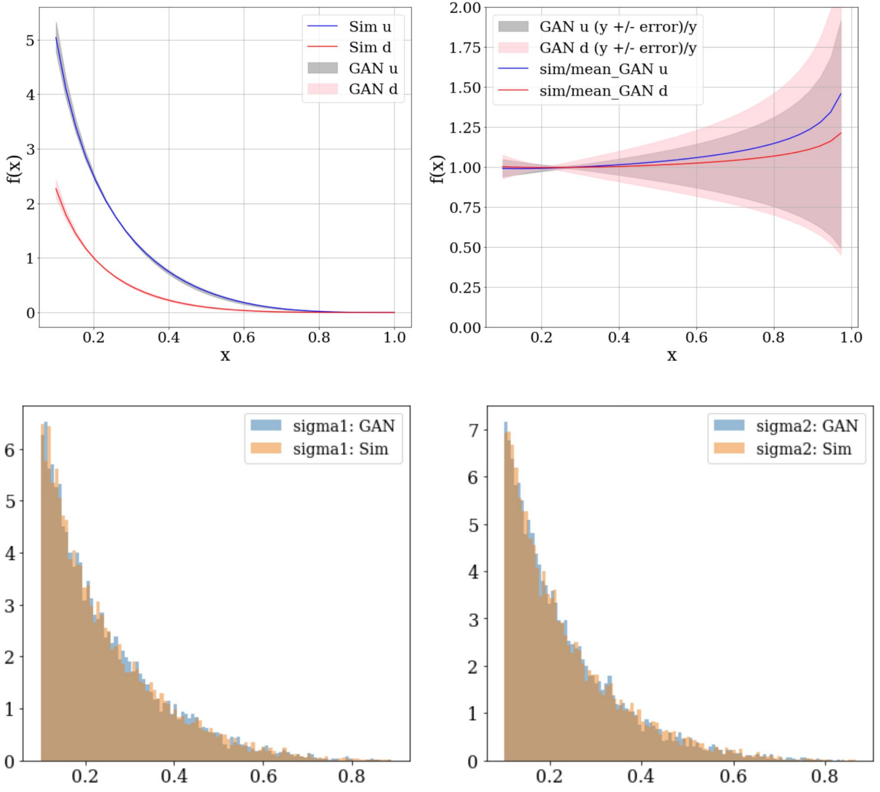
n event: *n* events generated per parameter set.
Discriminator trained on *n* events in batch.



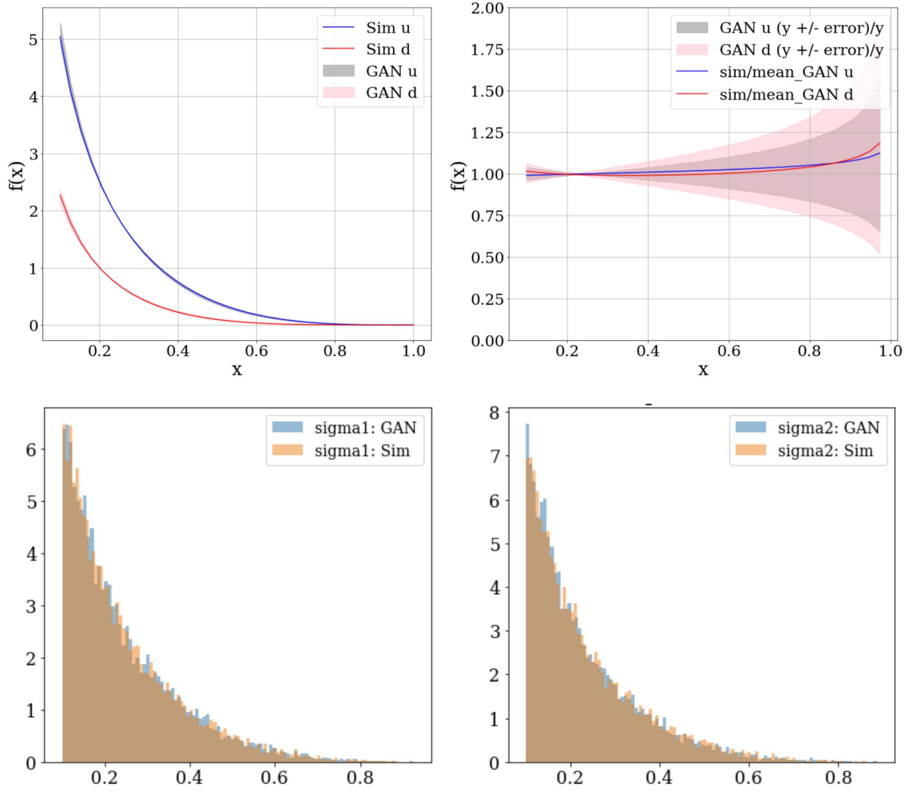
1 event vs. *n* events per parameter set



Training with *n* events - Preliminary Results

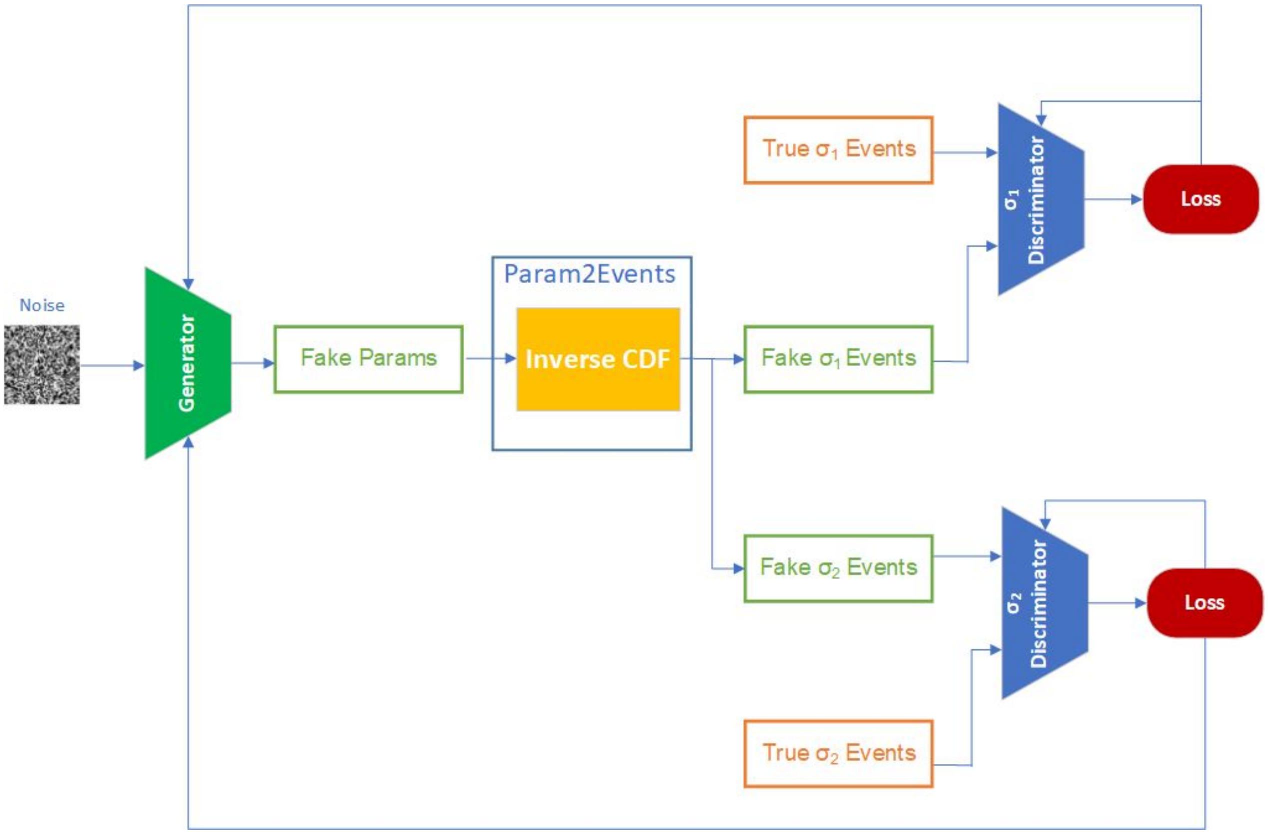


Num events per parameter set = 10
Batch size = 100
Num epochs = ~30,000

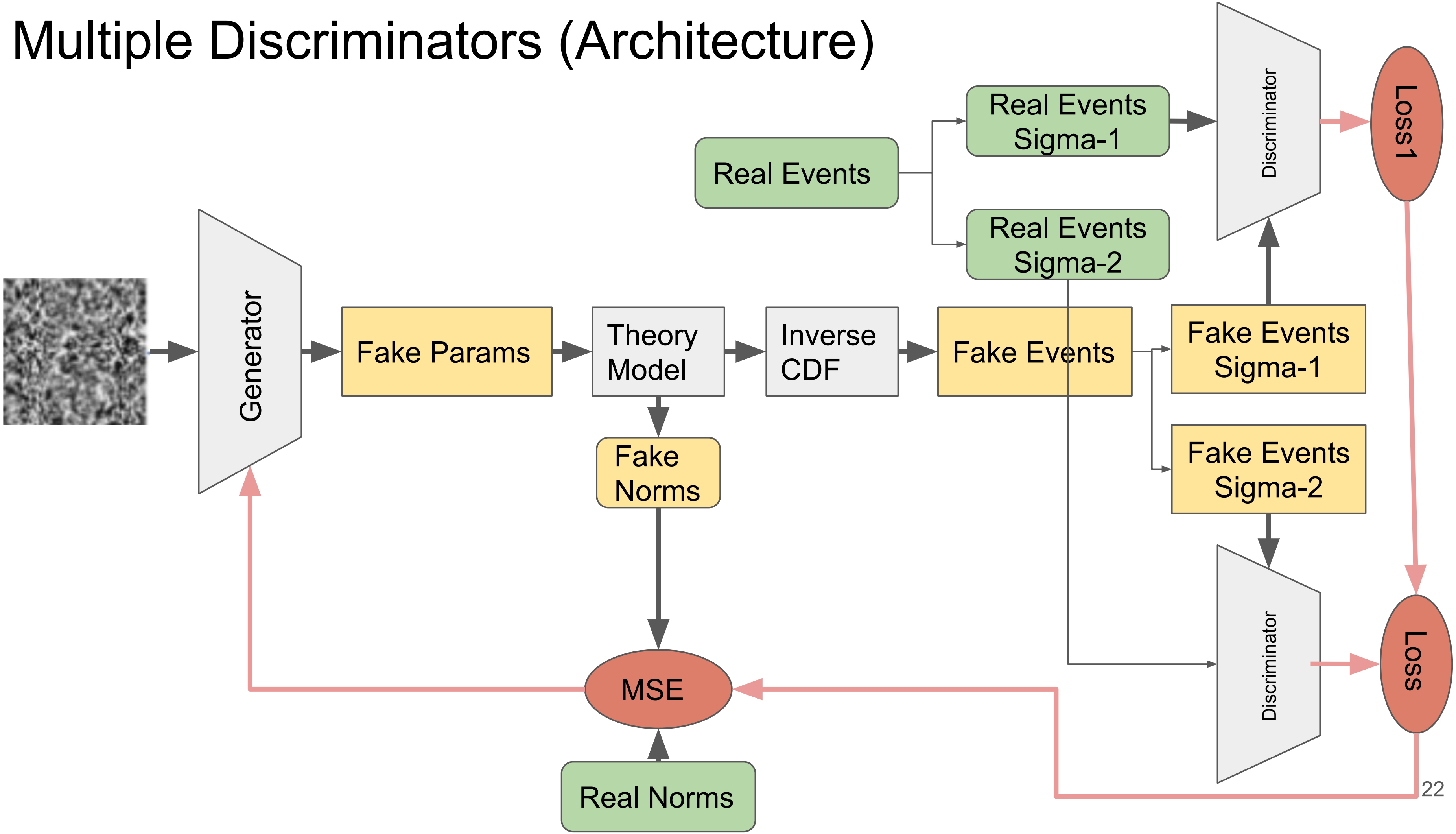


Num events per parameter set = 1
Batch size = 100
Num epochs = ~200,000

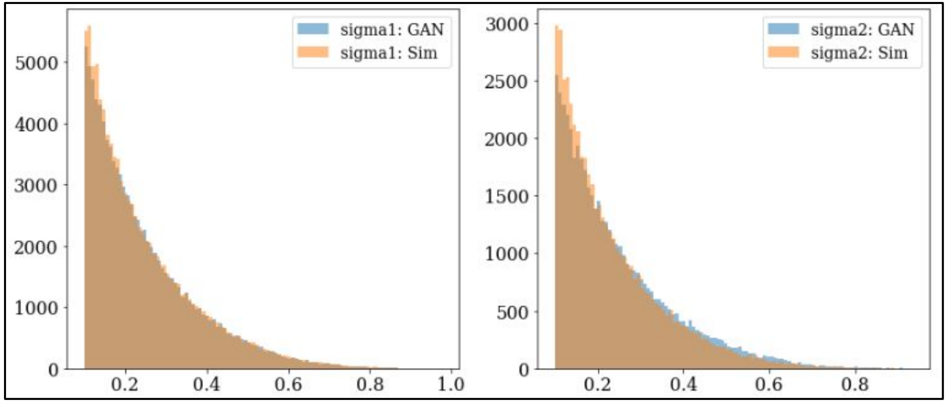
Two Discriminators (Architecture)

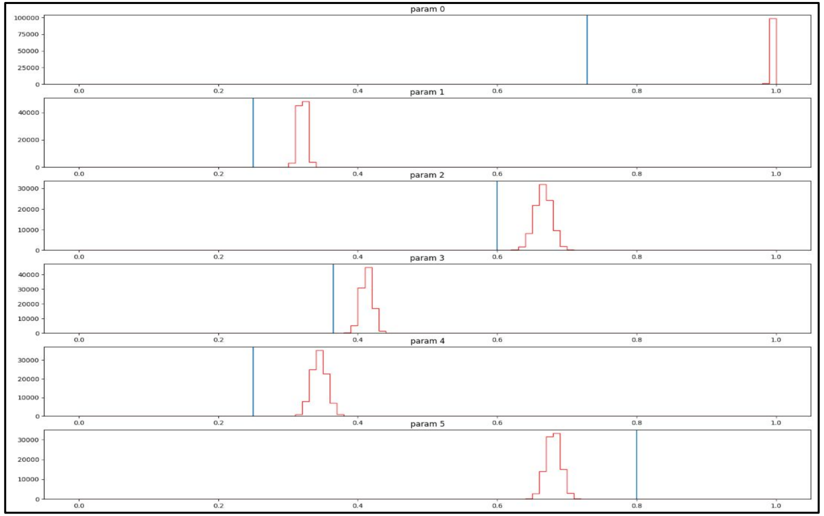


Multiple Discriminators (Architecture)



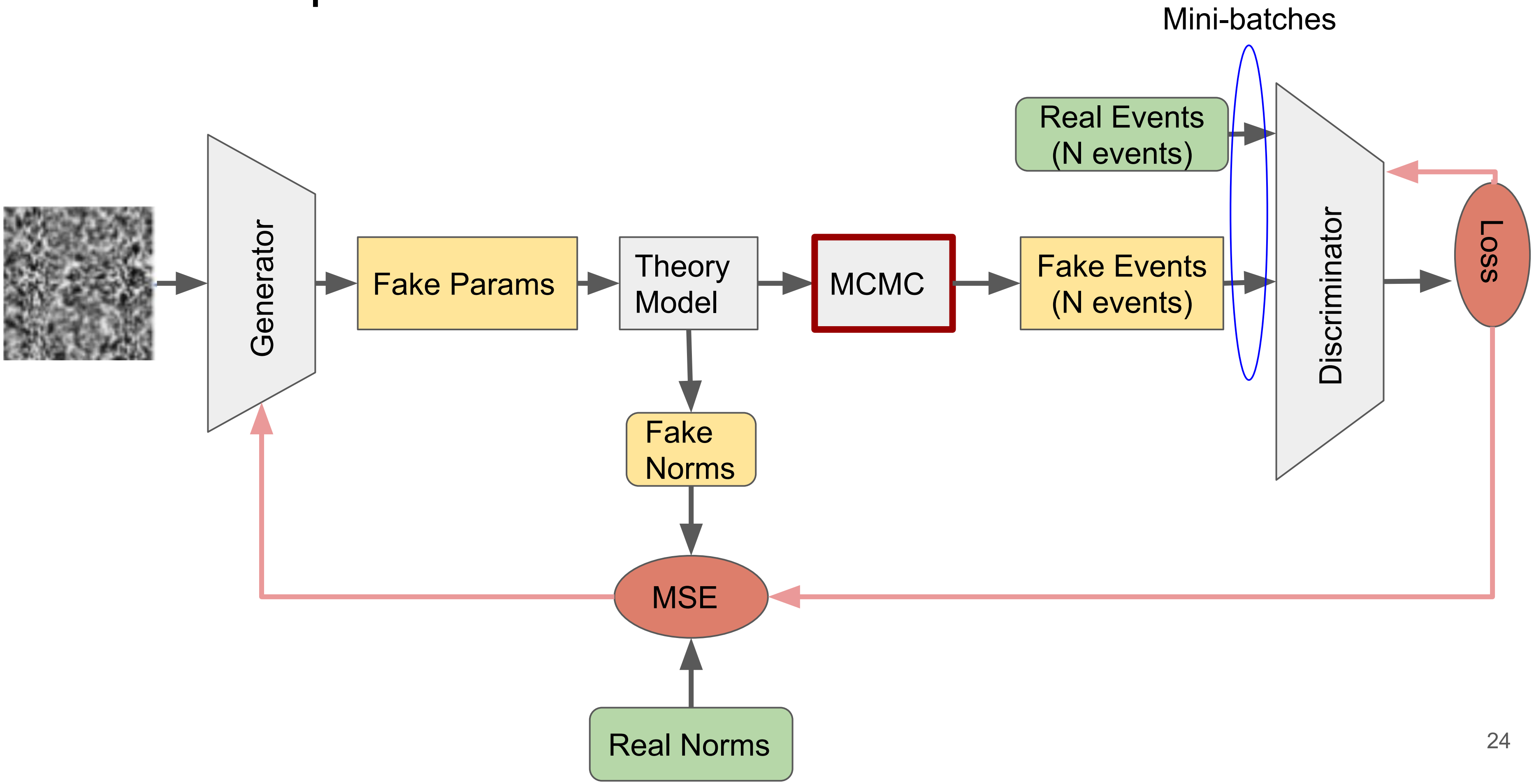
Multiple Discriminators Approach





Less sensitive as compared to training with single discriminator

MCMC: Replace Inverse CDF with MCMC



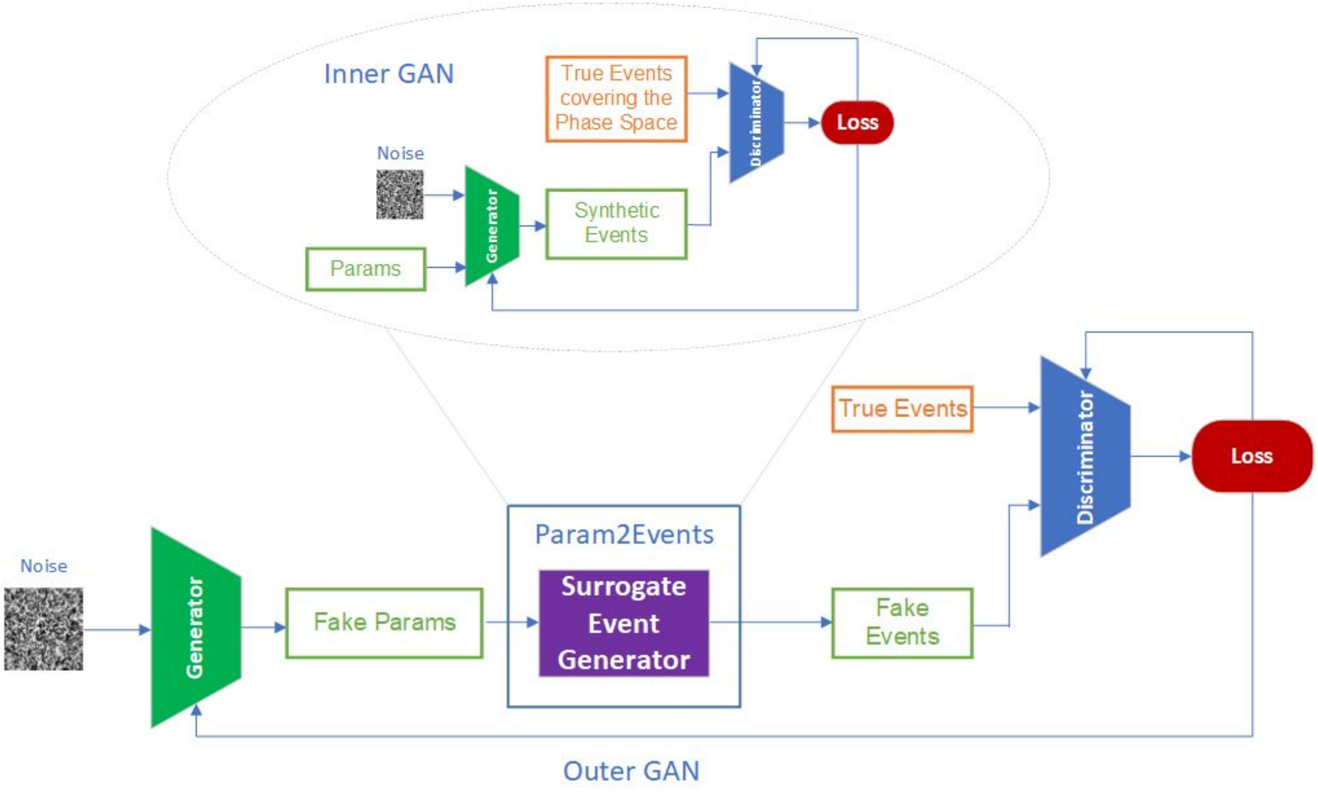
MCMC

Implementation Challenges:

1. No longer one event per step
2. If … then … else … : not favored by current framework
3. Gradient through MCMC
4. Warm-up problem??

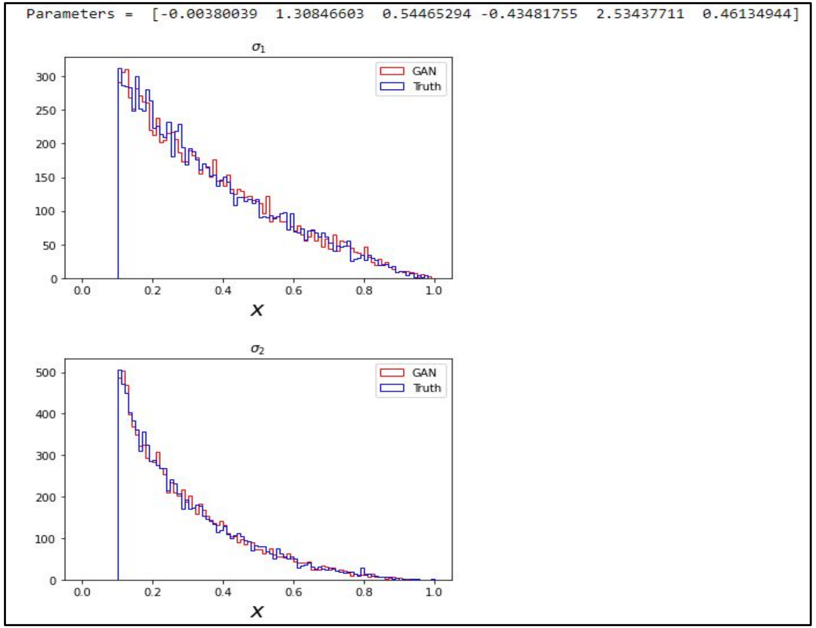
Tensorflow has an MCMC package

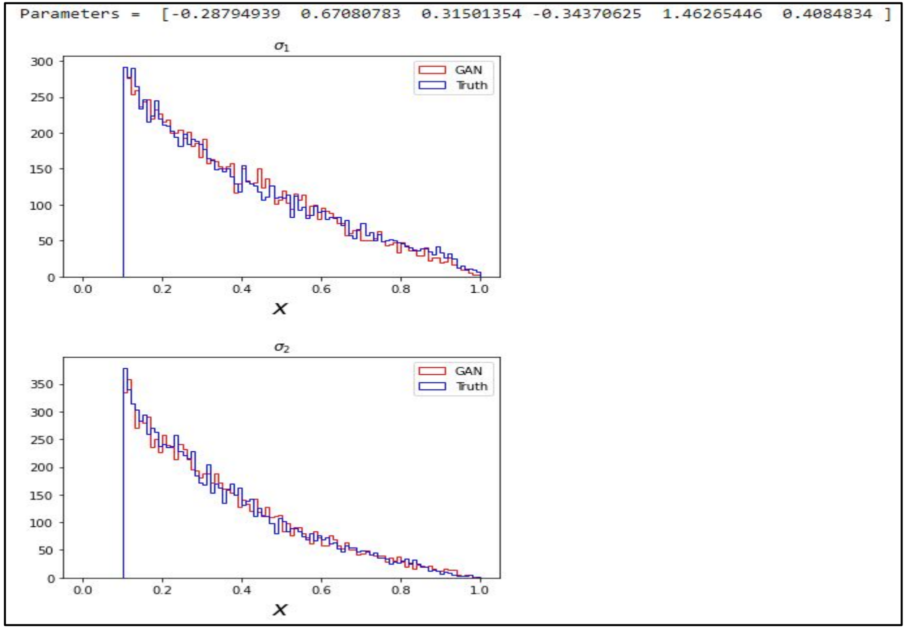
Surrogate Event Generator (GAN)



Surrogate Event Generator (GAN)

Preliminary results





Not as precise as inverse CDF yet

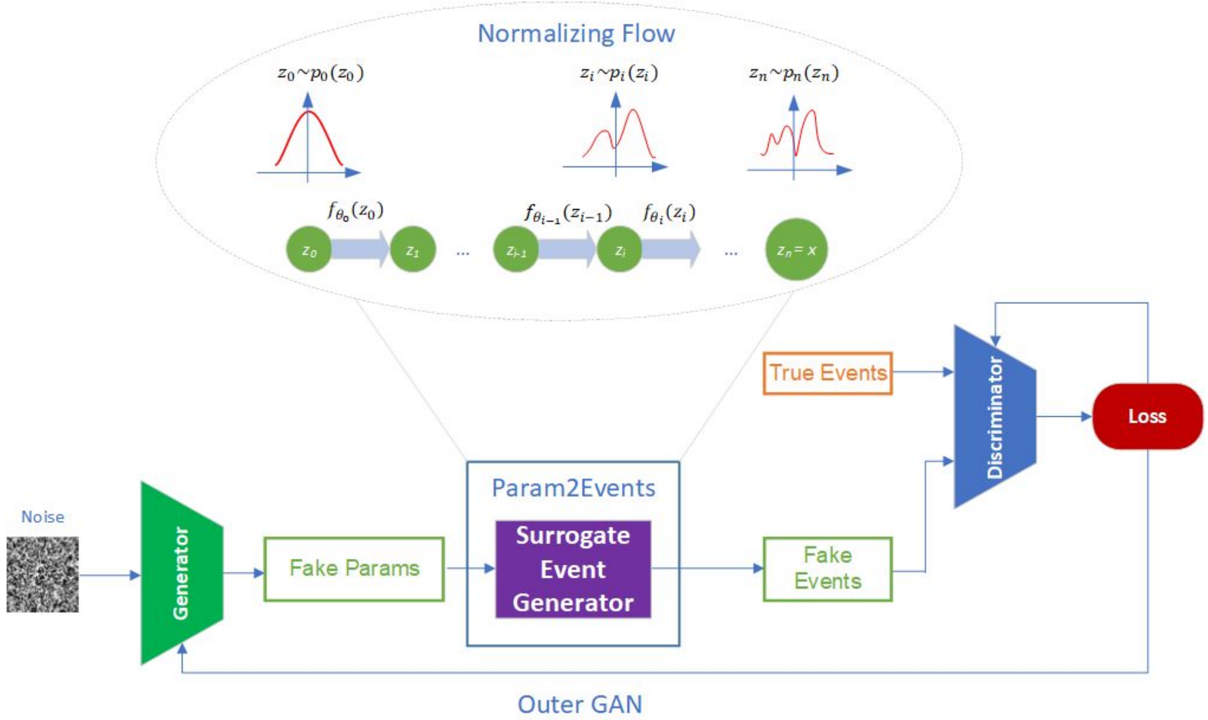
Surrogate Event Generator - Normalizing Flow

Advantages:

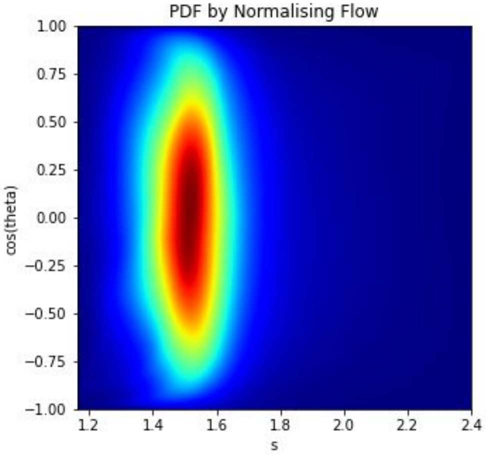
- NF can be supervised by PDF and/or events
- Can generate PDF

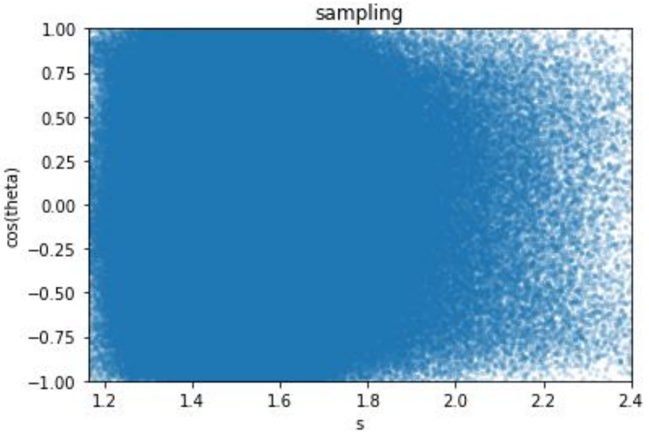
Disadvantage:

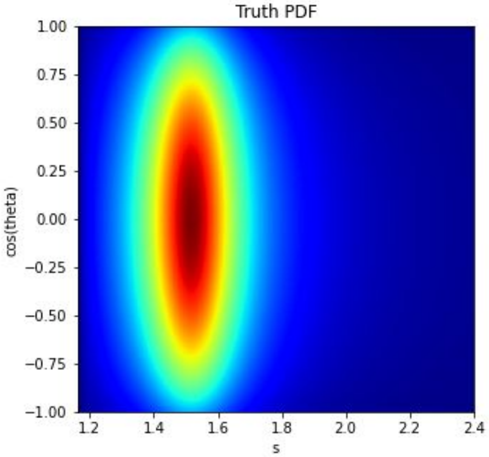
- Not as good approximation as GAN

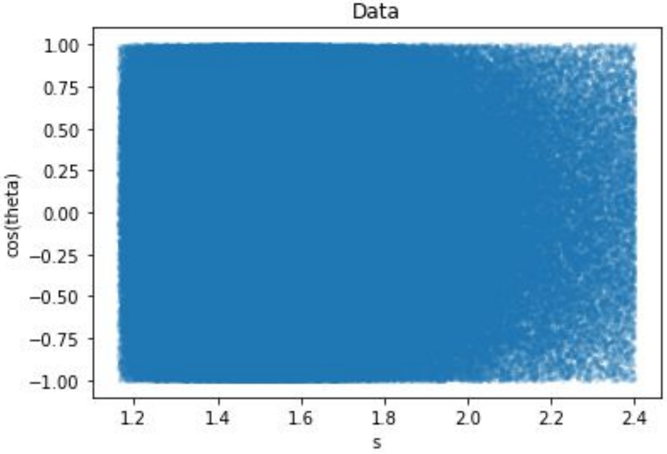


Normalizing Flow – A Toy Problem

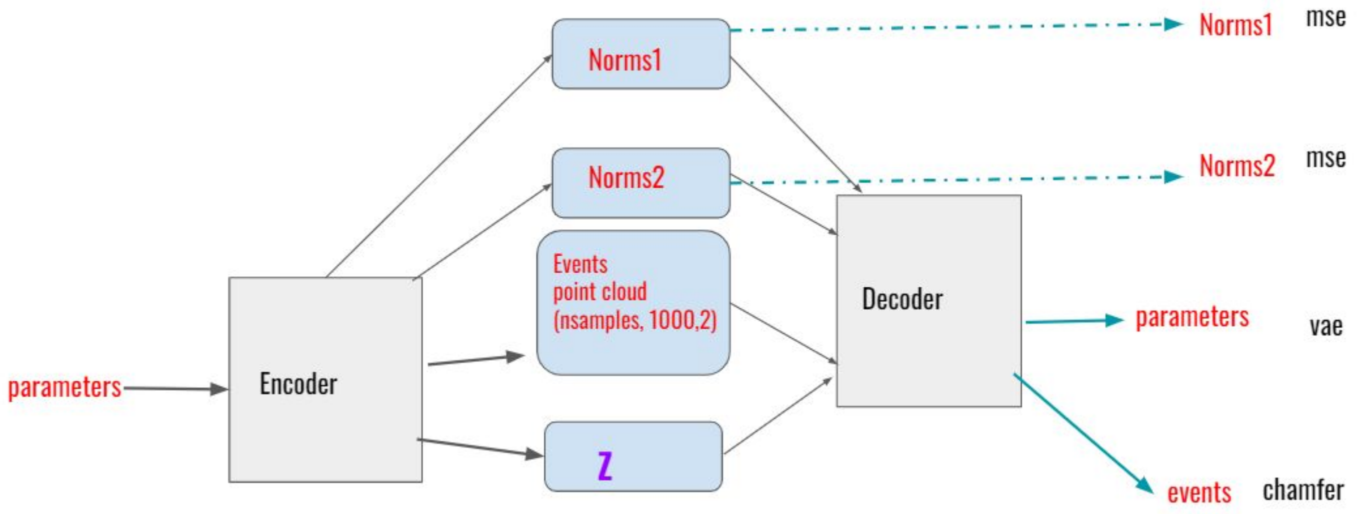




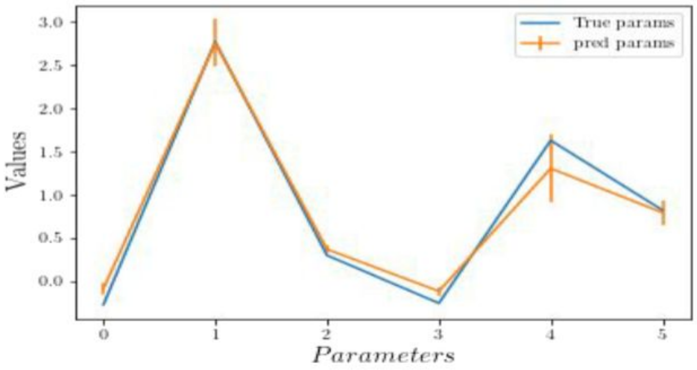


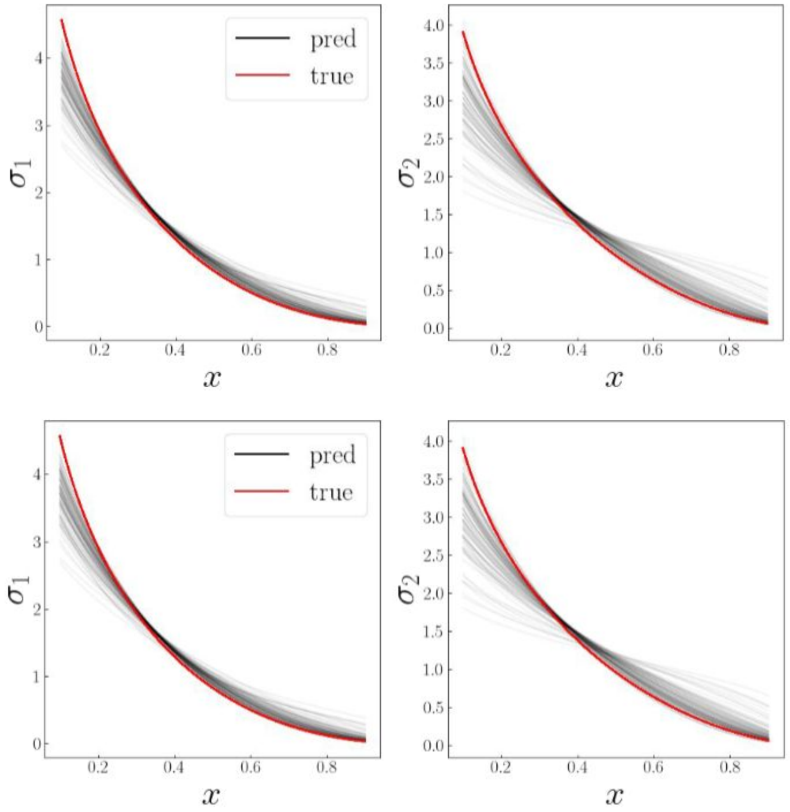


Variational AutoEncoder (VAE)

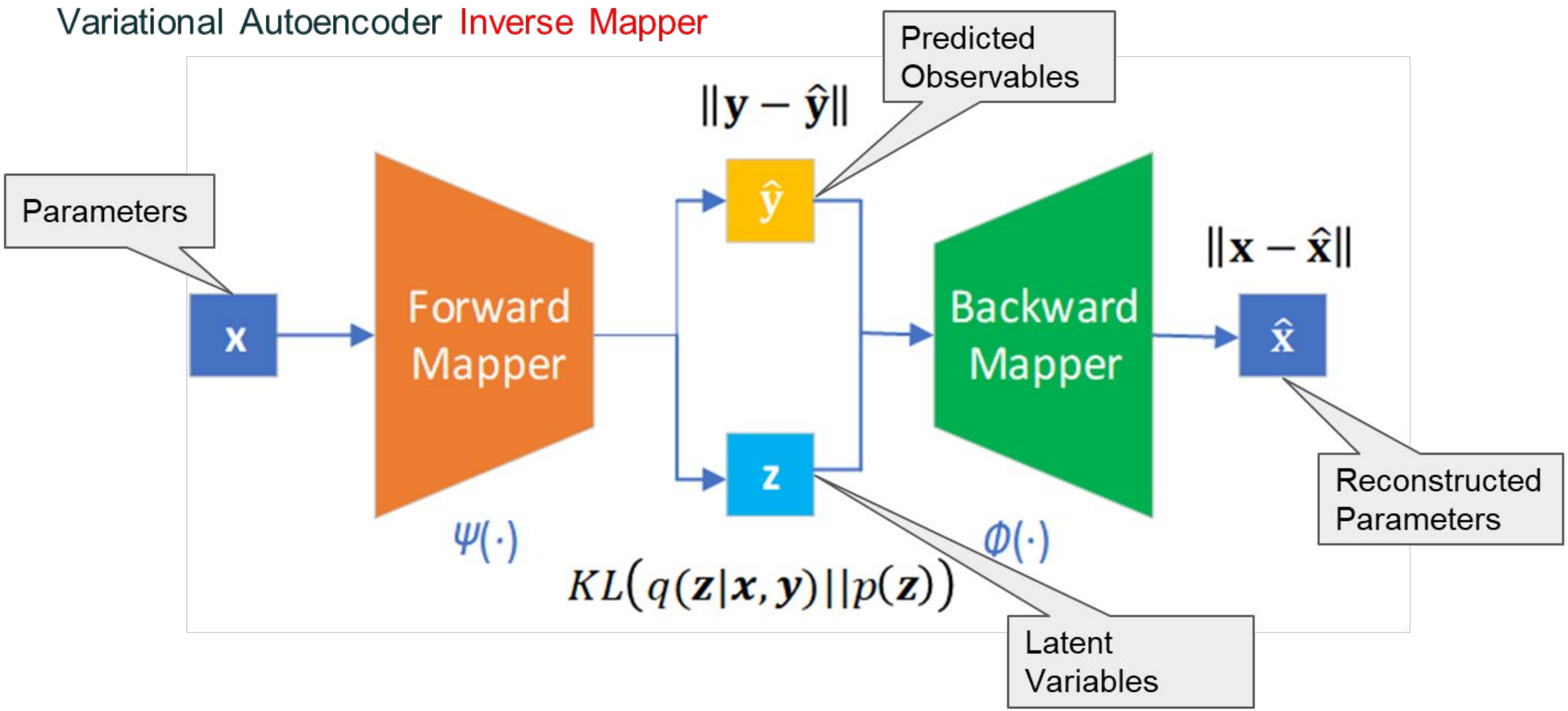


Variational AutoEncoder (VAE) Preliminary Results



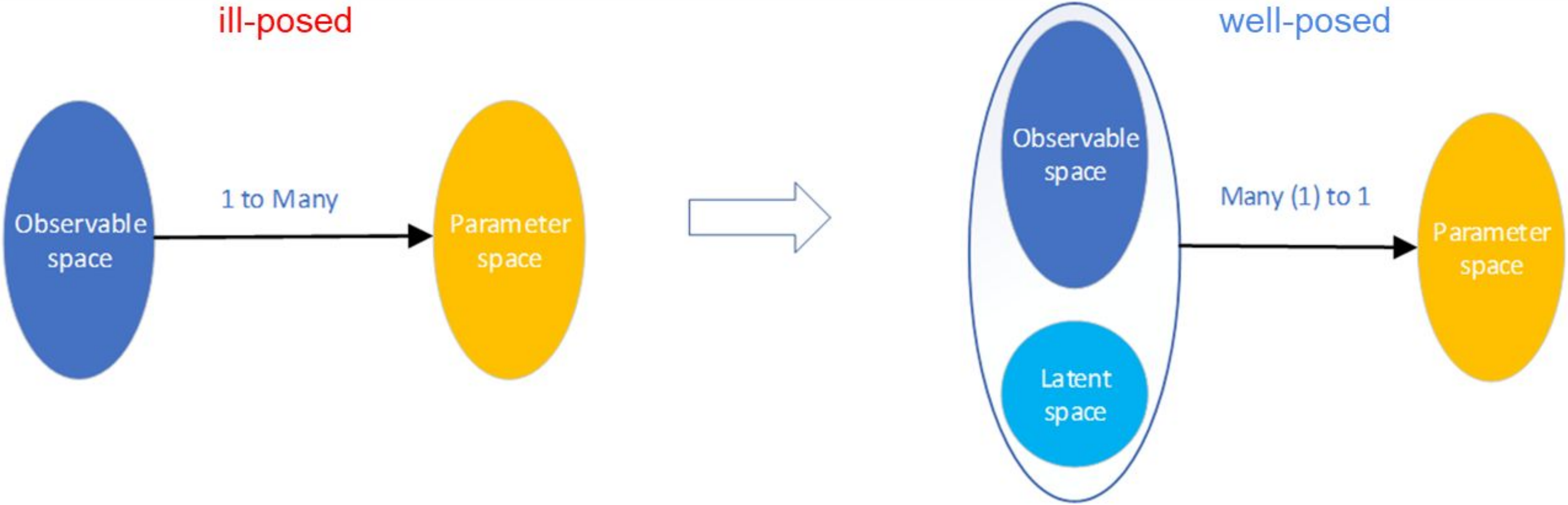


Multiple Solutions in Ill-posed Inverse Problems



VAIM: Fundamental Idea

Inverse Problem	VAIM



Automated Machine Learning: Necessity

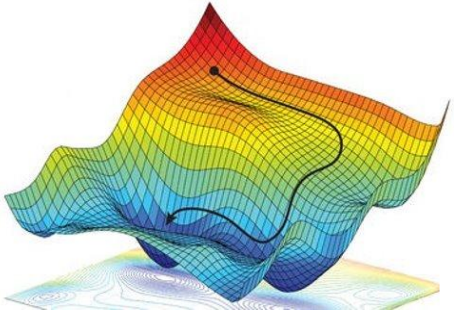
Requirements for efficient AI utilization:

- The neural architecture search and hyperparameter search algorithms needed to circumvent the exhaustive manual tuning of the modules.
- Single best-prediction (GAN) model may not be enough. Ensemble of well-performing ML-models can provide uncertainty quantification (both epistemic and aleatoric)
- Crucial to effectively utilize the super-computing resources.

Optimization space

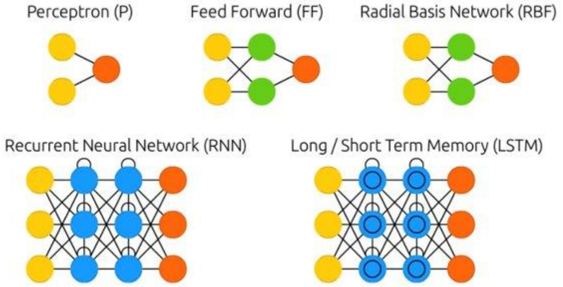
1) Algorithm hyper-parameter space

- Optimizer: SGD, RMSprop, Adam…
- Learning rate
- Minibatch size
- Learning rate scheduler



2) Architecture variable space

- Number of layers
- Layers: Fully Connected, Convolution …
- Activation function

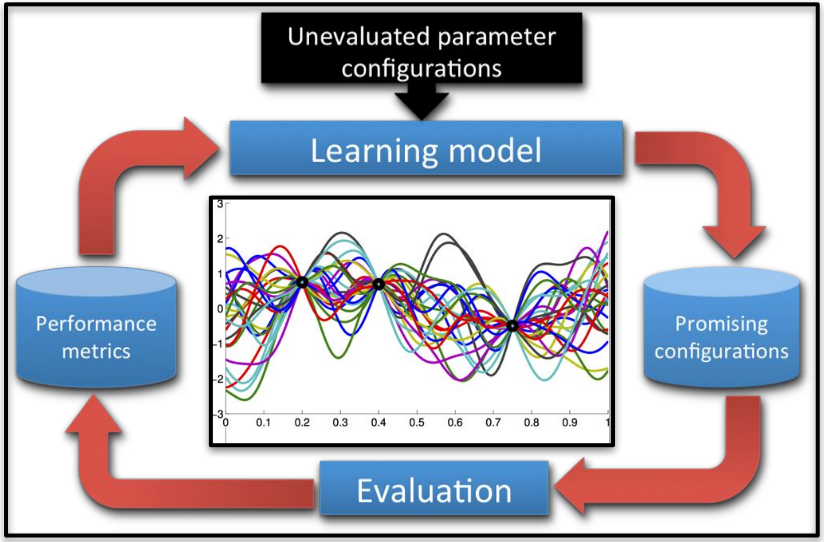


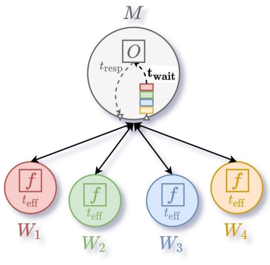
$$h_a^*, h_m^* = \underset{(h_a, h_m) \in H_a \times H_m}{\arg\max} \; \mathcal{M}_{w^*}^{val}(h_a, h_m)$$

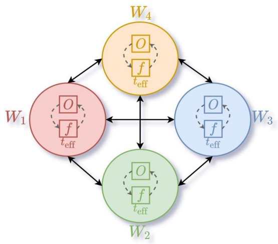
$$\text{s. t. } w^* = \underset{w}{\arg\min} \, \mathcal{L}_{h_a, h_m}^{train}(w),$$

DeepHyper: An AutoML package

- ML model/pipeline included as a black-box function.
- Dynamically updated surrogate model
  - (Hyperparameter/input space) validation inaccuracy/output space)
- Asynchronous model evaluation
- Search strategies:
  - Hyperparameter Search: Random, Bayesian optimization
  - Neural Architecture Search: Random, Bayesian, Genetic algorithms
  - Joint Neural and Hyperparameter Search: Genetic+Bayesian optimization
- Parallelization:
  - HPC interfacing: Ray, MPI
  - Schemes: Centralized, Decentralized

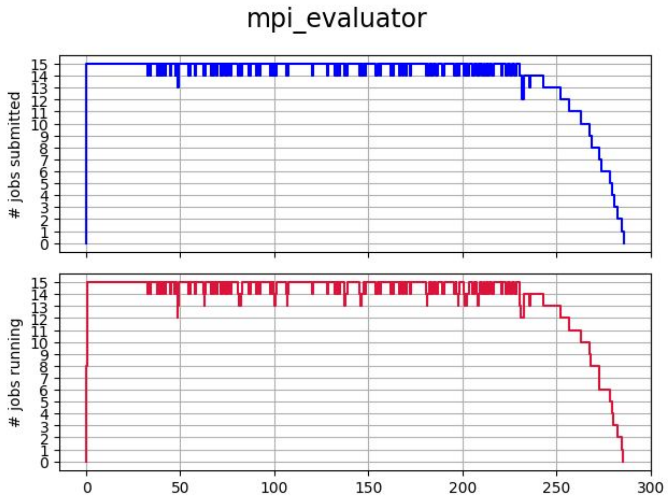


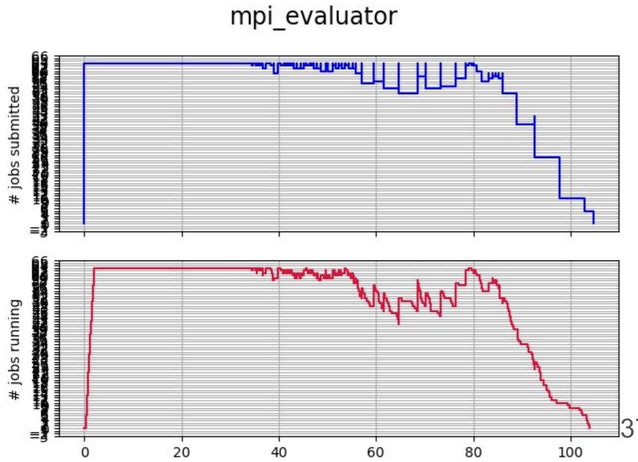




DeepHyper tests

- Tested on Perlmutter Supercomputer at the NERSC, LCRC-Swing, Leadership machines
- Centralized Bayesian Optimization (CBO) scheme (single manager monitors multiple workers)
- HPO on upto O(1000) nodes, where each GPU node has 4x NVIDIA A100 GPUs.
- Fig shows the performance of MPI evaluator (on a smaller number of nodes) for hyperparameter optimization of our neural network.
    - The top panel: number of submitted jobs represents the evaluator's performance in managing the workers.
    - The bottom panel: number of simultaneous jobs shows the true usage of the Perlmutter resources.





Path Forward

- Replace inverse CDF with MCMC
- Converge multiple discriminator GAN
- Complete the workflow by incorporating experimental module
- HPO pipeline
- Develop a surrogate model for Module-2 (No need to make theory code differentiable!)
- Closure test on inclusive DIS

- What are the expectations from different WG?
- List of standard libraries to use?
  - TF/**Torch**, which version?
  - Version numbers for other python libraries