Generative Models

Sascha Diefenbacher,

ML4FP School 2024

Generative Models in Media

- Learn underlying distribution of data
- Produce realistic new samples
- Recently gained attention for AI art generation (Dall-E 2, Imagen, Midjourney)

GOOGLE
IMAGEN AI
DALL·E 2

particle shower in a calorimeter in the style of egon schiele, Dall-E 2

Classification

High Dim. Data

Classification

High Dim. Data

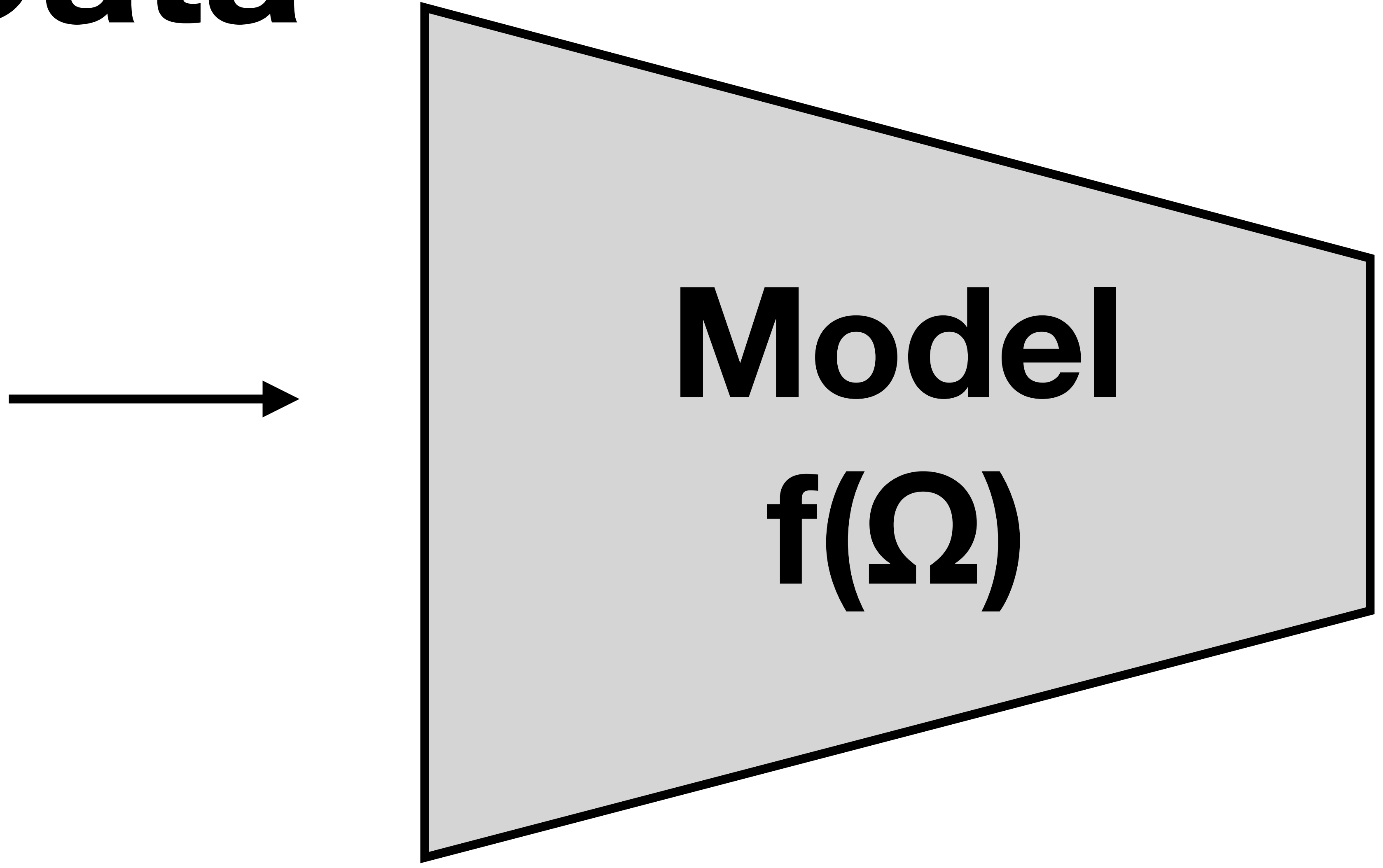
Low Dim. Label

Dog

Cat

Classification

High Dim. Data



Classification

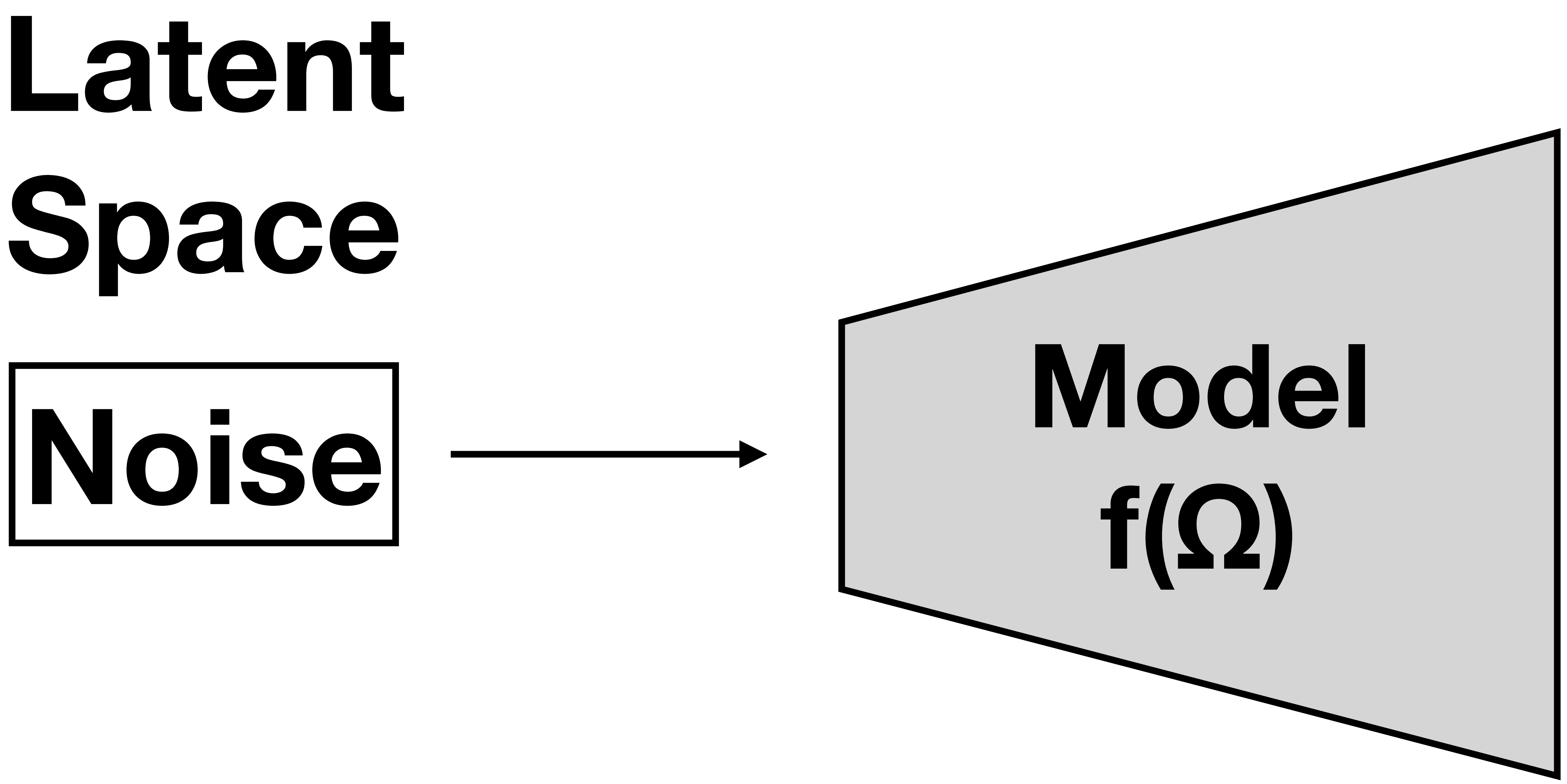
High Dim. Data

Model
f(Ω)

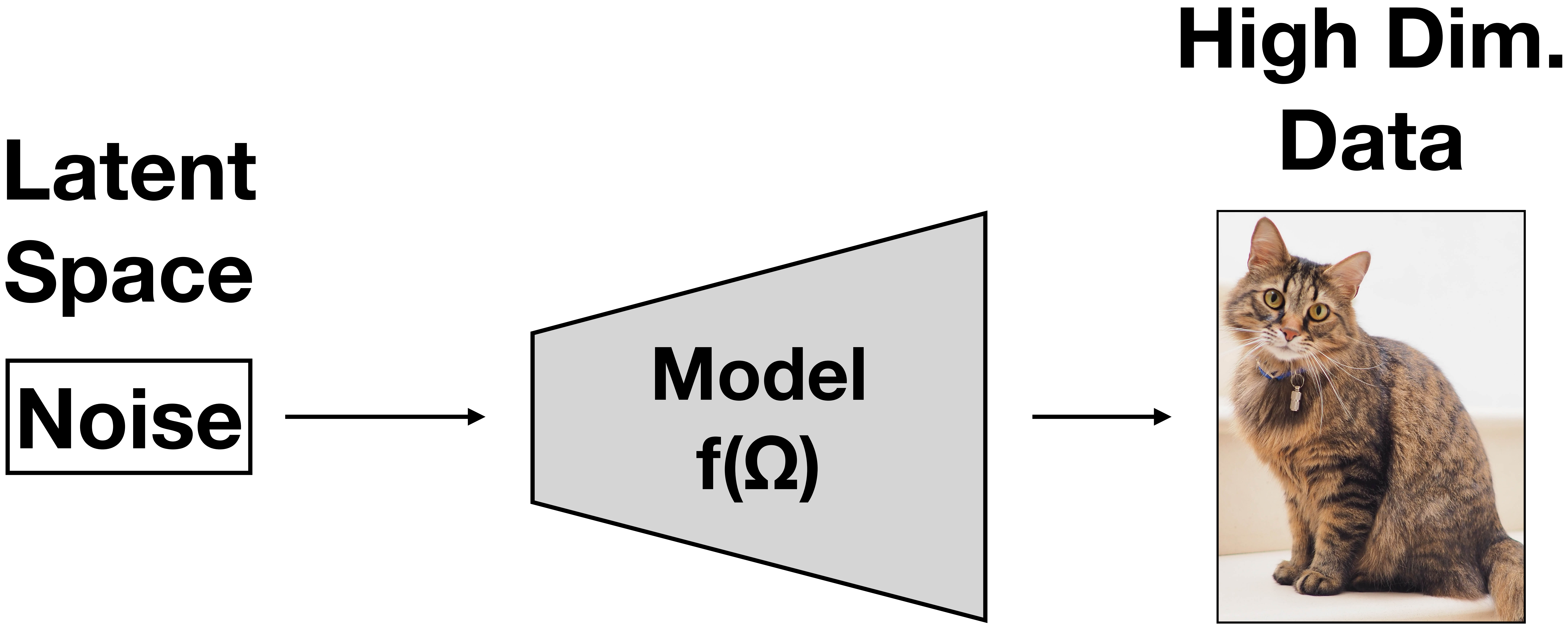
Prediction

Cat/Dog?

Generation



Generation



Generation

Training a generative model

1. Start from random noise
2. Use network to transform noise to data
3. Use loss function that gauges quality of generated data

Generation

Training a generative model

1. Start from random noise ✓
2. Use network to transform noise to data ✓
3. Use loss function that gauges quality of generated data

This is where it gets tricky

Classification Loss Functions

- Easy to understand model outputs
  - Predictions for a given data sample
  - Accuracy, Area Under Curve, directly measure network performance

Classification Loss Functions

- Easy to understand model outputs
  - Predictions for a given data sample
  - Accuracy, Area Under Curve, directly measure network performance
  - Directly compare network output to true prediction
  - Wide range of available loss functions
    - Mean Squared Error
    - Cross-Entropy

Generative Loss Functions

- How do you measure the model performance?
- How is this expressed mathematically (and differentiable)

Generative Loss Functions

- How do you measure the model performance?
- How is this expressed mathematically (and differentiable)

http://thesecatsdonotexist.com

Generative Difficulties

- Image Set:
  - Does the new set have the same properties as the data?

Training Data:

Generative Difficulties

- Image Set:
  - Does the new set have the same properties as the data?

Training Data:

Generated Data:

Generative Difficulties

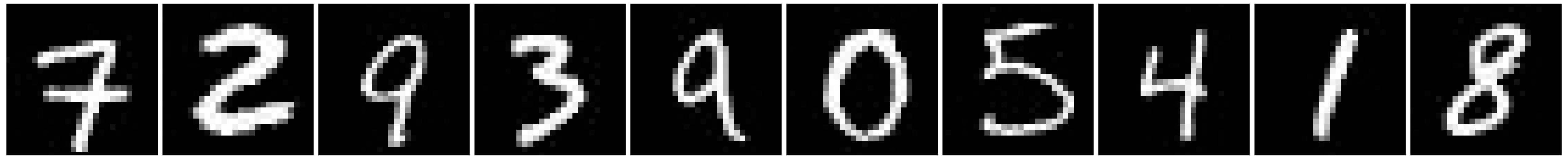
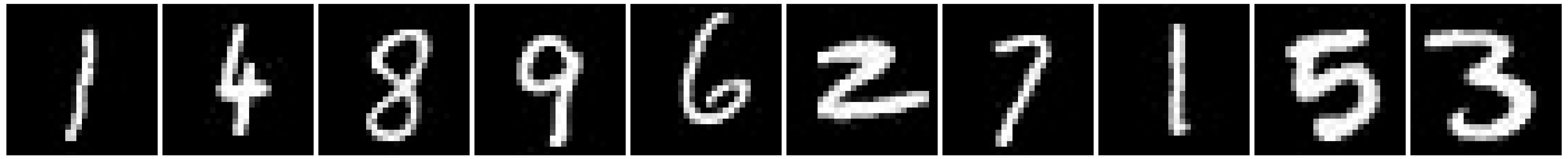


Image Set



Ideal generative outcome

Generative Difficulties

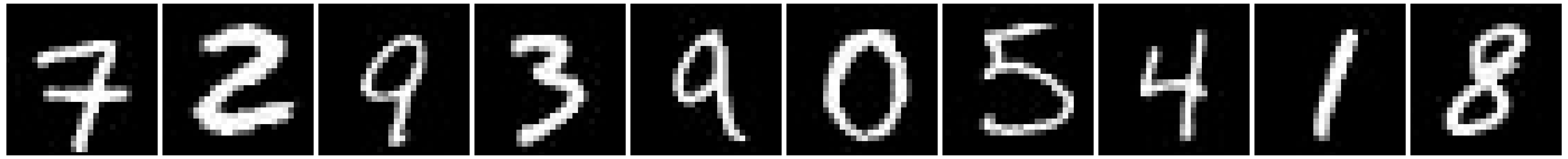
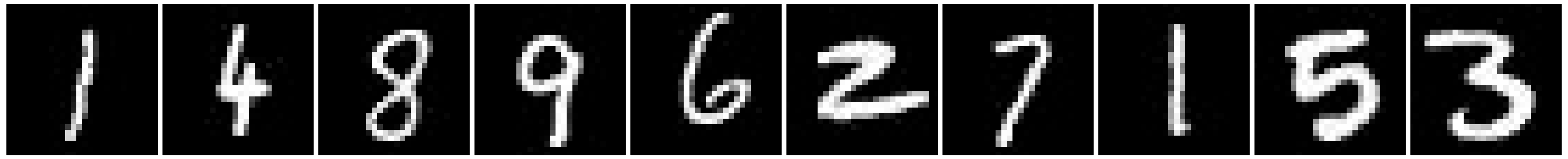
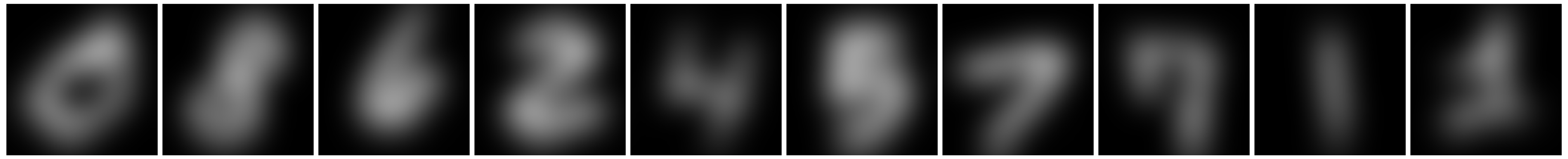


Image Set



Ideal generative outcome



Low sample quality

Generative Difficulties

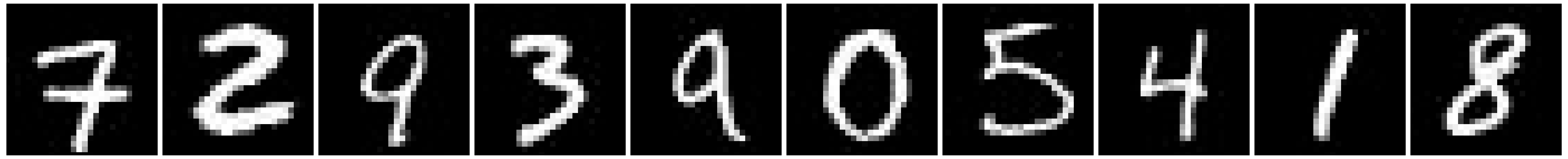
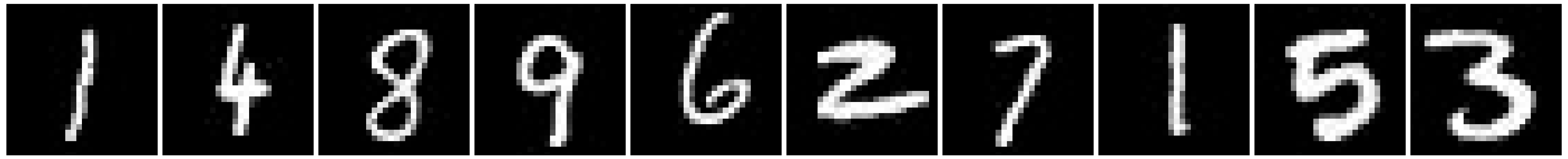
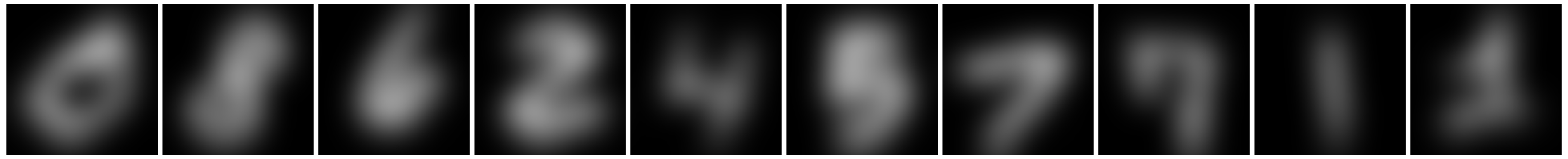


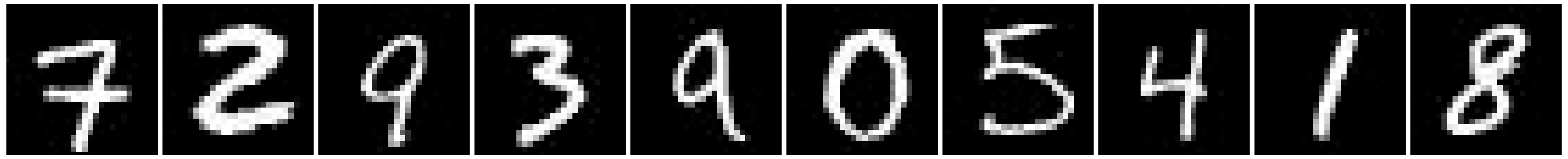
Image Set



Ideal generative outcome



Low sample quality



Overfitting

Generative Difficulties

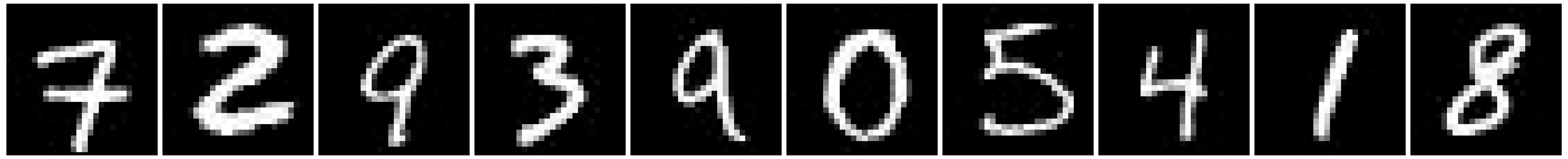
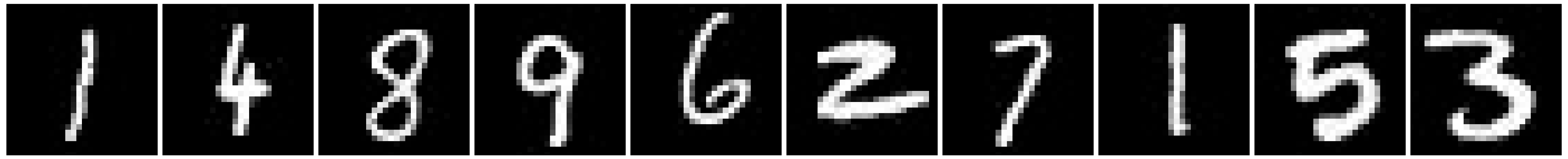
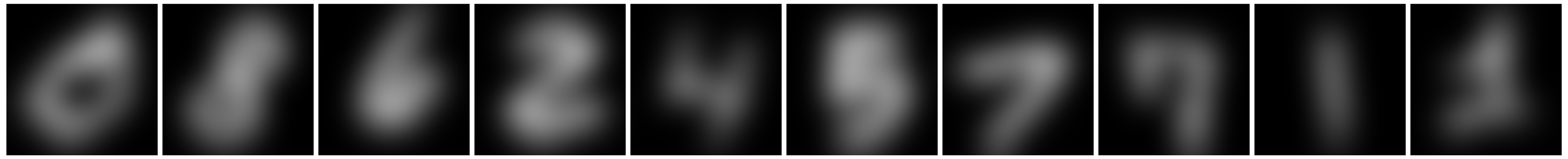


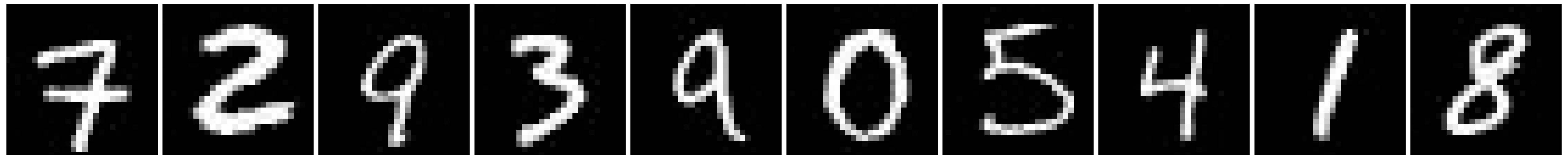
Image Set



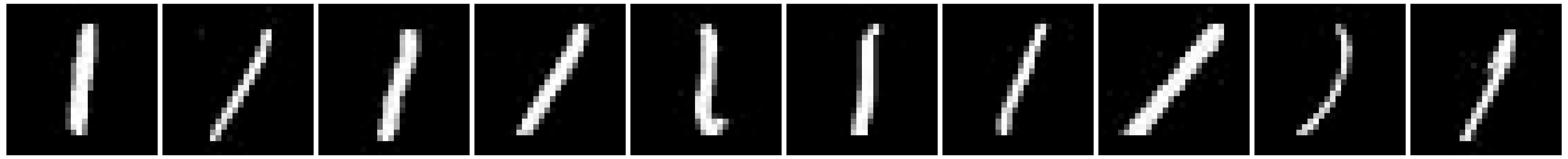
Ideal generative outcome



Low sample quality



Overfitting



Mode collapse

Generative Difficulties

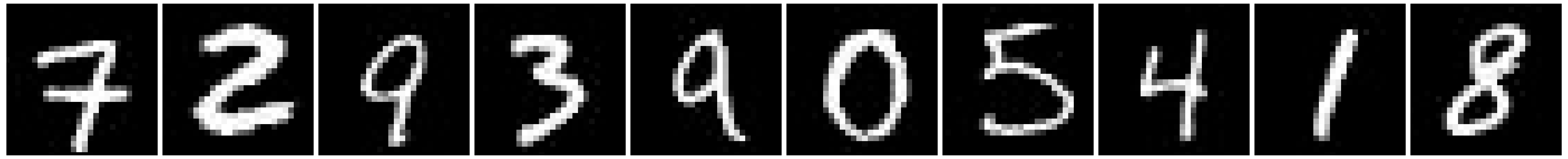
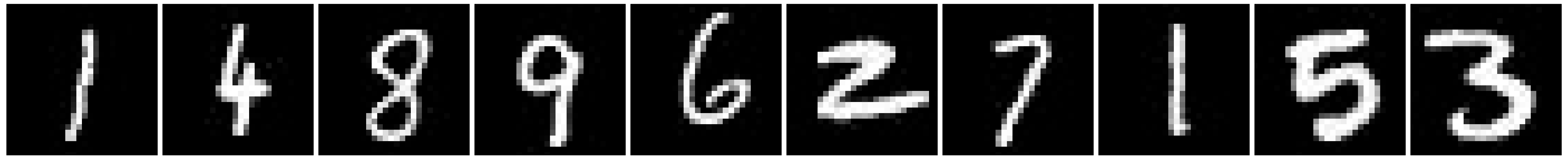
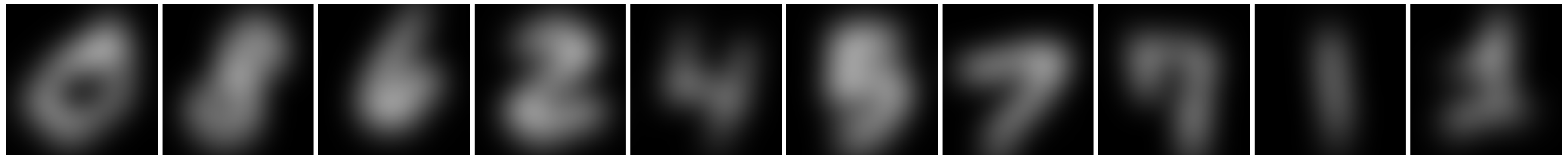


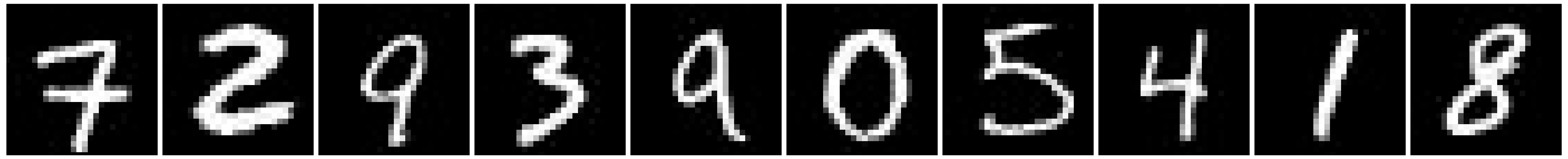
Image Set



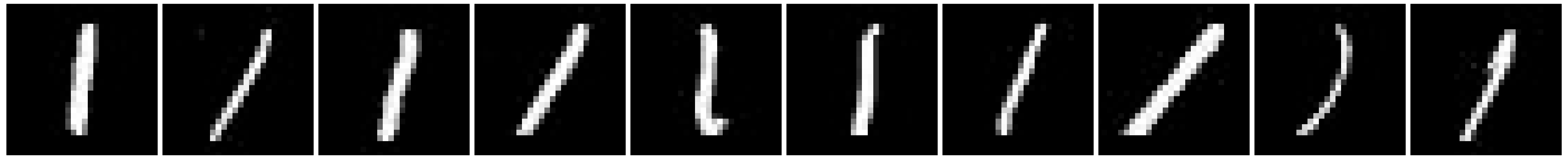
Ideal generative outcome



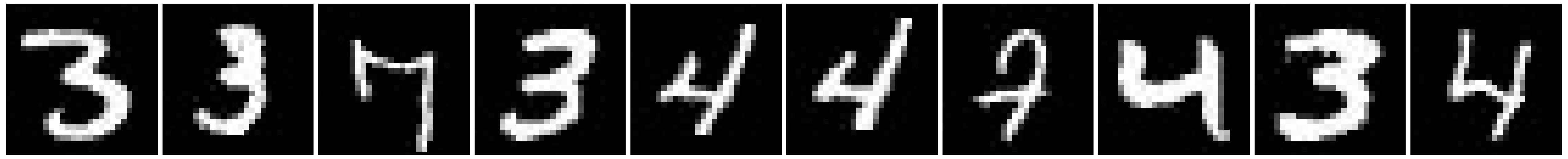
Low sample quality



Overfitting



Mode collapse



Incorrect composition

Generative Difficulties

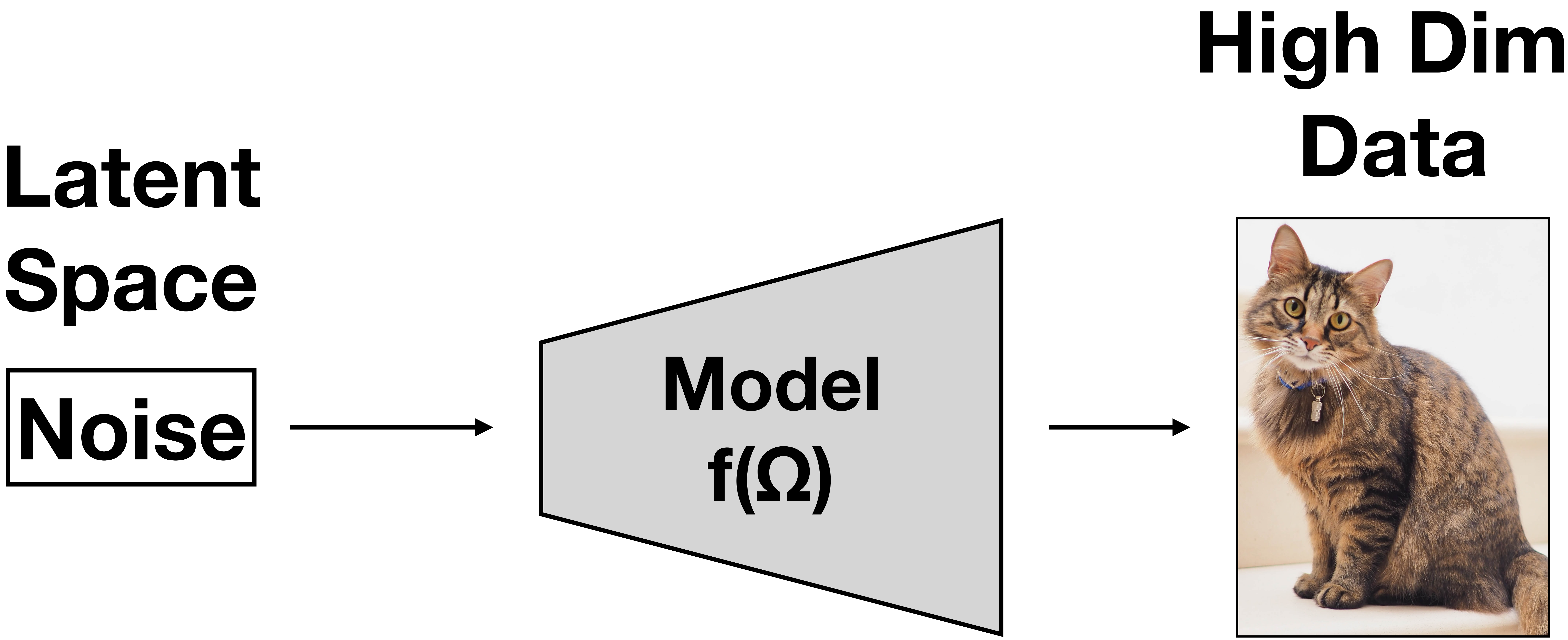
GANs
Generative Adversarial Networks

Score-based

NFs
Normalizing Flows

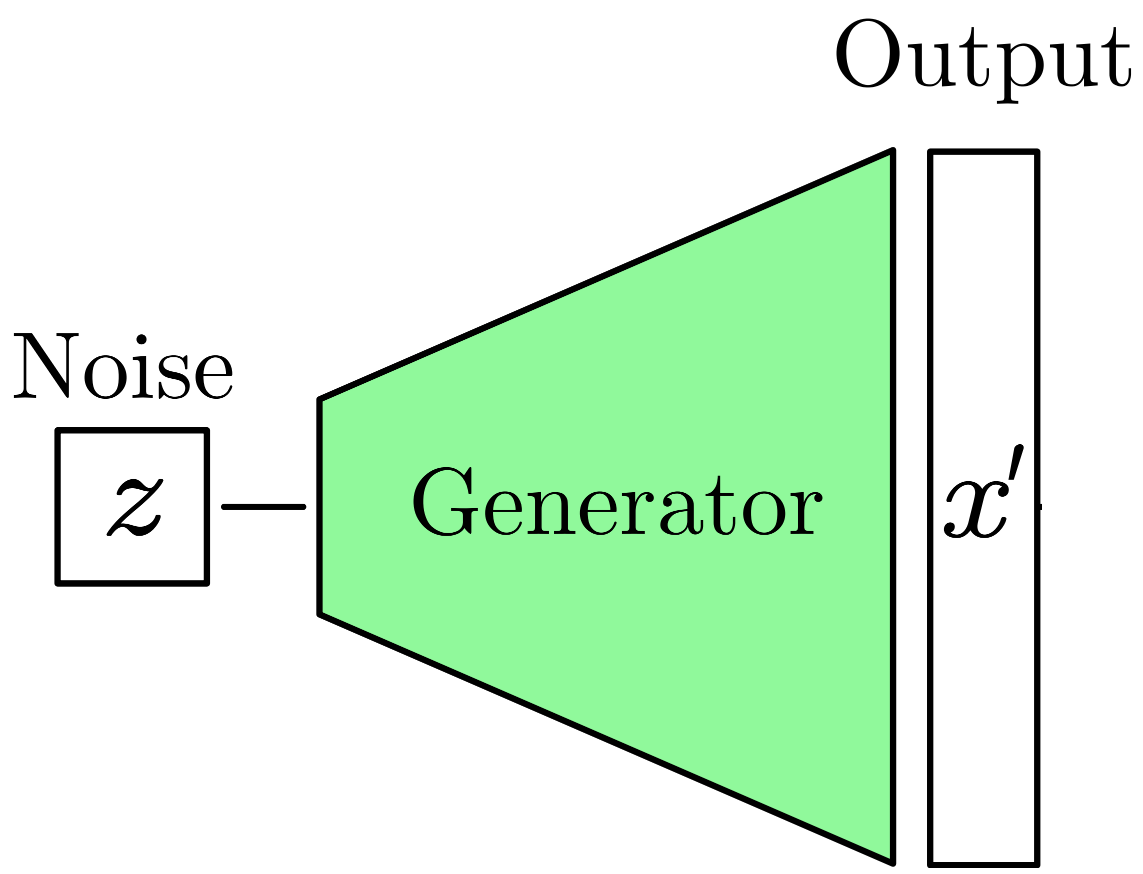
VAEs
Variational Autoencoders

Generative Adversarial Network



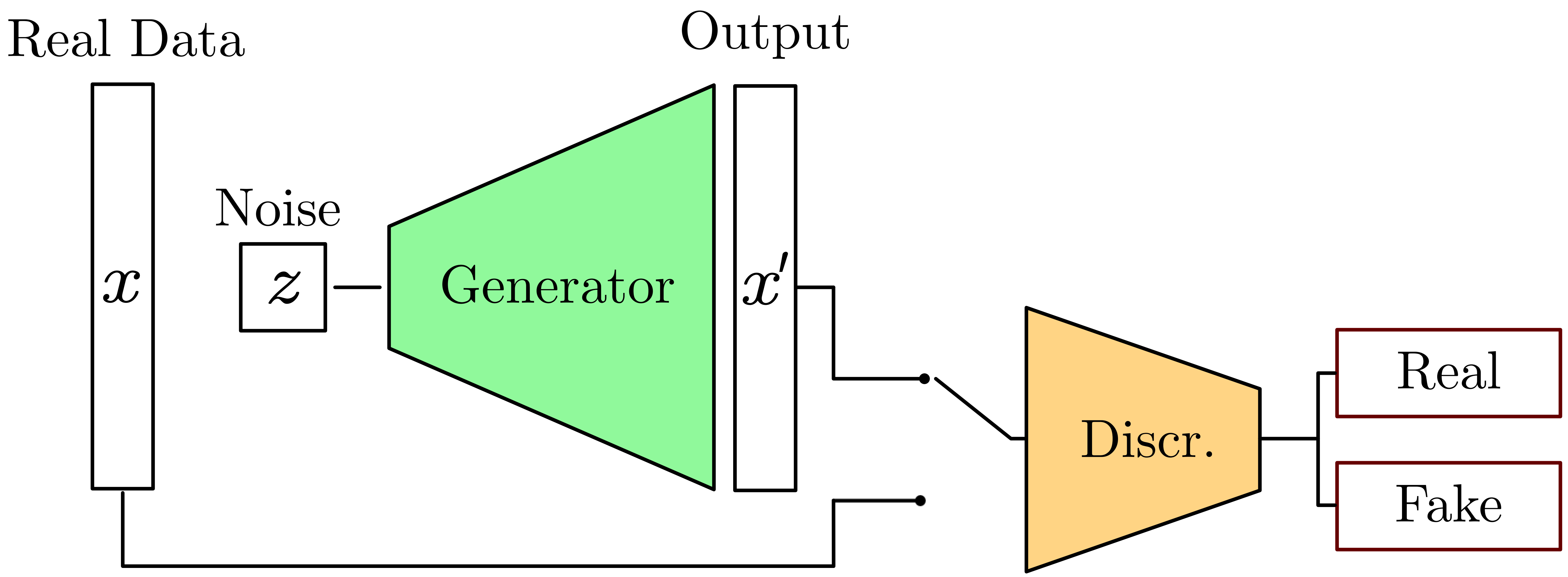
Generative Adversarial Network

- Generator Network G(z)=x
  - Maps noise Z to Data X



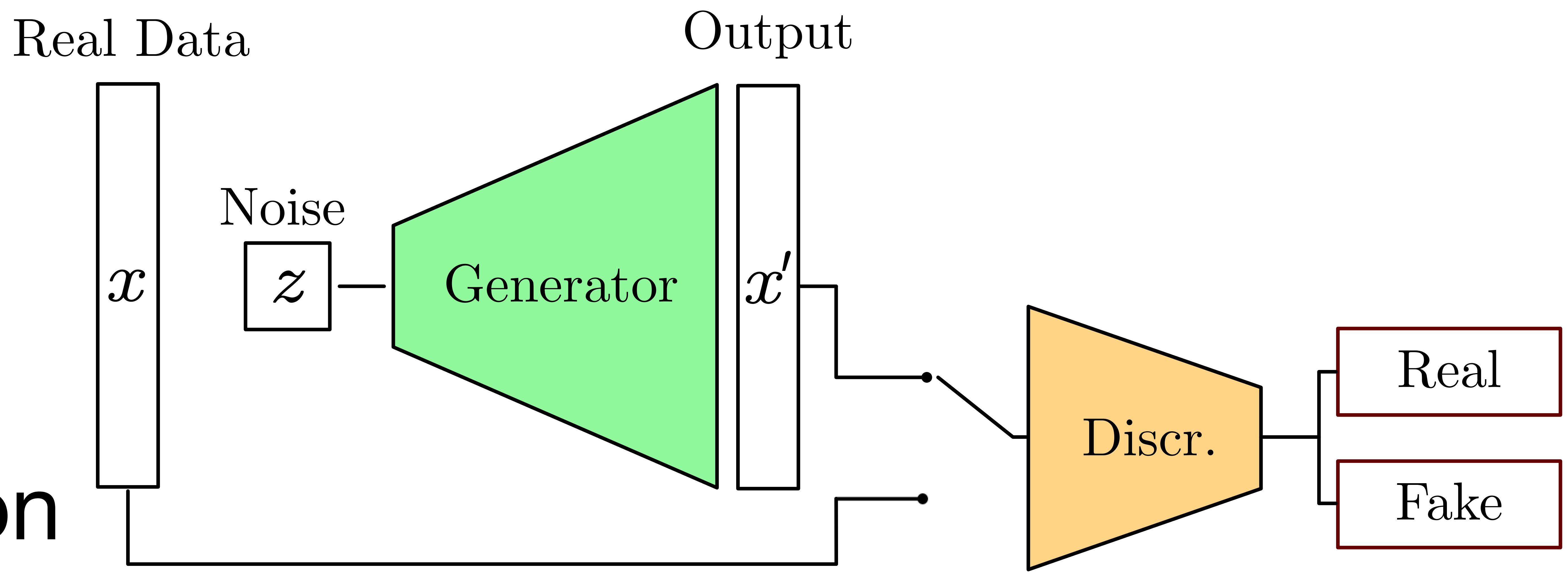
Generative Adversarial Network

- Generator Network G(z)=x
  - Maps noise Z to Data X
- Discriminator D(G(z)) and D(x)
  - Learns difference between real and fake

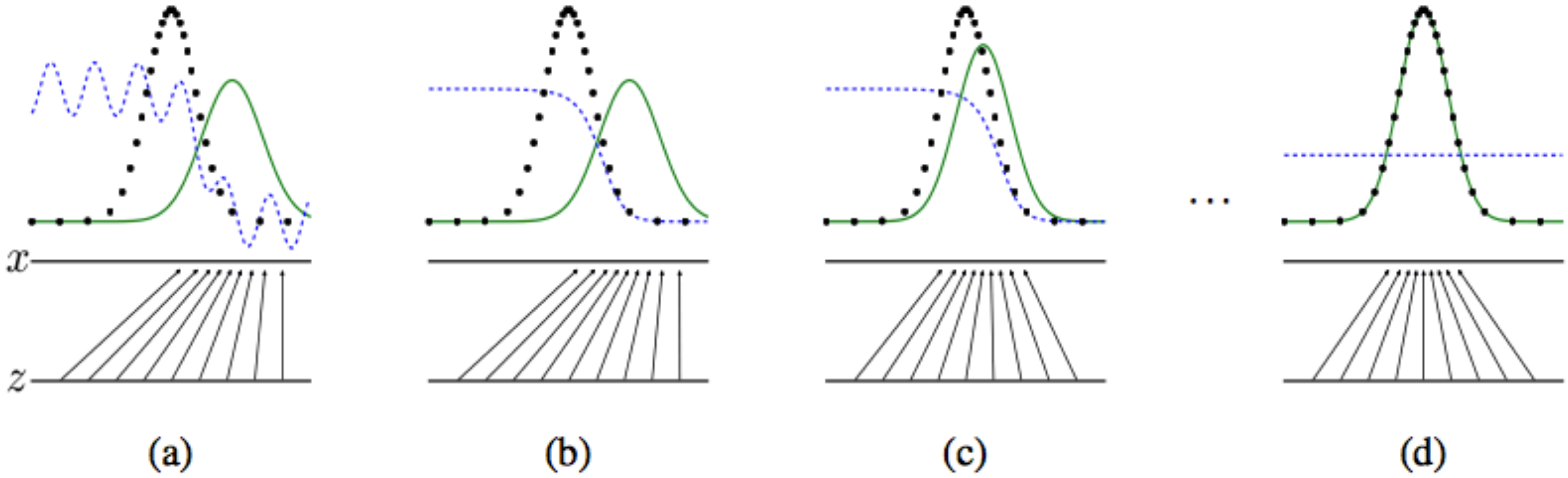


Generative Adversarial Network

- Generator Network G(z)=x
  - Maps noise Z to Data X
- Discriminator D(G(z)) and D(x)
  - Learns difference between real and fake
- D(G(z)) is differentiable function measuring performance
- Use D(G(z)) as loss to update G



Generative Adversarial Network



Goodfellow et al.- arXiv:1406.2661

- Discriminator of GAN approximates Jensen Shannon Divergence
- Guides Generated distribution to match real distribution

Generative Adversarial Network

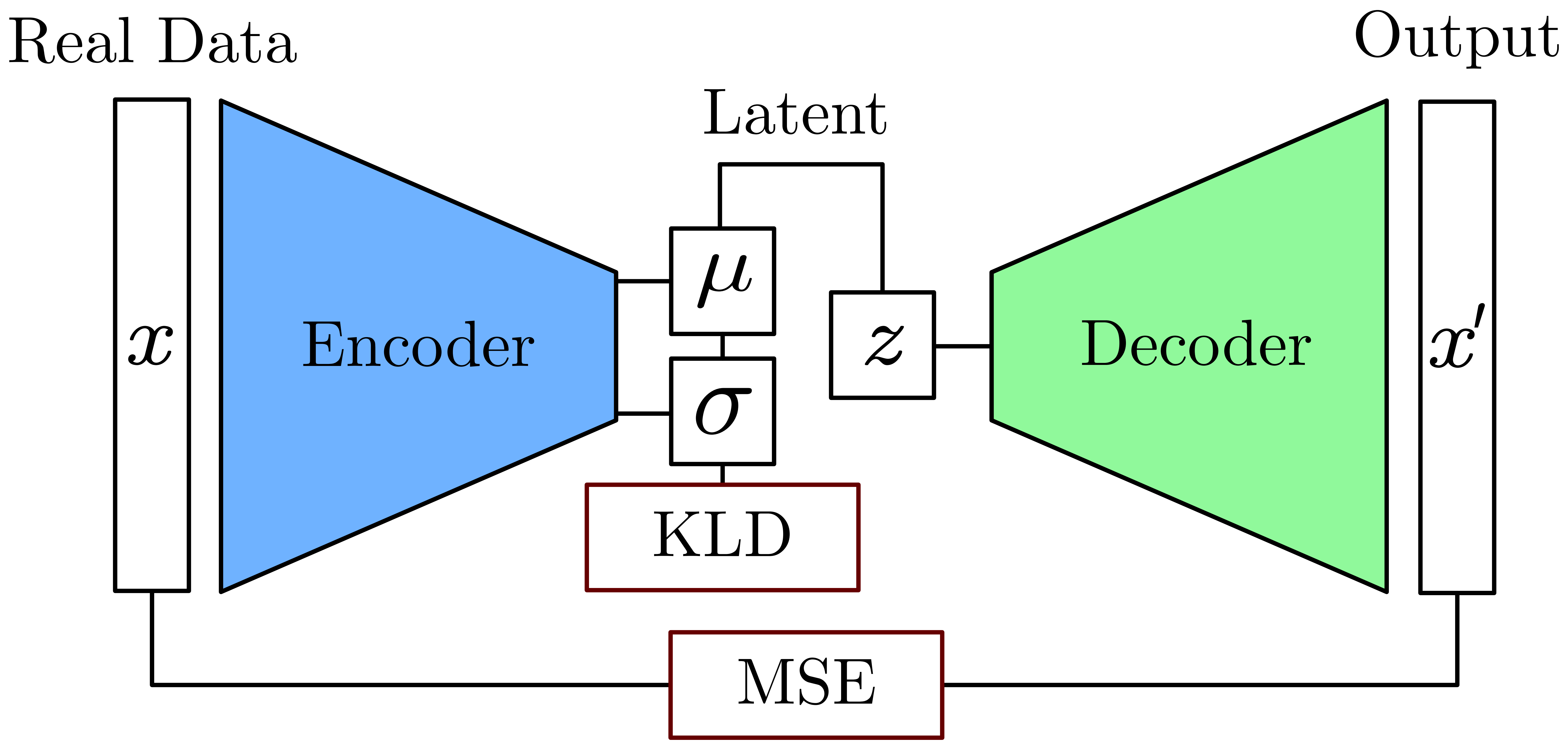
Upsides

- Intuitive approach
- Easy to introduce additional constraints
- Well explored with several improvements (WGANs, normalisations)

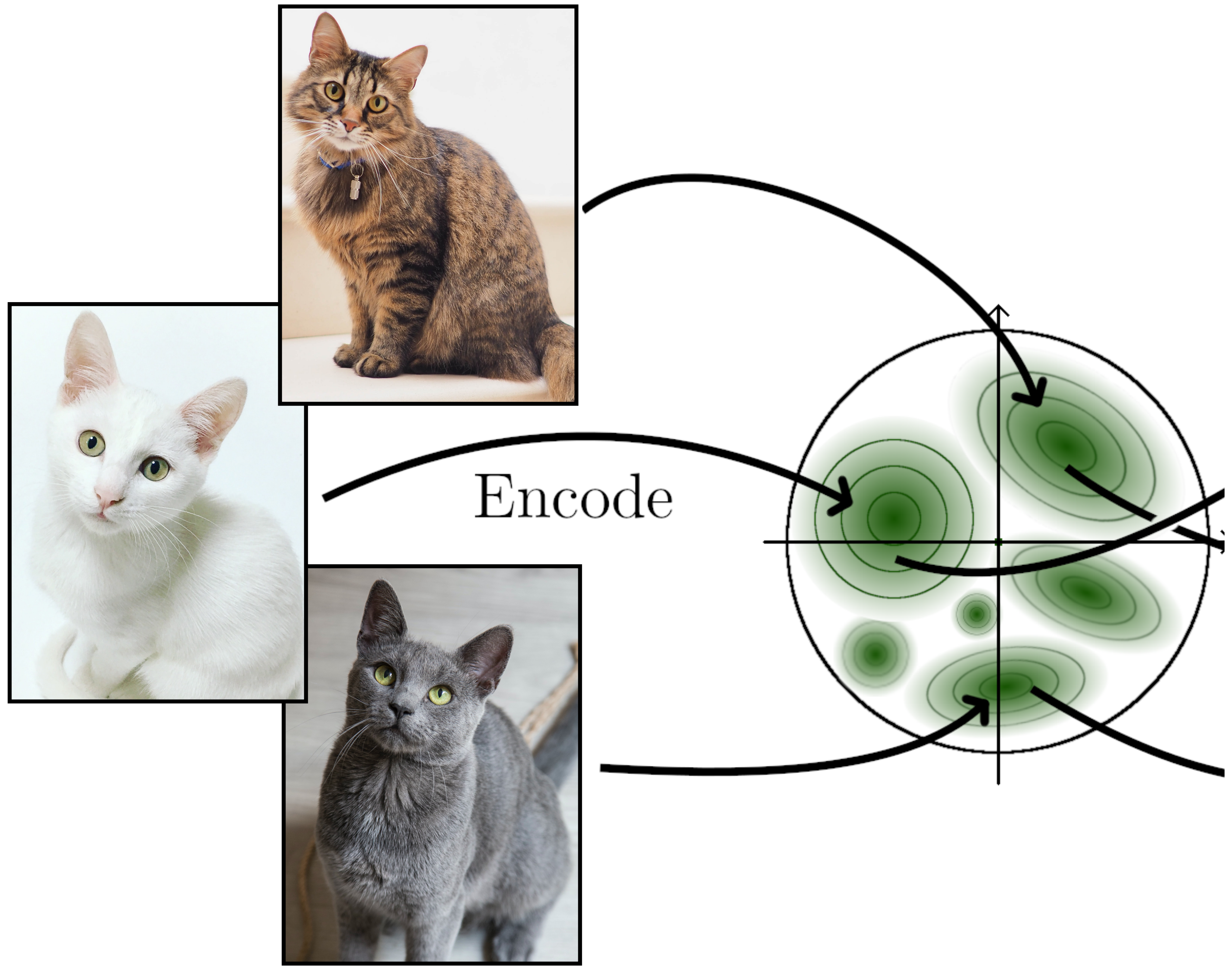
Difficulties

- Difficult to train
- Gen. and disc. needs to be balanced
- Can fail to converge
- Prone to mode collapse

Variational AutoEncoder

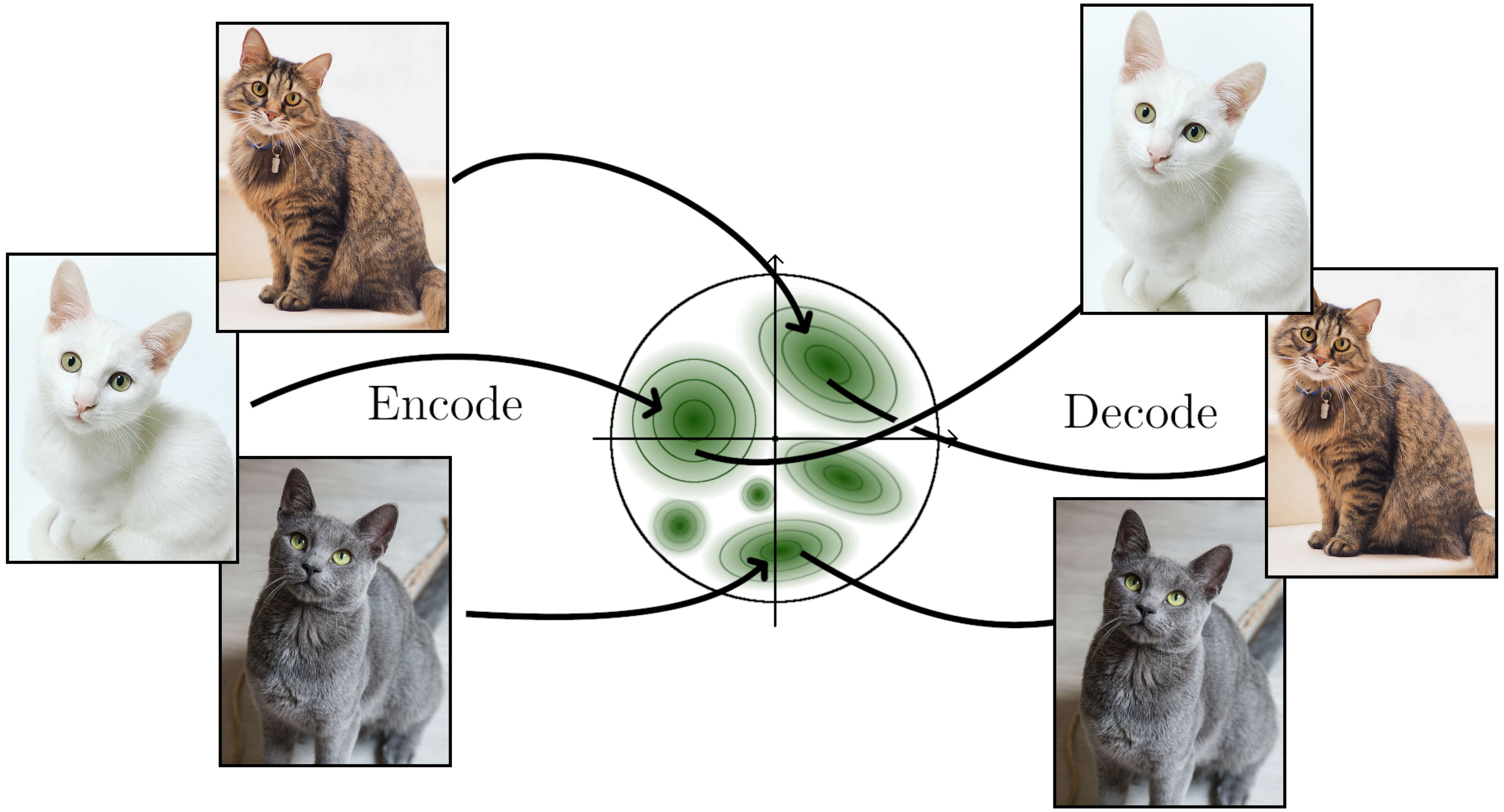


AutoEncoder



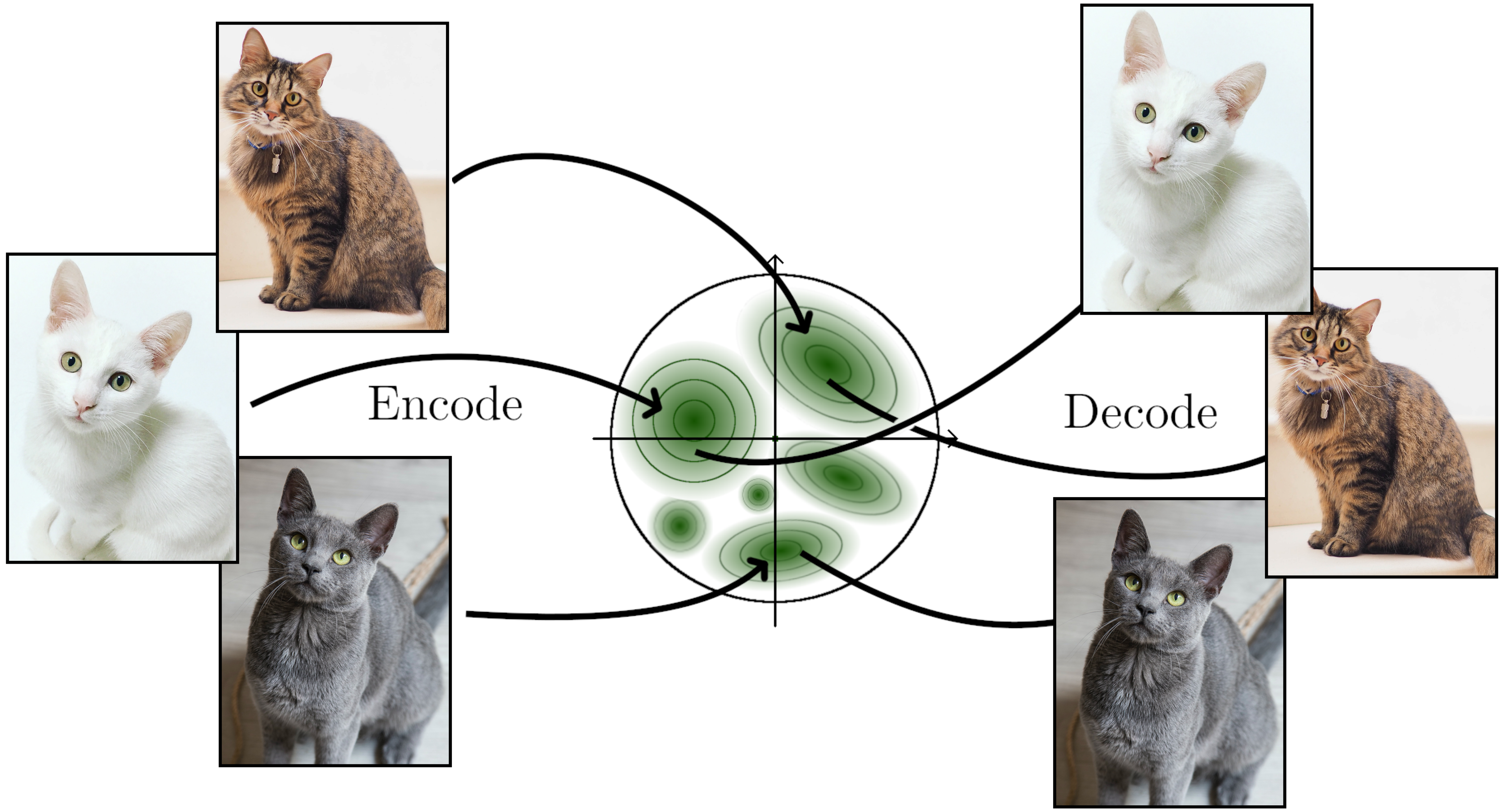
- Encoding function E(x)=z map high dimensional data X to low dimensional latent space Z

AutoEncoder



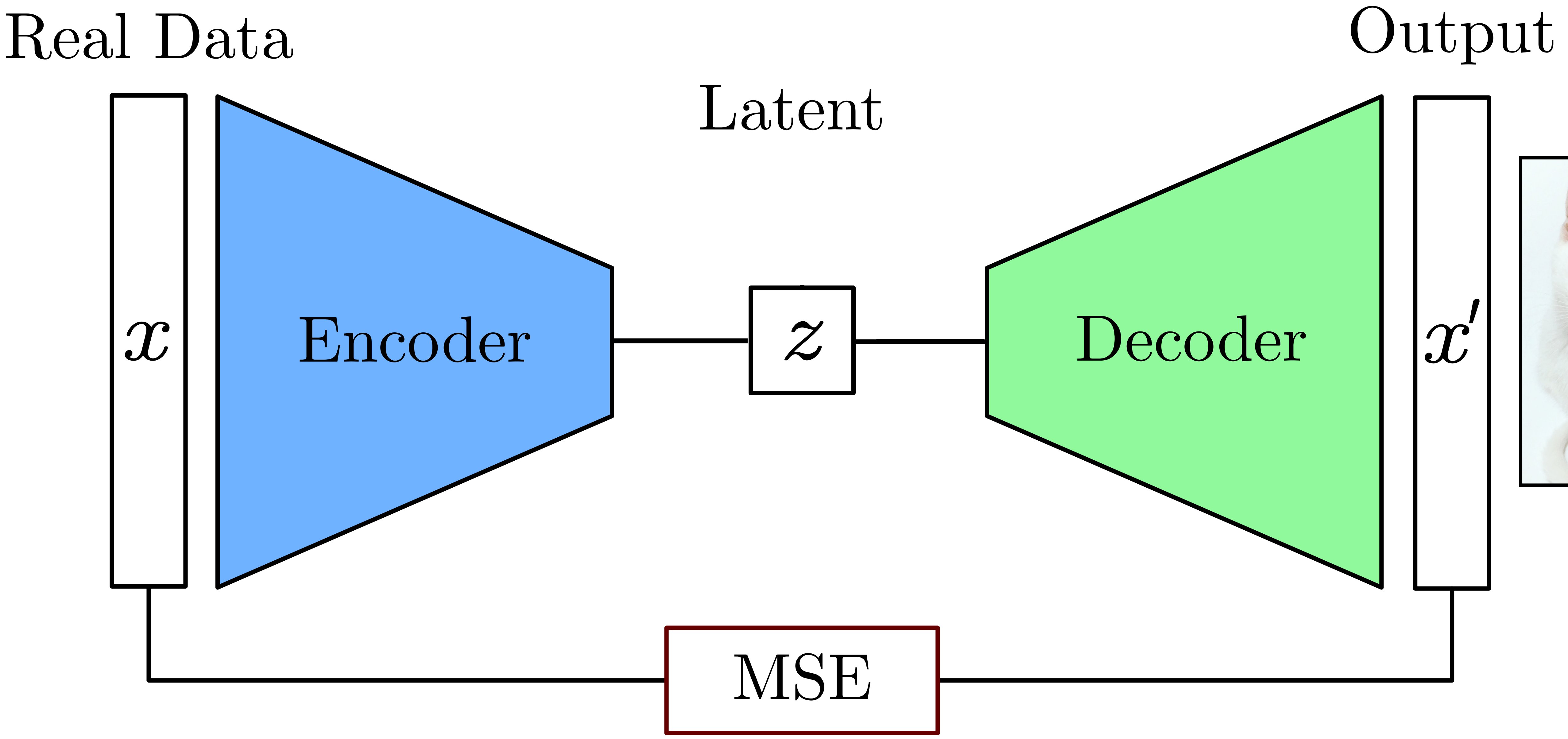
- Encoding function E(x)=z map high dimensional data X to low dimensional latent space Z
- Decoding function D(z)=x map latent space Z back to data X

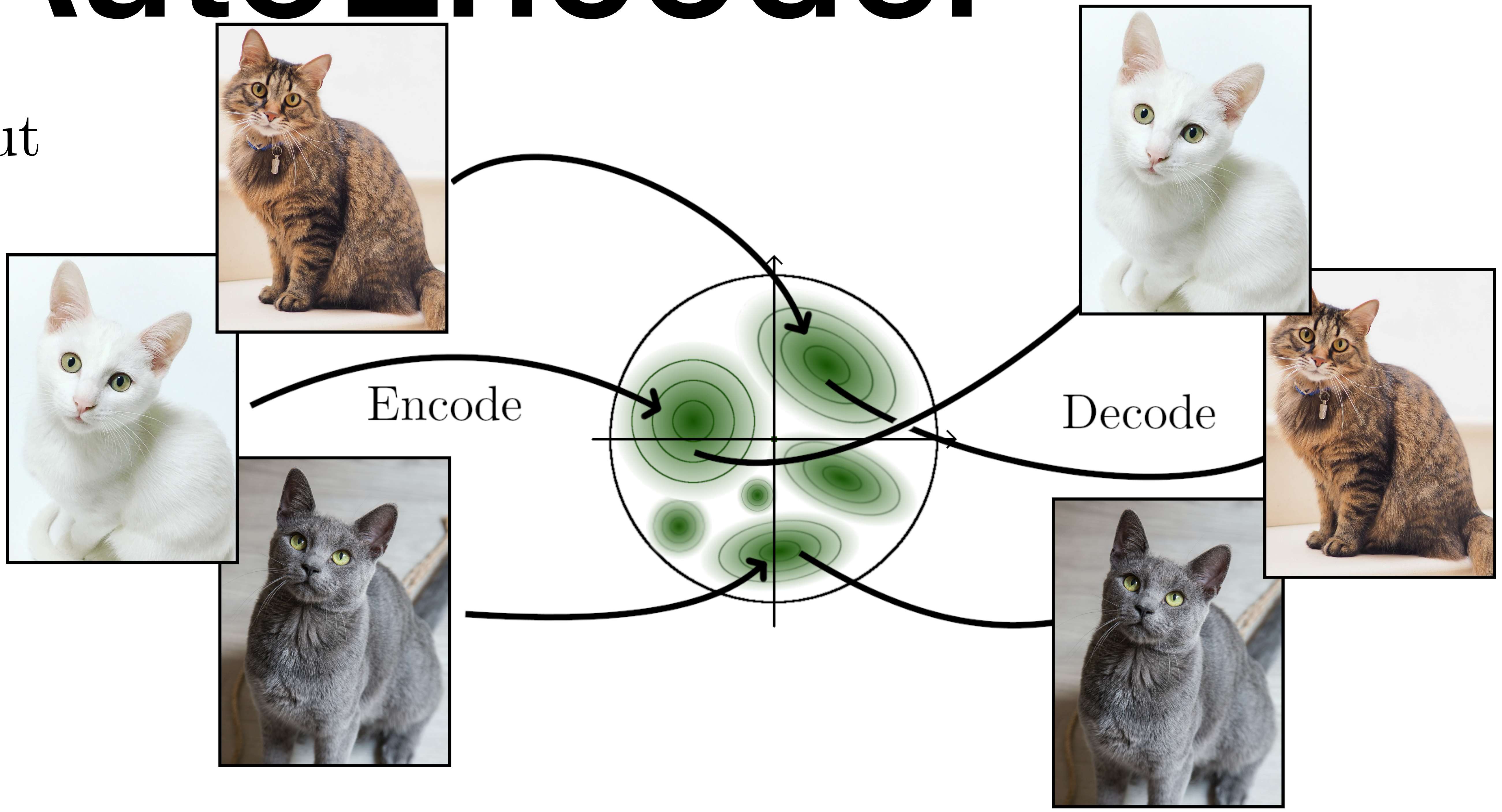
AutoEncoder



- Encoding function E(x)=z map high dimensional data X to low dimensional latent space Z
- Decoding function D(z)=x map latent space Z back to data X
- Compare Input and Output pixel by pixel with mean squared error

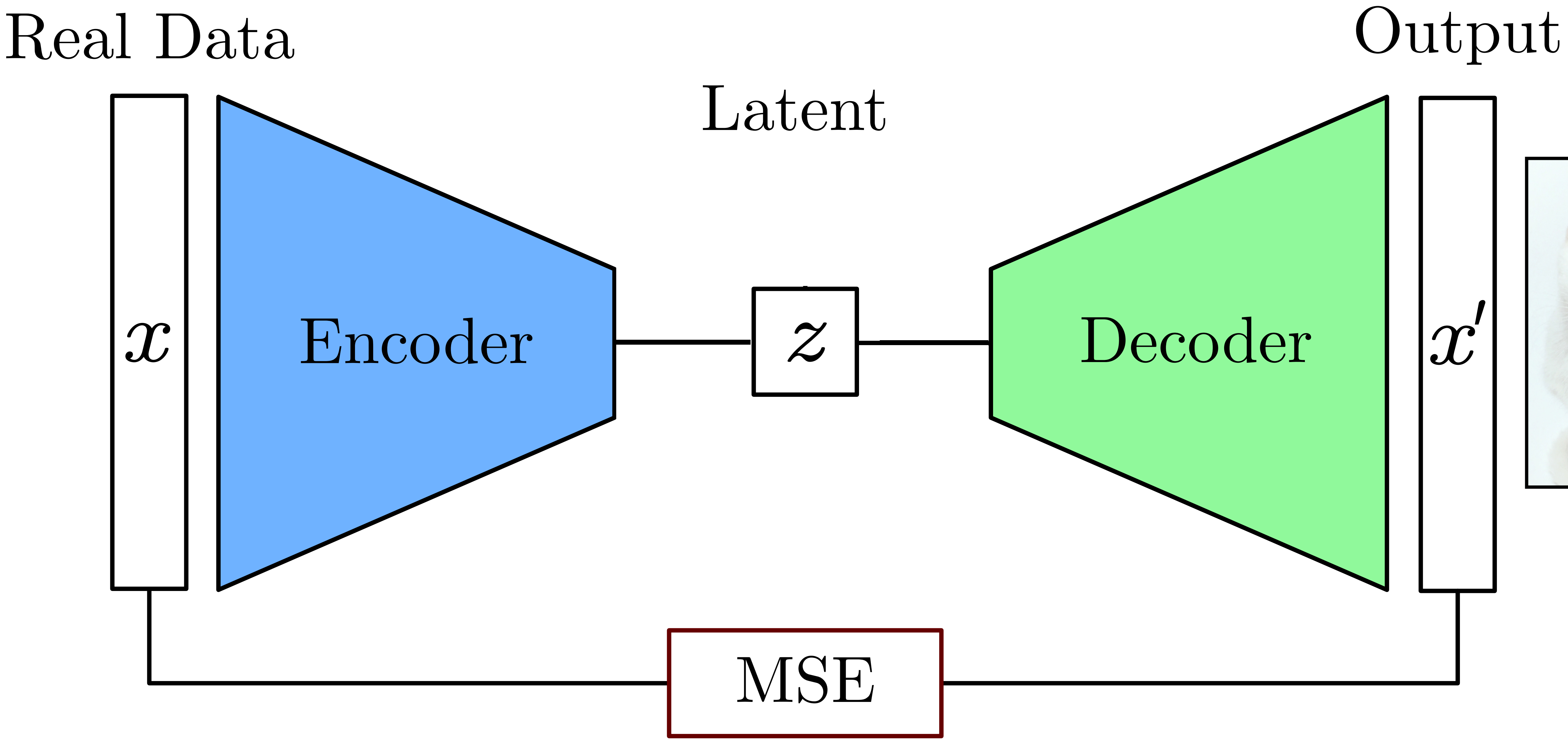
Variational AutoEncoder

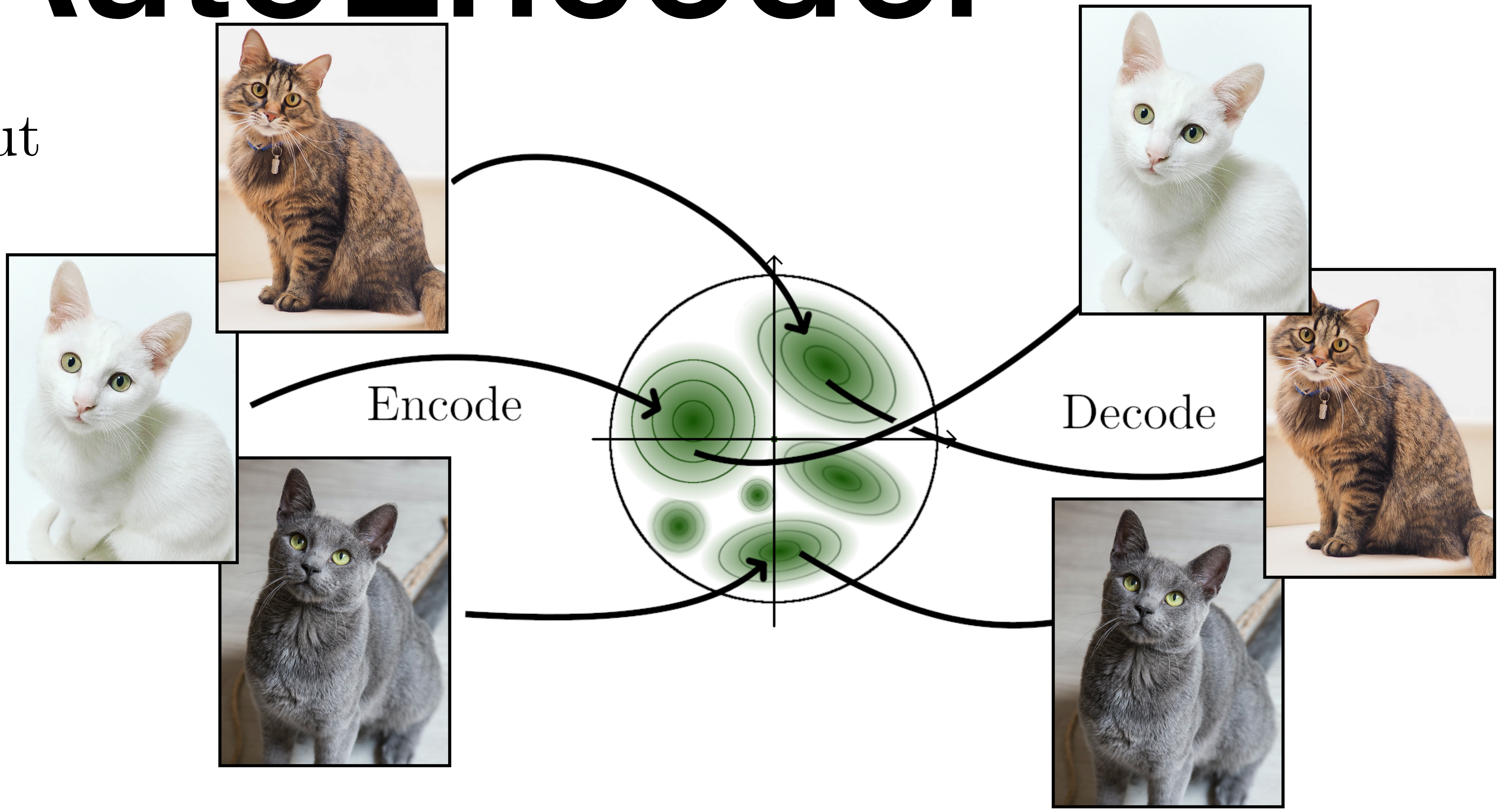




- Sample for Z and pass it to D(Z) ➡ Generate new samples

Variational AutoEncoder

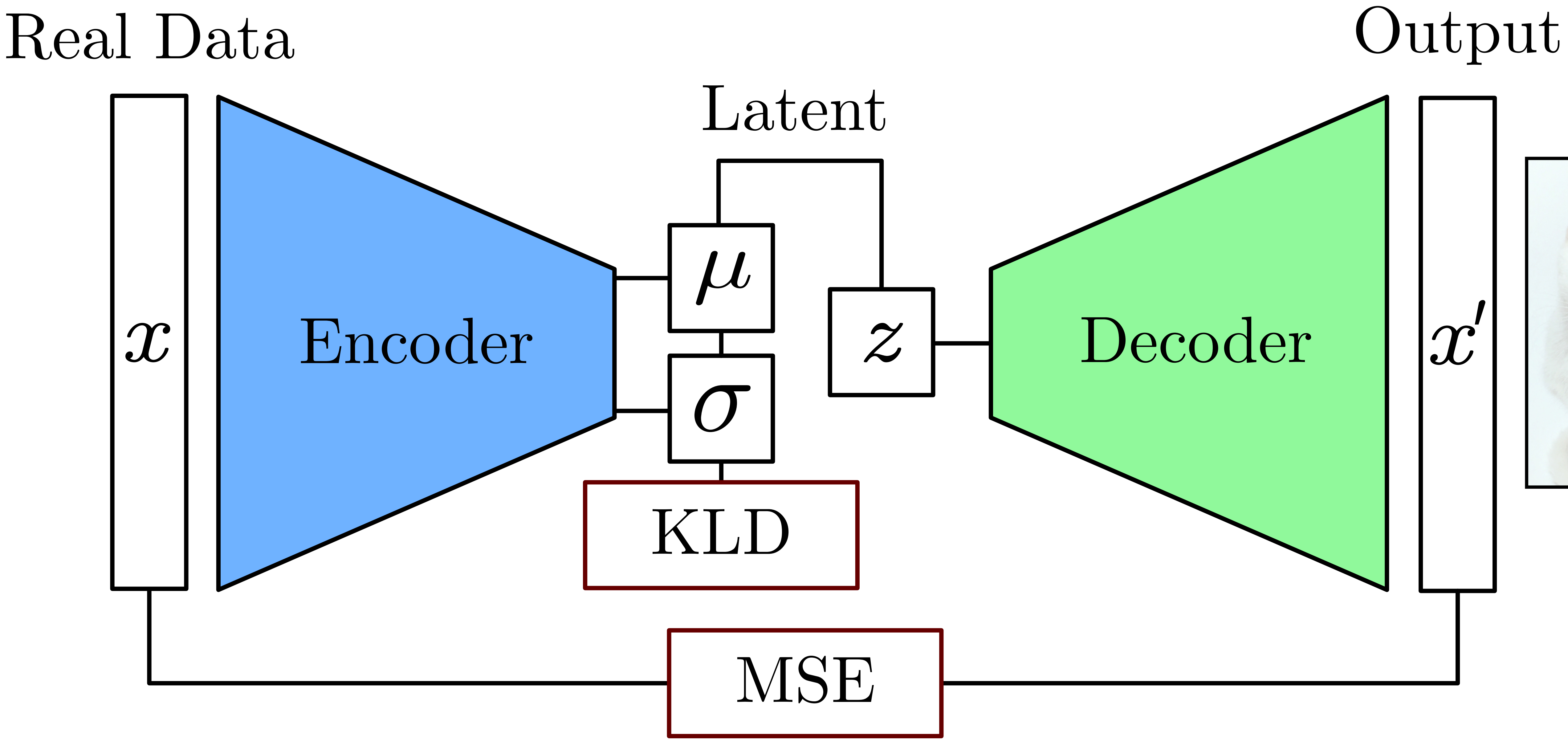


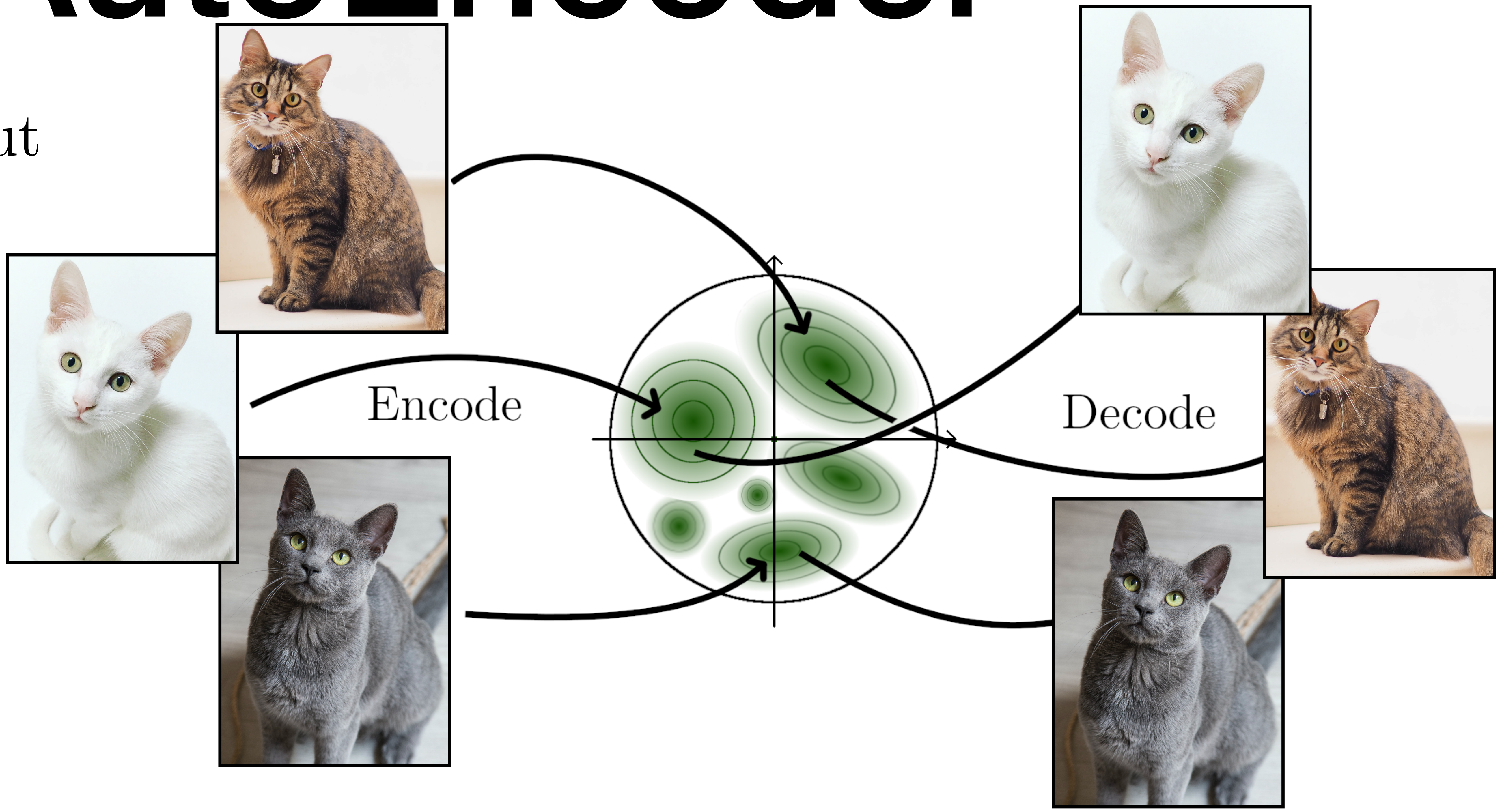


- Sample for Z and pass it to D(Z) ➡ Generate new samples
- Problem: Need regularised later space to sample form

➡ Variational AutoEncoder

Variational AutoEncoder





- Latent space: Series of Gaussians, regularised match N(μ=0, σ=1)
  - Using Gaussians lets us use Kullback–Leibler divergence

$$\sum_{i=1}^{n} \sigma_i^2+\mu_i^2-\log(\sigma_i)-1$$

  - Compare Input and Output again using MSE

Variational AutoEncoder

Upsides

- Directly evaluates log likelihood
- Stable in training

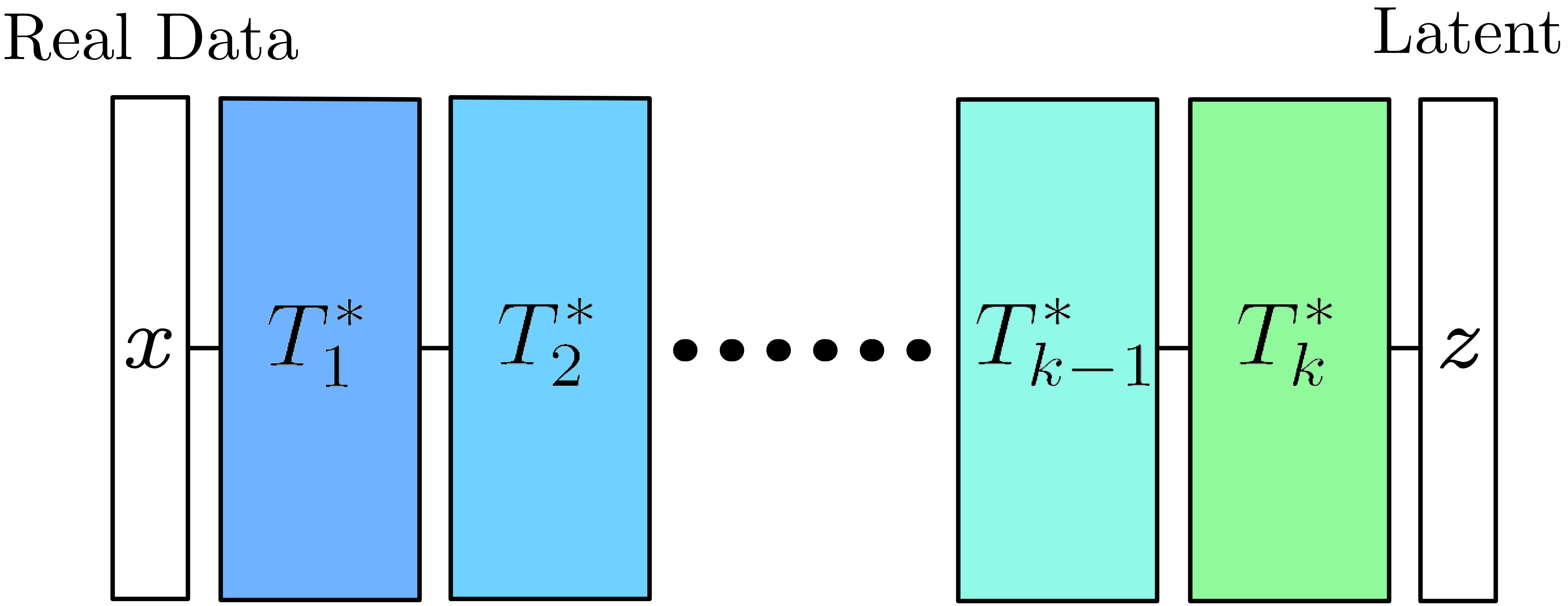
Difficulties

- MSE loss insufficient for certain data sets
- Needs to balance KLD and MSE loss terms

Normalizing Flows

- Variational AutoEncoder: map data to normal distribution and back using two networks
- Can we do this with a single network instead?

➡ Normalising Flow



Normalizing Flows

- Train invertible model T^{*-1} to map data to Normal distribution
- Well understood loss function:

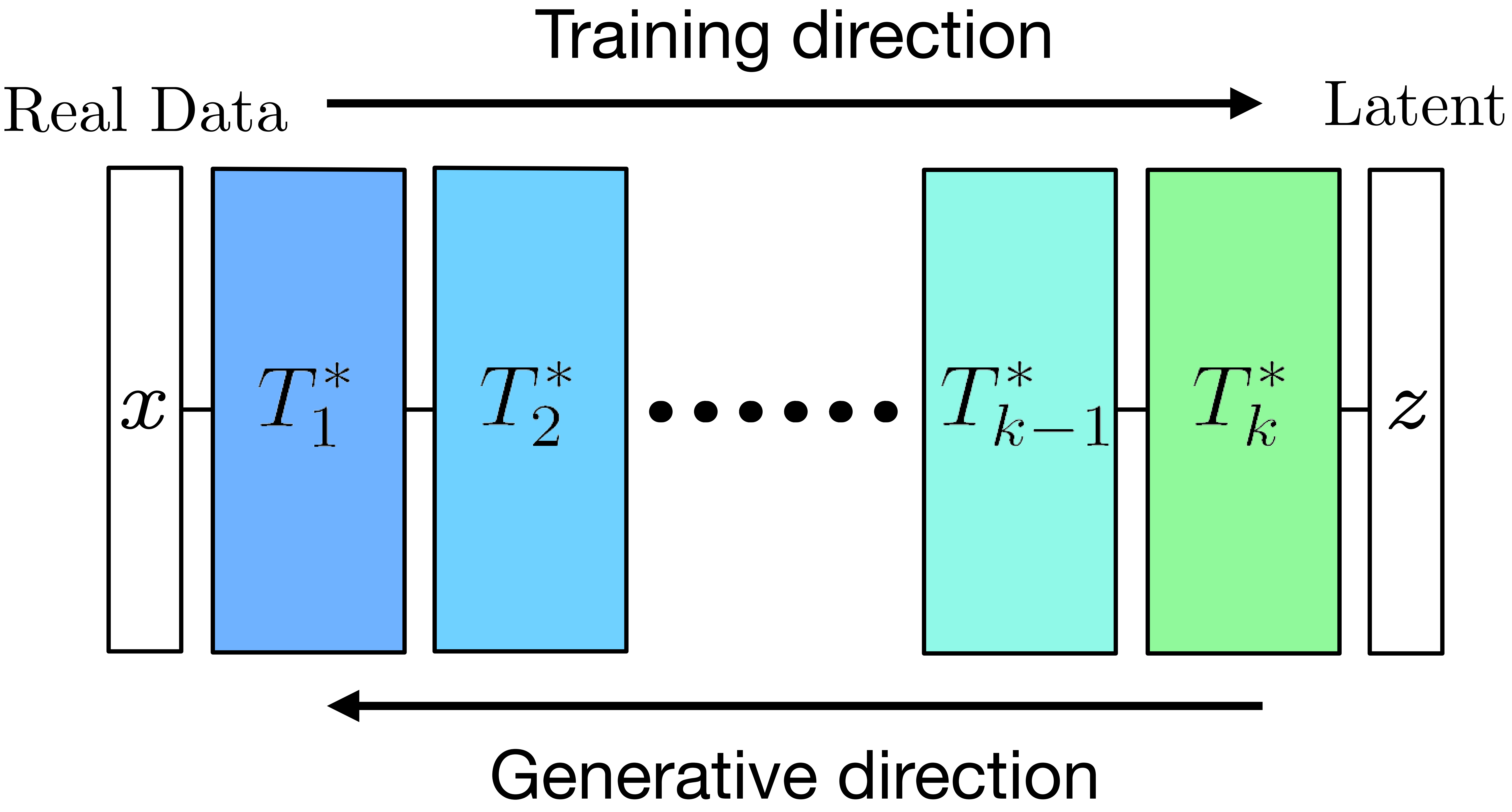
$$\mathcal{L}_{\text{Flow}} = \frac{1}{N}\sum_{n=1}^{N} \boxed{\log(p_z(T^{*-1}(\mathbf{x_n}, \theta)))} + \boxed{\log(|\det J_{T^{-1}}(\mathbf{x_n}, \theta)|)}$$

Latent Normal distribution

How well does the transformed sample match the latent distribution

Jacobean of transformation

Normalizing Flows



Normalizing Flows

- Train invertible model T^{*-1} to map data to Normal distribution
- Well understood loss function:

$$\mathcal{L}_{\text{Flow}} = \frac{1}{N}\sum_{n=1}^{N} \boxed{\log(p_z(T^{*-1}(\mathbf{x_n}, \theta)))} + \boxed{\log(|\det J_{T^{-1}}(\mathbf{x_n}, \theta)|)}$$

Latent Normal distribution

How well does the transformed sample match the latent distribution

Jacobean of transformation

Normalizing Flows

- Train invertible model T^{*-1} to map data to Normal distribution
- Well understood loss function:

$$\mathcal{L}_{\text{Flow}} = \frac{1}{N}\sum_{n=1}^{N} \log(p_z(T^{*-1}(\mathbf{x_n}, \theta))) + \boxed{\log(|\det J_{T^{-1}}(\mathbf{x_n}, \theta)|)}$$

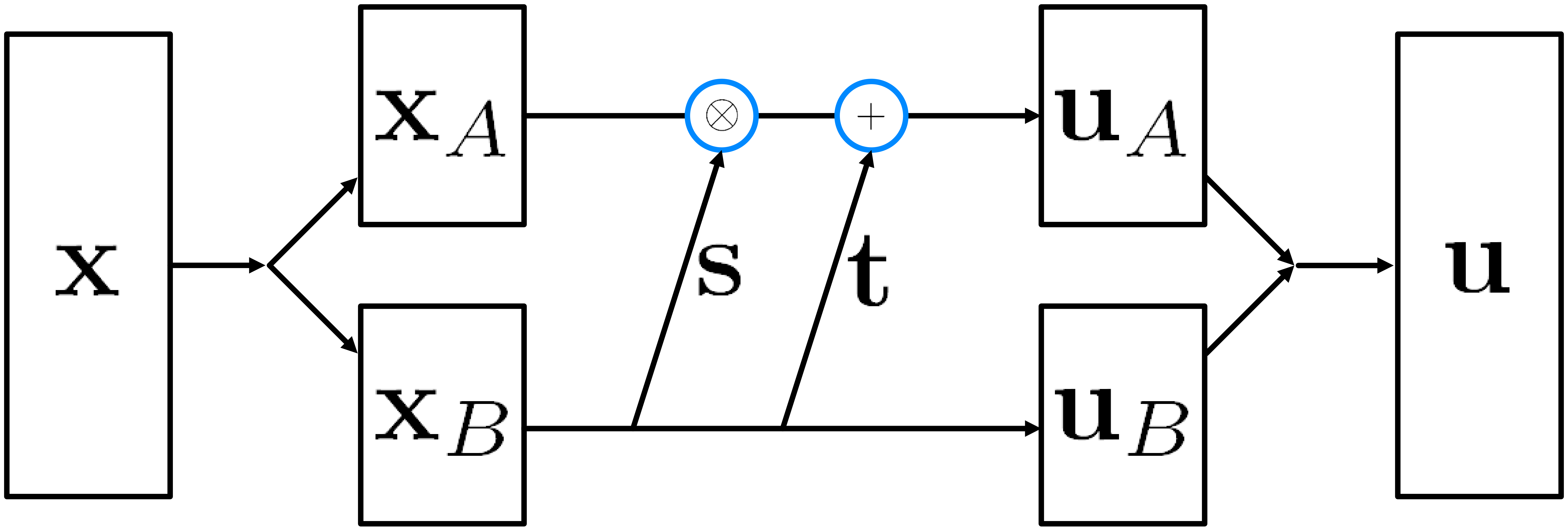
Latent Normal distribution

How well does the transformed sample match the latent distribution

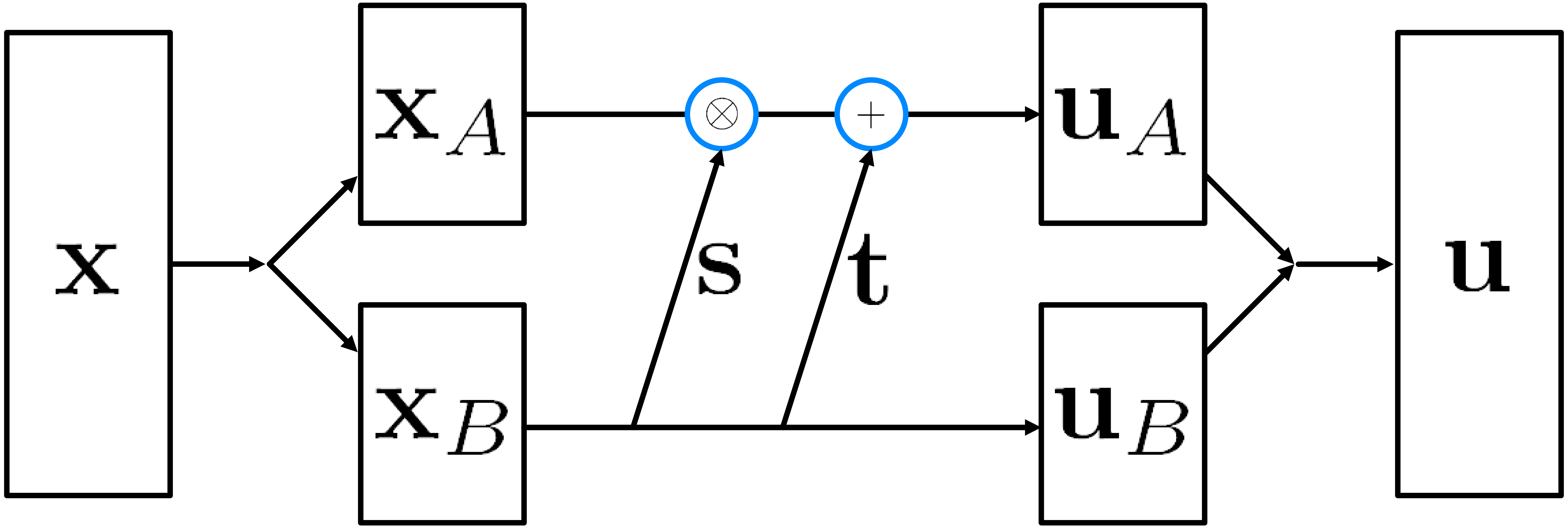
Jacobean of transformation

Invertible Neural Networks

- Dense layer: matrix multiplication
- Invertible if square matrix with det > 0
- Difficult in practice, use Coupling layers instead



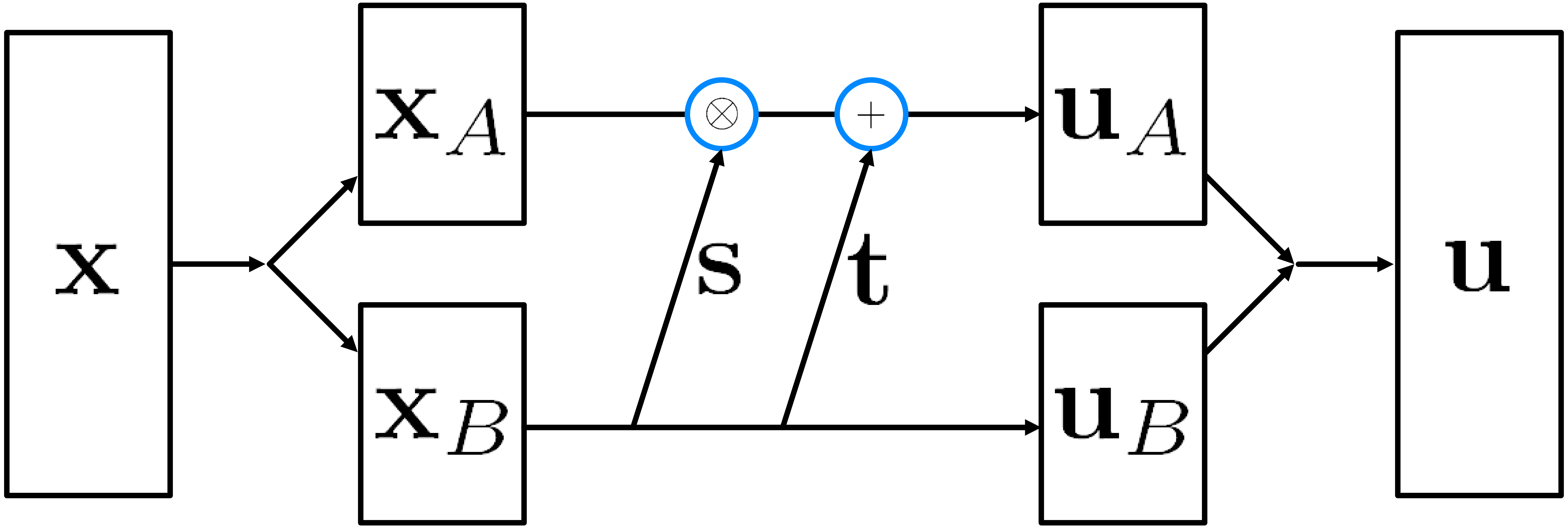
Invertible Neural Networks



$$\mathbf{u}_A = \mathbf{x}_A \otimes \mathbf{s}(\mathbf{x}_B) + \mathbf{t}(\mathbf{x}_B)$$

$$\mathbf{u}_B = \mathbf{x}_B$$

Invertible Neural Networks



$$\mathbf{u}_A = \mathbf{x}_A \otimes \mathbf{s}(\mathbf{x}_B) + \mathbf{t}(\mathbf{x}_B)$$

$$\mathbf{u}_B = \mathbf{x}_B$$

$$\mathbf{x}_A = \frac{\mathbf{u}_A - \mathbf{t}(\mathbf{u}_B)}{\mathbf{s}(\mathbf{u}_B)}$$

$$\mathbf{x}_B = \mathbf{u}_B$$

Normalizing Flows

Upsides

- Directly evaluates log likelihood
- Stable in training
- High generative quality
- Easy to train and use

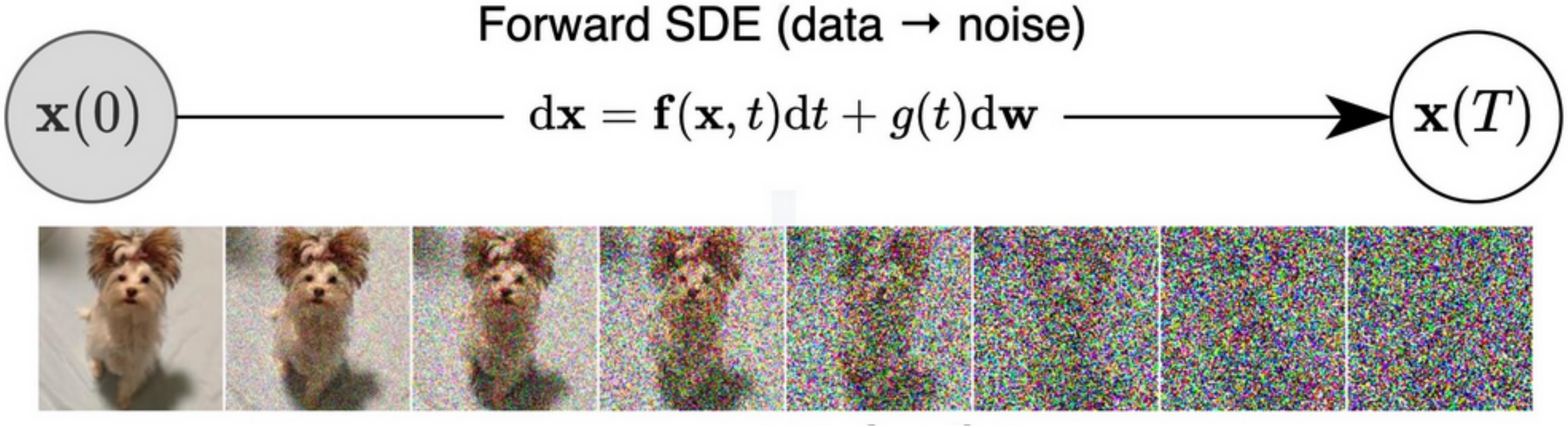
Difficulties

- Fixed dimensionality through entire flow
- Slow generation times for large models/data

Score-Based Models

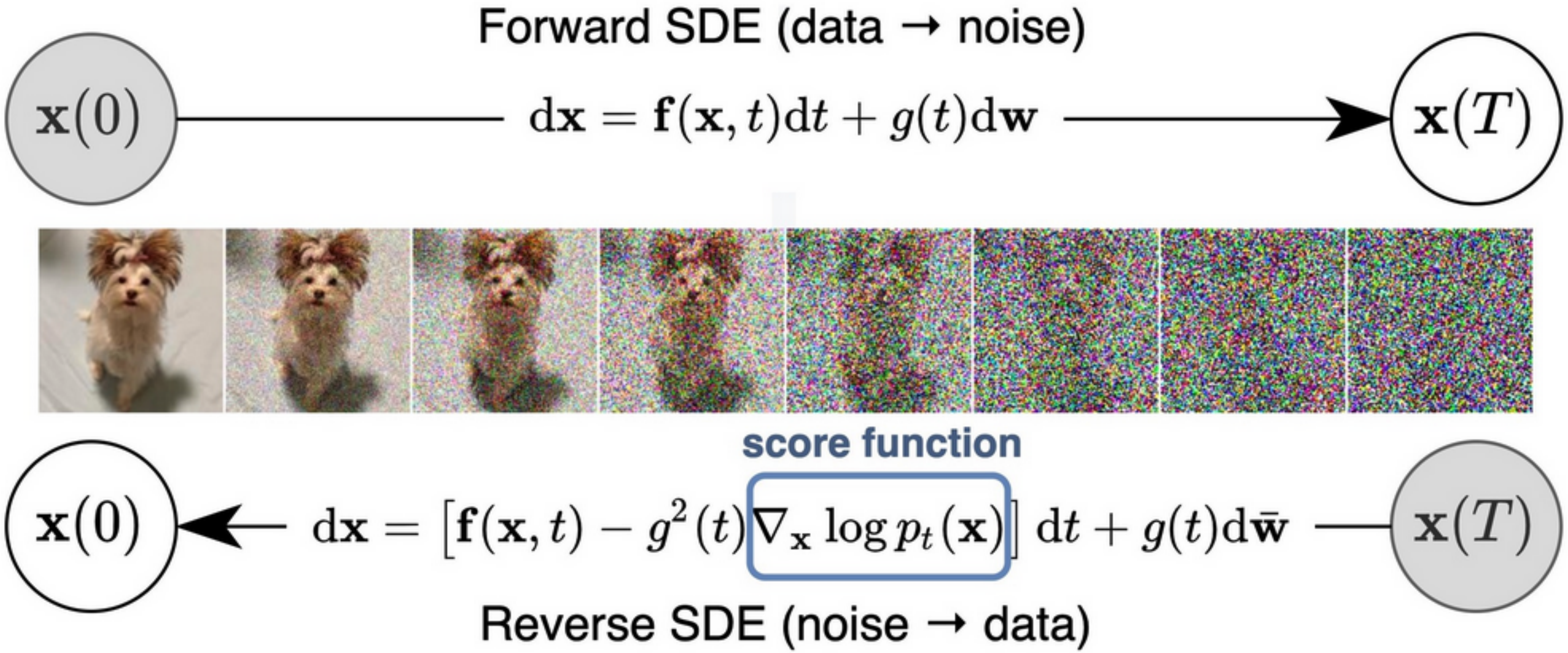
- Normalizing Flow: learn map of data to Normal distribution
- Do we need to explicitly learn this?
- Map data to Normal distribution by repeatedly adding noise

➔ Diffusion Models



Score-Based Models

- Repeatedly add noise to data point
- Learn to undo noise at every step
- Possible using Stochastic Differential Equation and score function



Score-Based Models

- Approximating the score function of the data
- Equivalent to approximating the score function of a smearing func.

Score func.

$$\frac{1}{2}\mathbb{E}_{p(\mathbf{x})}\left[\|\mathbf{s}_\theta(\mathbf{x},t) - \nabla_\mathbf{x} \log p(\mathbf{x})\|_2^2\right].$$

Score func. of smearing

$$\frac{1}{2}\mathbb{E}_t\mathbb{E}_{p_t(x)}\left[\lambda(t)\,\|\mathbf{s}_\theta(\mathbf{x_t},t) - \nabla_{\mathbf{x_t}} \log p_t(x_t|x_0)\|_2^2\right]$$

For gaussian smearing

$$\nabla_{\tilde{\mathbf{x}}} \log p_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = \frac{\mathbf{x}-\tilde{\mathbf{x}}}{\sigma^2} \sim \frac{\mathcal{N}(0,1)}{\sigma}$$

Score-Based Models

Upsides

- Superior generation quality
- State of the Art for generative models

Difficulties

- Conceptually complex
- Difficult to train for non-standard data-sets

Physics Use Cases

Generative Models can

- Learn underlying distribution from a given dataset
- Quickly sample from learned distribution

What HEP uses can such models have?

- Fast simulation

Physics Use Cases

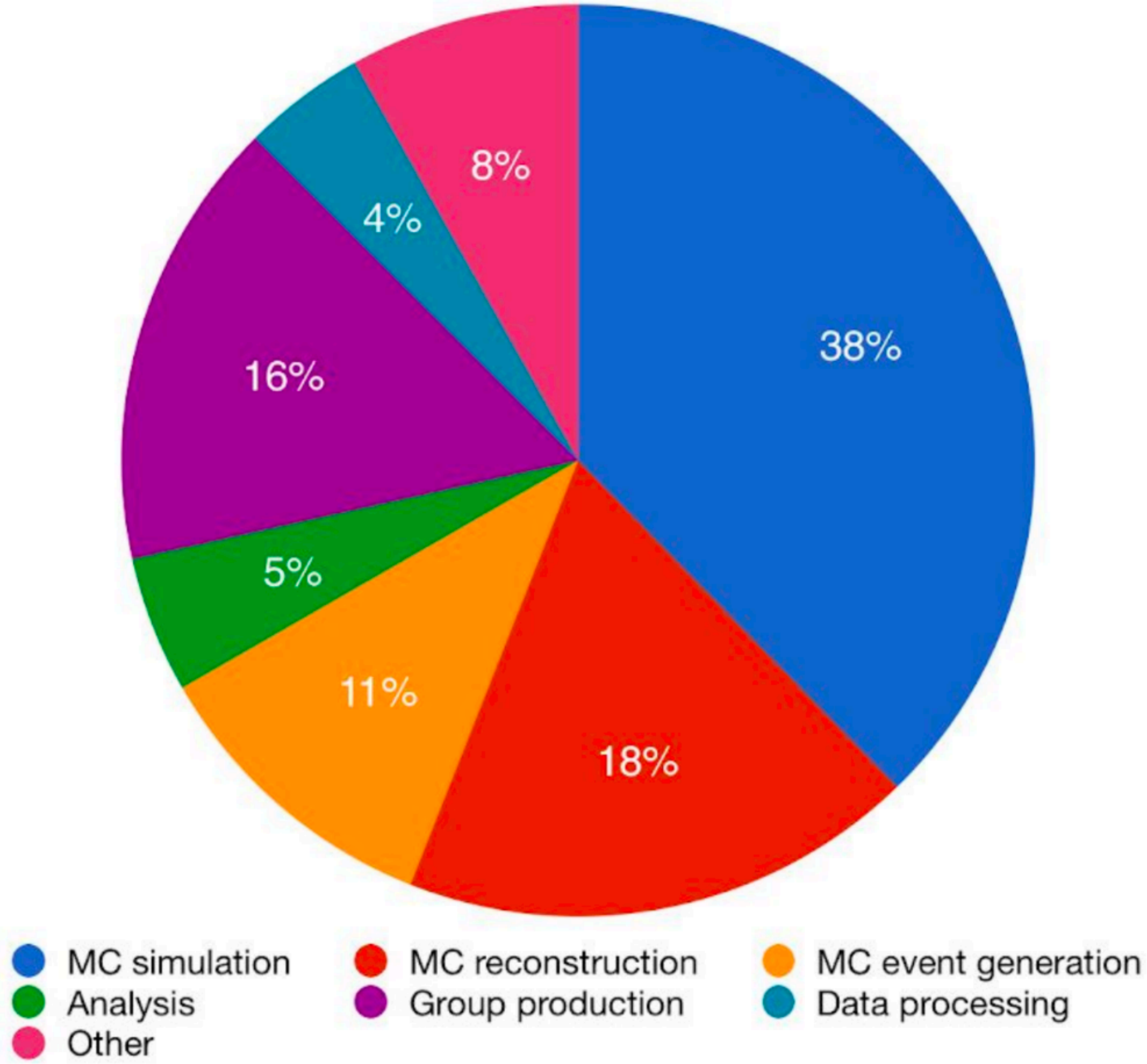
Generative Models can

- Learn underlying distribution from a given dataset
- Quickly sample from learned distribution

What HEP uses can such models have?

- Fast simulation
- Anomaly detection
- Many more!

Generative Simulation



- MC simulation large part of computing
- Speed up:
  - Train ML model on small dataset
  - Draw majority of samples form ML model

    ➔ Amplify original data set

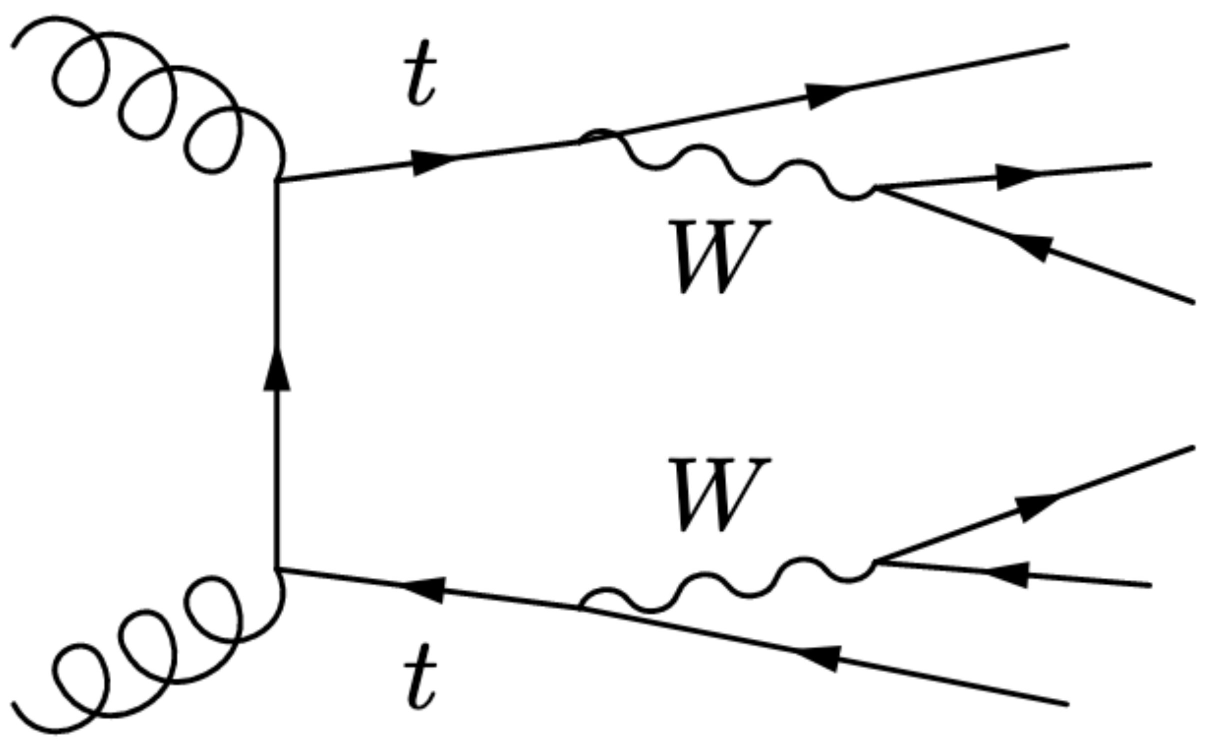
    ➔ Significantly faster

Catmore et. al. **ATLAS HL-LHC Computing Conceptual Design Report,** [CERN-LHCC-2020-015 ; LHCC-G-178]()

Butter et al.: **Amplifying Statistics using Generative Models:** NeurIPS ML4PS 2020, [2008.06545]()

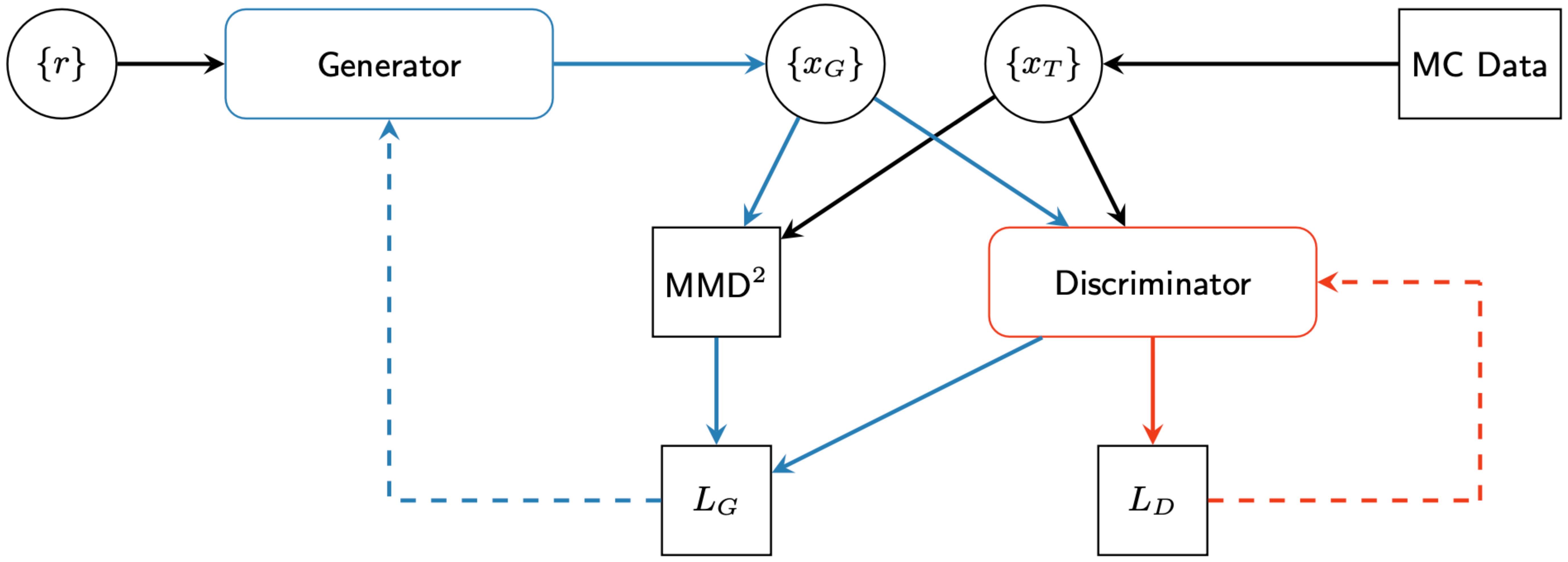
Generative Simulation

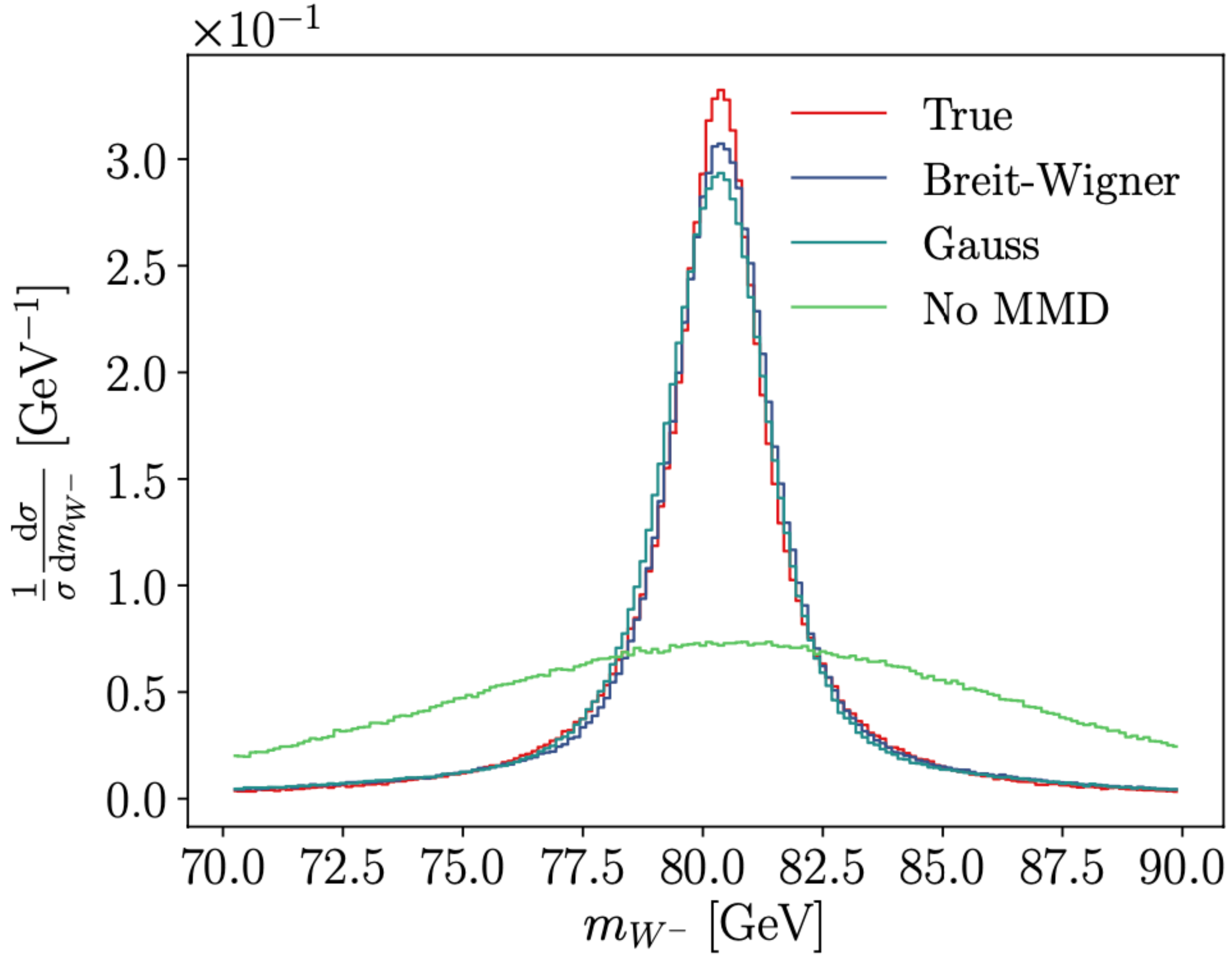
Event Generation



Event Generation

- Generation of event 4-momenta, ordered list, O(10) dimensions
- Using GAN model with additional MMD loss for mass peaks

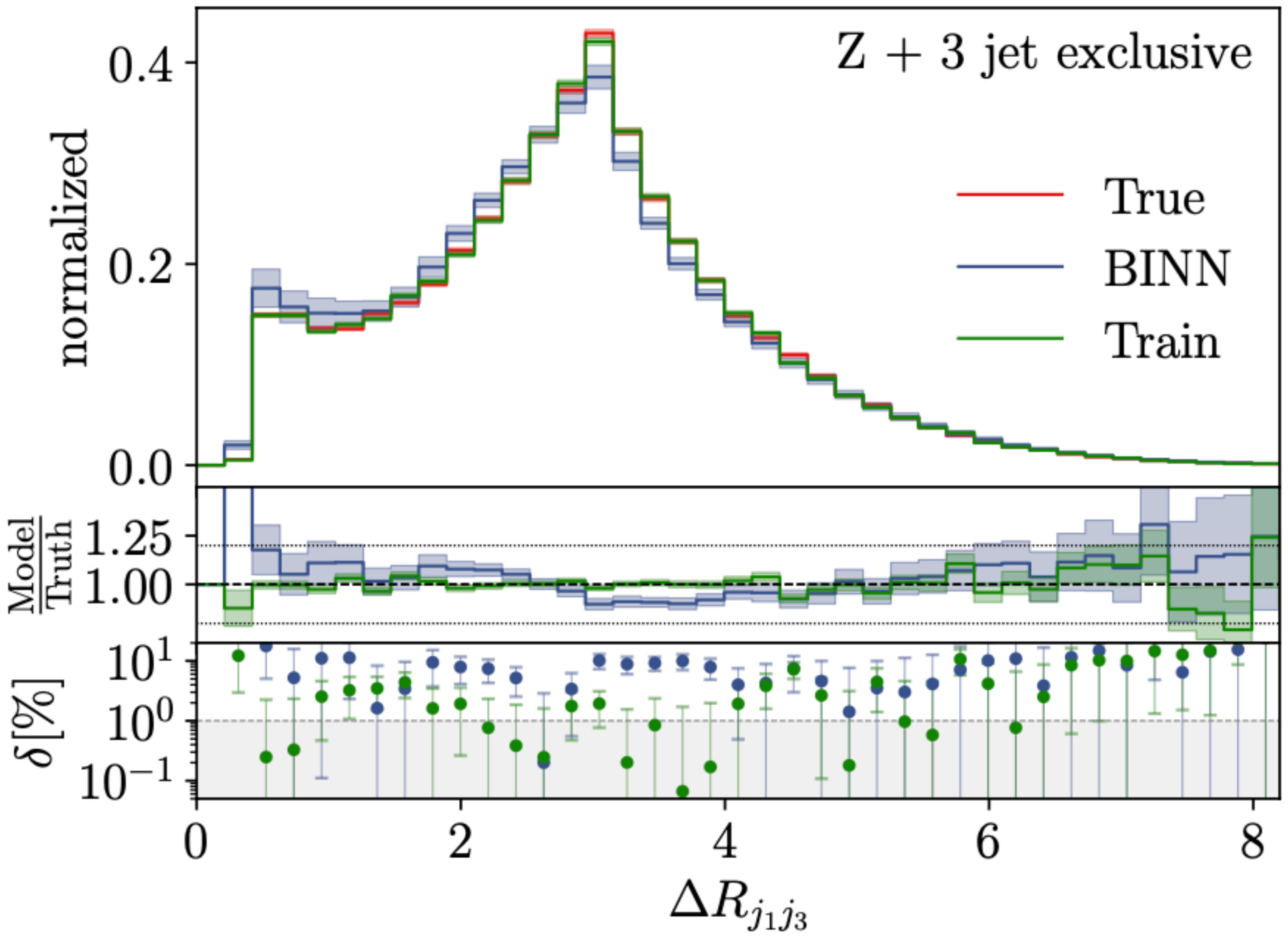


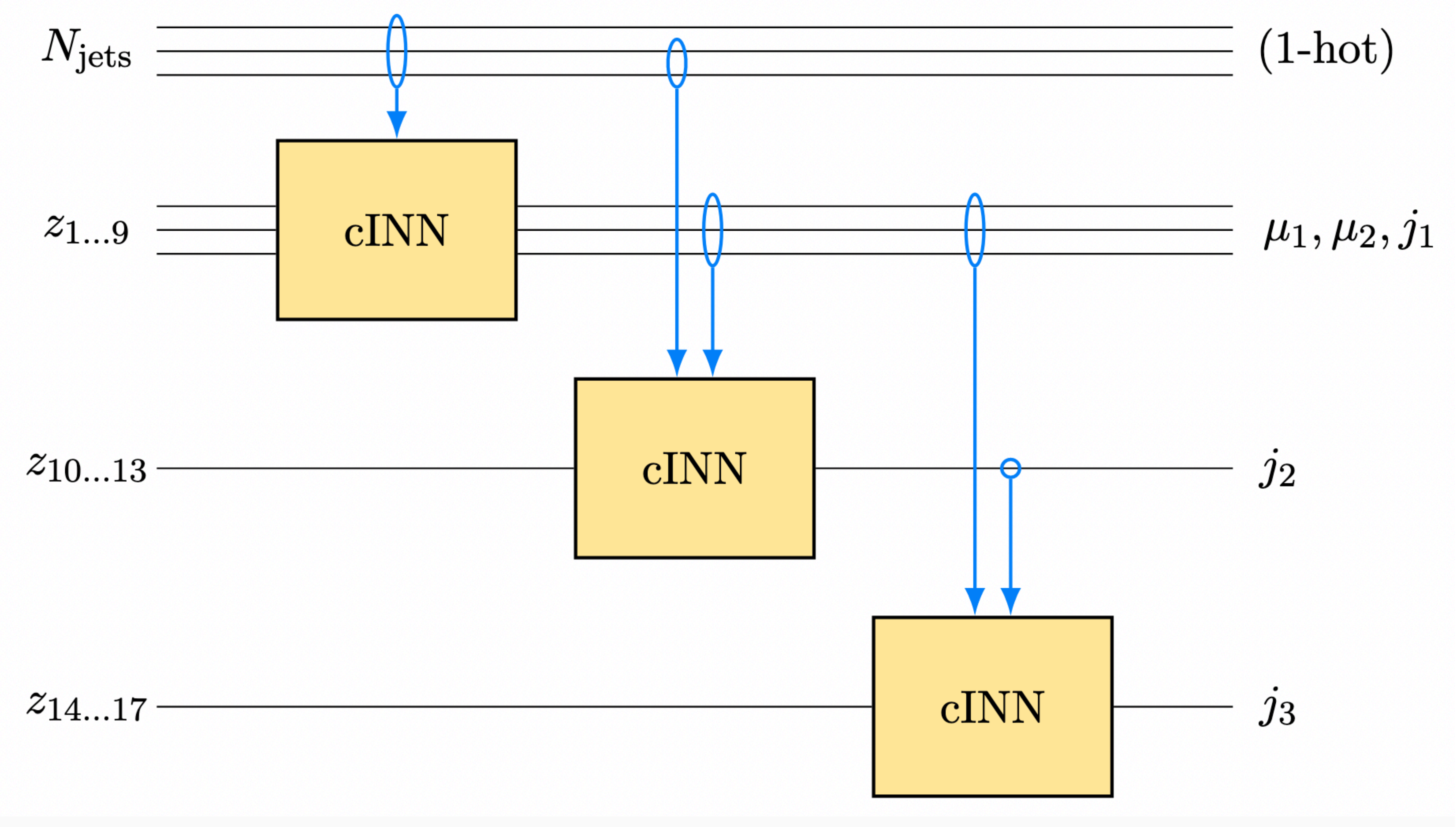


Butter et al.: **How to GAN LHC Events:** (2019), [1907.03764]

Event Generation

- Normalizing flow approach improves precision
- Bayesian network enables uncertainty estimation

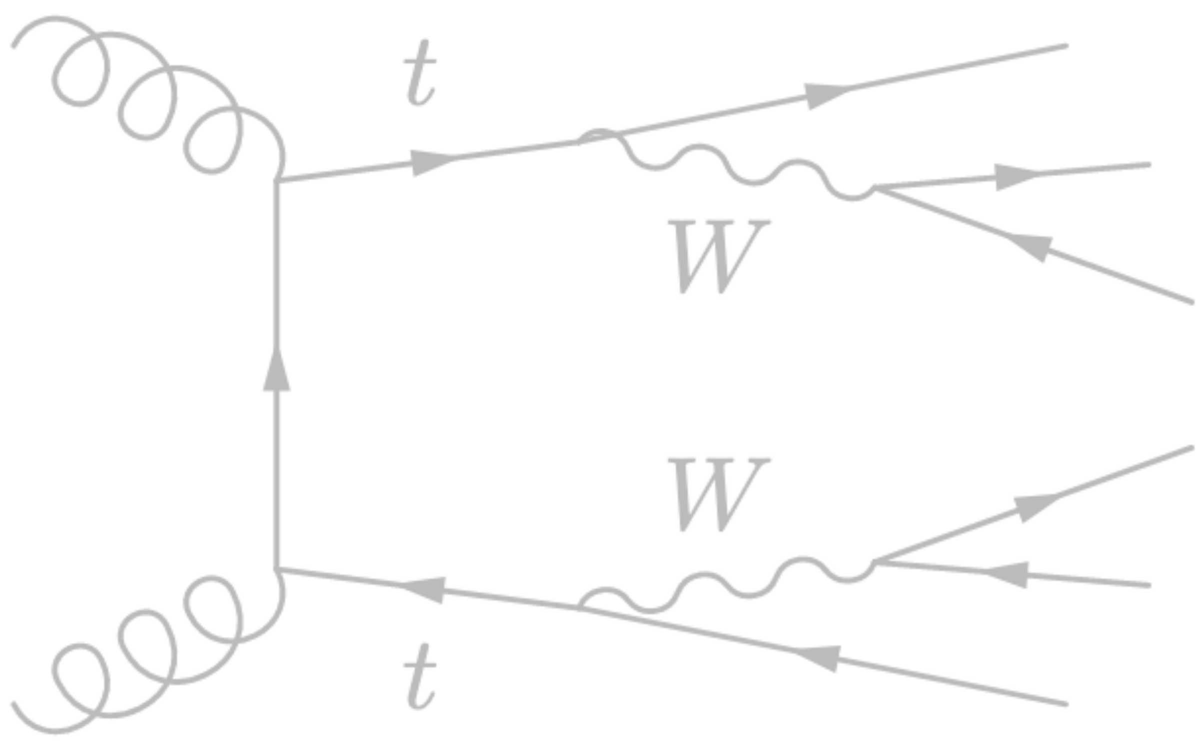




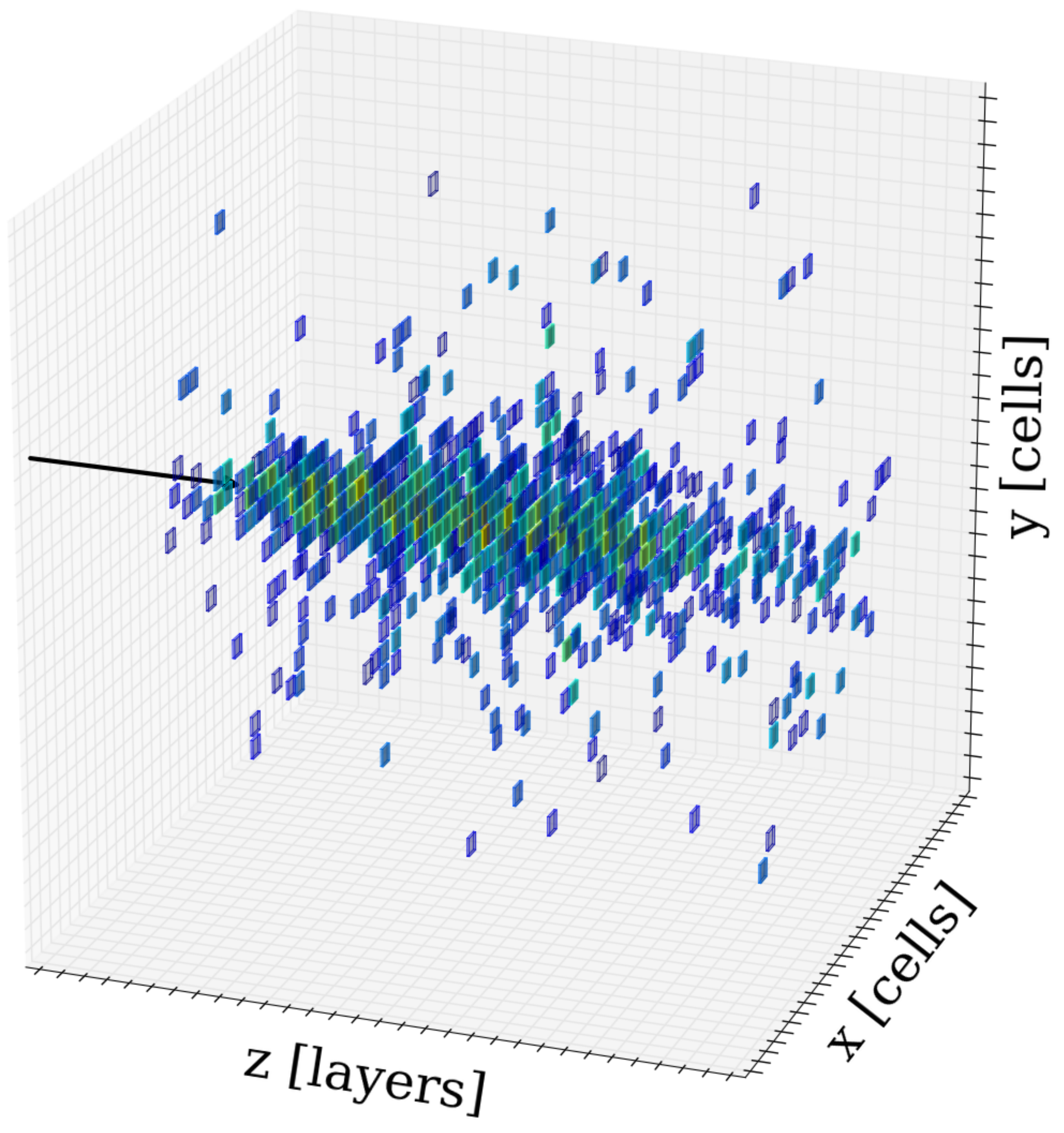
Butter et al.: **Generative Networks for Precision Enthusiasts** (2021), 2110.13632

Generative Simulation

Event Generation

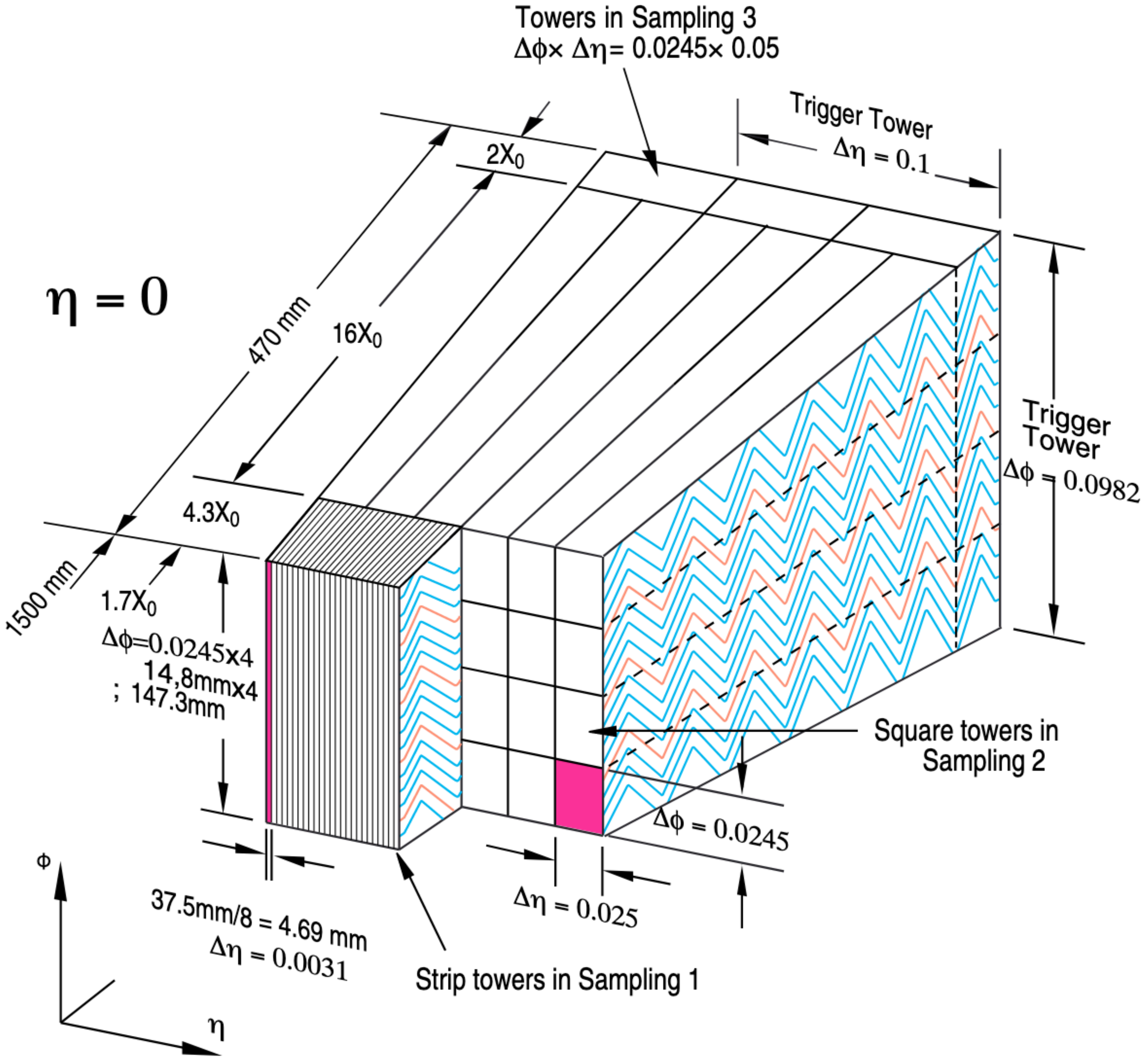


Detector Simulation



Detector Simulation

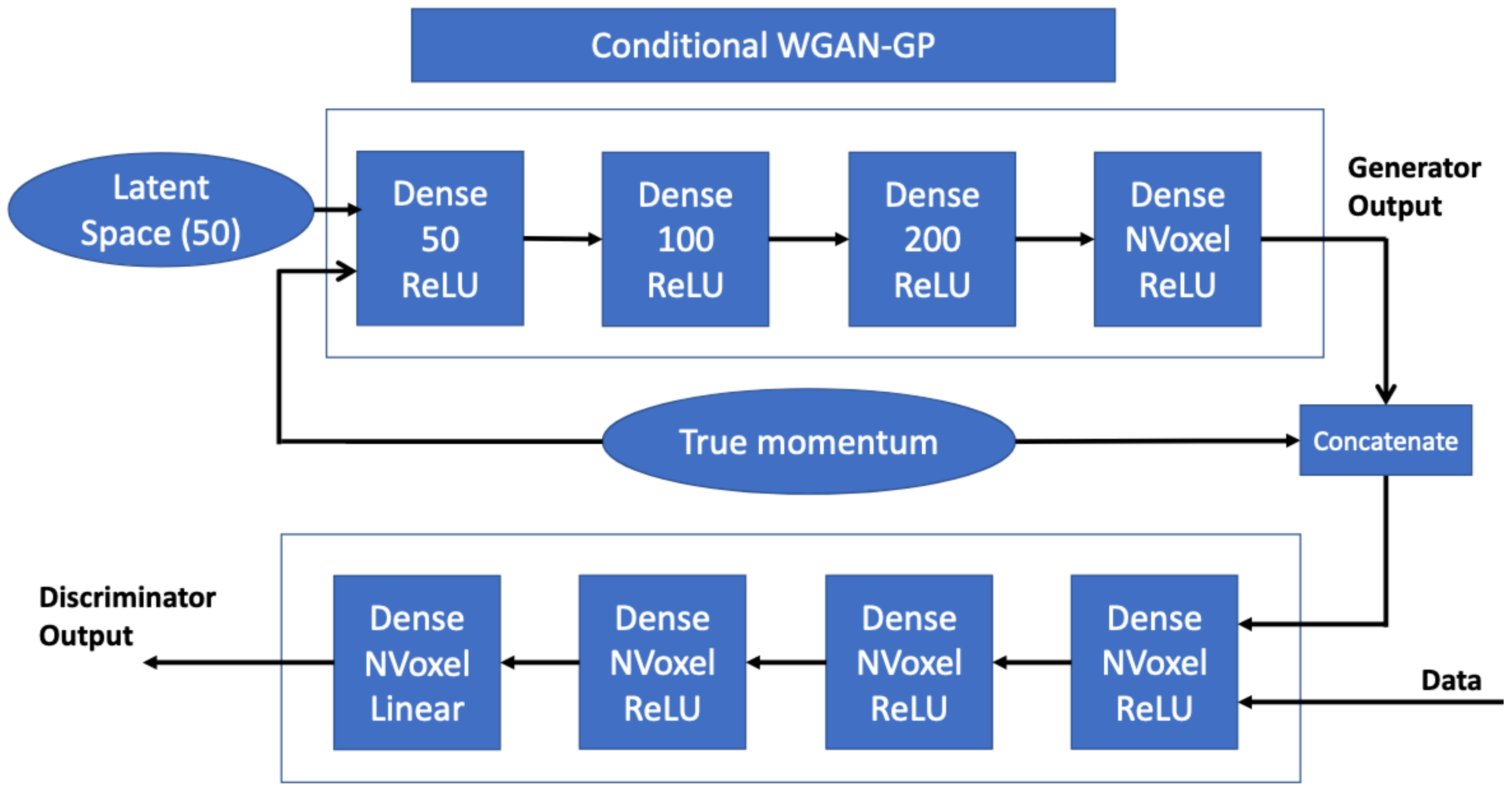
- Fixed output geometry O(100-10,000) dimensions
- Common in detector simulation, e.g. ATLAS FastCaloGAN
- Already in use for fast simulation of calorimeter showers

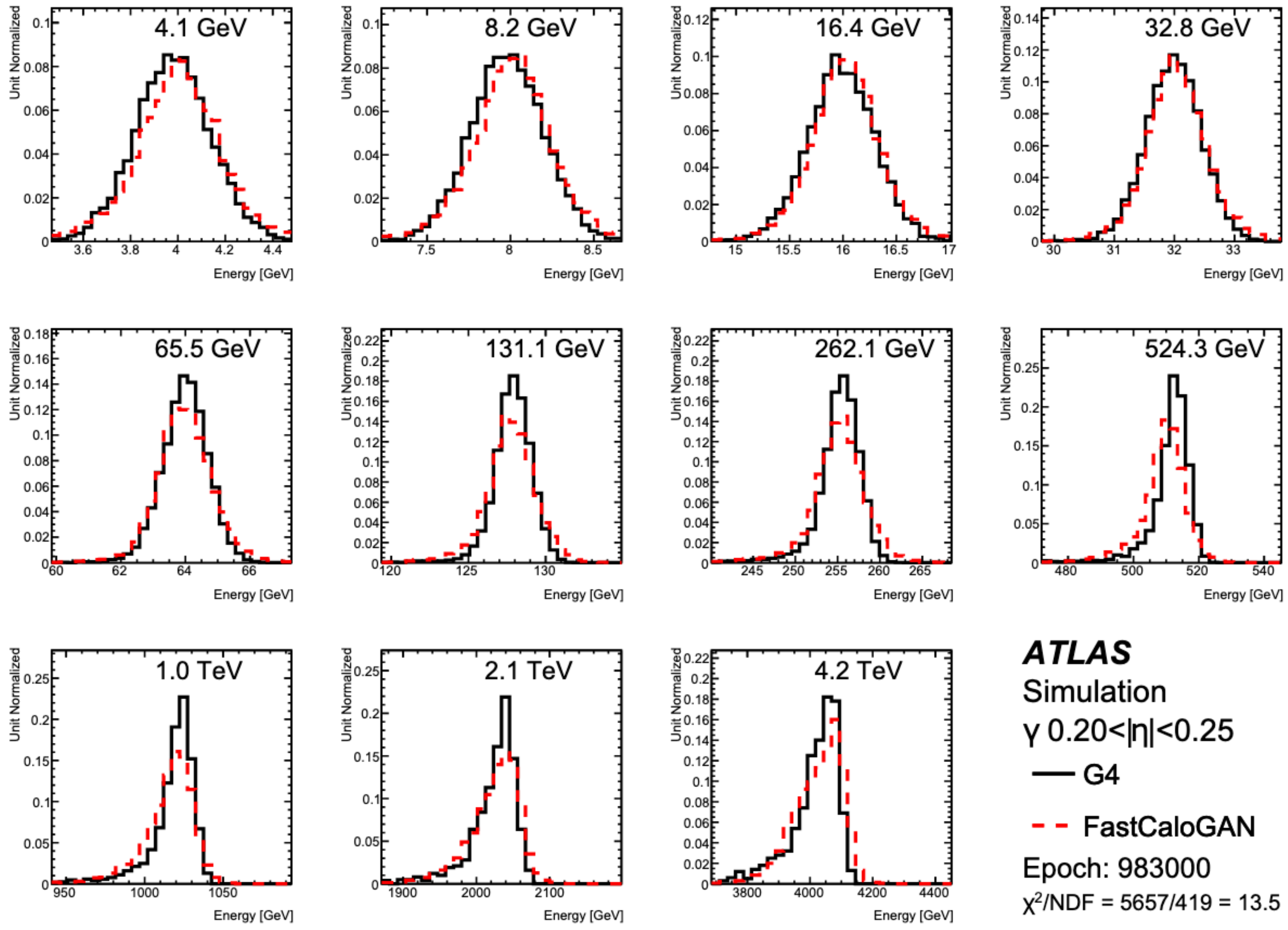


ATLAS Collaboration, **AtlFast3: the next generation of fast simulation in ATLAS** (2021), [2109.02551]

Detector Simulation

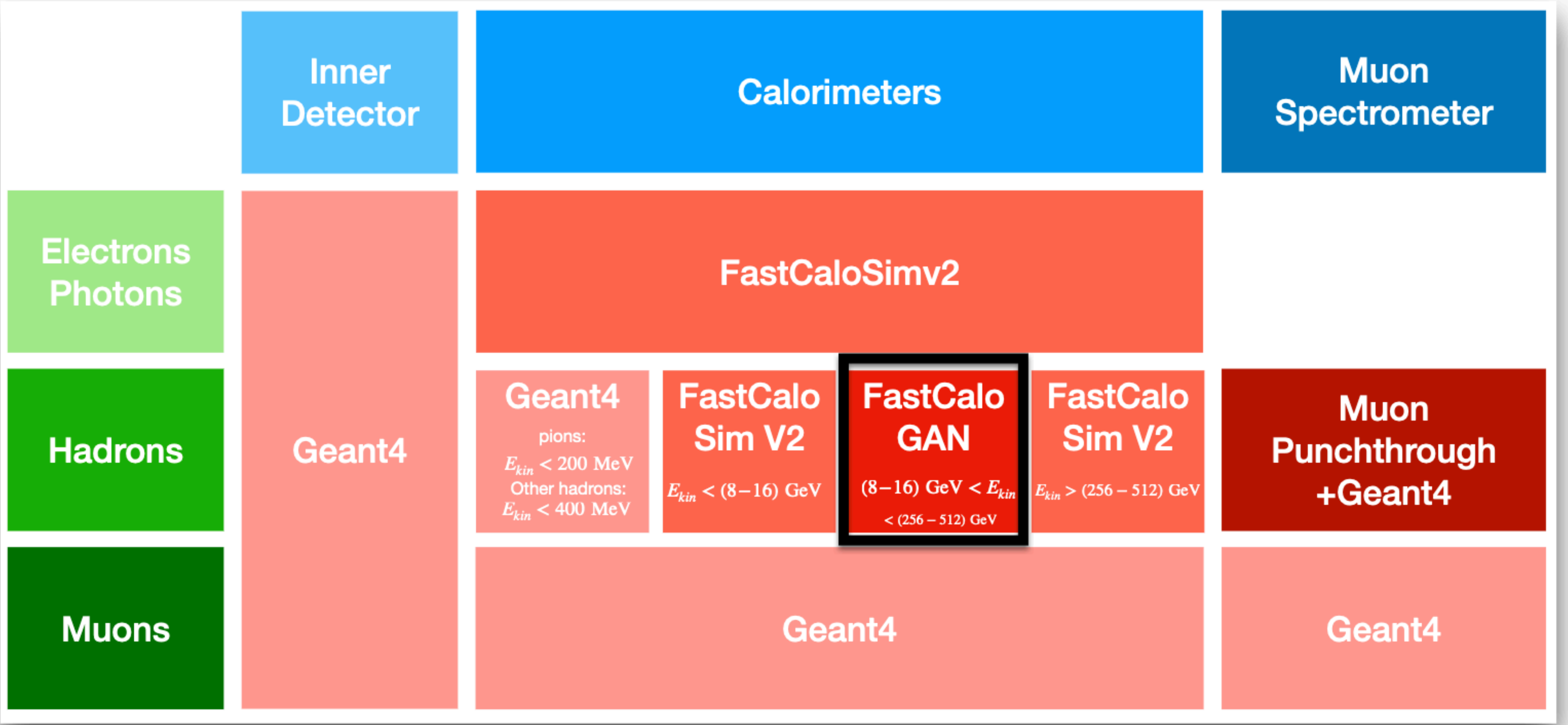
- ATLAS FastCaloGAN
- 300 total networks for particle types and η slices





ATLAS Collaboration, **AtlFast3: the next generation of fast simulation in ATLAS** (2021), [2109.02551](2109.02551)

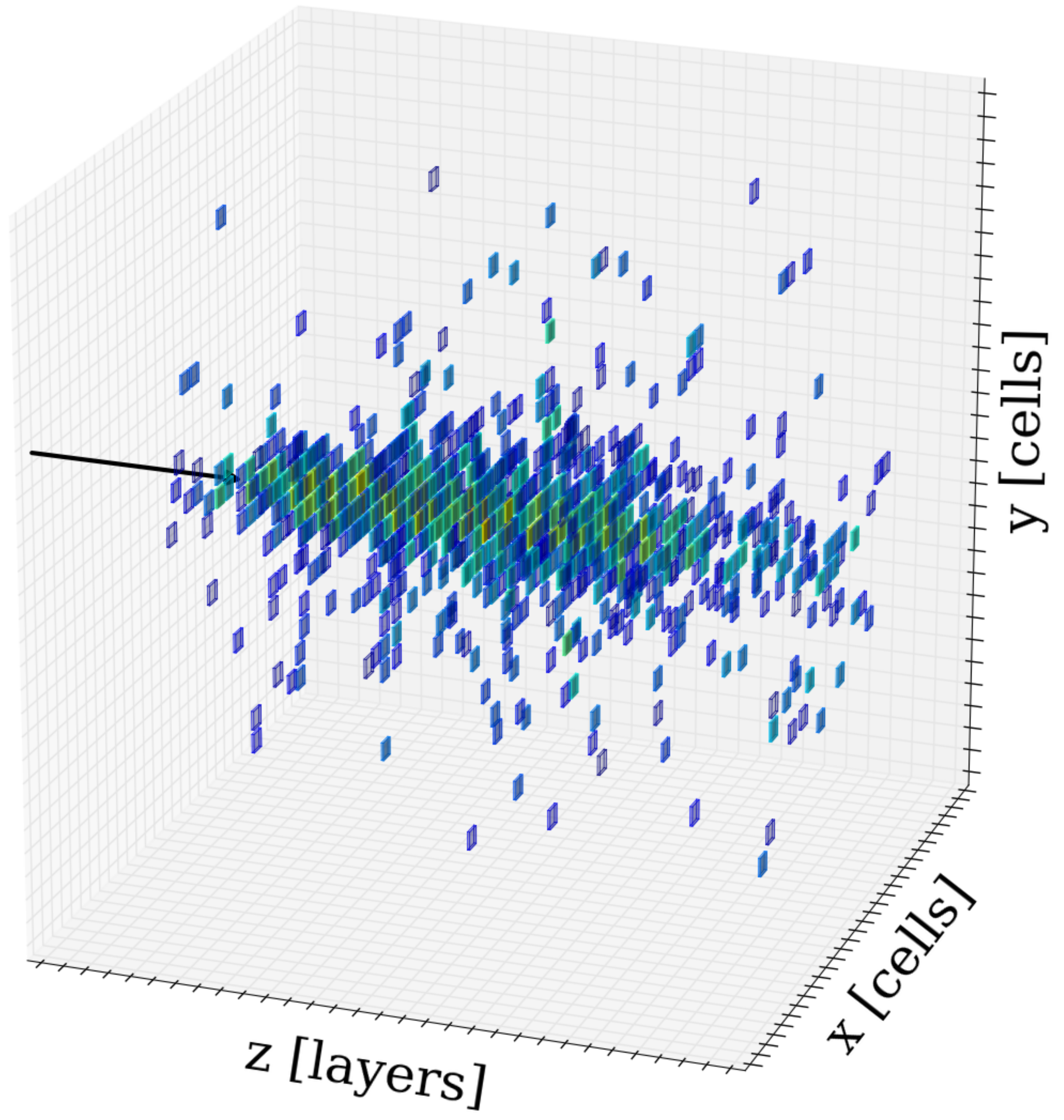
Detector Simulation



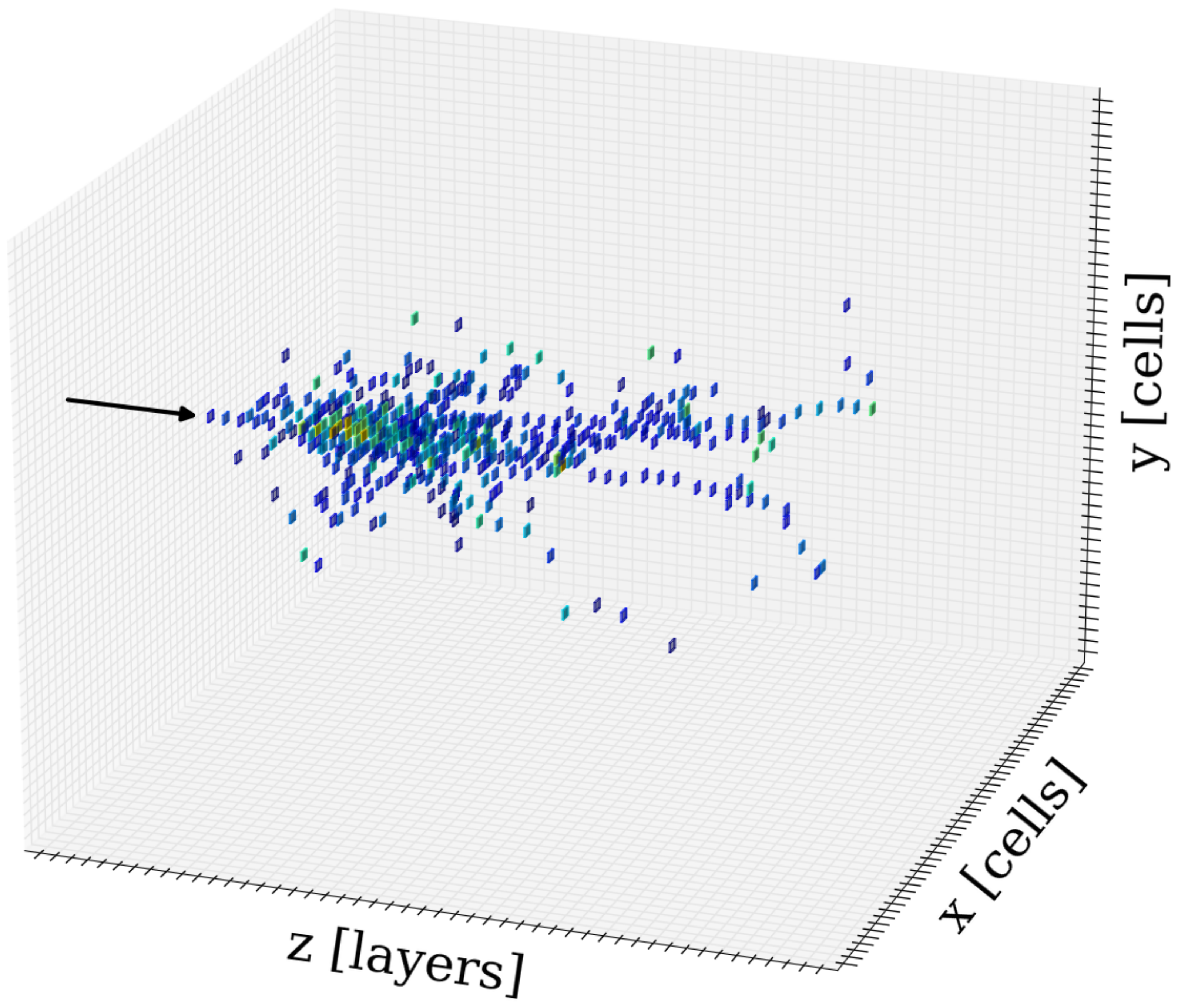
ATLAS Collaboration, **AtlFast3: the next generation of fast simulation in ATLAS** (2021), 2109.02551

High Granularity Calorimeter

Photon shower



Charged pion shower



Training data:

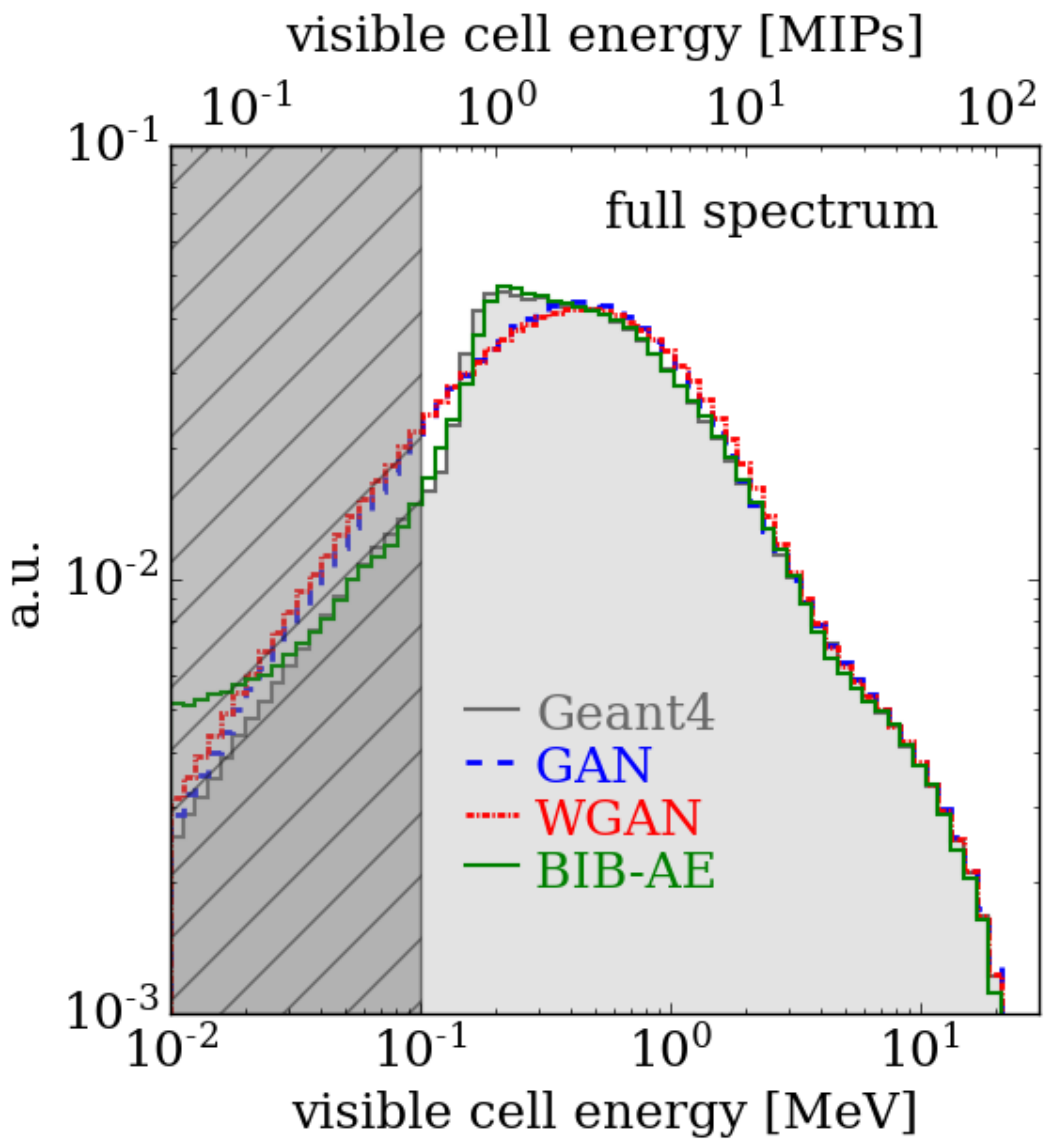
- Photons / charged Pions
- 1 million / 500k showers
- 10 to 100 GeV
- Fixed incident point & angle
- Project to grid
- 30×30×30 / 25×25×48

Buhmann et al.: **Getting High: High Fidelity Simulation of High Granularity Calorimeters with High Speed** (2020) 2005.05334

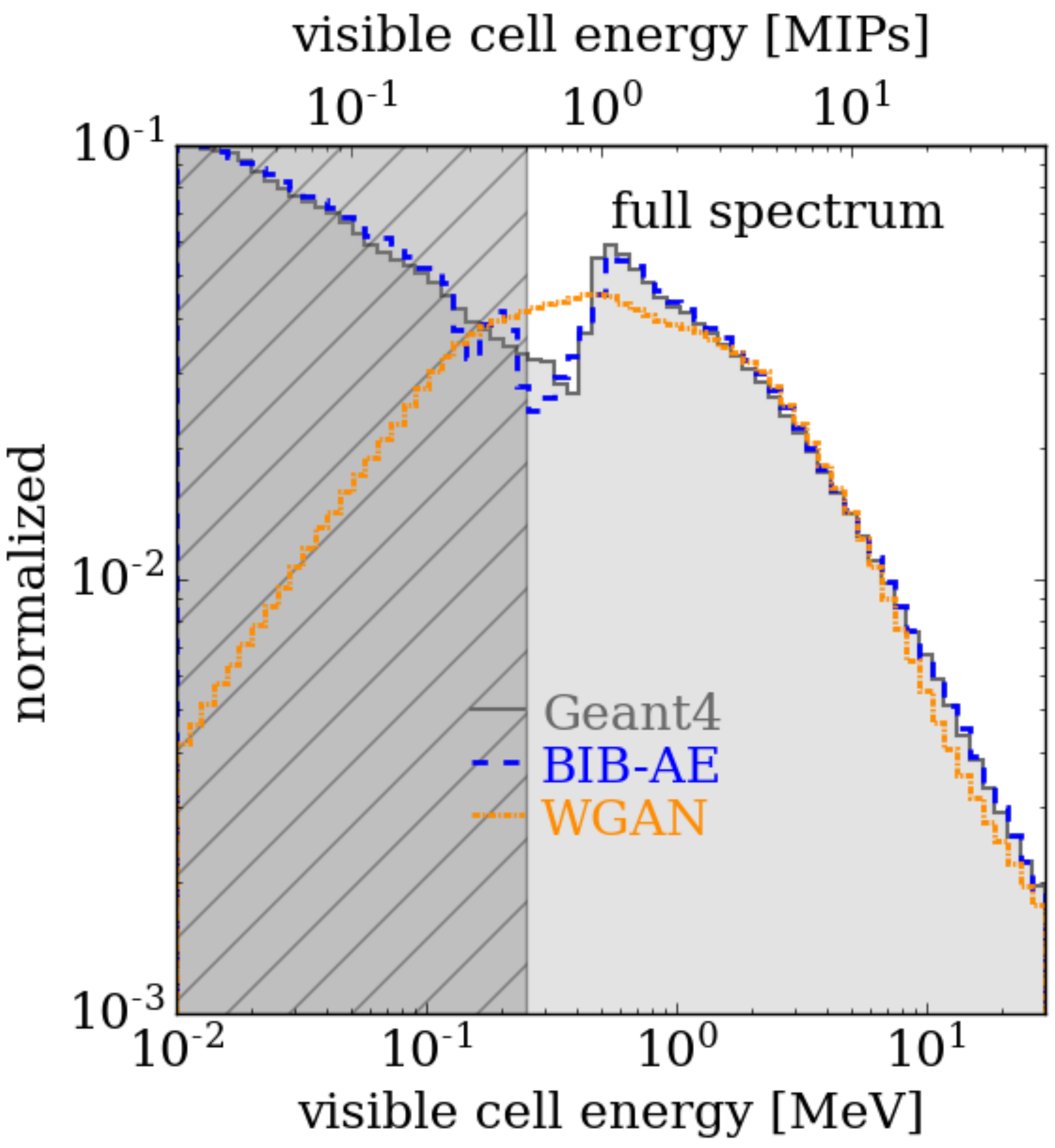
Buhmann et. al. **Hadrons, Better, Faster, Stronger:** (2021) 2112.09709

High Granularity Calorimeter

Photons



Pions



Anomaly Detection

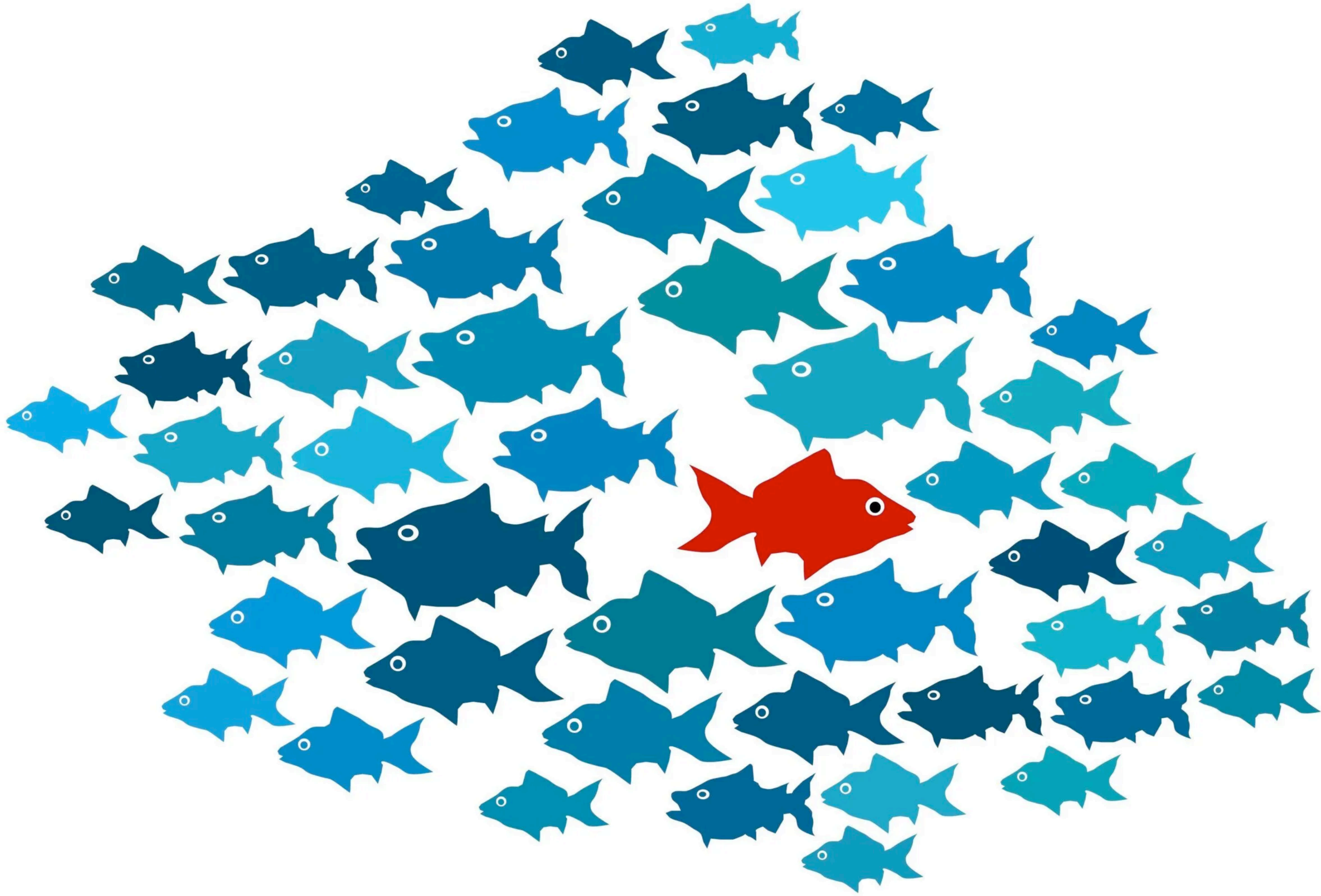
So far: no new discoveries of particles beyond the Standard Model

- Maybe not enough data

  → Previous ML methods

- Not looking for the right theory

  → Anomaly detection

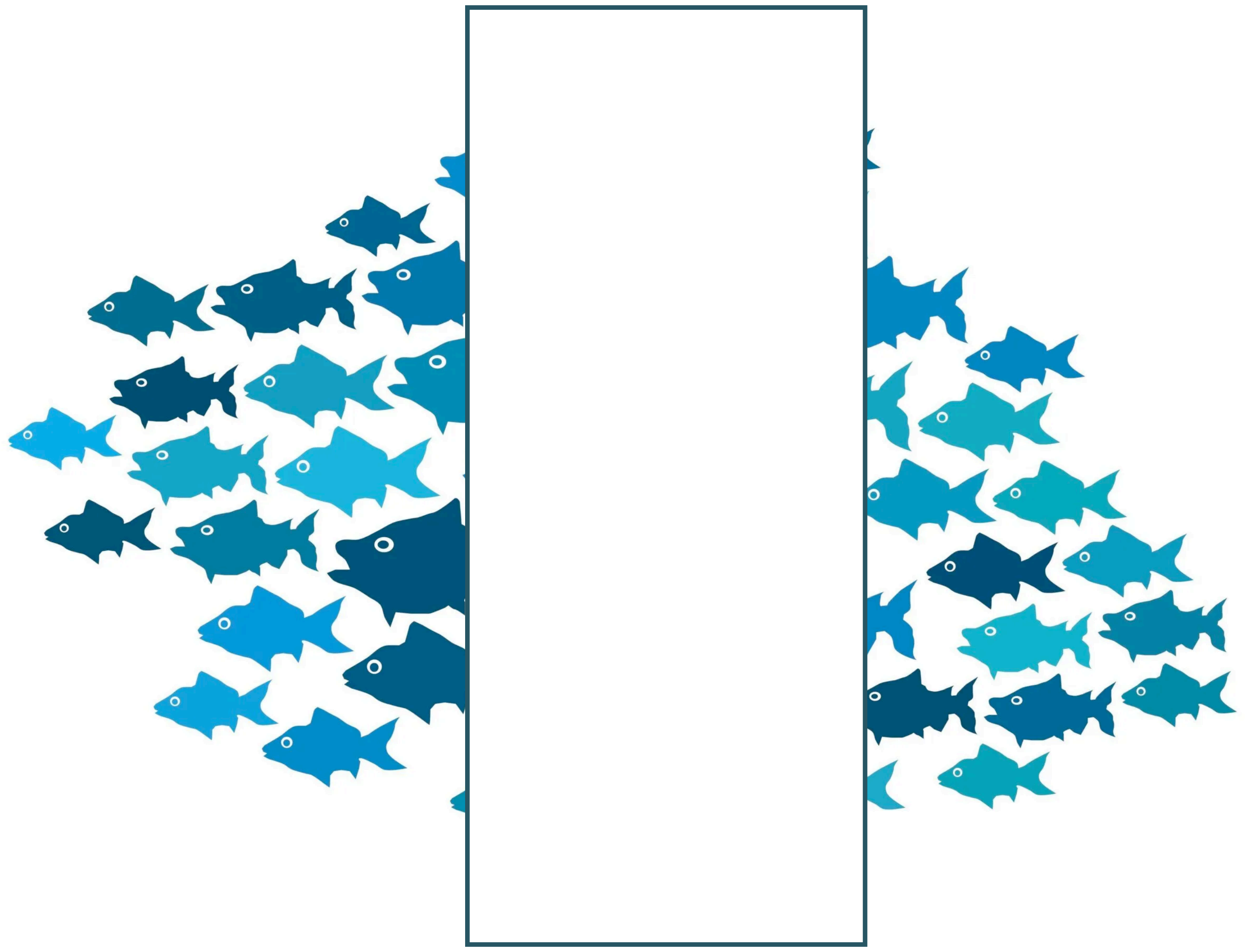


https://www.tatvic.com/blog/detecting-real-time-anomalies-using-r-google-analytics-360-data/

Anomaly Detection

Semi-Supervised AD:

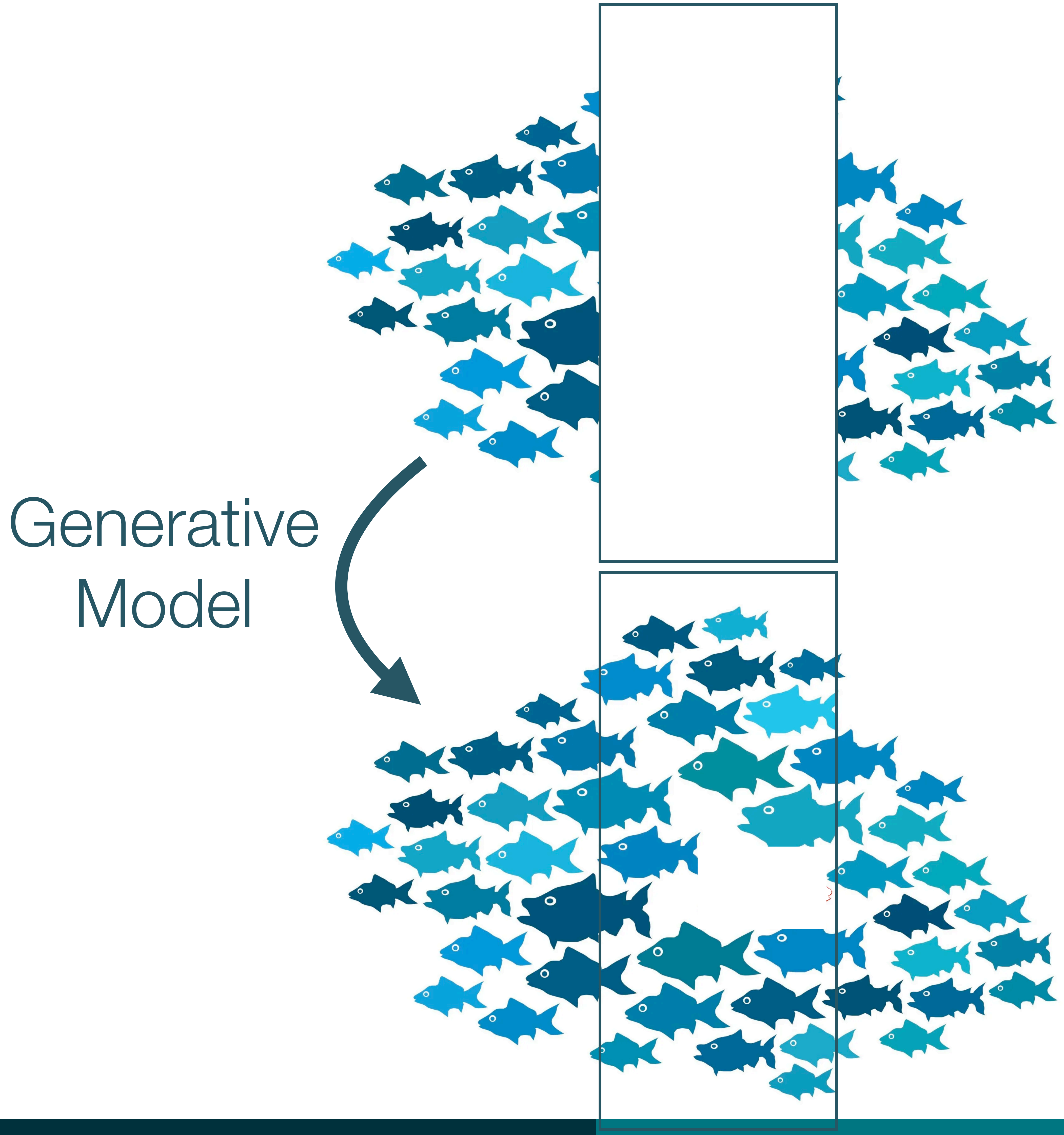
- Find unexpected samples
- Cut out part of data
- Use generative model to learn data



Anomaly Detection

Semi-Supervised AD:

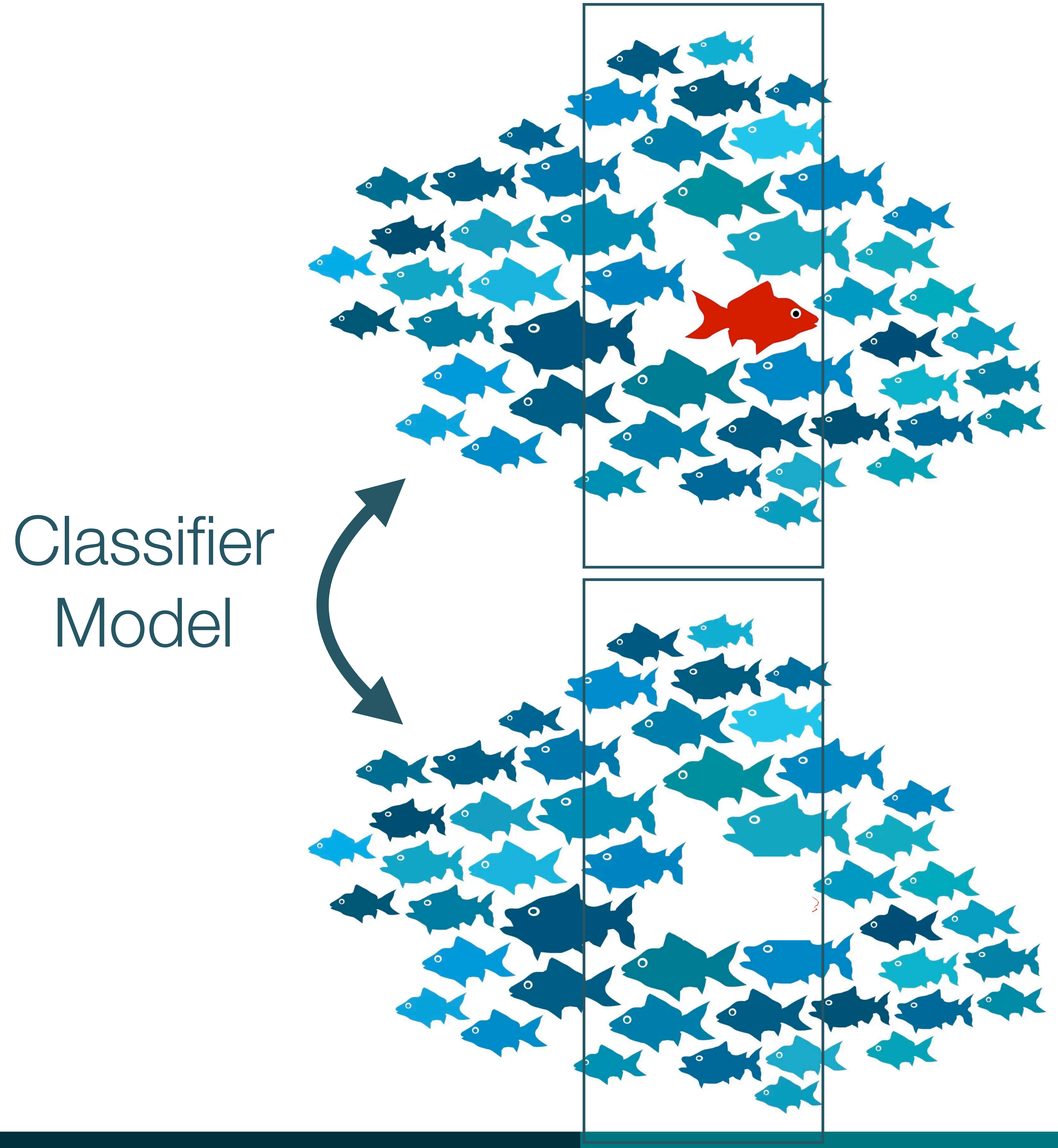
- Find unexpected samples
- Cut out part of data
- Use generative model to learn data
- Predict cut out part



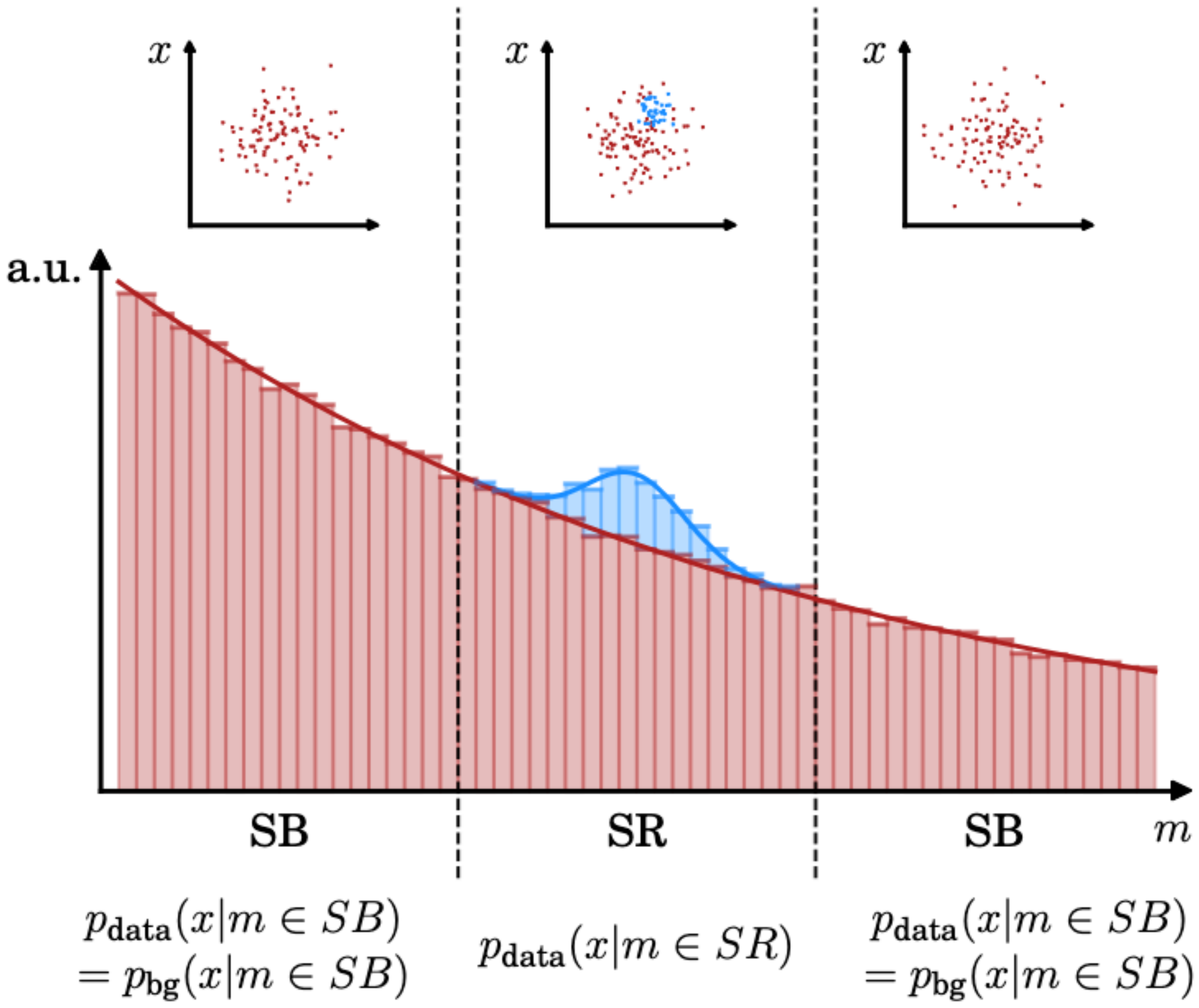
Anomaly Detection

Semi-Supervised AD:

- Find unexpected samples
- Cut out part of data
- Use generative model to learn data
- Predict cut out part
- Compare prediction and cut out



Anomaly Detection

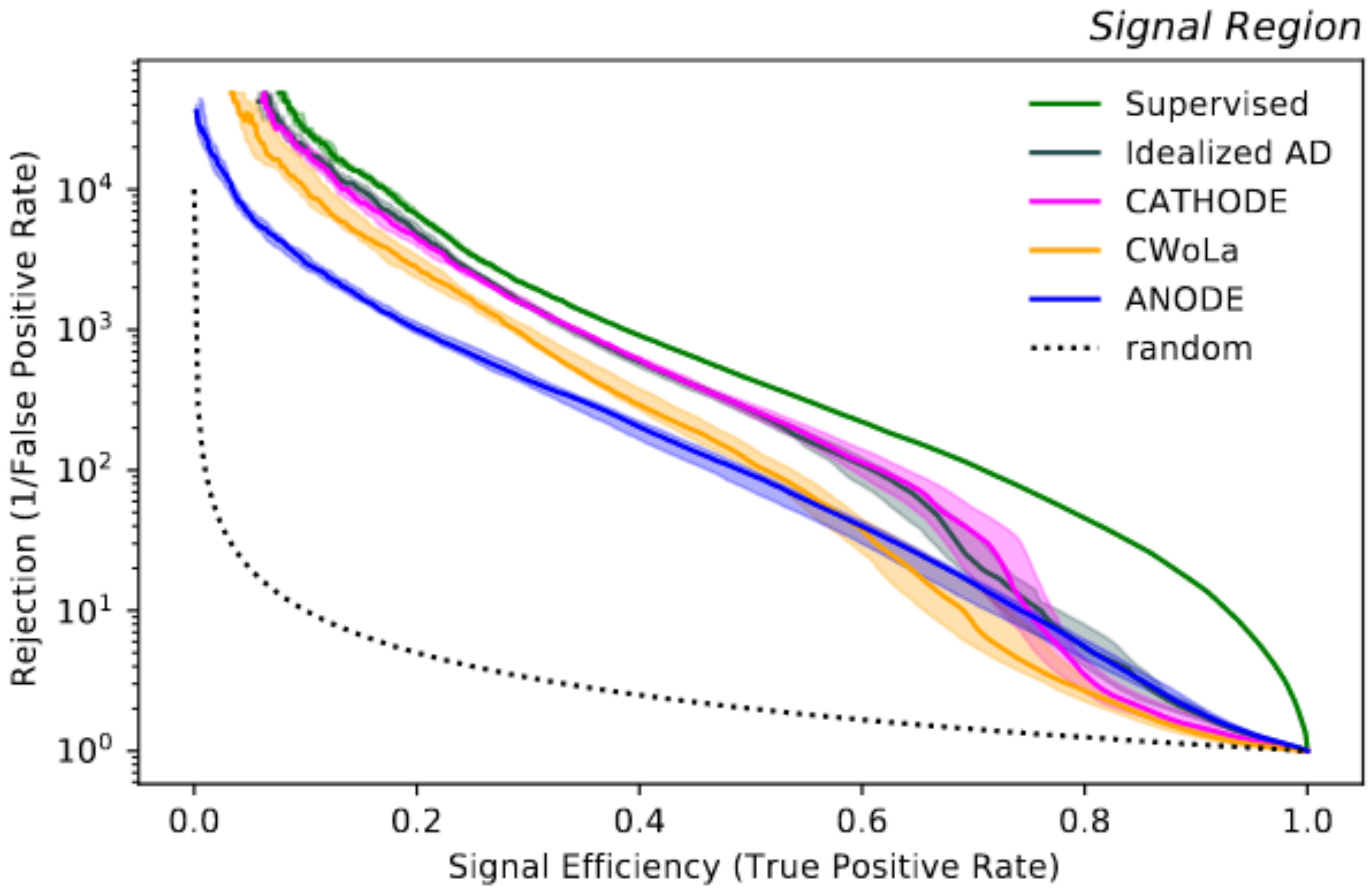


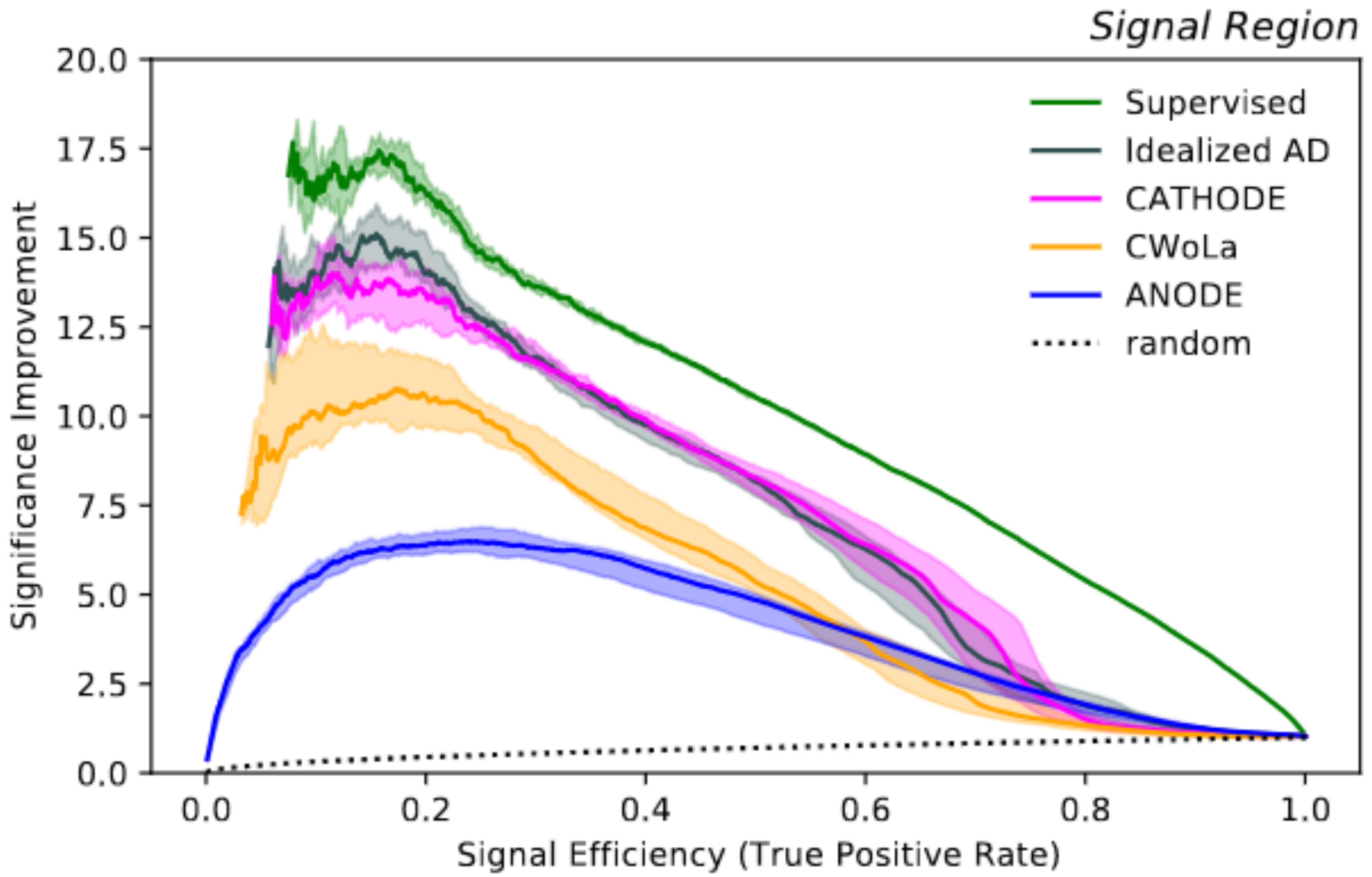
Classifying Anomalies THrough Outer Density Estimation (CATHODE)

- Divide data into Signal Region (SR) and Side Bands (SB)
- SR contains suspected signal (is scanned over in real search
- Train Normalizing Flow on SB
- Extrapolate into SR

Hallin et. al. **Classifying Anomalies THrough Outer Density Estimation (CATHODE),** [2109.00546](https://arxiv.org/abs/2109.00546)

Anomaly Detection

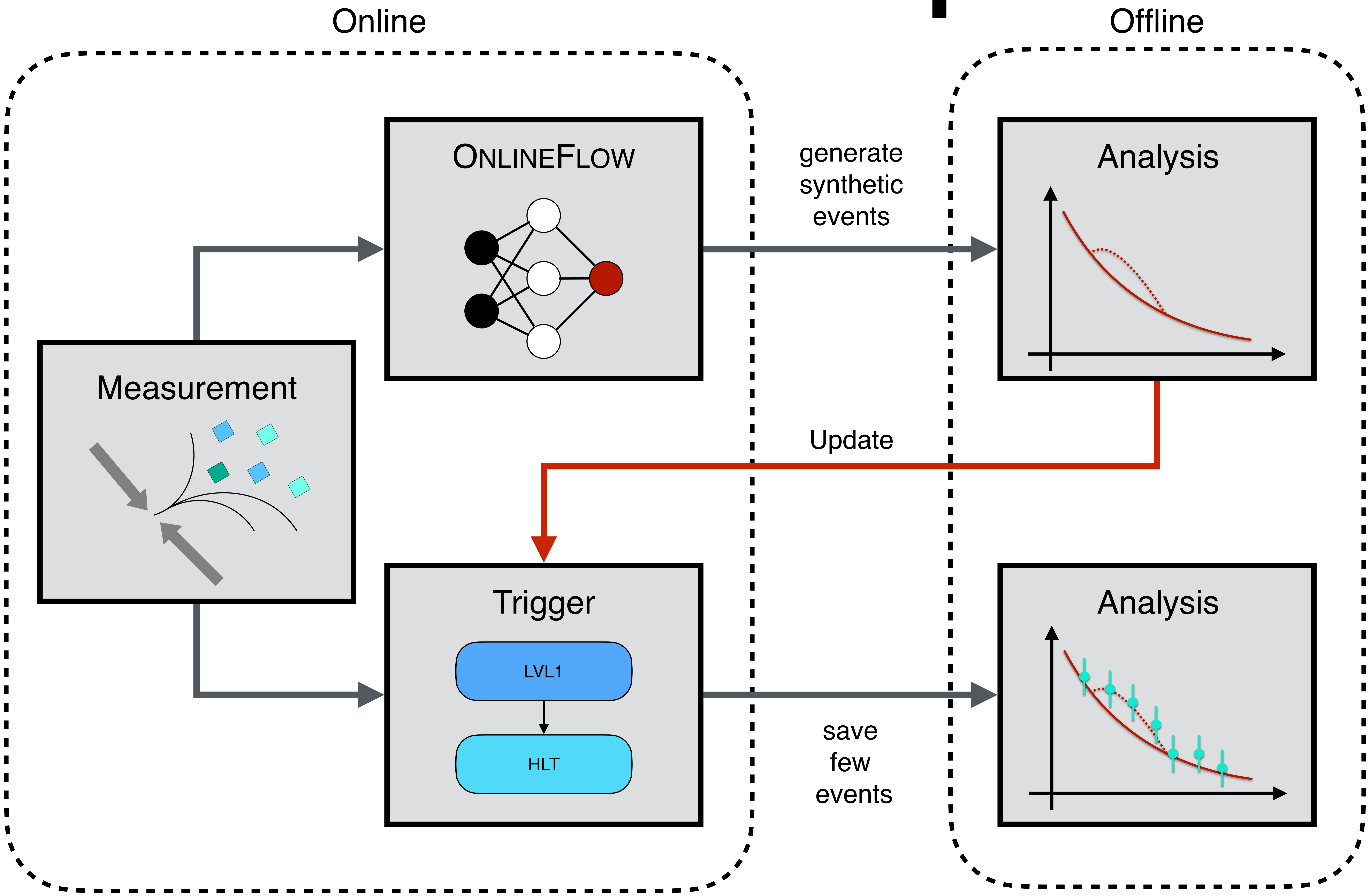




- Train Classifier on real SR vs. extrapolates SR
- Learns to find signal events

Hallin et. al. **Classifying Anomalies THrough Outer Density Estimation (CATHODE),** [2109.00546](2109.00546)

Online Data Compression



Hands-On Tutorial

https://github.com/ml4fp/2024-lbnl

git clone https://github.com/ml4fp/2024-lbnl.git

Or (if already cloned)

```
git add -u
git commit -m ‘past tutorials’
git pull
```

Introduction

Common point of criticism: Information in new samples

- Assume generative model trained on N events
- Used to generate M >> N new events

Info (N real points) = Info (M new points)

Little advantage to be gained from generative model

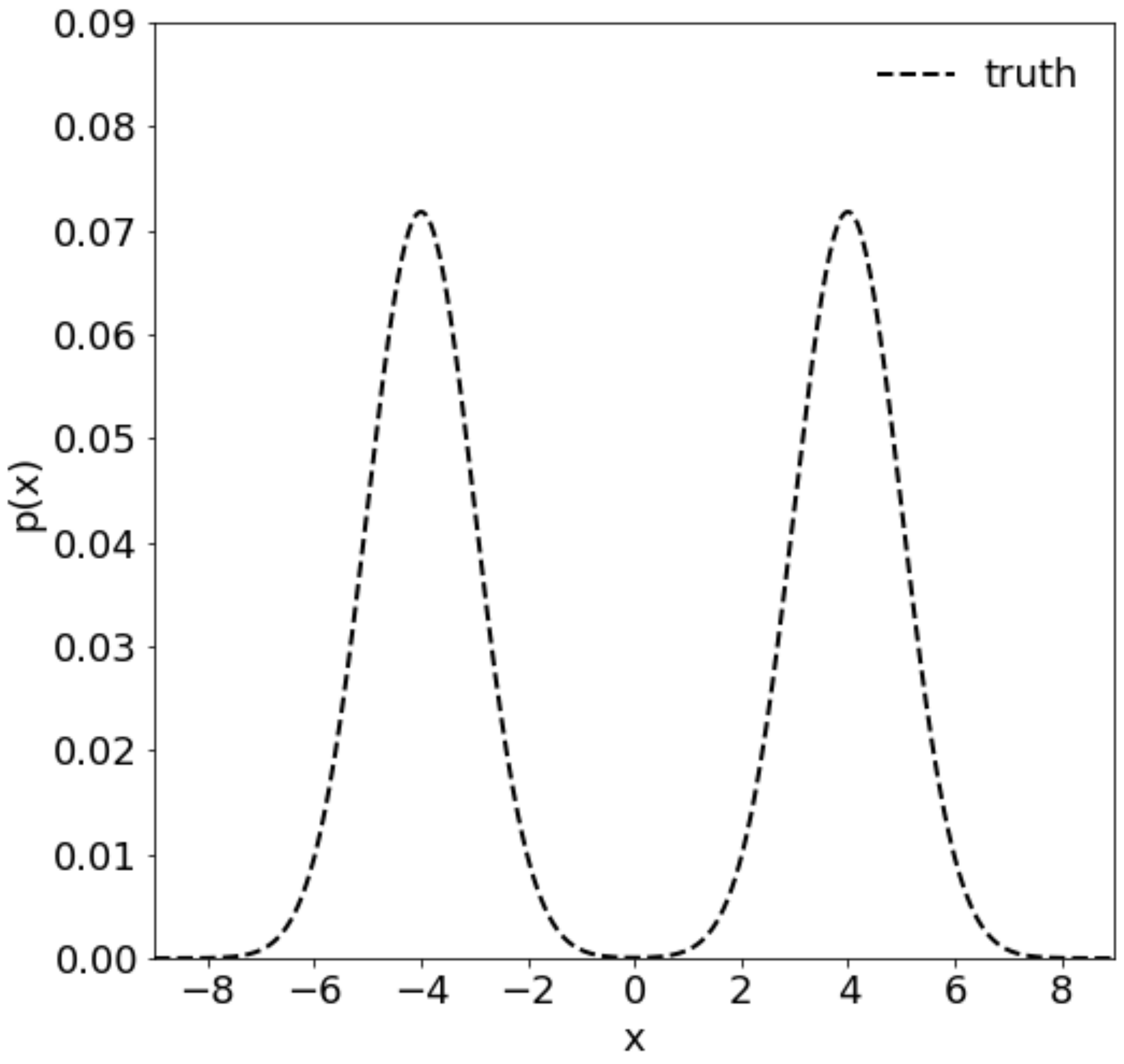
Info (N real points) < Info (M new points)

generative model can speed up simulations

1-D Toy Model

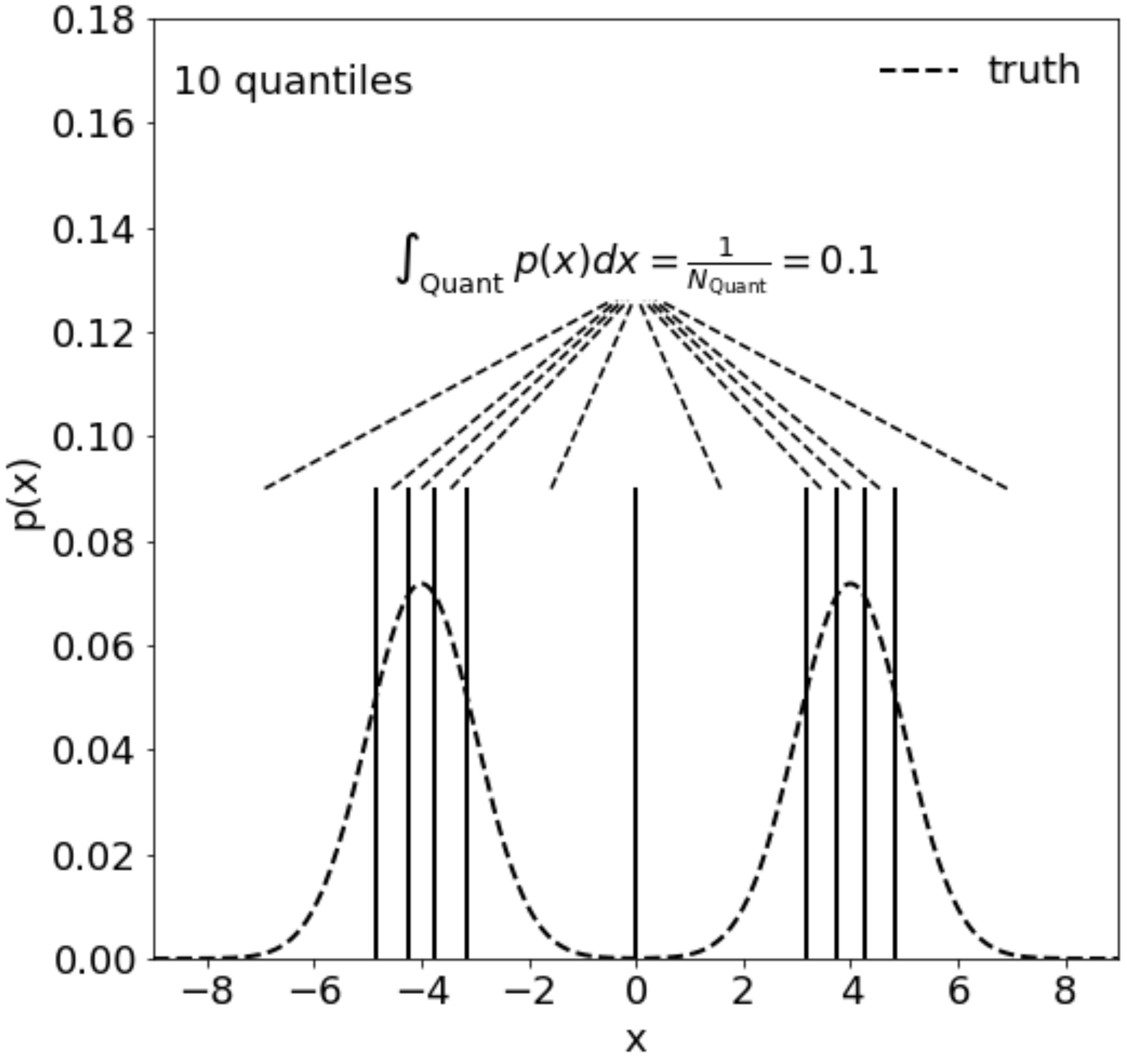
- Camel back function: double peak Gaussian

$$p(X) = \frac{1}{2}(N_{-4,1}(x) + N_{4,1}(x))$$



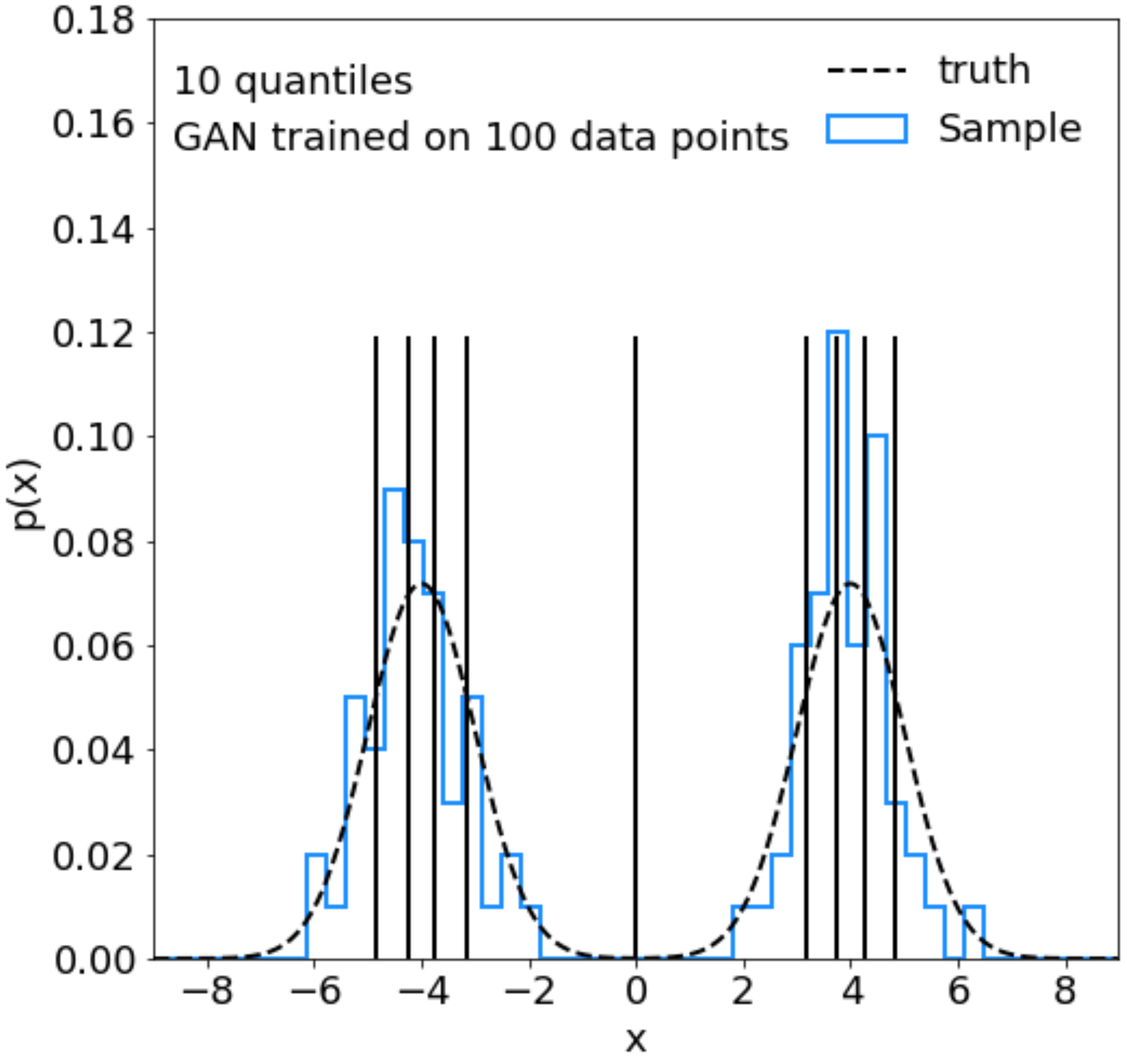
Quantiles

- Measurement how well function is described
- Define N quantiles on true distribution
- Each quantile contains equal probability

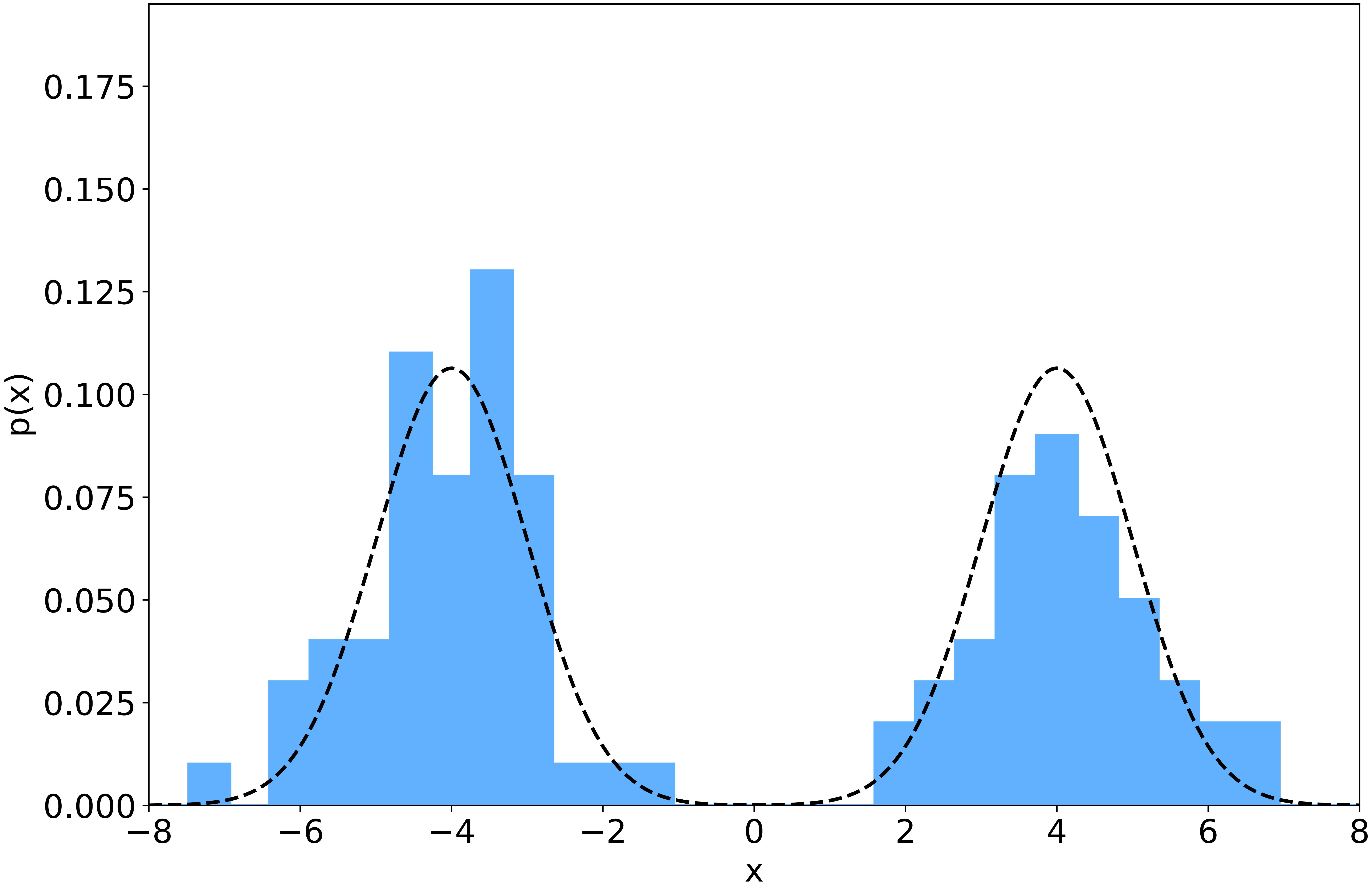


Training Sample

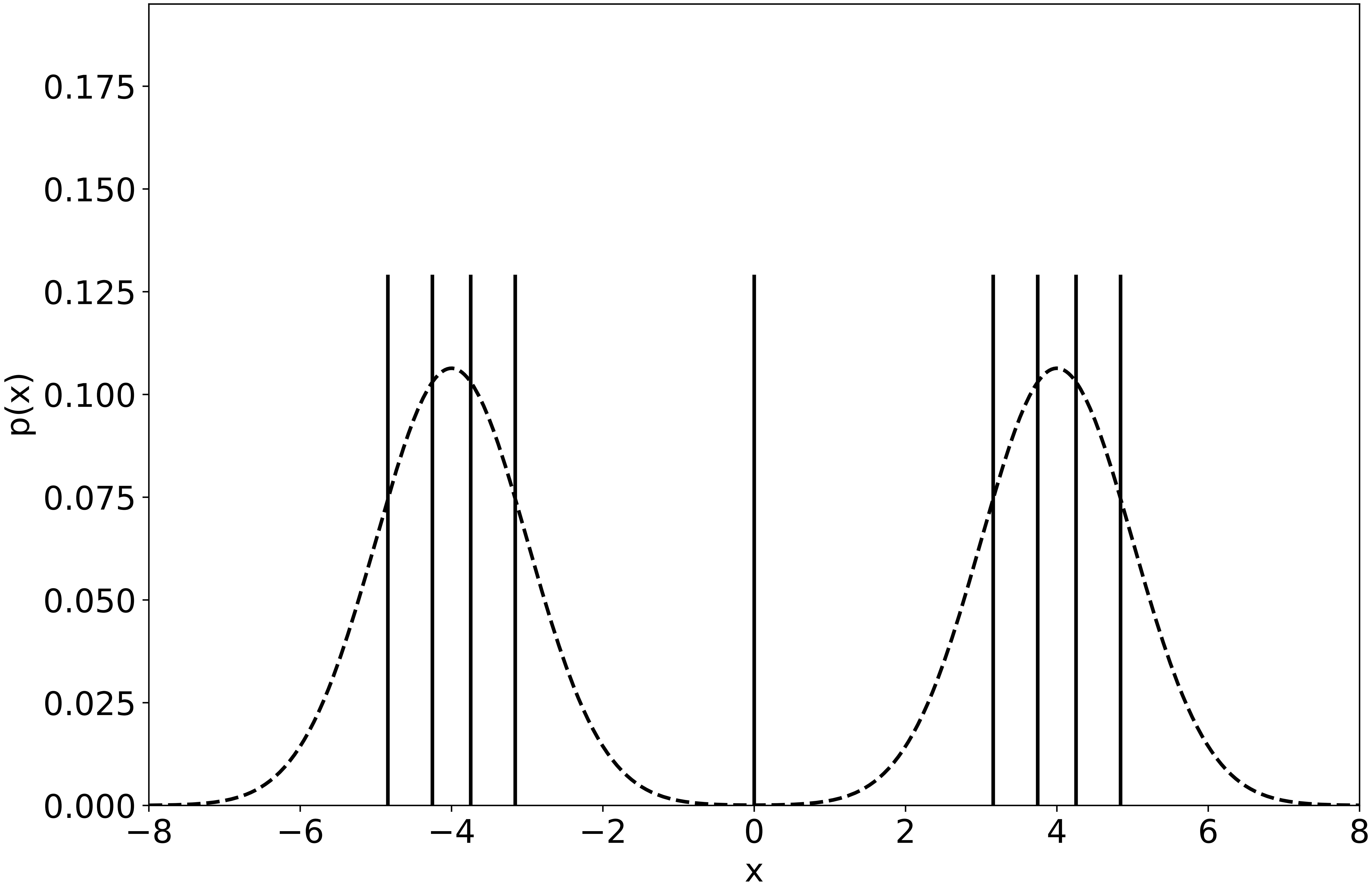
- Draw 100 points from true camel back distribution
- This is designated as the (training) sample
- Calculate fraction of points in each quantile
- Baseline comparison



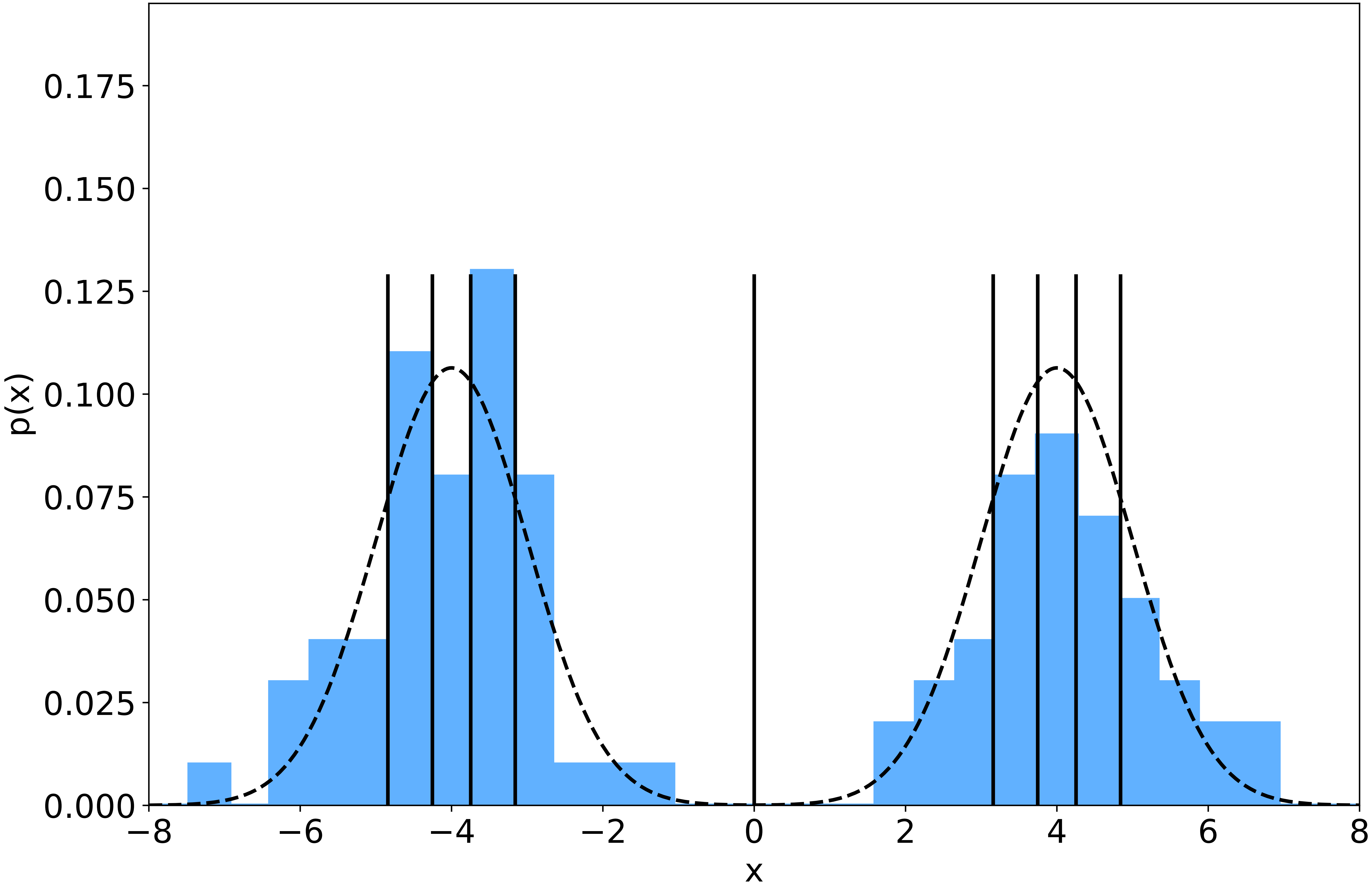
Quantile Measure



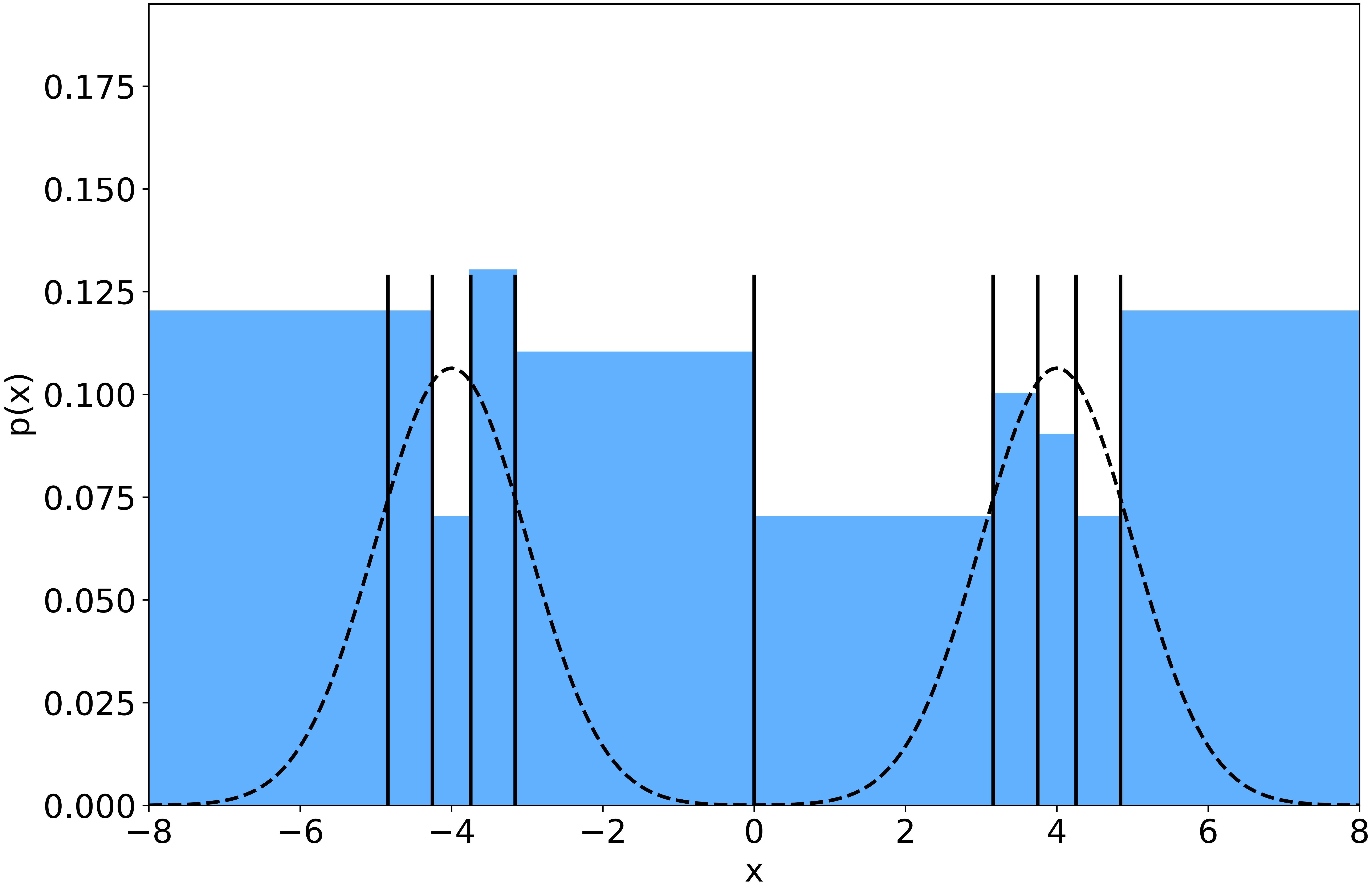
Quantile Measure



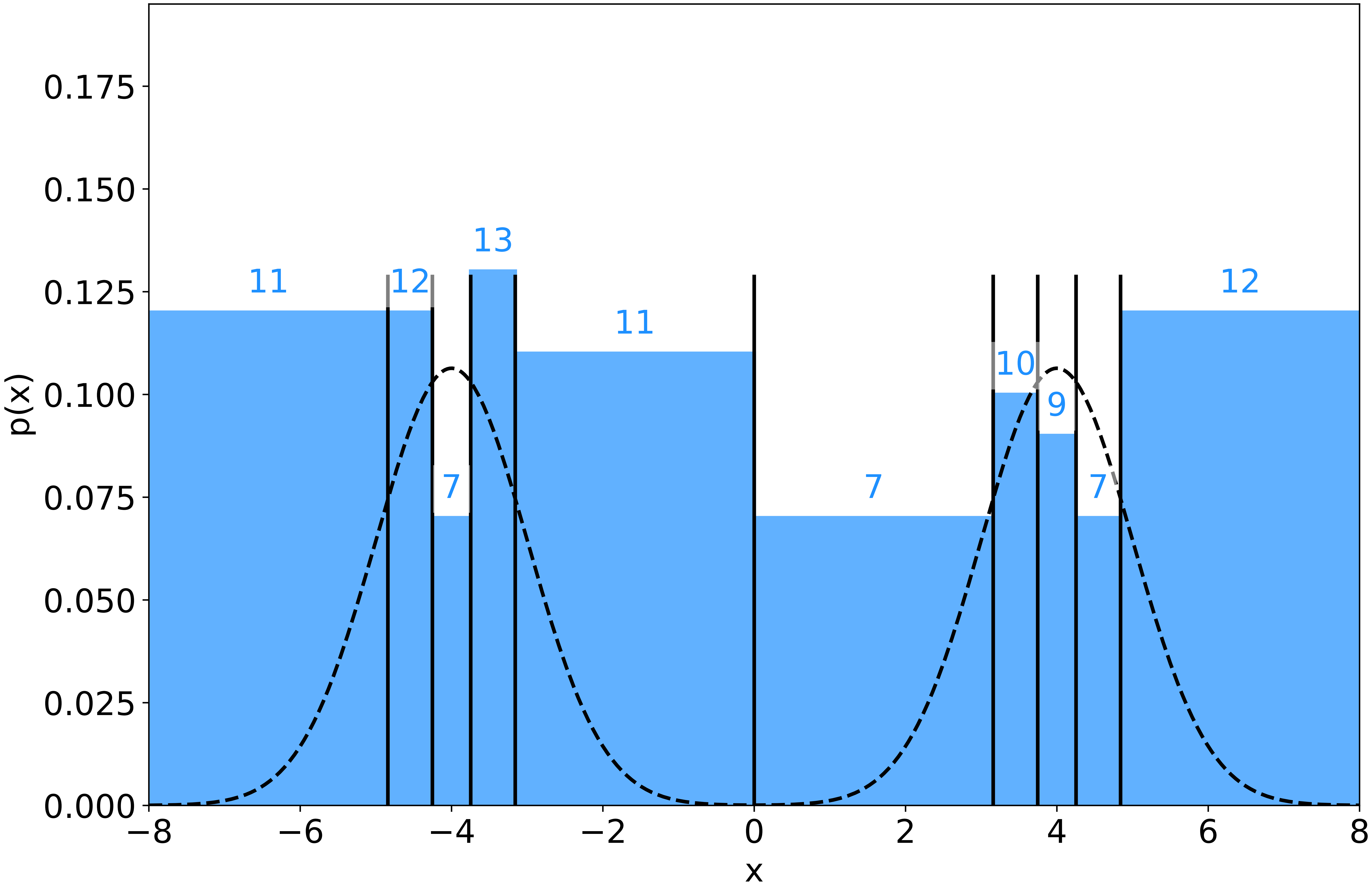
Quantile Measure



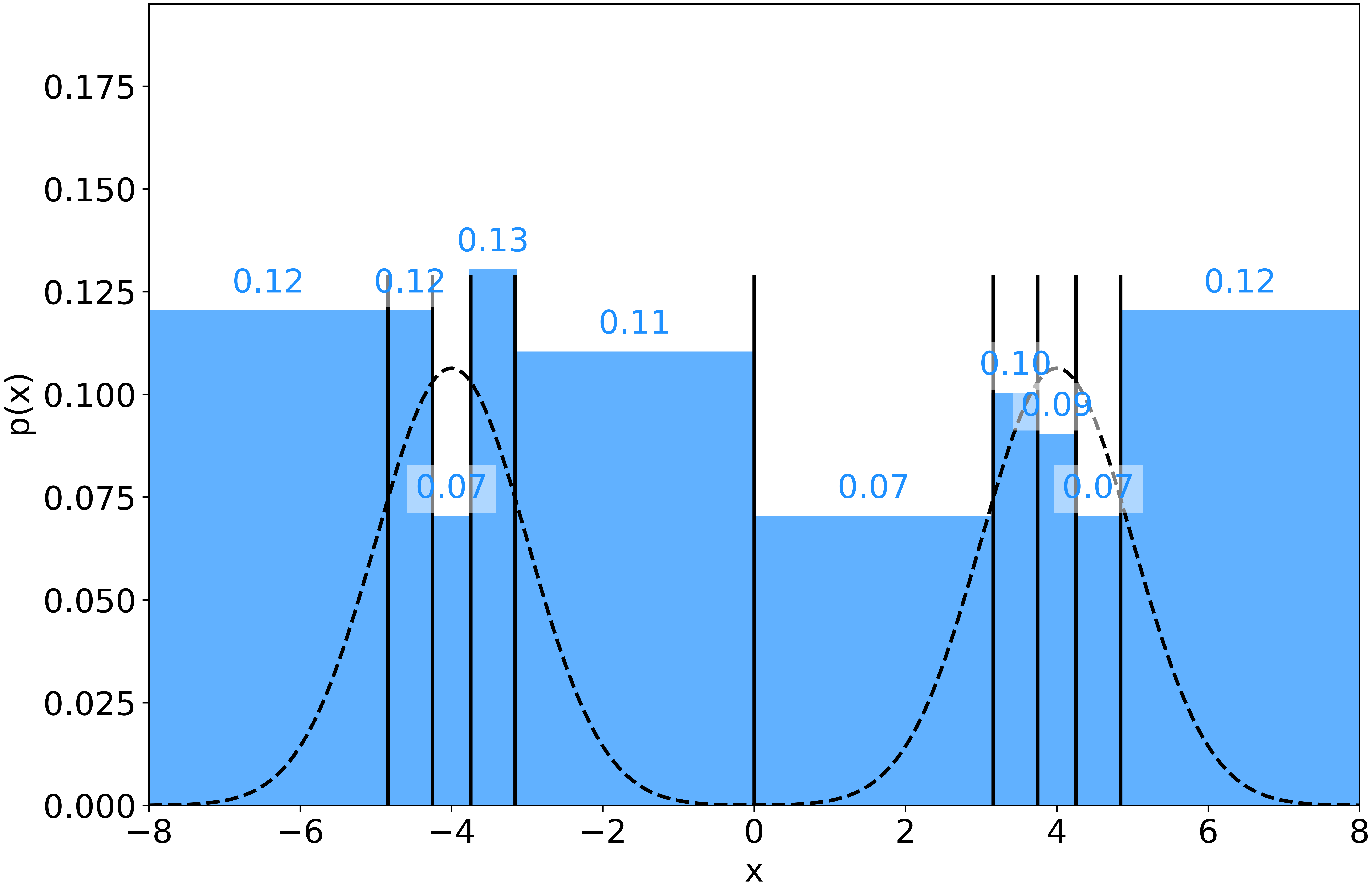
Quantile Measure



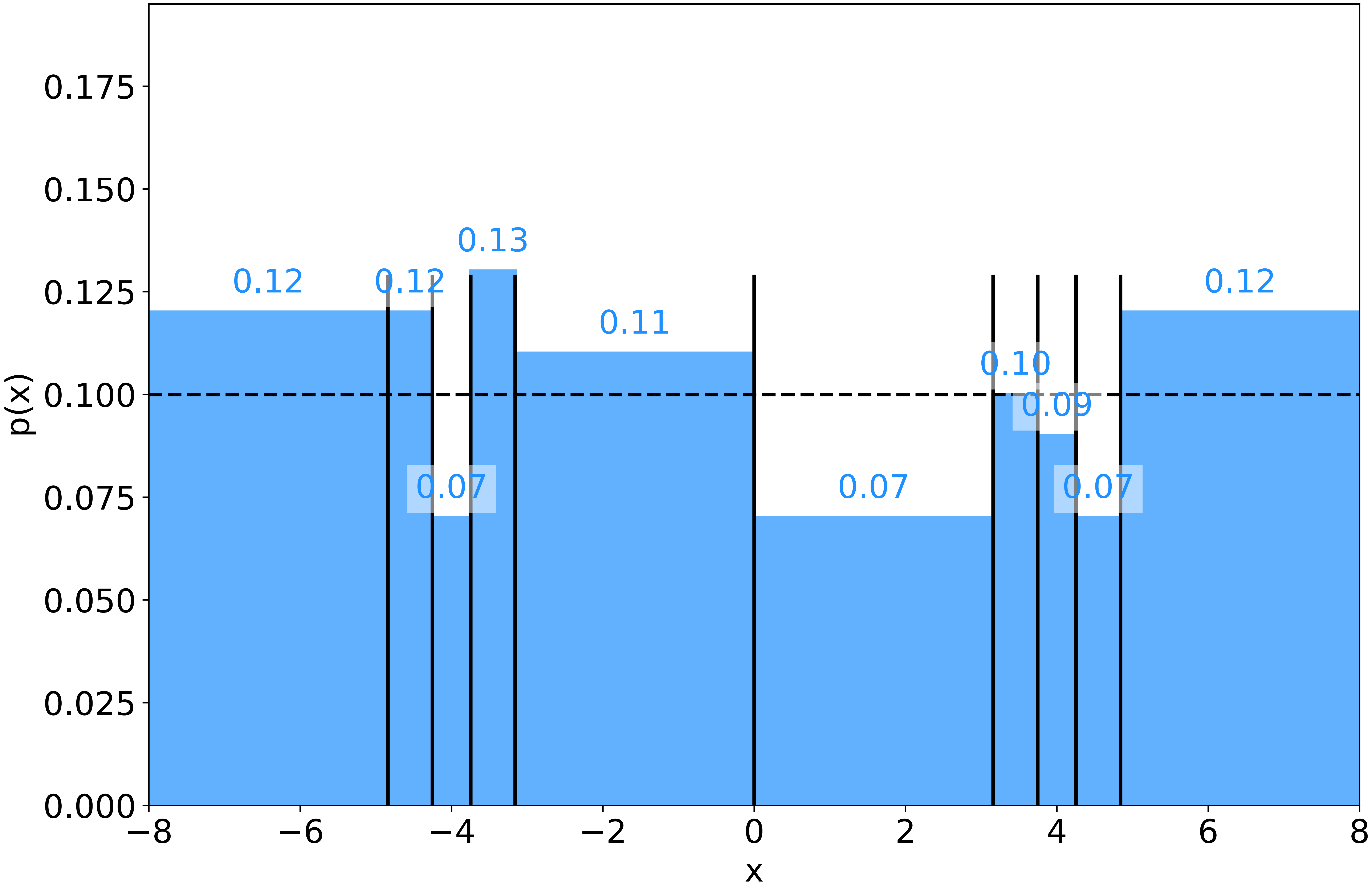
Quantile Measure



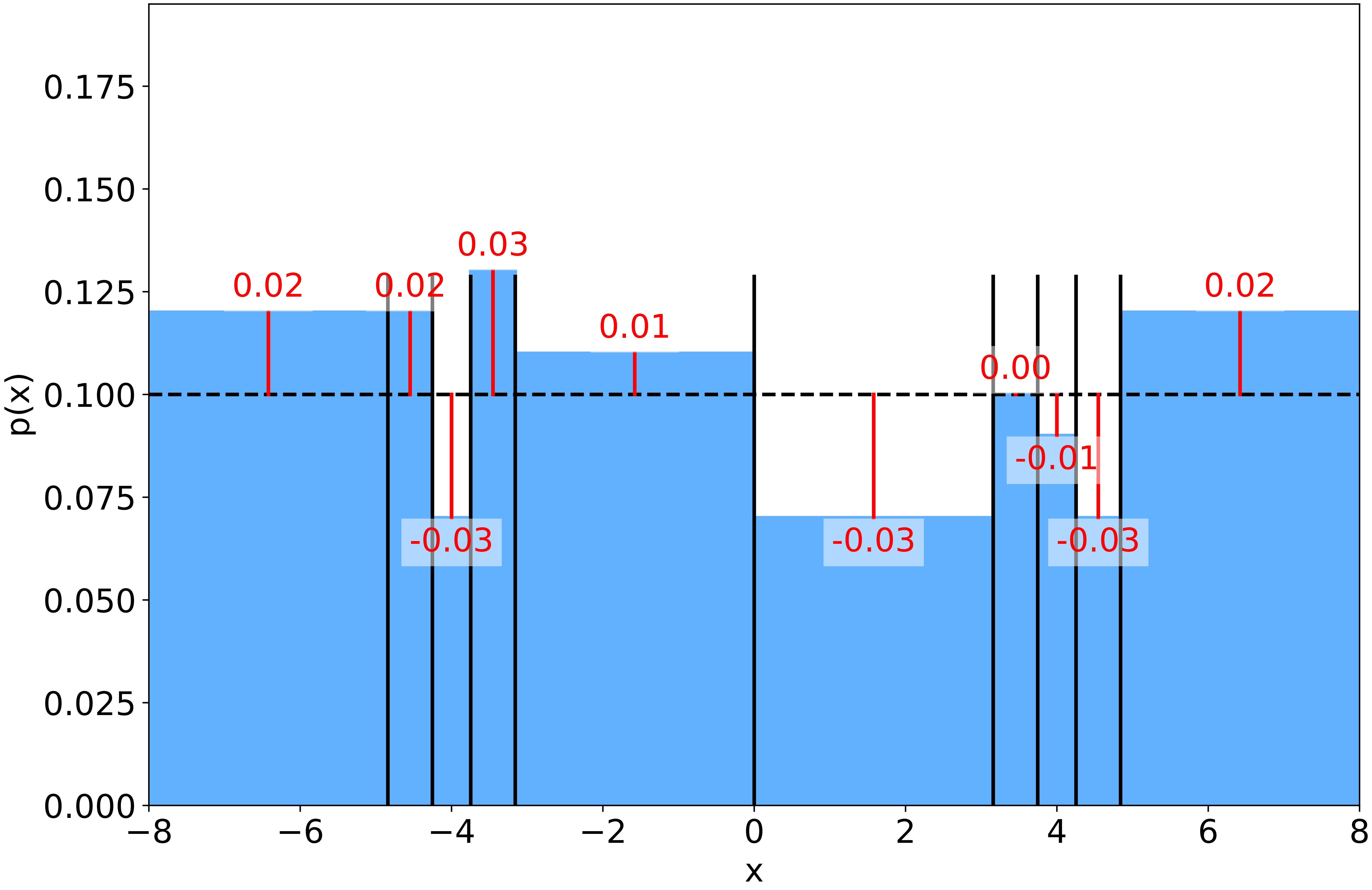
Quantile Measure



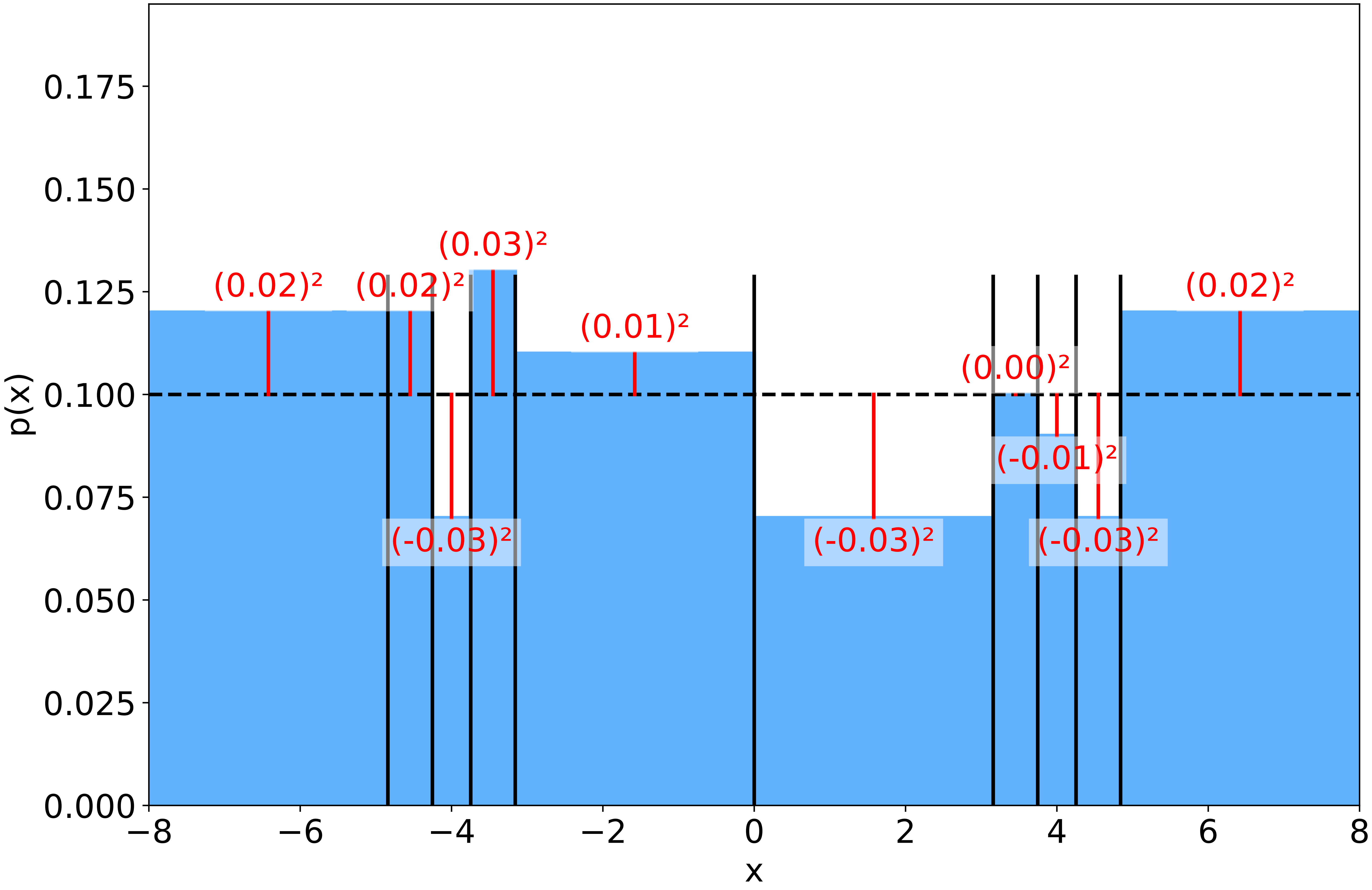
Quantile Measure



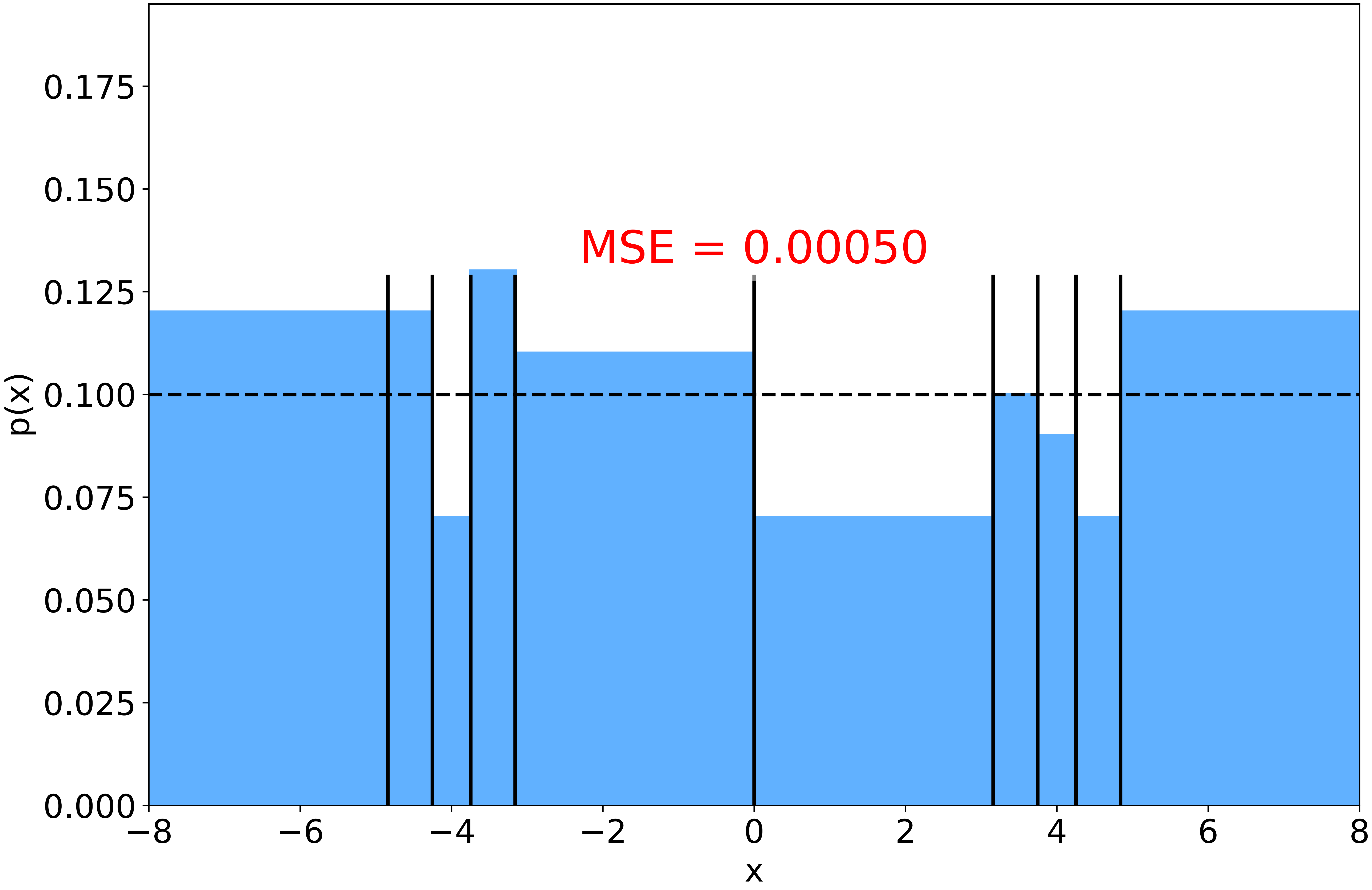
Quantile Measure



Quantile Measure



Quantile Measure

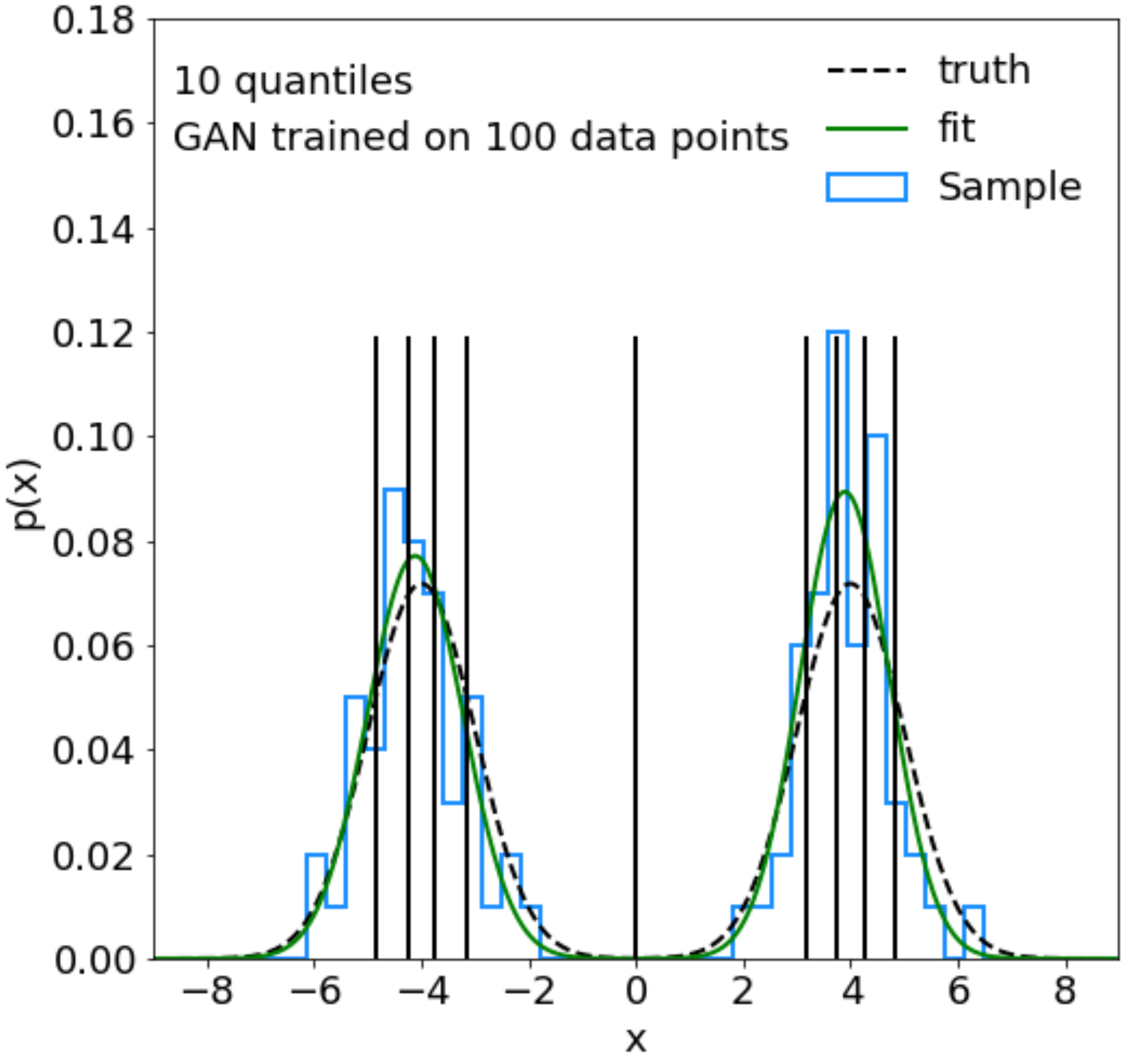


Parameter Fit

- Fit 5 parameter camel back function to training samples

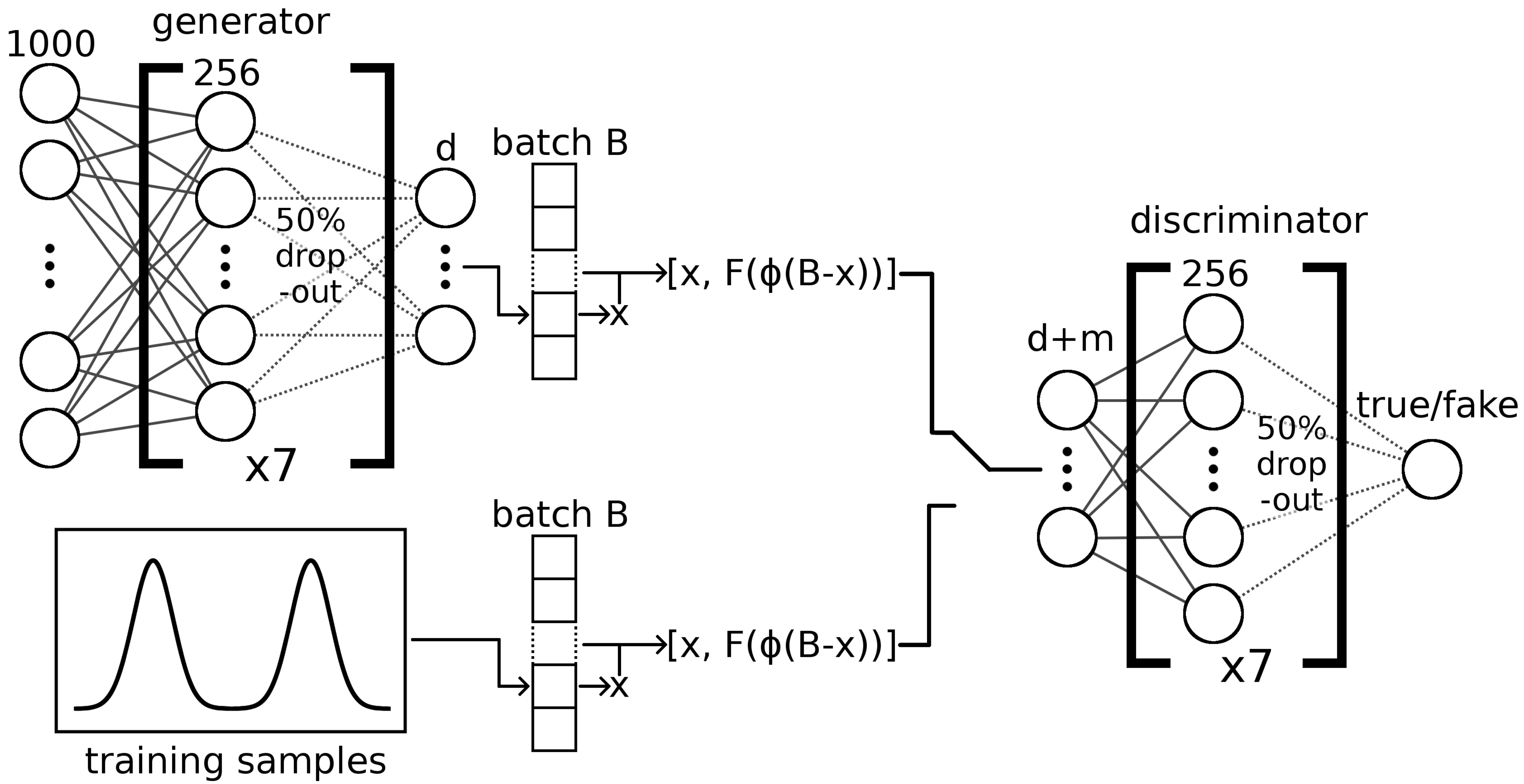
$$p(X) = a\,N_{\mu_1,\sigma_1}(x) \\ +(1-a)N_{\mu_2,\sigma_2}(x)$$

- Analytically calculate integral for each quantile
- Gives upper performance benchmark



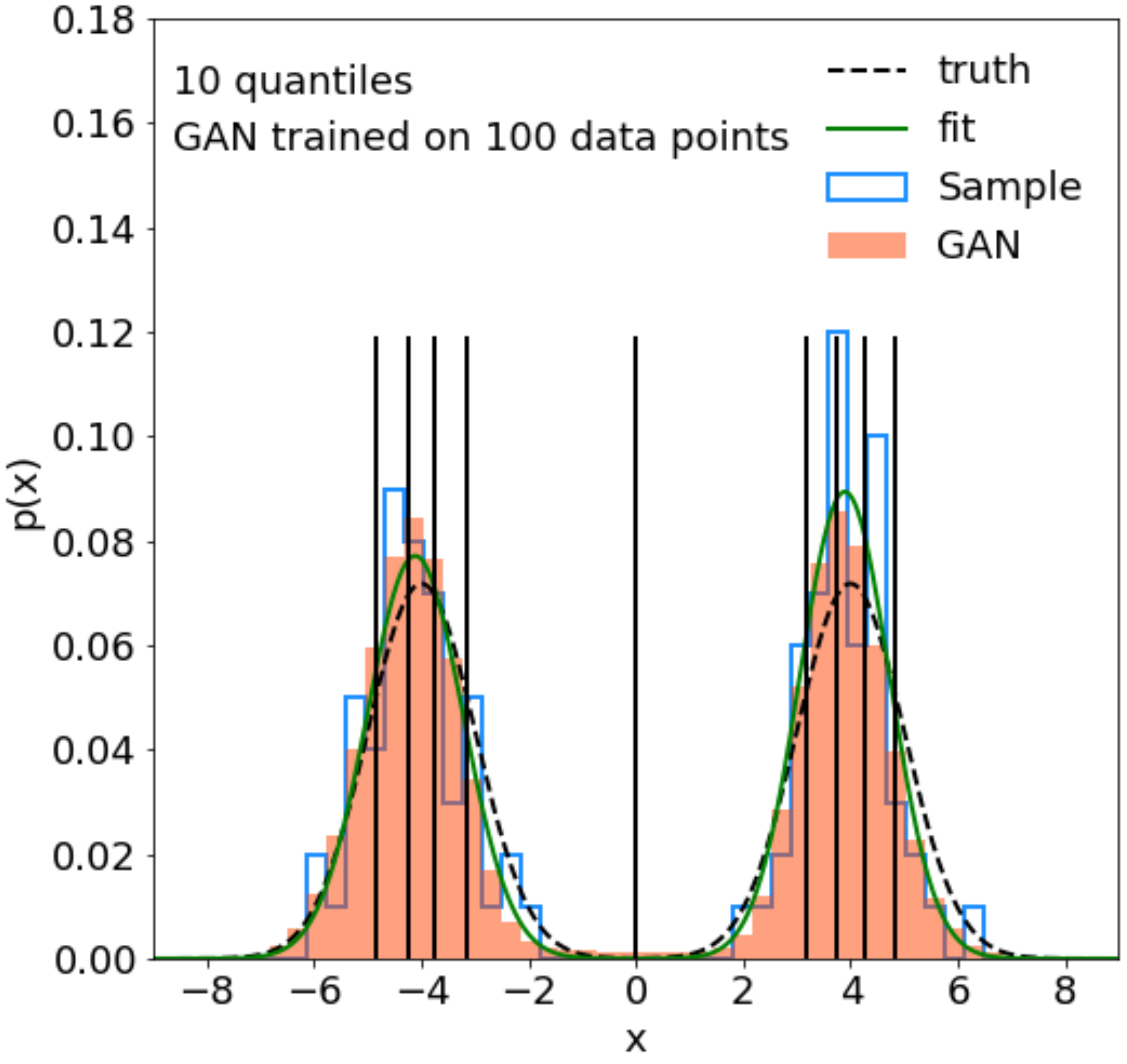
Generative Adversarial Network

- Train GAN on 100 data points from training sample
- Mode-collapse and overfitting problematic
  - Dropout
  - Added training noise
  - Batch-statistics



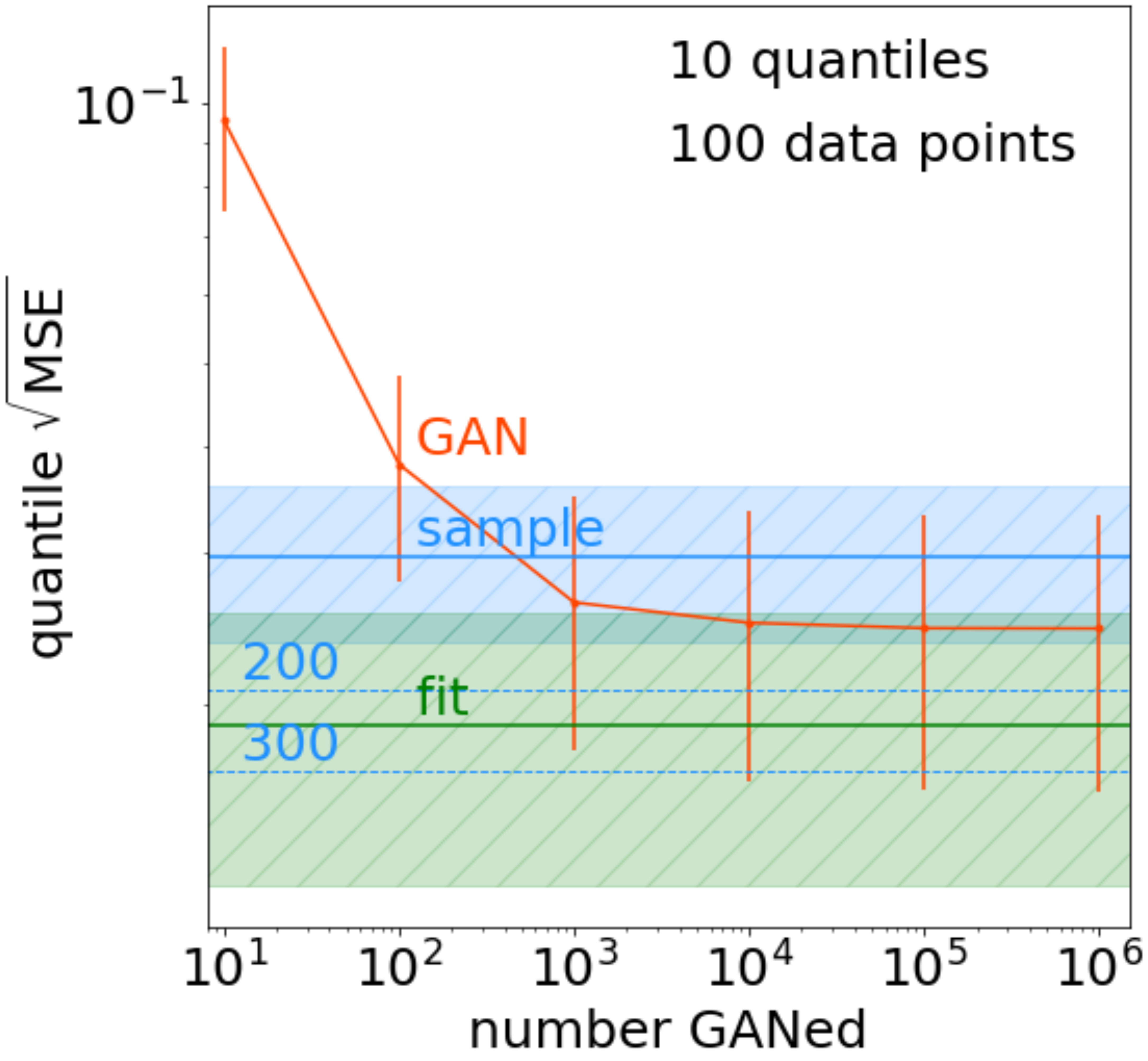
Generative Network

- Generate $O(10^7)$ data points using GAN
- Calculate fraction of points in each quantile
- Compare to train and fit



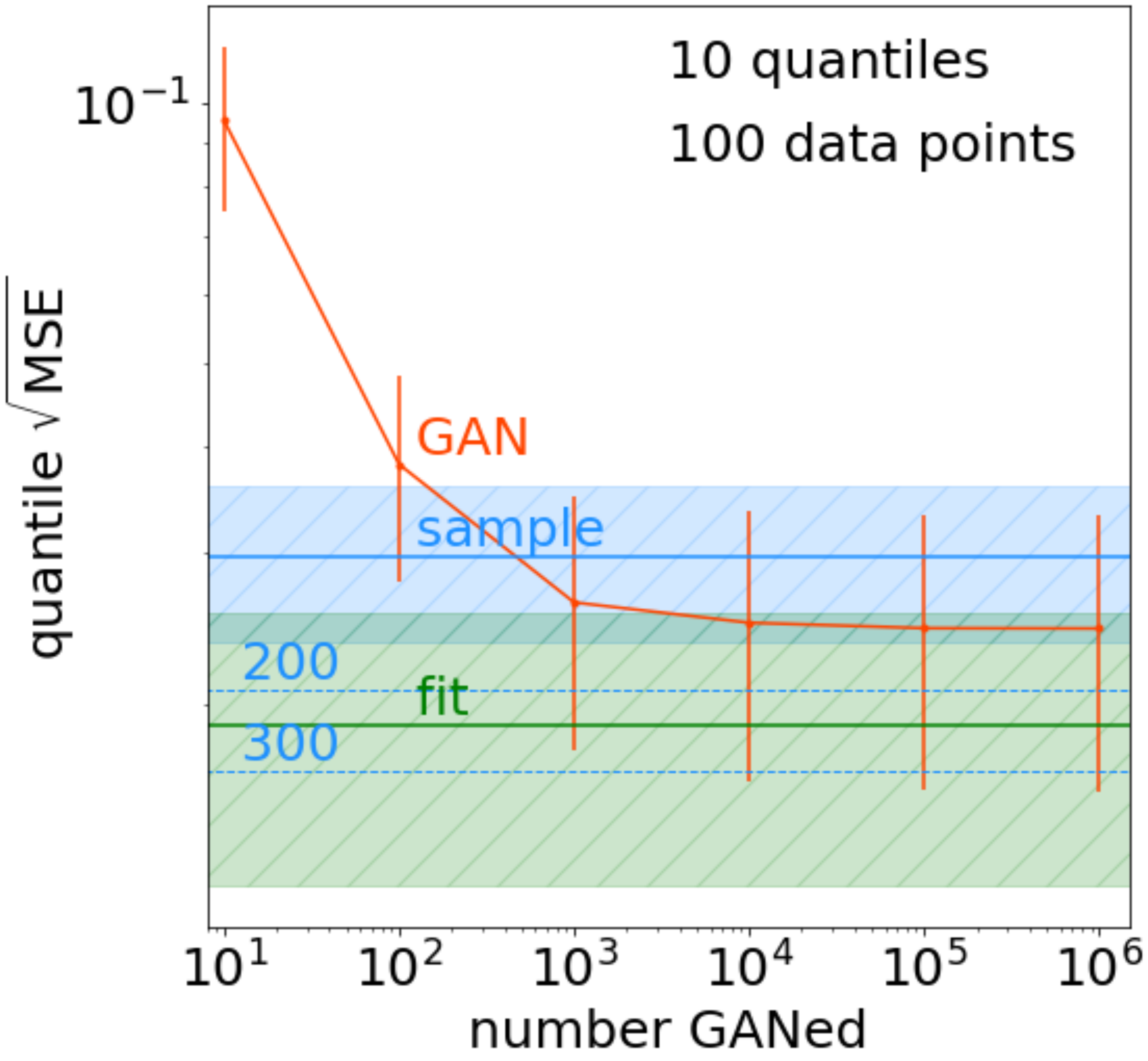
Generative Network

- For 100 training samples, 100 fits and 100 GANs compare MSE
- GAN describes distribution better than training data
- Needs 10,000 GANed points to match 150 true points
- Shifts statistical uncertainty to systematic uncertainty



Generative Network

- How is this possible?
- In terms of information:
  - sample: only data points
  - fit: data + true function
  - GAN: data + smooth, continuous function
- This allows the GAN to interpolate



Intermediate Conclusion

Common point of criticism: Information in new samples

- Assume generative model trained on N events
- Used to generate M >> N new events

Info (N real points) = Info (M new points)

Little advantage to be gained from generative model

Info (N real points) < Info (M new points)

generative model can speed up simulations