Some Open Problems in Generative Methods

Barnabas Poczos

Carnegie Mellon University

Machine Learning Department

Collaborators

Aarti Singh

Zoltan Szabo

Shashank Singh

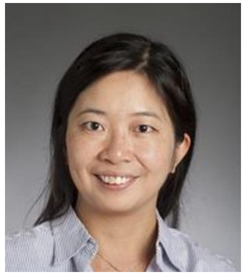
Arthur Gretton

Ananya Uppal

Alexander Smola

Manfred Paulini

Junier Oliva



Shirley Ho

Francois Lanosse

Michelle Ntampaka

Chun-Liang Li

Danica Sutherland

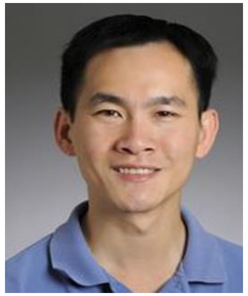
Michael Andrews

Rachel Mandelbaum

Manzil Zaheer

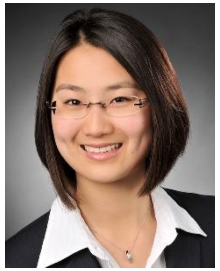
Siamak Ravanbhaksh

Jeff Schneider



Hy Trac

Ruslan Salakhutdinov



Yang Zhang

Generative Methods

Generative methods are getting more and more popular:

- VAE
- GAN
- Normalizing flows
- Transformers
- Diffusion
- and many others …

Generative Methods

We will discuss some important questions:

- Different variants/improvements of existing methods?
- How to generate other objects than images?
- How to create shallow (non deep learning based) generative methods?
    - Density functional estimation
    - Distribution regression/classification, distribution embedding
- How good are these generative methods? Convergence rates?
- Open problems?

Generative Methods

Goal:

Given a training dataset, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \sim p_{\text{data}}$,

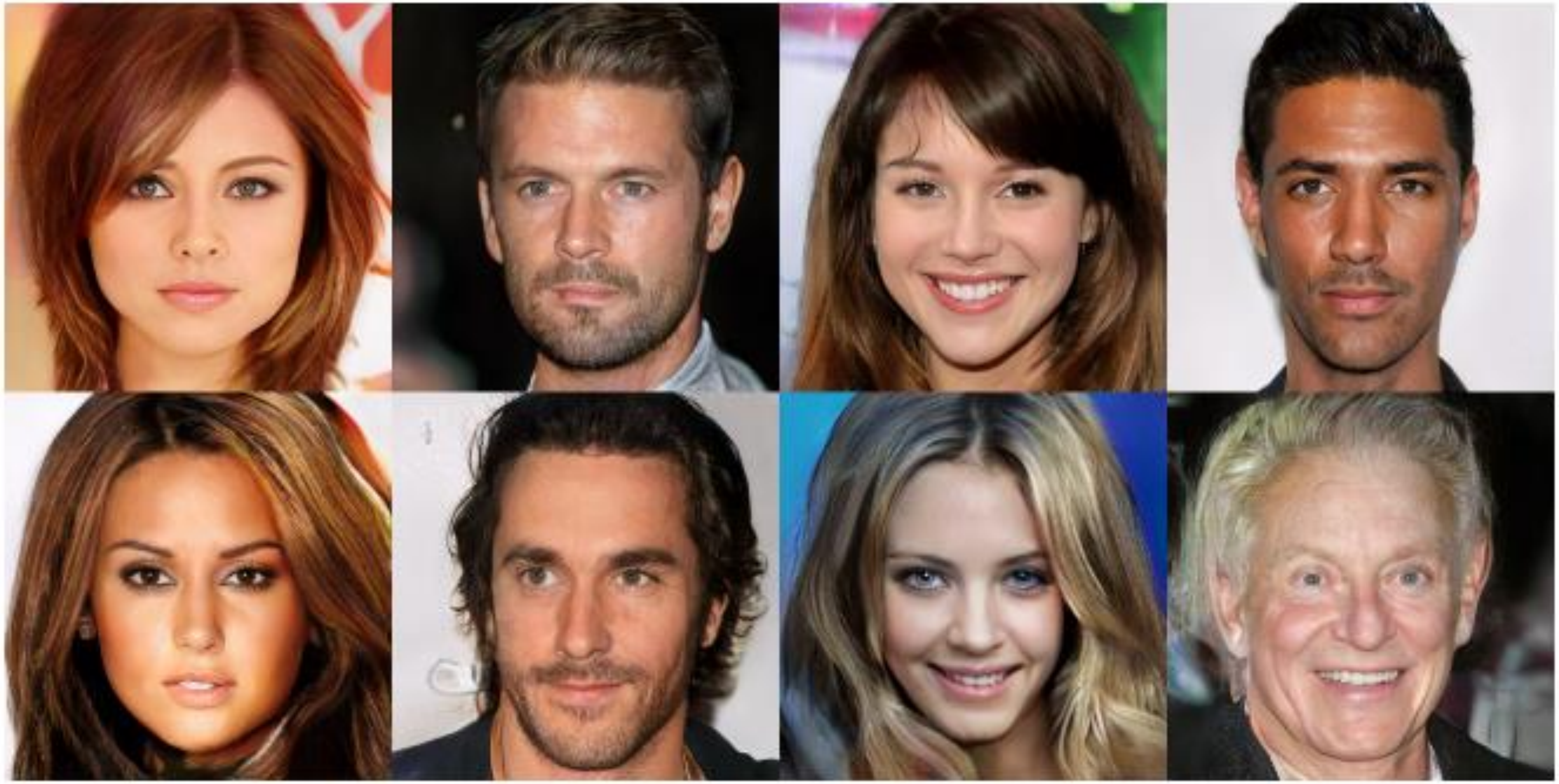
generate more data $\boldsymbol{x}_{n+1}, \ldots, \boldsymbol{x}_{n+m}$ from the same distribution p_{data} .

[without estimating the distribution/density of the data]

We will start the discussions with GANs

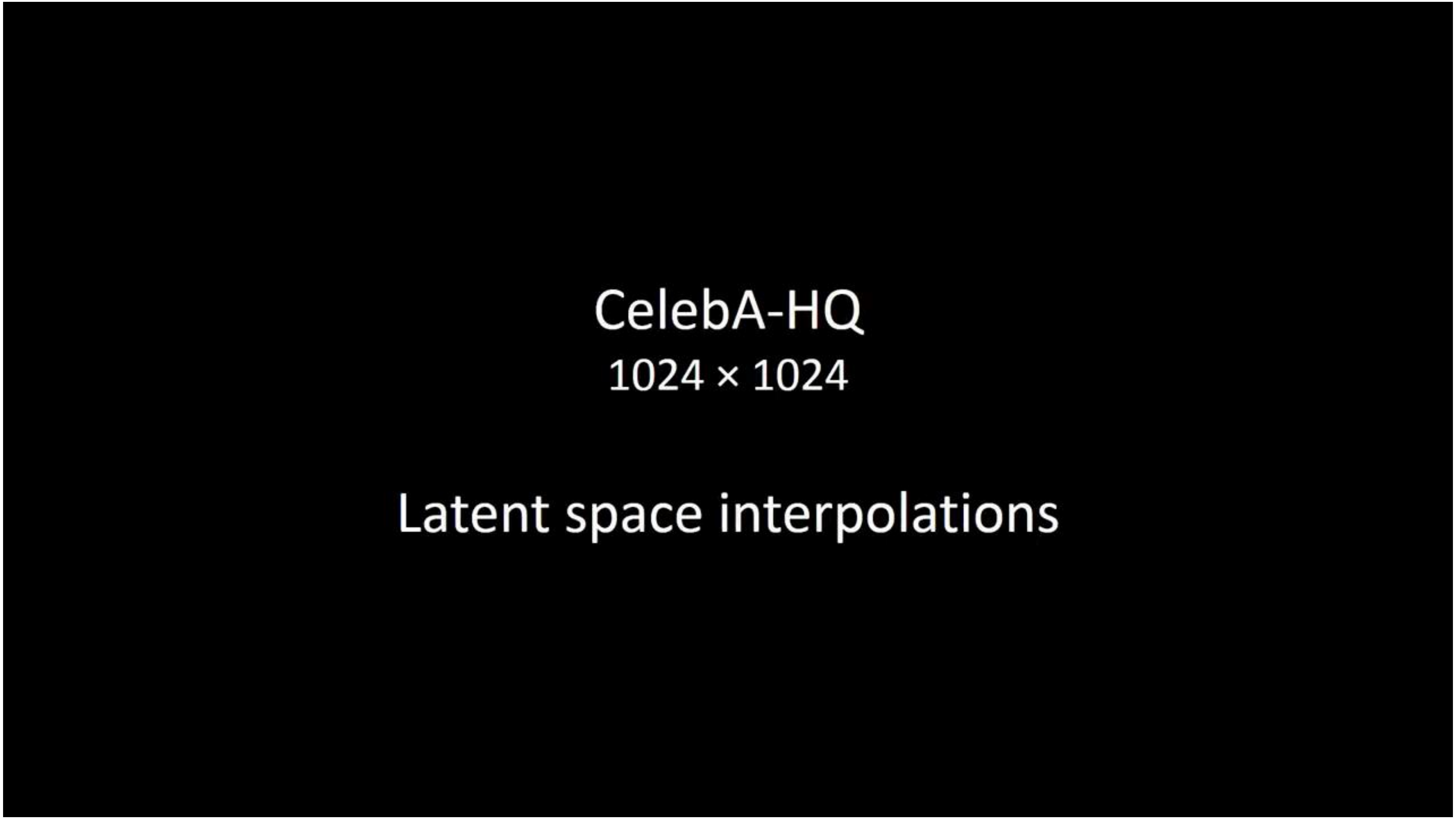
Generative Adversarial Networks
A Brief Summary

Generating Adversarial Networks



Generated fake celebrity images

Tero Karras, Timo Aila, Samuli Laine, ICLR 2018

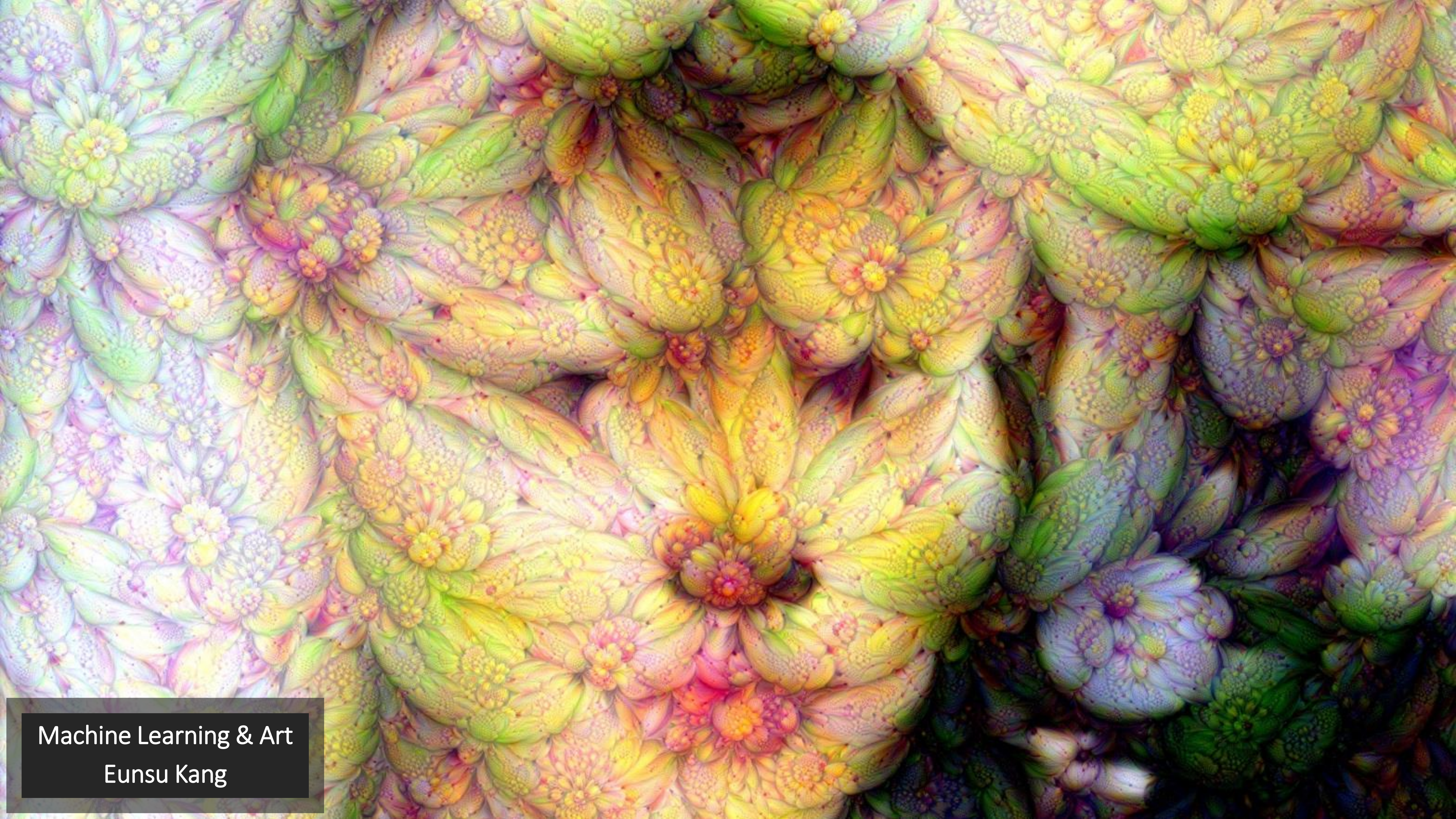


CelebA-HQ

1024 × 1024

Latent space interpolations

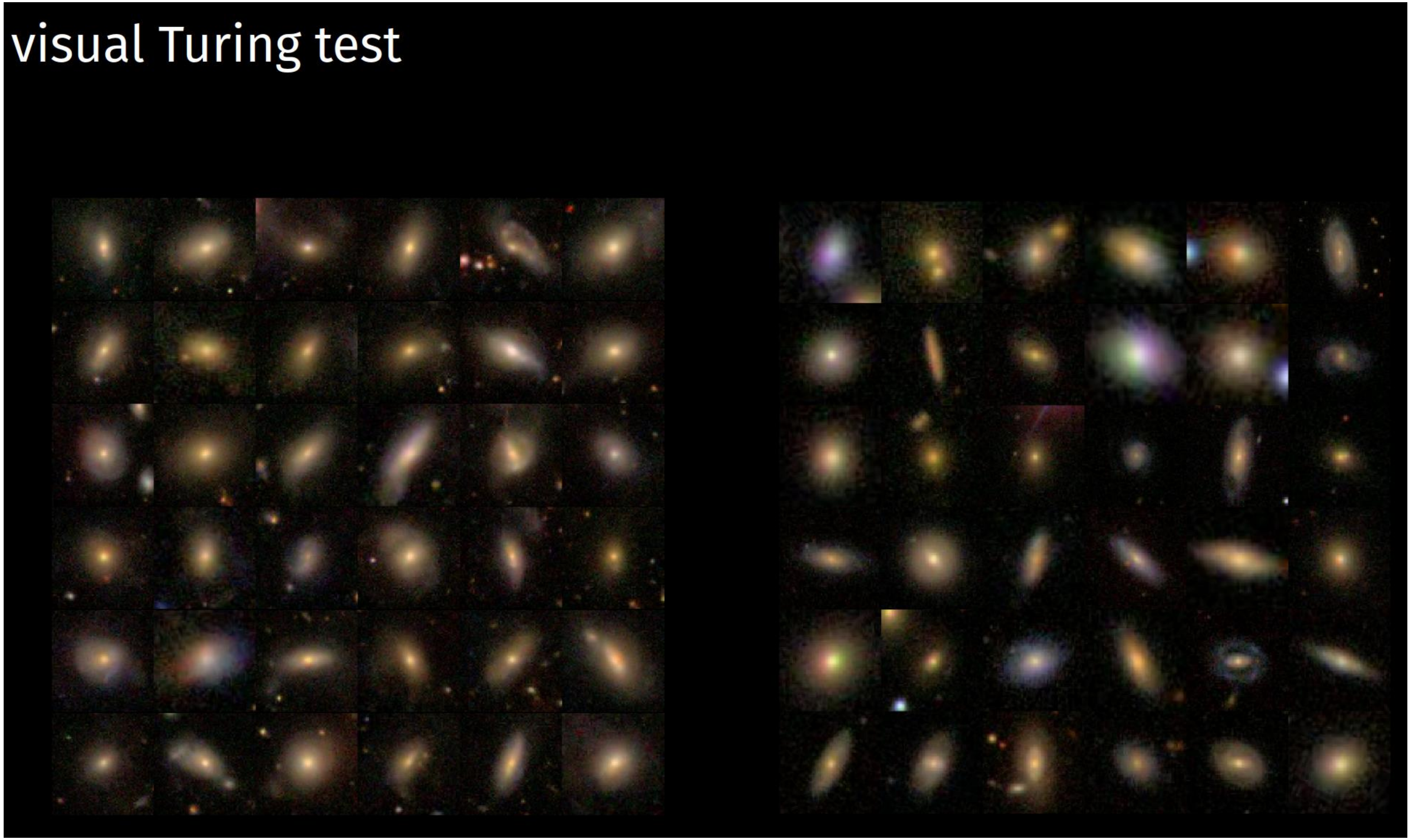
Tero Karras, Timo Aila, Samuli Laine, ICLR 2018



Machine Learning & Art

Eunsu Kang

Ravanbakhsh, M., Lanusse, F., Mandelbaum, R., Schneider, J., and Poczos, B., AAAI 2017



visual Turing test

Mock **Real**

Generative Adversarial Networks

Goodfellow et al, Generative Adversarial Nets, 2014

Generator:

- We define a prior on input noise variables $p_{\boldsymbol{z}}(\boldsymbol{z})$. (e.g. $\boldsymbol{z} \sim \mathcal{N}(0, I)$)
- Then create a mapping to data space as $G(\boldsymbol{z}; \theta_g)$.

  Here G is a neural net with parameters θ_g.

  [In case of diffusion based merthods, G is a diffusion process starting from $\boldsymbol{z}$]

Discriminator:

- $D(\boldsymbol{x}; \theta_d)$ is a second neural net that outputs a single scalar in $[0, 1]$.
- $D(\boldsymbol{x}; \theta_d)$ represents the estimated probability that $\boldsymbol{x}$ came from the data rather than the generator G.

Generative Adversarial Networks

- We train D to maximize the probability of assigning the correct label to both training examples and samples from G:

  D wants $D(\boldsymbol{x})$ to be large when $\boldsymbol{x} \sim p_{\text{data}}$

  D wants $D(G(\boldsymbol{z}))$ to be small [since these are the generated sample points.]

Objective function of the discriminator:

$$\max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}\,(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

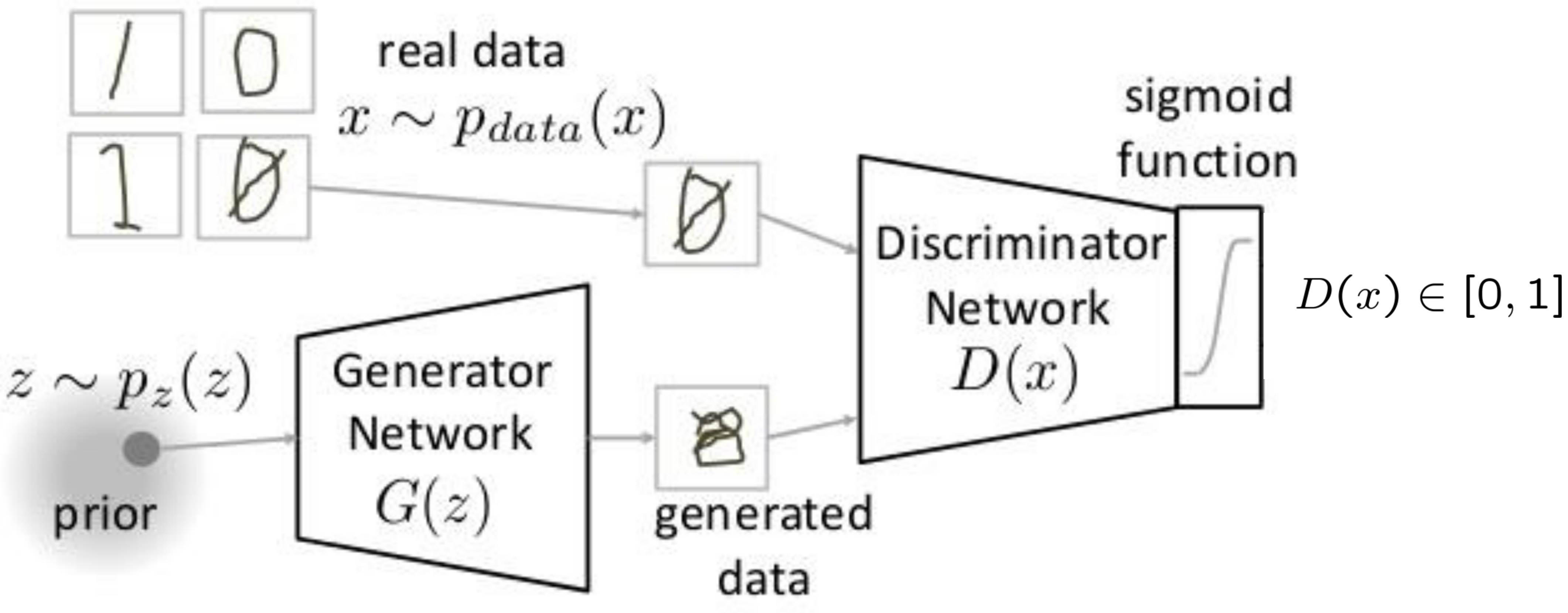
- We simultaneously train G to trick the discriminator D:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}\,(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Generating Adversarial Networks

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

$$V(D, G) := \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$



Generating Adversarial Networks

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Lemma: For G fixed, the optimal discriminator D is $D_G^*(\boldsymbol{x}) = \dfrac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$

Lemma: $C(G) := V\left(D_G^*, G\right)$

$$= -\log(4) + KL\left(p_{\text{data}} \| \frac{p_{\text{data}} + p_g}{2}\right) + KL\left(p_g \| \frac{p_{\text{data}} + p_g}{2}\right)$$

$$= -\log(4) + JS\left(p_{\text{data}} \| \frac{p_{\text{data}} + p_g}{2}\right)$$

The original GAN is trying to minimize the Jensen-Shannon divergence between the distributions of the generated data p_g and the training data p_{data}.

Lemma: This minimax game has a global optimum for $p_g = p_{\text{data}}$.

Other Versions?

Adversarial Losses

The GAN loss function is equivalent to $\min_G JS\left(p_{\text{data}} \,\|\, \frac{p_{\text{data}} + p_g}{2}\right)$

However, there are many other divergences/distances between distributions that we could try to minimize instead:

- Kolmogorov-Smirnov distance
- Lp loss
- Maximum mean discrepancy (MMD)
- Energy distance
- Wasserstein distance
- Renyi-alpha divergence
- Kantorivich – Rubinstein distance
- Total variation distance
- Sobolev distance
- Dudley metric
- Neural network distance
- …

They have very different properties:

- Distance/divergence, bounded/unbounded, continuity, differentiability, statistical power, …

Wasserstein GAN

Arjovsky et al,Wasserstein GAN, 2017

Let $\mathbb{P}_r$ and $\mathbb{P}_g$ denote the distributions of the real and generated data.

The Earth-Mover distance or Wasserstein-1

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\big[\, \|x - y\| \,\big] \;,$$

where $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively $\mathbb{P}_r$ and $\mathbb{P}_g$.

Intuitively, $\gamma(x, y)$ indicates how much "mass" must be transported from x to y in order to transform the distributions $\mathbb{P}_r$ into the distribution $\mathbb{P}_g$.

The EM distance then is the ``cost'' of the optimal transport plan.

Wasserstein GAN

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\big[\, \|x - y\| \,\big] \, ,$$

The Earth-Mover distance is not tractable

However, from the Kantorovich-Rubinstein duality we have that

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \le 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)]$$

where the supremum is over all the 1-Lipschitz functions $f : \mathcal{X} \to \mathbb{R}$.

Similarly, $K \cdot W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \le K} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)]$

where the supremum is over all the K-Lipschitz functions $f : \mathcal{X} \to \mathbb{R}$.

Wasserstein GAN

$$K \cdot W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq K} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)]$$

where the supremum is over all the K-Lipschitz functions $f : \mathcal{X} \to \mathbb{R}$.

Therefore, if we have a parameterized family of functions $\{f_w\}_{w \in \mathcal{W}}$ that are all K-Lipschitz for some K, we could consider solving the problem

$$\max_{w \in \mathcal{W}} \mathbb{E}_{x \sim \mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z \sim p(z)}[f_w(g_\theta(z)]$$

this would yield a calculation of $W(\mathbb{P}_r, \mathbb{P}_\theta)$ up to a multiplicative constant.

How to get $\mathcal{W}$, a family of K-Lipschitz functions for some K?

Consider neural networks with bounded weights.

WGAN objective: $\min_\theta \max_{w \in \mathcal{W}} \mathbb{E}_{x \sim \mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z \sim p(z)}[f_w(g_\theta(z)]$

MMD GAN

Li et al, MMD GAN, 2017

Given two distributions $\mathbb{P}$ and $\mathbb{Q}$, and a kernel k, the square of MMD distance is defined as

$$M_k(\mathbb{P},\mathbb{Q}) = \mathbb{E}_{\mathbb{P}}[k(x,x')] - 2\mathbb{E}_{\mathbb{P},\mathbb{Q}}[k(x,y)] + \mathbb{E}_{\mathbb{Q}}[k(y,y')].$$

Lemma: Let k be a characteristic kernel. Then $M_k(\mathbb{P},\mathbb{Q}) = 0$ iff $\mathbb{P} = \mathbb{Q}$.

An example of characteristic kernel is the Gaussian kernel $k(x,x') = \exp(\|x - x'\|^2)$.

In practice we use finite samples from distributions to estimate MMD distance.

Given $X = \{x_1, \cdots, x_n\} \sim \mathbb{P}$ and $Y = \{y_1, \cdots, y_n\} \sim \mathbb{Q}$, one estimator of $M_k(\mathbb{P},\mathbb{Q})$ is

$$\hat{M}_k(X,Y) = \frac{1}{\binom{n}{2}} \sum_{i \neq i'} k(x_i, x_i') - \frac{2}{\binom{n}{2}} \sum_{i \neq j} k(x_i, y_j) + \frac{1}{\binom{n}{2}} \sum_{j \neq j'} k(y_j, y_j').$$

MMD GAN

$$\widehat{M}_k(X,Y) = \frac{1}{\binom{n}{2}} \sum_{i \neq i'} k(x_i, x_i') - \frac{2}{\binom{n}{2}} \sum_{i \neq j} k(x_i, y_j) + \frac{1}{\binom{n}{2}} \sum_{j \neq j'} k(y_j, y_j').$$

MMD GAN objective function:

$$\min_{\theta} \max_{k \in \mathcal{K}} M_k(\mathbb{P}_{\mathcal{X}}, \mathbb{P}_{\theta}),$$

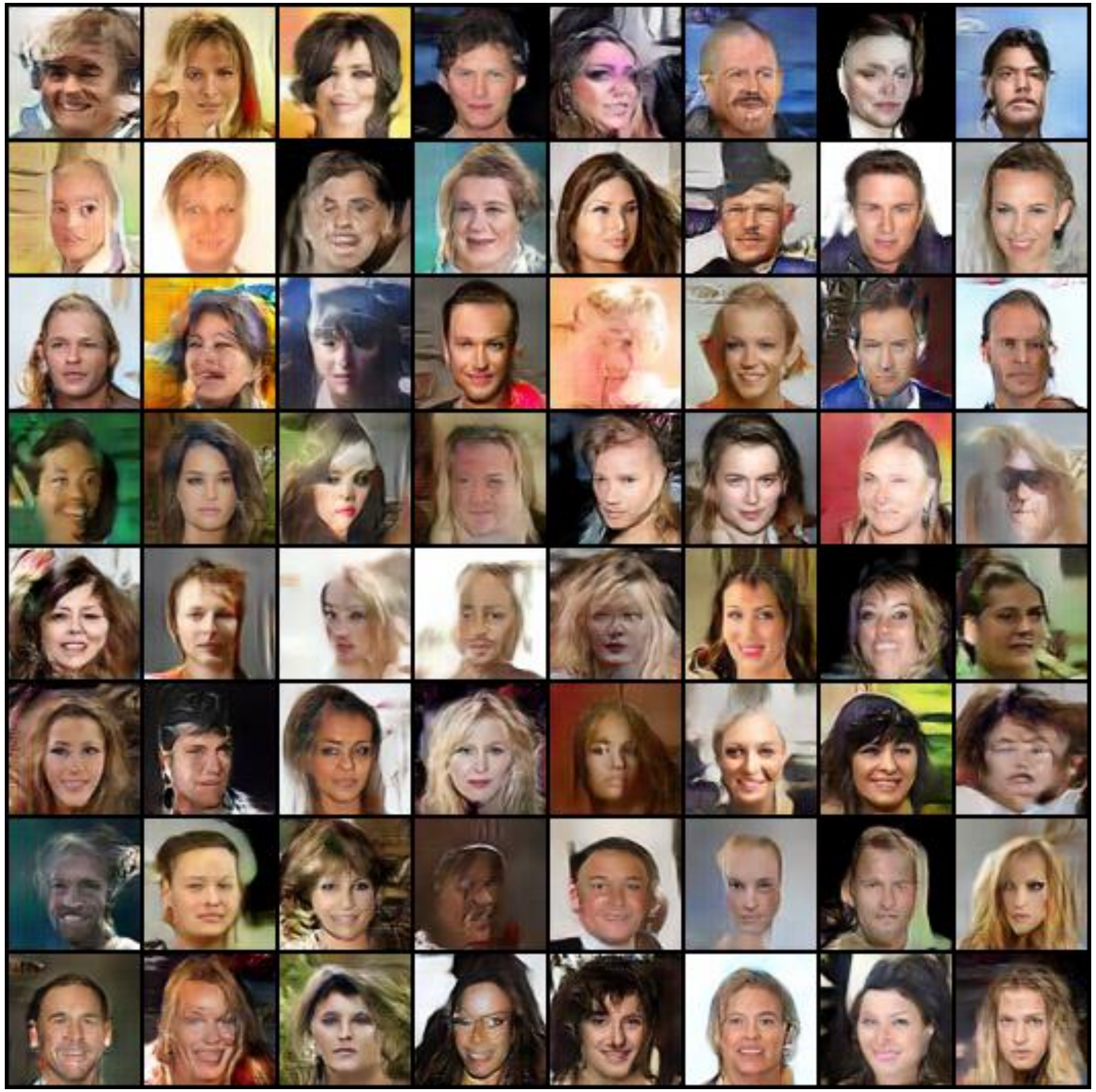
where

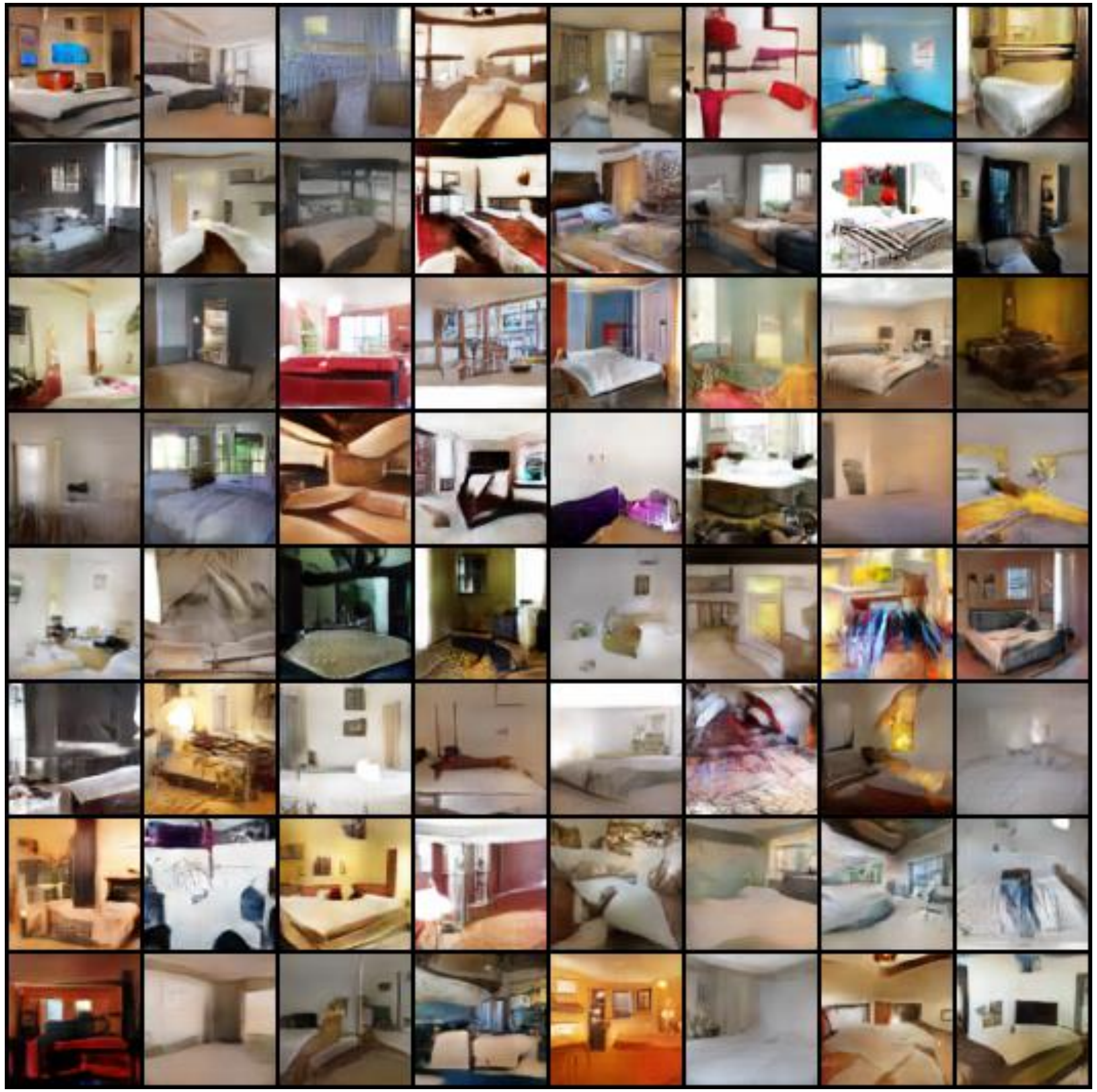
- $\mathbb{P}_{\mathcal{X}}$ is the distribution of the training data
- $\mathbb{P}_{\theta}$ is the distribution of samples generated by the generative neural network.
- θ is the parameters of the generative neural network.
- $\mathcal{K}$ is a set of characteristic kernels.

e.g. combining Gaussian kernels with injective functions f_ϕ:

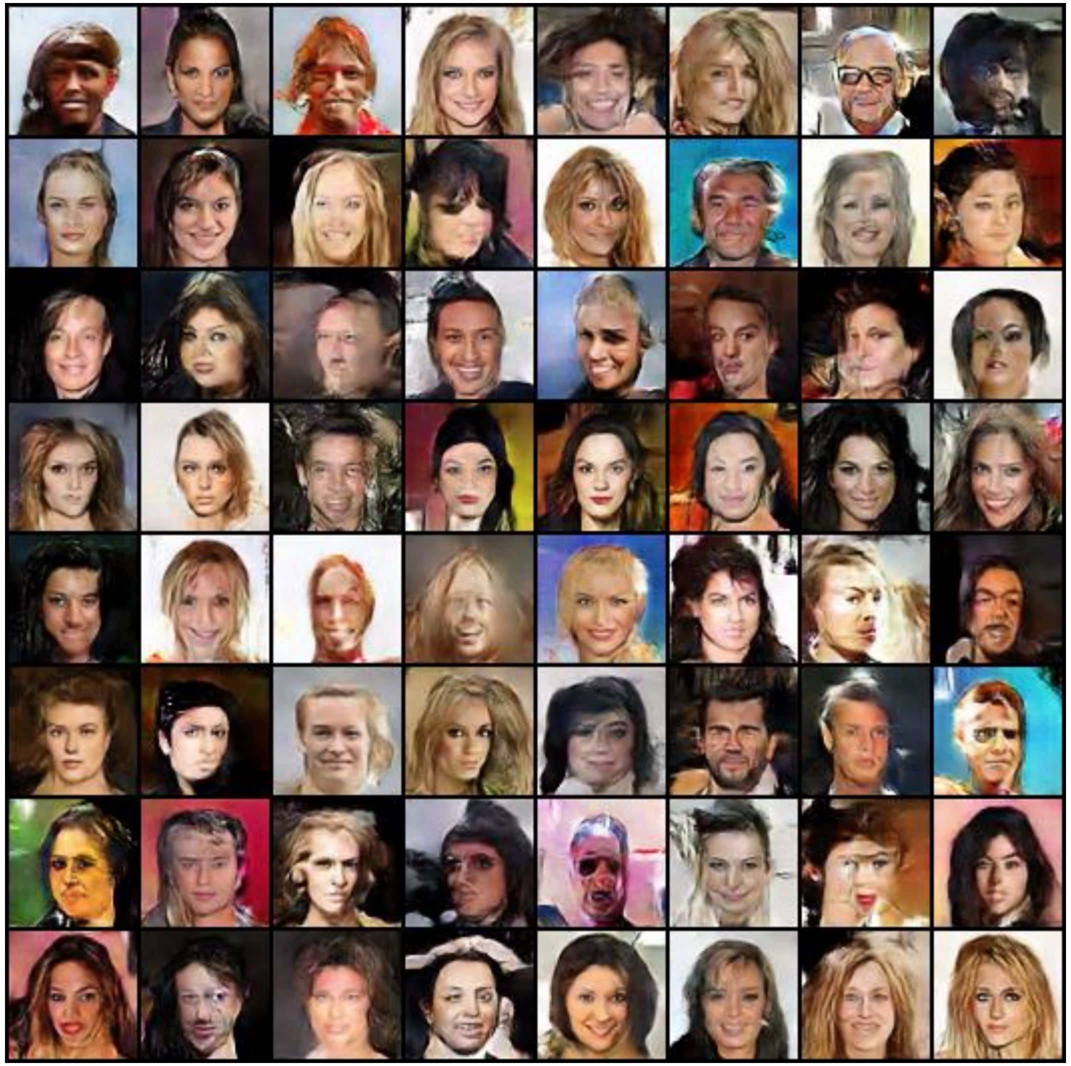
$$\tilde{k}(x,y) = \exp(-\|f_\phi(x) - f_\phi(y)\|^2).$$

MMD GAN

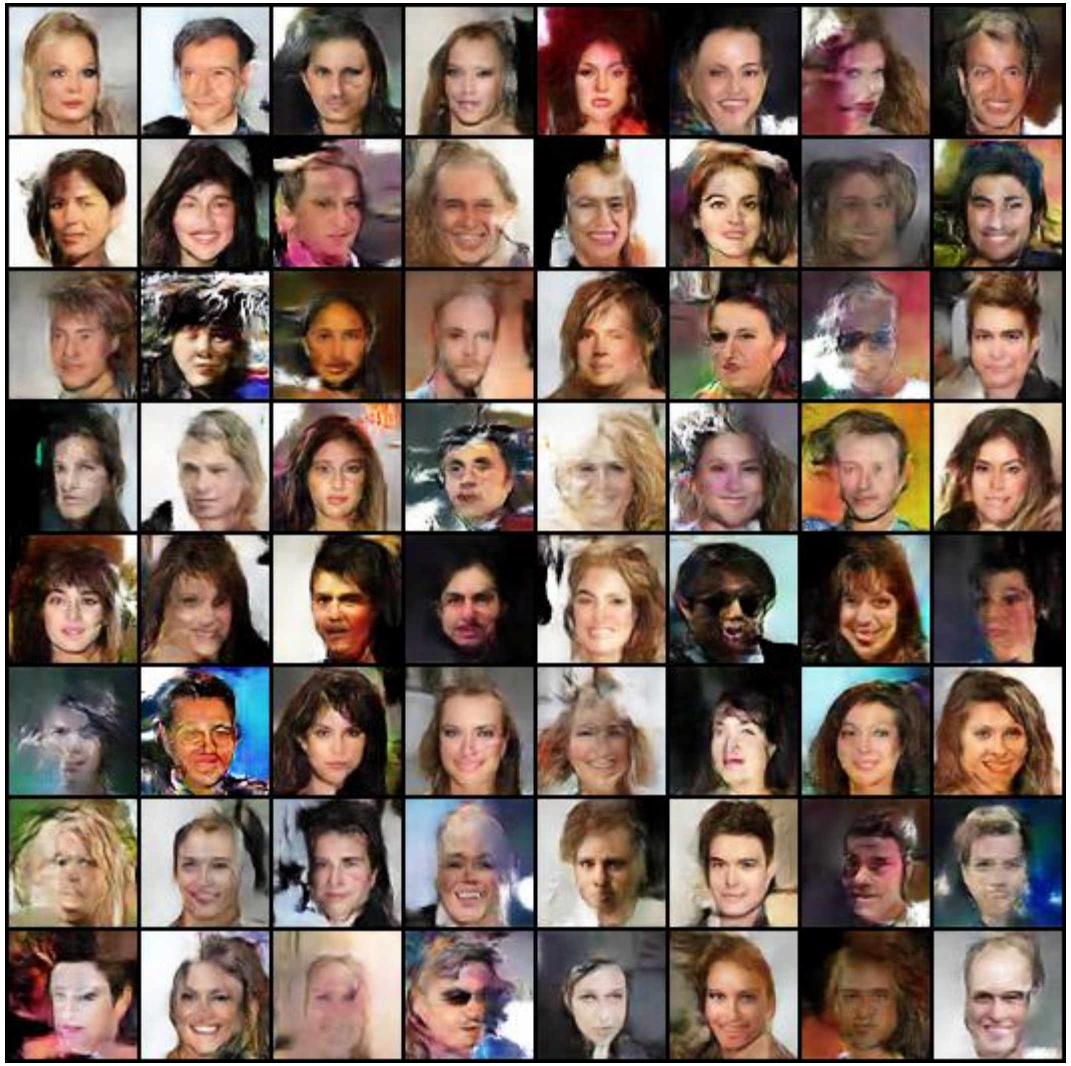




MMD GAN vs WGAN



MMD GAN



WGAN

Take me Home!

Depending on the distance/divergence used between distributions, we can create new GAN methods

These all have different properties, and some divergences have never been tried: e.g. Renyi-alpha GAN?

How to Generate More Complicated Objects?

Point Cloud GAN

Li et al, Point Cloud GAN, 2018

The previous methods generated a sample point from a distribution.

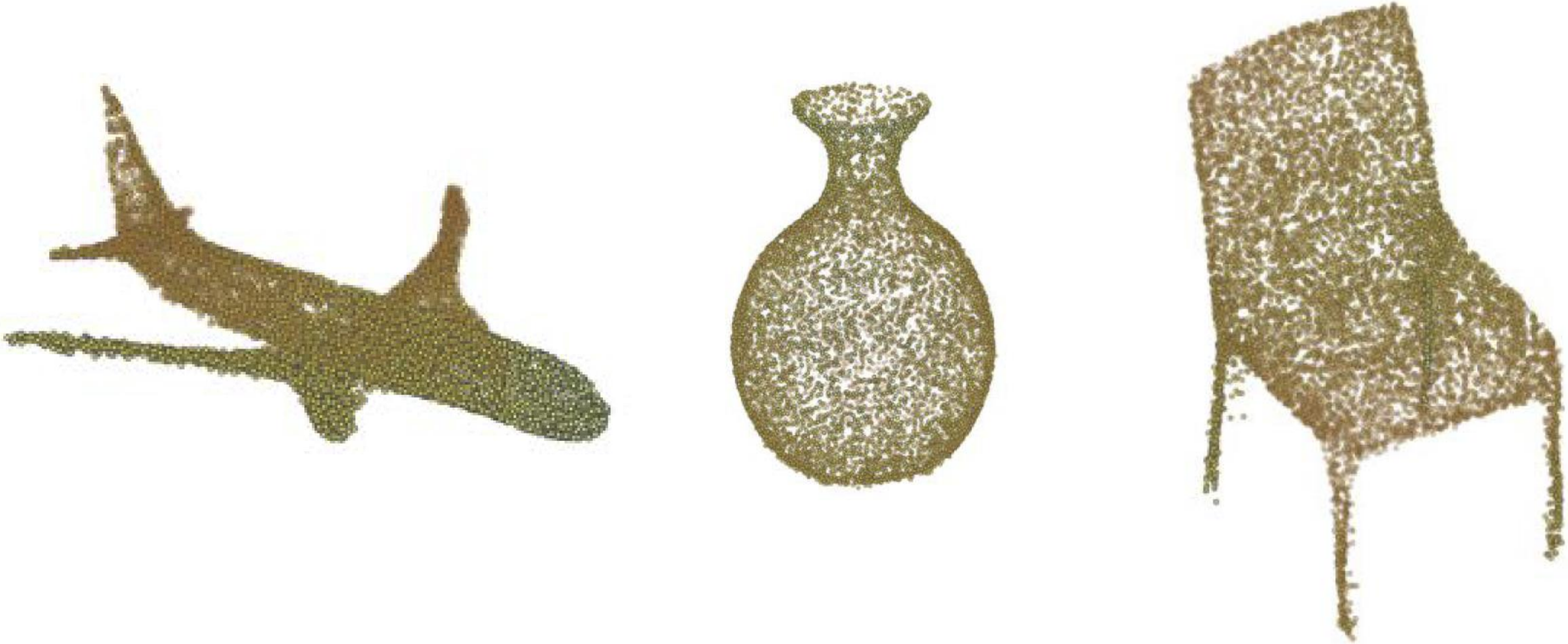
Question: **Can we create a hierarchical data generation process?**

- Generate a sample point x
- Based on x, generate sample points from another conditional distribution p(y|x)

Applications:

- Point cloud generation
- 3D mesh generation
- Autoregressive data generation
- …

Point Cloud GAN



Point Cloud GAN

$$p(X,\theta) = \underbrace{p(\theta)}_{\text{object}} \underbrace{\prod_{i=1}^{n} p(x_i|\theta)}_{\text{points for object}}$$

Issues:

Although GANs have been extended to learn conditional distributions, they require the conditioning variable θ to be observed, such as the one-hot label or a given image.

What should θ be?

Naïvely modeling θ to be a one-hot vector, to indicate which object the points belong to in the training data, cannot generalize to unseen test data.

We need a richer representation for θ, which is an unobserved random variable. Thus, we need to infer θ during the training.

Point Cloud GAN

$$p(X,\theta) = \underbrace{p(\theta)}_{\text{object}} \quad \underbrace{\prod_{i=1}^{n} p(x_i|\theta)}_{\text{points for object}}$$

Solution:

If we knew the feature θ of a given object, we could use conditional GAN:

- We define a prior on input noise variables $p_{\boldsymbol{z}}(\boldsymbol{z})$. (e.g. $\boldsymbol{z} \sim \mathcal{N}(0, I)$, $z \in \mathbb{R}^{d_1}$)
- Let the new generated point be $x = G_x(z,\theta)$, where $z \sim p(z)$,

  $G_x(z,\theta)$ is a generative neural network that takes $z \in \mathbb{R}^{d_1}$ and $\theta \in \mathbb{R}^{d_2}$ as inputs.

Since we don't know vector θ, we need to infer it from the point clouds:

We need to create an inference network Q, that takes a point cloud as input $X = \{x_1, \ldots, x_n\}$, and outputs a vector $\theta \in \mathbb{R}^{d_2}$.

Luckily such neural network exists: DeepSets.

Point Cloud GAN

$$p(X,\theta) = \underbrace{p(\theta)}_{\text{object}} \underbrace{\prod_{i=1}^{n} p(x_i|\theta)}_{\text{points for object}}$$

Hierarchical sampling:

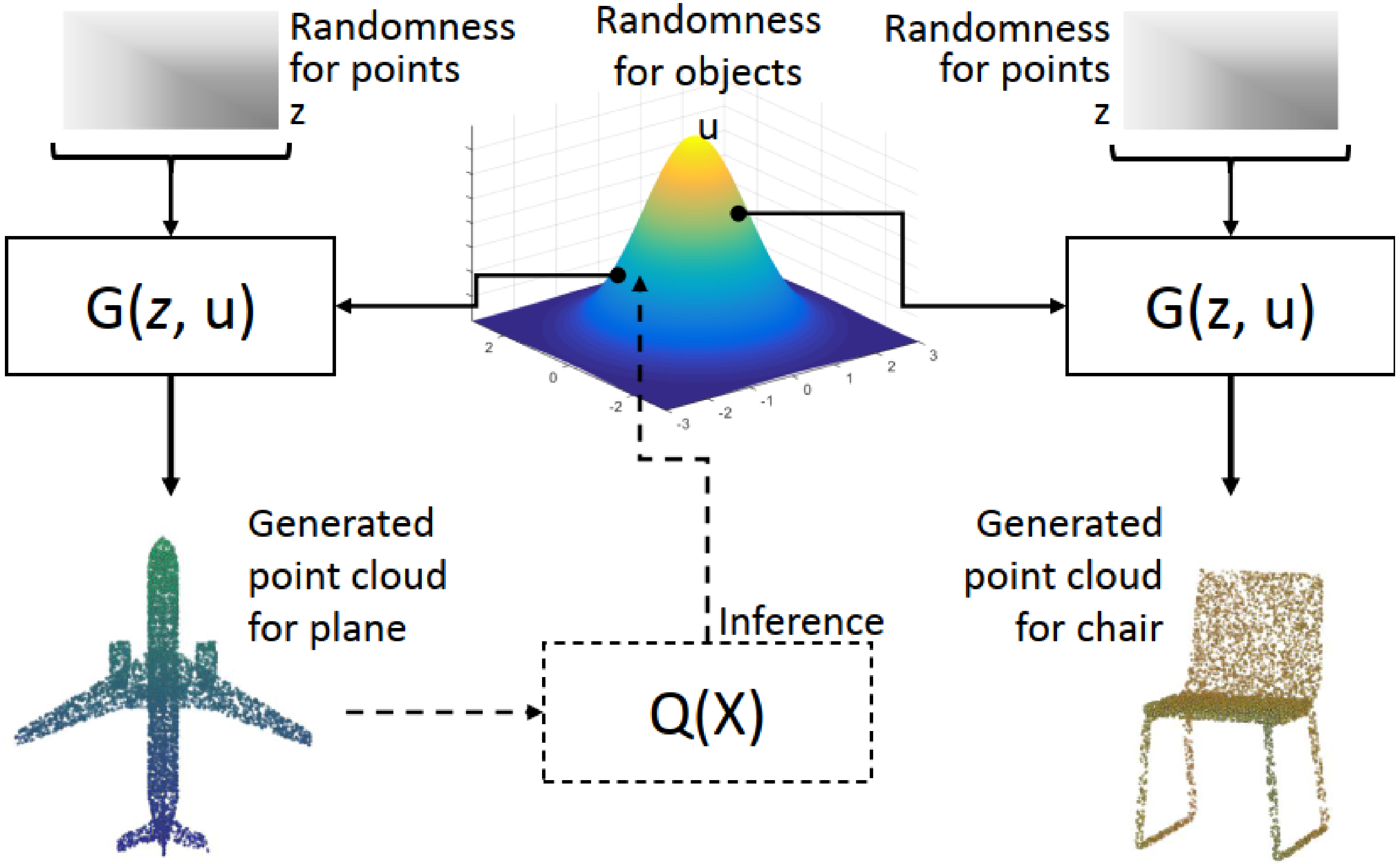
Given a point cloud $X = \{x_1, \ldots, x_n\}$, we can generate more points from this object with this: $x = G_x(z, Q(X))$

To create a fully hierarchical model, all that left is to create another generative neural network $G_\theta(u)$ that can map noise $u \in \mathbb{R}^{d_3}$ into $Q(X) \in \mathbb{R}^{d_2}$ for some point cloud $X = \{x_1, \ldots, x_n\}$.

The full generative process for sampling one point cloud:

$\{x_i\}_{i=1}^n = \{G(z_i, u)\}_{i=1}^n = \{G_x(z_i, G_\theta(u))\}_{i=1}^n$, where $z_1, \ldots, z_n \sim p(z)$, and $u \sim p(u)$.

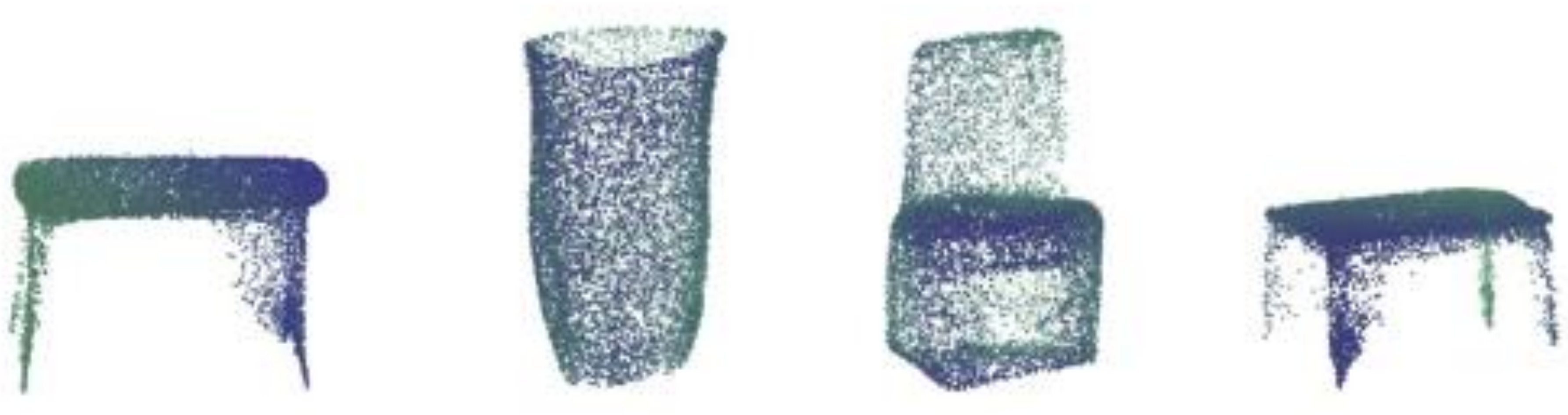
Point Cloud GAN



The full generative process for sampling one point cloud:

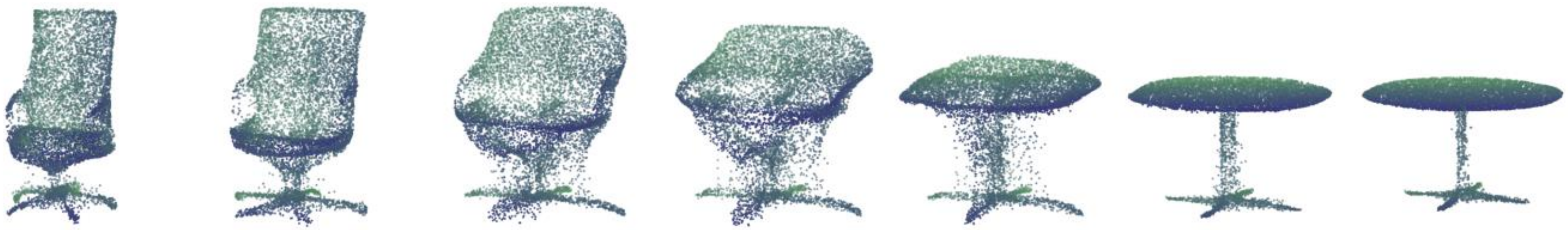
$$\{x_i\}_{i=1}^{n} = \{G(z_i, u)\}_{i=1}^{n} = \{G_x(z_i, G_\theta(u))\}_{i=1}^{n}, \text{ where } z_1, \ldots, z_n \sim p(z), \text{ and } u \sim p(u).$$

Point Cloud GAN



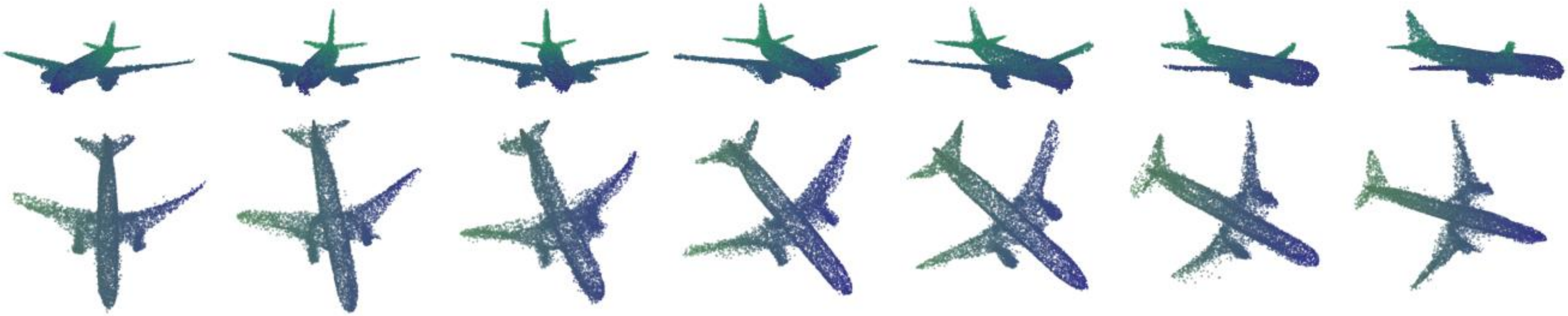
Randomly sampled objects and corresponding point cloud from the hierarchical sampling

Point Cloud GAN



Interpolating between a table and a chair point clouds using the latent space representation.

Point Cloud GAN



Interpolating between different rotations of an airplane, using the latent space representation.

Shallow Generative Methods

- Divergence Estimation
- ML on Sets
    - Regression & Classification
    - Manifold Learning

Important problems in Statistics and ML

Given a dataset,

1. Estimate some properties of the unknown distribution of the data
   (*Entropy, mutual information, KL divergence*, …)

2. Sample more points from this unknown distribution
   (Generative AI)

Density Functional Estimation

Density Functionals

- ❑ Entropy $-\int p \log p$
- ❑ KL Divergence $\int p \log \frac{p}{q}$
- ❑ Mutual Information $\int p_{XY} \log \frac{p_{XY}}{p_X p_Y}$

Fernandes & Gloor: Mutual information is critically dependent on prior assumptions**: would the correct estimate of mutual information please identify itself?**

BIOINFORMATICS Vol. 26 no. 9 2010, pages 1135–1139

Divergences between distributions

Euclidean: $D(p,q) = (\int (p(x) - q(x))^2 dx)^{1/2}$

Kullback-Leibler: $D(p,q) = KL(p,q) = \int p(x) \log \frac{p(x)}{q(x)} dx$

Renyi: $D(p,q) = R_\alpha(p\|q) = \frac{1}{\alpha - 1} \log \int p^\alpha q^{1-\alpha}$

RÉNYI DIVERGENCE ESTIMATION

without density estimation

Using $X_{1:n} = \{X_1, \ldots, X_n\} \sim p \quad Y_{1:m} = \{Y_1, \ldots, Y_m\} \sim q$

Estimate divergence $$R_\alpha(p\|q) \doteq \frac{1}{\alpha - 1} \log \int p^\alpha q^{1-\alpha}$$

How should we estimate them?

Naïve plug-in approach using density estimation

- ❑ histogram
- ❑ kernel density estimation
- ❑ k-nearest neighbors [D. Loftsgaarden & C. Quesenberry. 1965.]

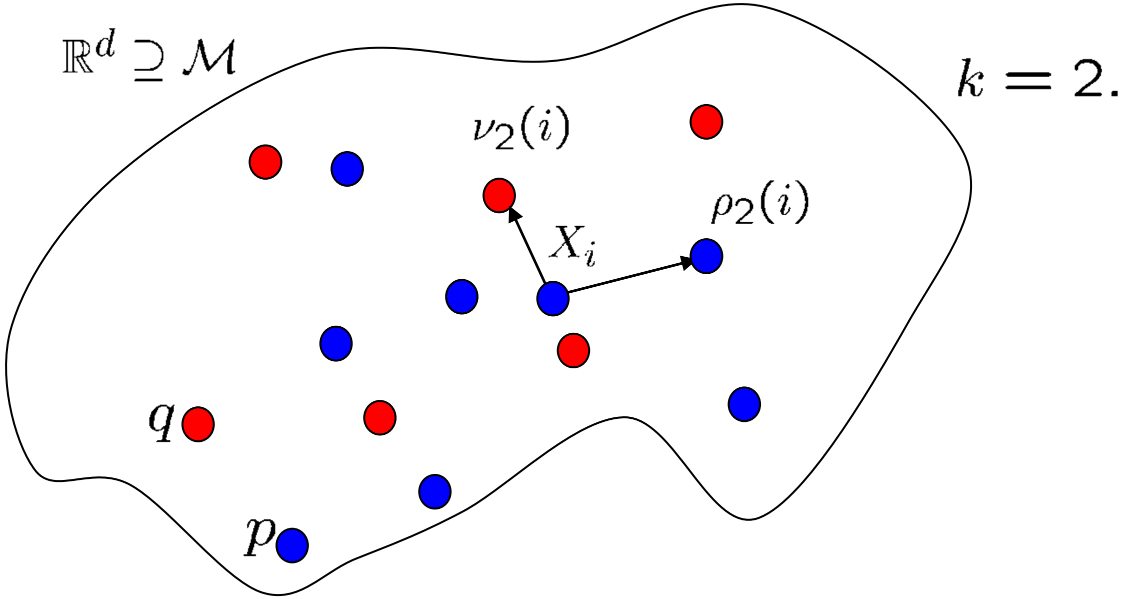
Density: nuisance parameter

Density estimation: difficult, **curse of dimensionality!?**

How can we estimate them directly, without estimating the density?

The estimator

$$R_\alpha(p\|q) \doteq \frac{1}{\alpha-1}\log\int p^\alpha q^{1-\alpha}$$



$k \geq 1$, fixed.

$\rho_k(i)$: the distance of the k-th nearest neighbor of X_i in $X_{1:n}$

$\nu_k(i)$: the distance of the k-th nearest neighbor of X_i in $Y_{1:m}$

$$D_\alpha(p\|q) \doteq \int p^\alpha q^{1-\alpha}$$

$$\widehat{D}_\alpha(X_{1:n}\|Y_{1:m}) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{(n-1)\rho_k^d(i)}{m\nu_k^d(i)}\right)^{1-\alpha}\frac{\Gamma(k)^2}{\Gamma(k-\alpha+1)\Gamma(k+\alpha-1)}$$

Asymptotically unbiased

The estimator

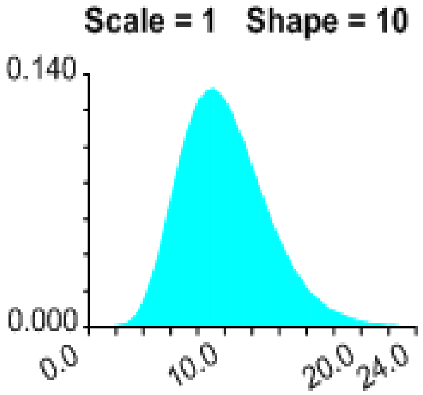
$$\widehat{D}_\alpha(X_{1:n}\|Y_{1:m}) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{(n-1)\rho_k^d(i)}{m\nu_k^d(i)}\right)^{1-\alpha}\frac{\Gamma(k)^2}{\Gamma(k-\alpha+1)\Gamma(k+\alpha-1)}$$

We need to prove:

$$\frac{\Gamma(k-\alpha+1)\Gamma(k+\alpha-1)}{\Gamma(k)^2}\int p^\alpha q^{1-\alpha} = \lim_{n,m\to\infty}\mathbb{E}\left[\left(\frac{(n-1)\rho_k^d(1)}{m\nu_k^d(1)}\right)^{1-\alpha}\right]$$

The r.h.s. can be rewritten as

$$\lim_{n,m\to\infty}\mathbb{E}_{X_1\sim p}\left[\underbrace{\mathbb{E}\left[(n-1)^{1-\alpha}\rho_k^{d(1-\alpha)}(1)\middle|X_1=x\right]}_{\to p^{\alpha-1}(x)\frac{\Gamma(k+1-\alpha)}{\Gamma(k)}c^{\alpha-1}}\underbrace{\mathbb{E}\left[m^{\alpha-1}\nu_k^{d(\alpha-1)}(1)\middle|X_1=x\right]}_{\to q^{1-\alpha}(x)\frac{\Gamma(k+\alpha-1)}{\Gamma(k)}c^{1-\alpha}}\right]$$



Normalized k-NN distances converge to the Erlang distribution

$$\xi_n = (n-1)\rho_k^d(1) \to_d \xi$$

All we need is $\{\xi_n \to_d \xi\} \Rightarrow \{\mathbb{E}[\xi_n^{1-\alpha}] \to \mathbb{E}[\xi^{1-\alpha}]\}$

ENTROPY ESTIMATION

without density estimation

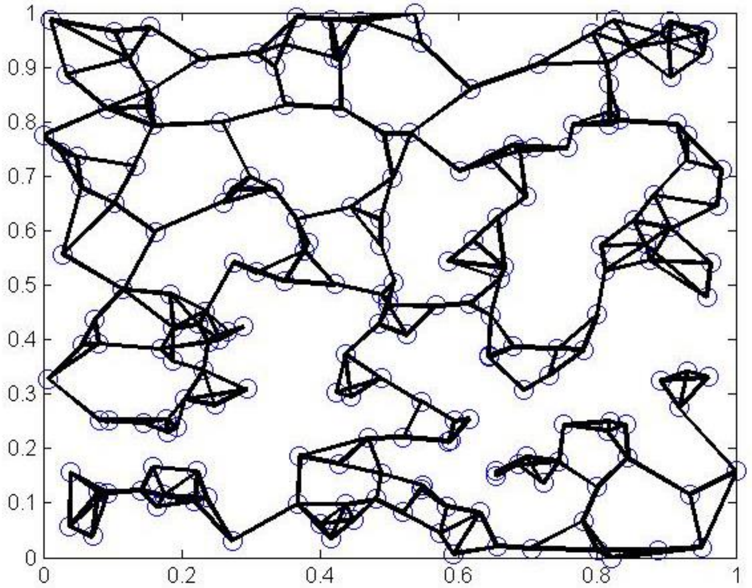
Using $X_{1:n} \doteq (X_1, \ldots, X_n)$ i.i.d. sample $\sim f$

Estimate Rényi entropy $R_\alpha = \dfrac{1}{1-\alpha} \log \int f^\alpha(\mathrm{x})\mathrm{dx}$

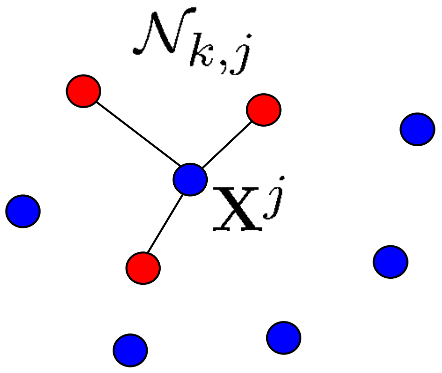
Rényi entropy estimators using kNN graphs

$\mathbf{X}^1, \ldots, \mathbf{X}^n \sim f$ i.i.d. samples in $\mathbb{R}^d$

Let $p \doteq d - d\alpha$, k fixed.



Let $\mathcal{N}_{k,j}$ be the set of the k nearest neighbours of $\mathbf{X}^j$ in $\{\mathbf{X}^1, \ldots, \mathbf{X}^n\}$



$k = 3$

Calculate: $L_n = \sum_{j=1}^{n} \sum_{\mathbf{V} \in \mathcal{N}_{k,j}} \|\mathbf{V} - \mathbf{X}^j\|^p$

$$\frac{1}{1-\alpha} \log \left(\frac{L_n}{n^{(d-p)/d} \beta} \right) \to H_\alpha(\mathbf{X})$$

MUTUAL INFORMATION ESTIMATION

without density estimation

Using $X_1, \dots, X_n$ i.i.d. sample $\sim f = (f_1, \dots, f_d)$

Estimate MI $I_\alpha \doteq \frac{1}{\alpha - 1} \log \int f^\alpha(x) \left(\prod_{i=1}^{d} f_i(x_i) \right)^{1-\alpha} \mathrm{d}x$

How can we get mutual information estimators from entropy estimators?

Trick: **Information is preserved under monotonic transformations.**

Let $(g_1(X_1), \ldots, g_d(X_d)) = (Z_1, \ldots, Z_d) = \mathbf{Z}$
where $g_j : \mathbb{R} \to \mathbb{R},\ j = 1, \ldots, d$, are monotone functions.

$$\boxed{I_\alpha(\mathbf{Z}) \doteq \frac{1}{\alpha - 1} \log \int_{\mathcal{Z}} \left(f_{\mathbf{Z}}(\mathbf{z}) \right)^\alpha \qquad d\mathbf{z} = I_\alpha(\mathbf{X})}$$

When the marginals of $\mathbf{Z}$ are uniform, $\Rightarrow I_\alpha(\mathbf{Z}) = -H_\alpha(\mathbf{Z})$

$$\Rightarrow I_\alpha(\mathbf{X}) = I_\alpha(\mathbf{Z}) = -H_\alpha(\mathbf{Z})$$

Monotone transform

Uniform margins

Transformation to get uniform margins

Monotone transformation leading to uniform margins?

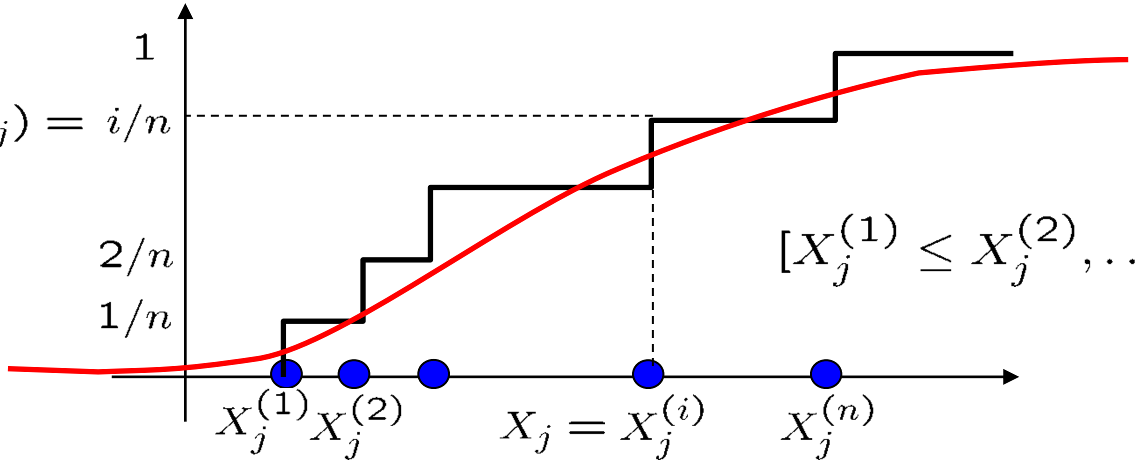
Prob theory 101: $X_j \sim F_j$ cont. $\Rightarrow F_j(X_j) \sim U[0,1]$

The **copula transformation:**

Let $\mathbf{X} = \boxed{[X_1, \ldots, X_d] \to [F_1(X_1), \ldots, F_d(X_d)]} = [Z_1, \ldots, Z_d] = \mathbf{Z}$

A little problem: we don't know F_i distribution functions…

Solution: Empirical distribution function (ranks are enough)



$[X_j^{(1)} \leq X_j^{(2)}, \ldots, \leq X_j^{(n)}] \doteq sort\{X_j^1, \ldots, X_j^n\}$

Extensions

Conditional Rényi Mutual Information:

$$I_\alpha(X,Y|Z) \doteq \int p_Z(z) D_\alpha(p(X,Y|Z=z) \| p(X|Z=z)p(Y|Z=z)|Z=z)$$

$$\widehat{I}_\alpha = \frac{1}{\alpha-1} \log \frac{1}{N} \sum_{n=1}^{N} \frac{(c_{xyz})^{(1-\alpha)}\, \rho_{xyz}^{d_{xyz}(1-\alpha)}(X_n;Y_n;Z_n)}{(c_{xz})^{(1-\alpha)}\, \rho_{xz}^{d_{xz}(1-\alpha)}(X_n;Z_n)} \frac{(c_z)^{(1-\alpha)}\, \rho_z^{d_z(1-\alpha)}(Z_n)}{(c_{yz})^{(1-\alpha)}\, \rho_{yz}^{d_{yz}(1-\alpha)}(Y_n;Z_n)} B^2,$$

where $B^2 = \frac{\Gamma^4(k)}{\Gamma^2(k-\alpha+1)\Gamma^2(k+\alpha-1)}$.

Open Questions

- What density functionals can we estimate without estimating the densities themselves?
  - entropy, divergences, mutual information,…

- When can we avoid the curse of dimensionality?

- How can we exploit manifold property in the data?

- What are the most practical “smoothness classes”?

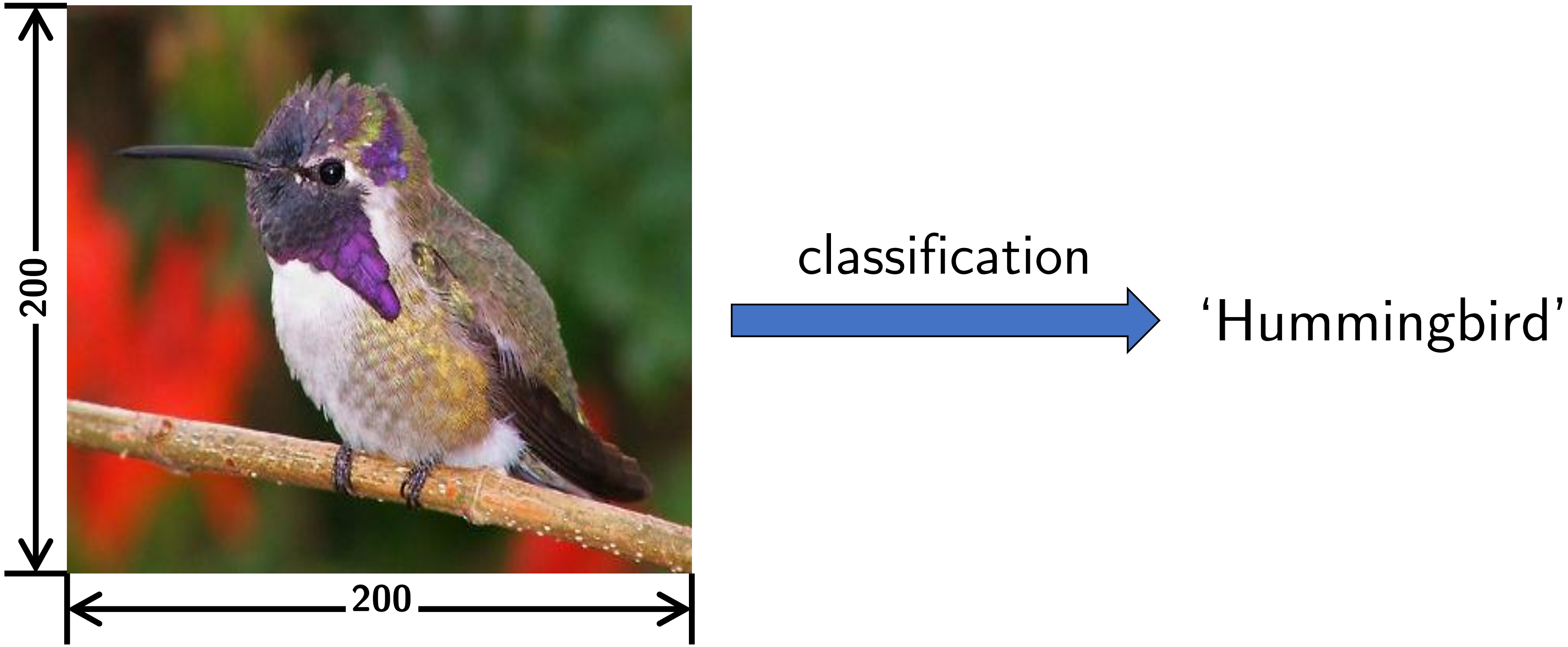
Take me Home!

Some density functionals
(e.g entropy, mutual information, divergences)
can be estimated directly,
without estimating the densities first!

ML on Sets

Motivation

Traditionally, machine learning handles data of the form of fixed dimensional vectors



classification

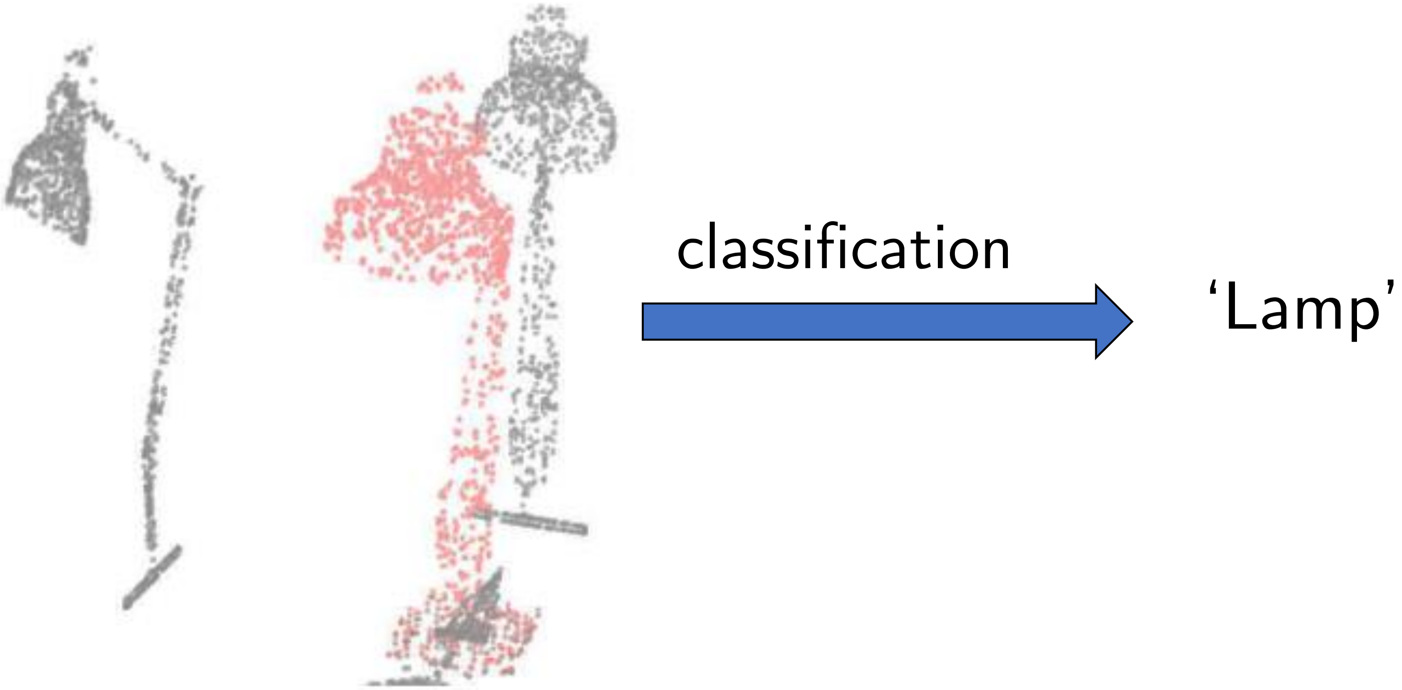
‘Hummingbird’

Motivation

What happens if the inputs are sets?

□ Unordered collection of objects

□ and the number of objects can vary

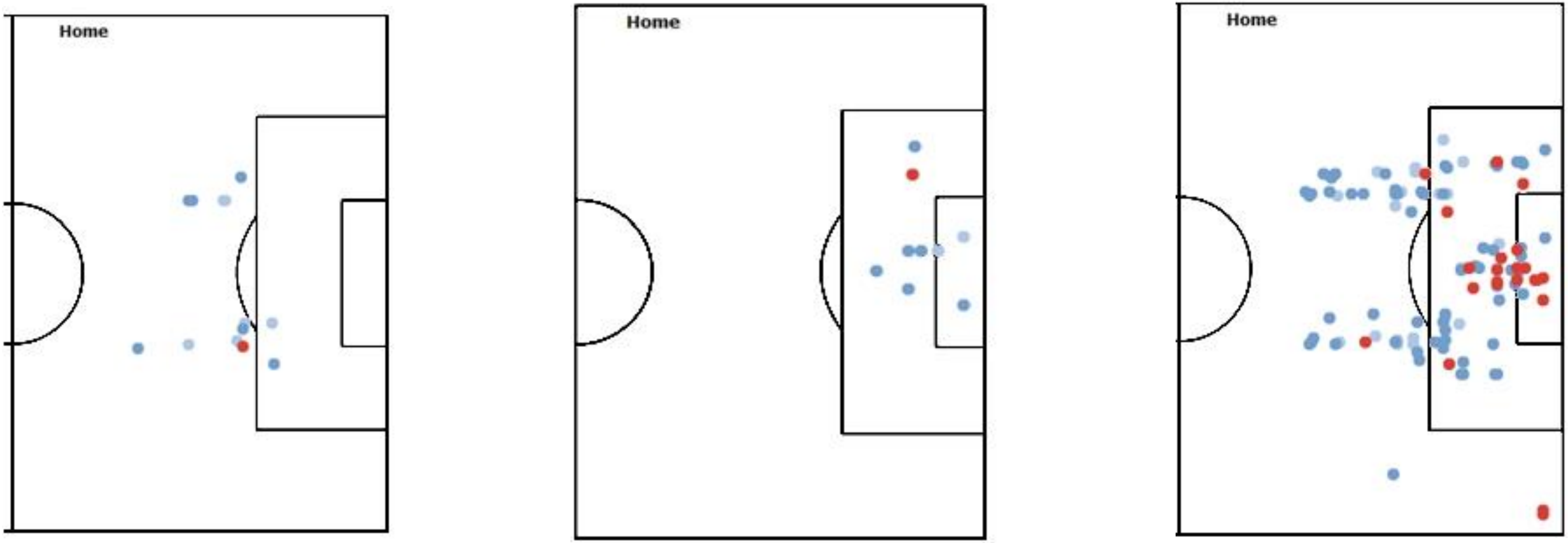


classification

‘Lamp’

Distributional Data

Manchester United 07/08



Owen Hargreaves

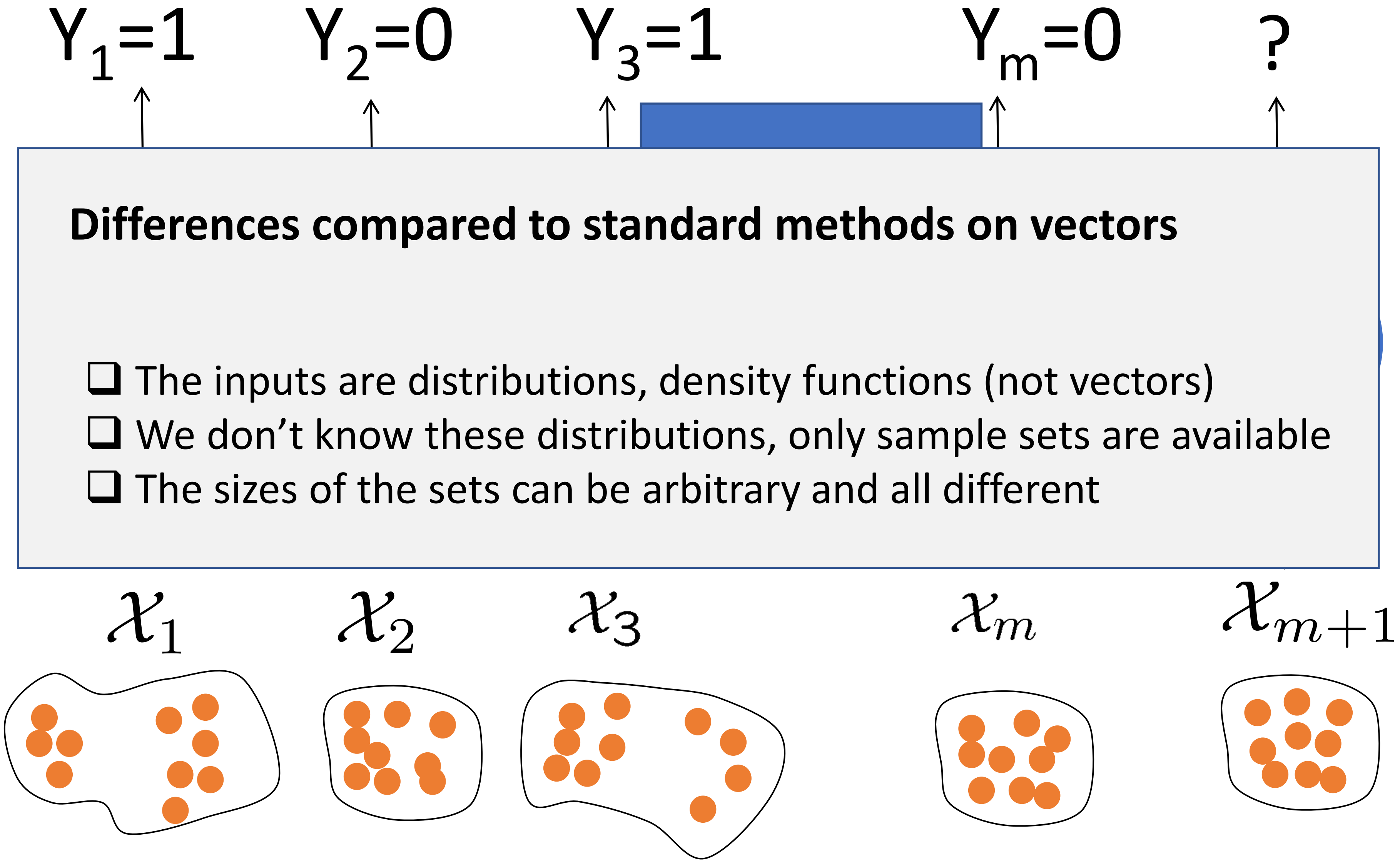
Rio Ferdinand

Cristiano Ronaldo

Shot Type
- ● Goals
- ● Shots on Goal
- ● Shots

www.juhokim.com/projects.php

Distribution Regression / Classification



Differences compared to standard methods on vectors

- □ The inputs are distributions, density functions (not vectors)
- □ We don’t know these distributions, only sample sets are available
- □ The sizes of the sets can be arbitrary and all different

Kernel / Support Vector Regression

Kernel / Support Vector Ridge Regression

Linear regression after feature transformation: $f(x) = \langle \mathbf{w}, \phi(x) \rangle$

Primal problem:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}\in\mathcal{K}} \sum_{i=1}^{n} \xi_i^2$$

subject to $y_i - \langle \underbrace{\phi(x_i)}_{\mathbf{x}_i}, \mathbf{w} \rangle = \xi_i, \ \forall i = 1, \dots, n$

and $\|\mathbf{w}\| \leq B$

Kernel / Support Vector Ridge Regression Algorithm

Dual problem:

Given $D = \{(x_i, y_i), i = 1, \ldots, n\}$ training data set.
$k(\cdot,\cdot)$ kernel, $\lambda > 0$ parameter. $\mathbf{y} \doteq (y_1, \ldots, y_n)^T \in \mathbb{R}^n$

- $\boldsymbol{G} \in \mathbb{R}^{n \times n} \doteq \{G_{ij}\}_{i,j}^{n,n}$,

  where $G_{ij} \doteq \overbrace{\langle \underbrace{\mathbf{x}_i}_{\phi(x_i)}, \underbrace{\mathbf{x}_j}_{\phi(x_j)} \rangle_{\mathcal{K}}}^{k(x_i,x_j)}$, Gram matrix.
- $\hat{\boldsymbol{\alpha}} = (\boldsymbol{G} + \lambda \boldsymbol{I}_n)^{-1}\mathbf{y}$
- $\hat{\mathbf{w}} = \sum_{i=1}^{n} \hat{\alpha}_i \phi(x_i)$.
- $f(x) = \langle \hat{\mathbf{w}}, \phi(x) \rangle = \sum_{i=1}^{n} \hat{\alpha}_i k(x_i, x)$

Kernel Estimation for Support Vector Machines

Kernel function: $K(\cdot,\cdot)$ is a positive semi-definite function.

Linear kernel: $K(p,q) = \int pq$

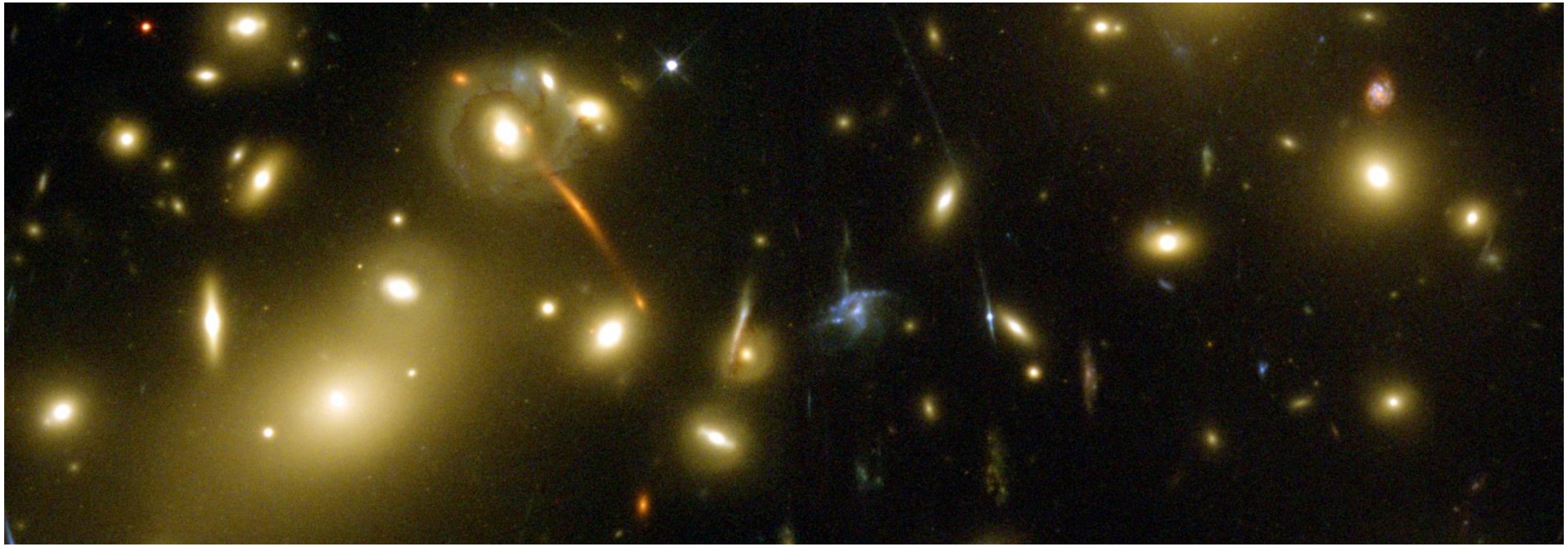
Polynomial kernel: $K(p,q) = (\int pq + c)^s$

Gaussian kernel: $K(p,q) = \mathrm{exp}(-\frac{1}{2\sigma^2}(\int (p-q)^2) = \mathrm{exp}(-\frac{1}{2\sigma^2}(\int p^2 + \int q^2 - 2\int pq).$

We only need to estimate $\int p^\alpha q^\beta$ terms.

Applications

Estimating Properties where Physics is too Complicated



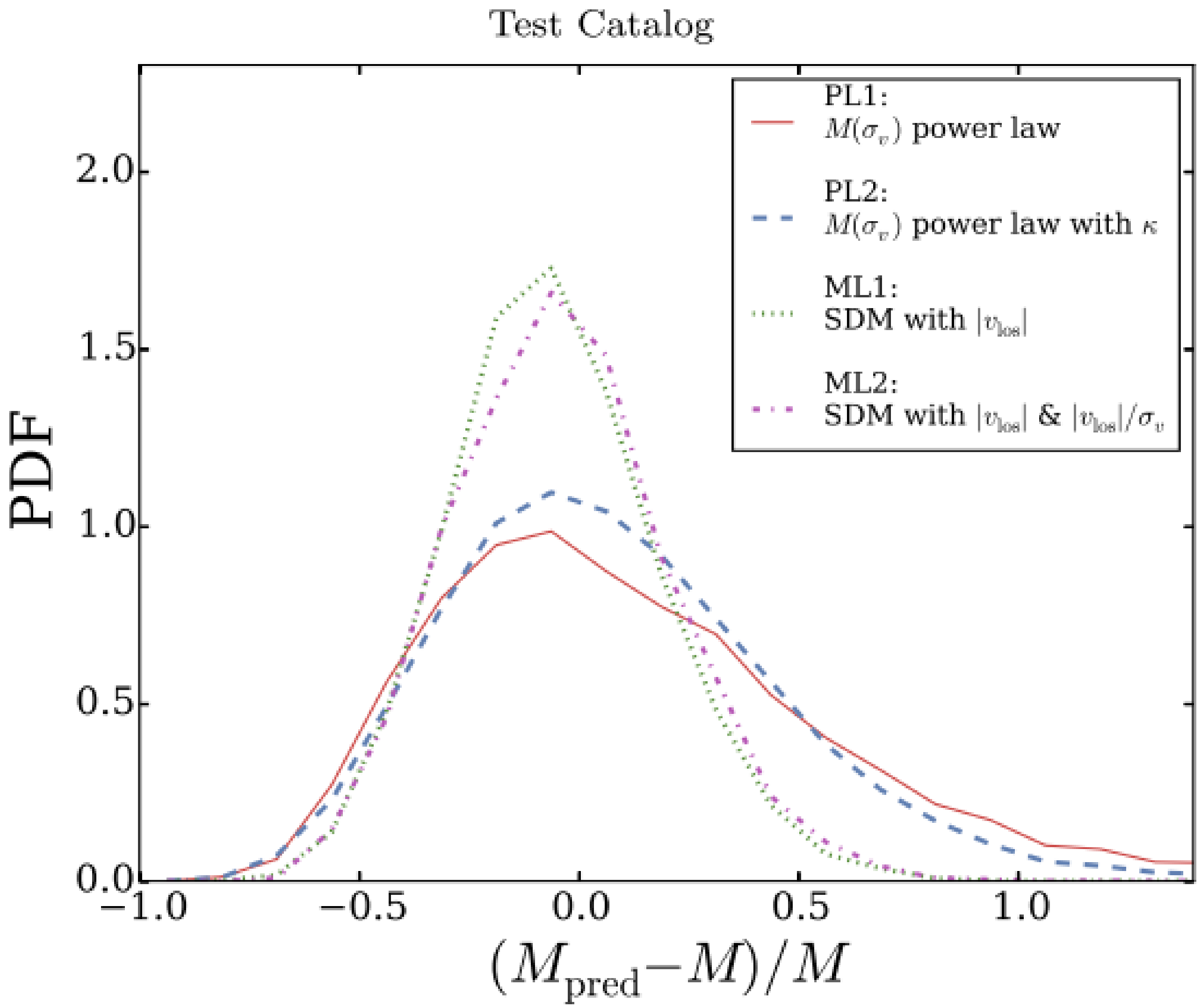
Goal: Estimate dynamical mass of galaxy clusters.

Importance: Galaxy clusters are being the largest gravitationally bound systems in the Universe. Dynamical mass measurements are important to understand the behavior of dark matter and normal matter.

Difficulty: We can only measure the velocity of galaxies not the mass of their cluster. Physicists estimate dynamical cluster mass from single velocity dispersion.

Our method: Estimate the cluster mass from the whole distribution of velocities rather than just a simple velocity distribution.

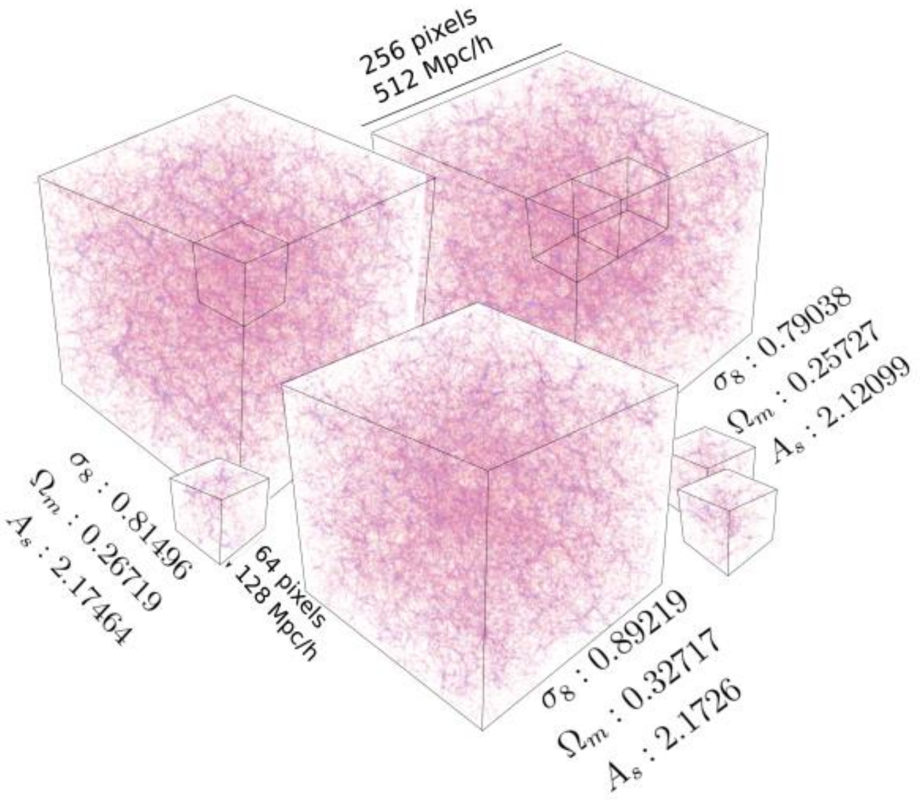
Estimating Properties where Physics is too Complicated

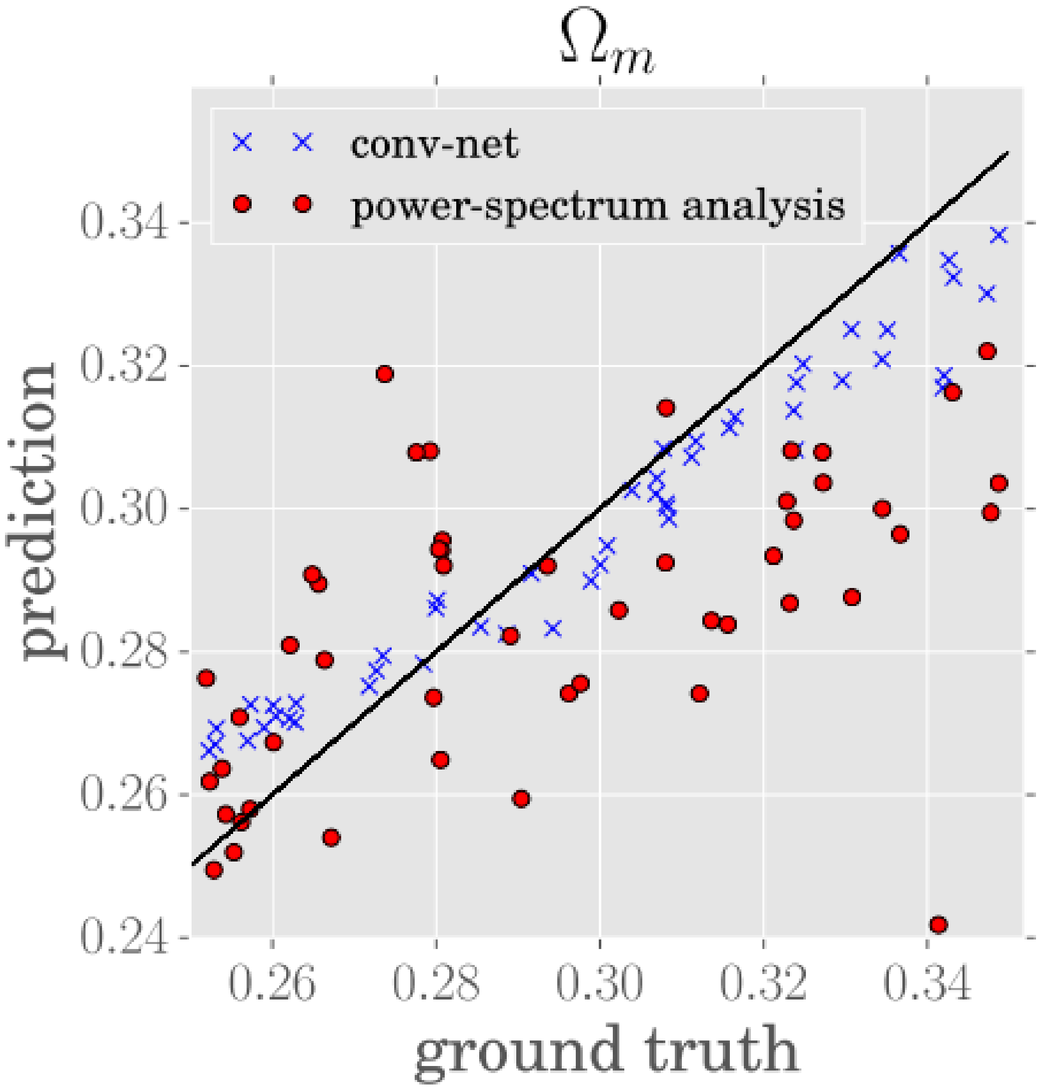


Michelle Ntampaka et al, A Machine Learning Approach for Dynamical Mass Measurements of Galaxy Clusters, APJ 2015

Find the parameters of Universe

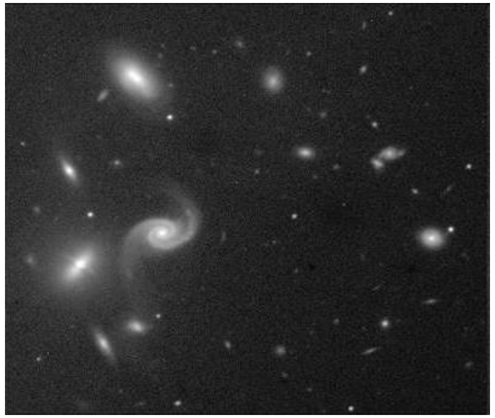
Given a distribution of particles, our goal is to predict the parameters of the simulated universe





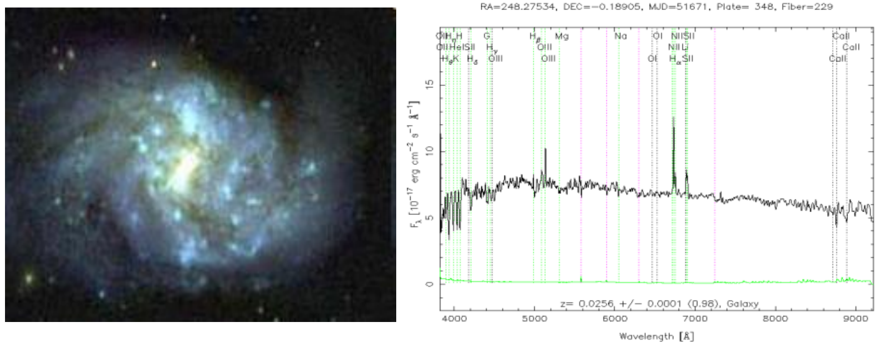
Ravanbakhsh, M., Oliva, J., Fromenteau, S., Price, L., Ho, S., Schneider, J., and Poczos, B., ICML 2016

Find interesting Galaxy Clusters

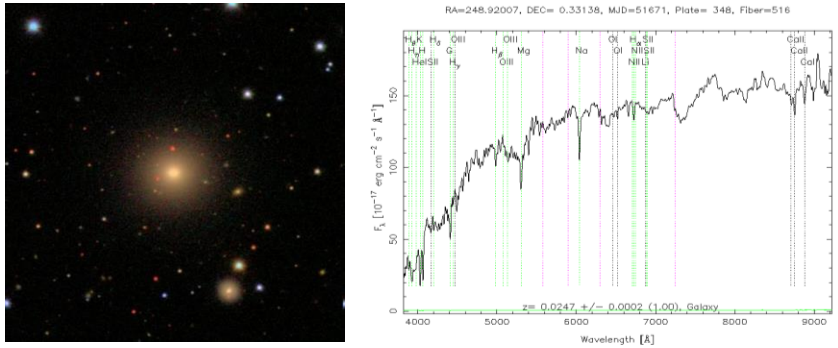


Sloan Digital Sky Survey (SDSS)

- ❑ continuum spectrum
- ❑505 galaxy clusters (10-50 galaxies in each)
- ❑7530 galaxies



Blue galaxy



Red galaxy

What are the most anomalous galaxy clusters?

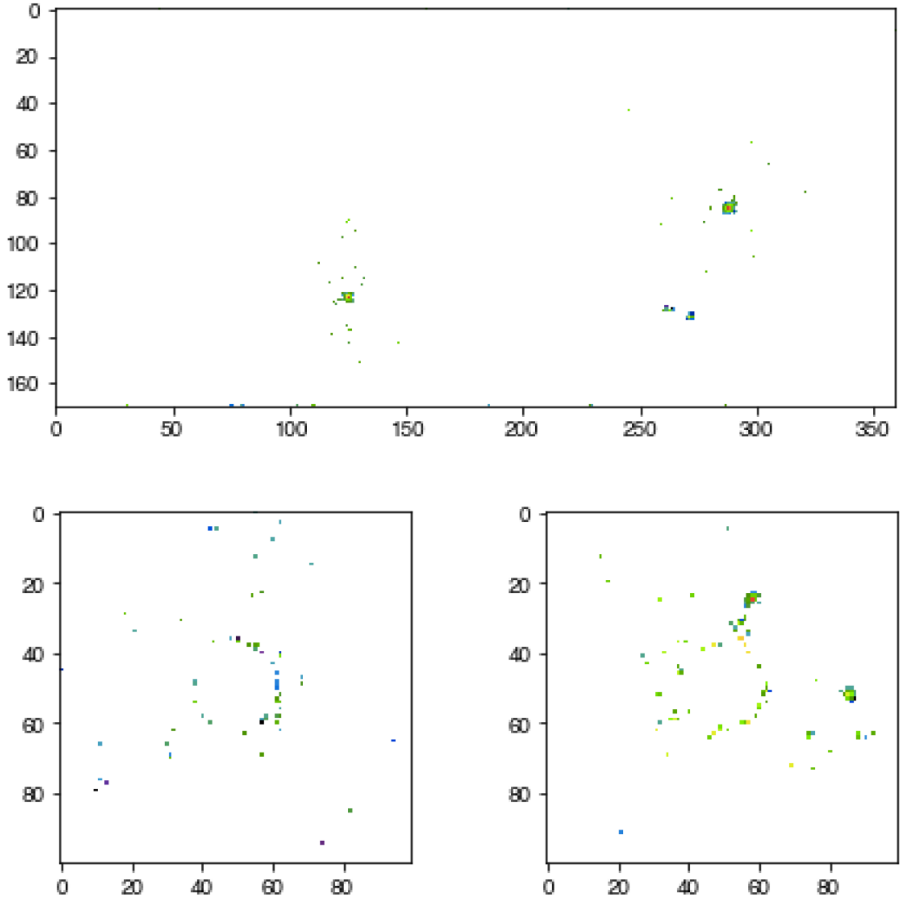
The most anomalous galaxy cluster contains mostly

- ❑ star forming blue galaxies
- ❑ irregular galaxies

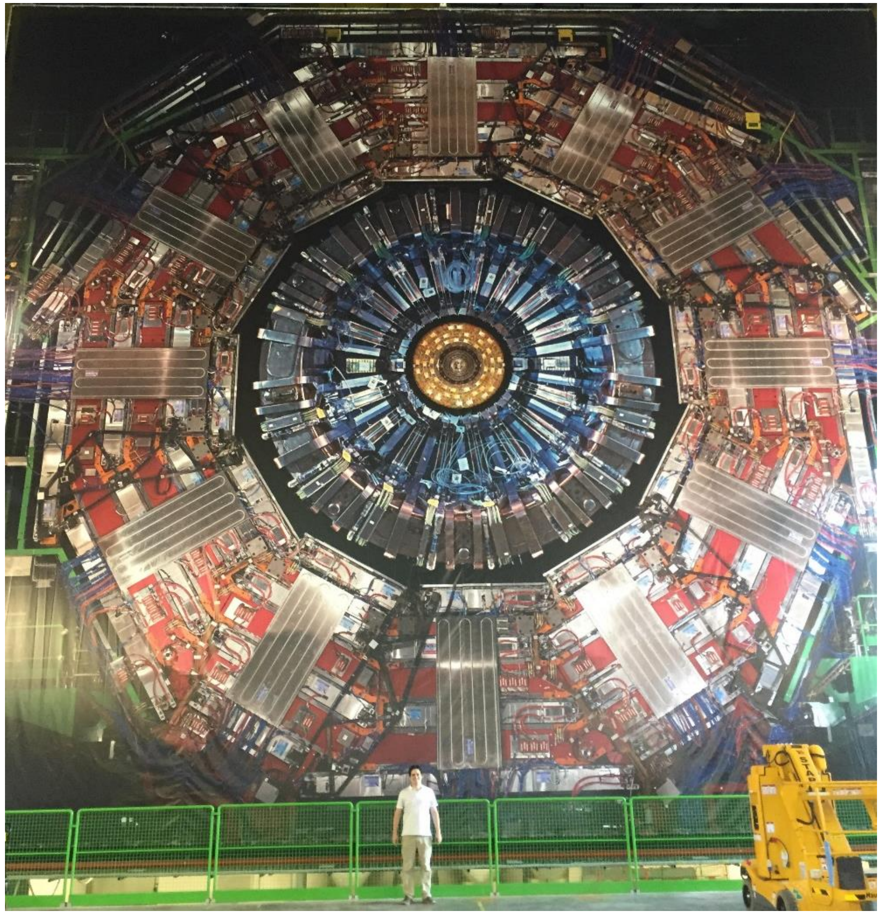
Image Credits: ESA, NASA

Point Cloud Applications – High Energy Physics

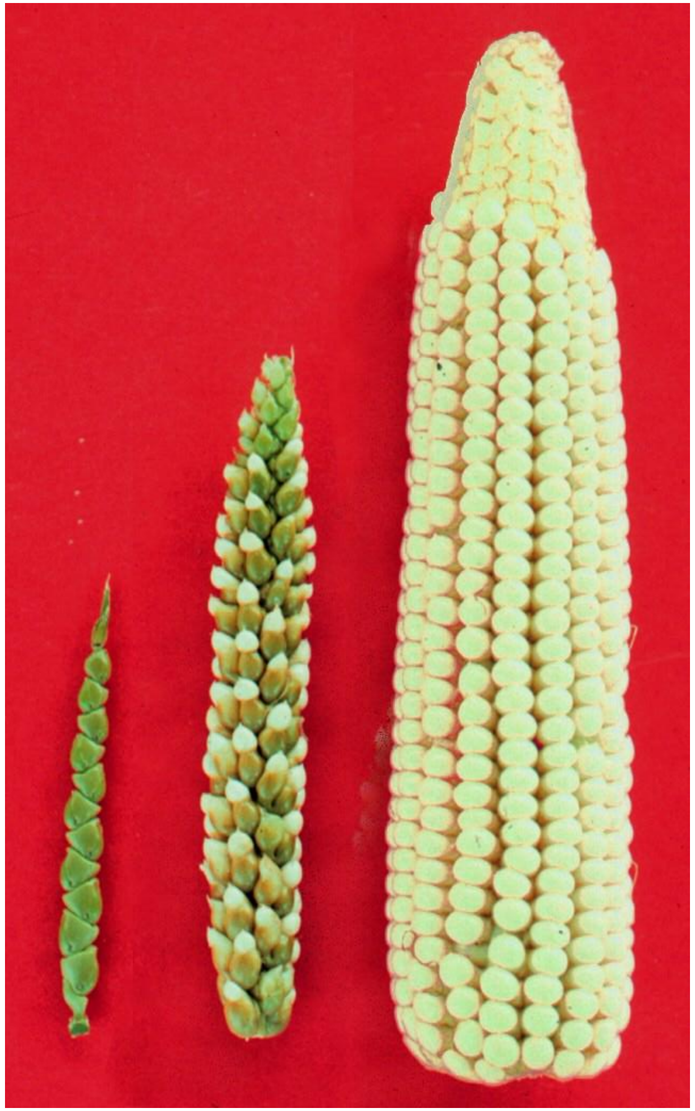
Compact Muon Solenoid data (CMS, LHC)

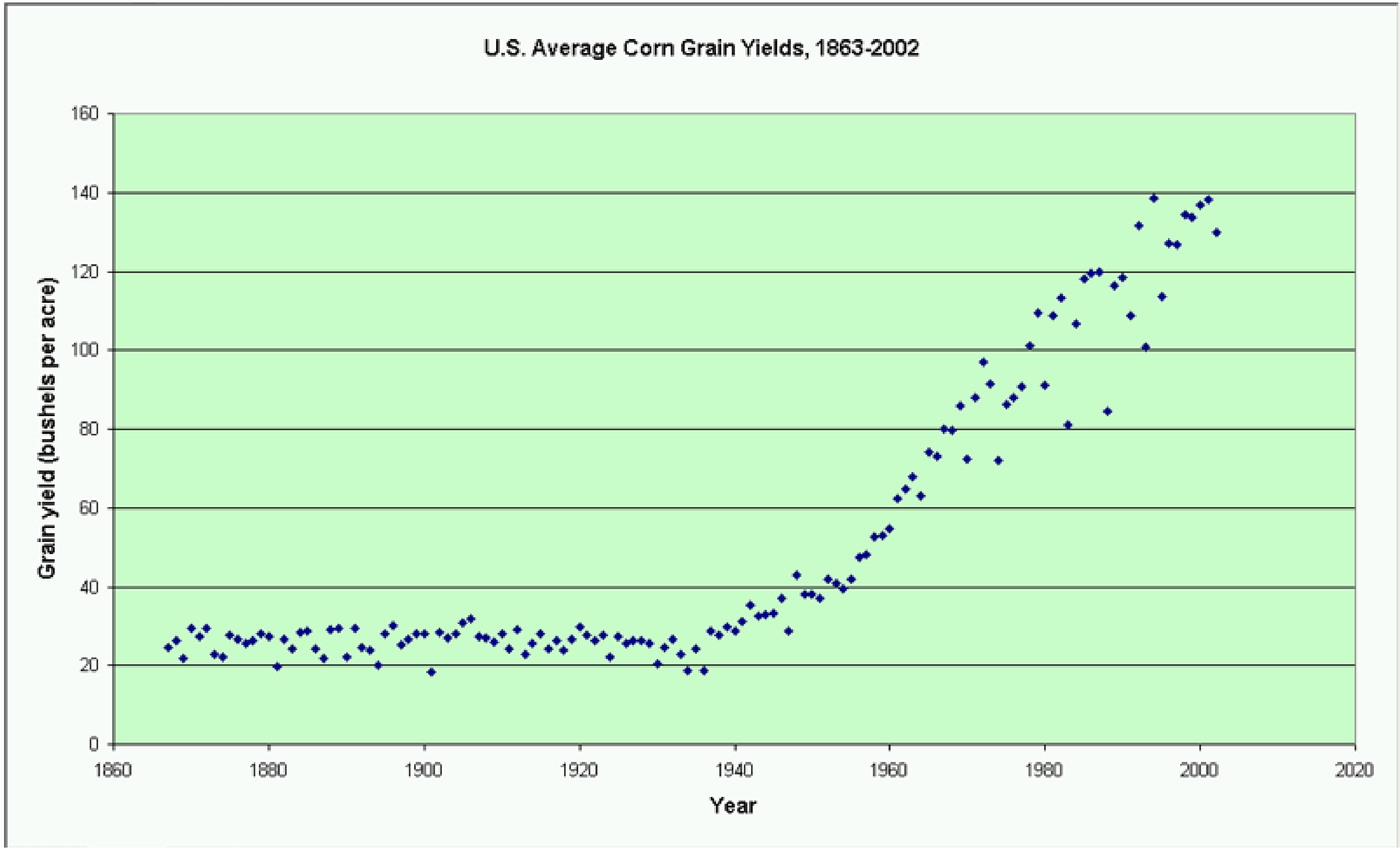


End-to-End Event Classification

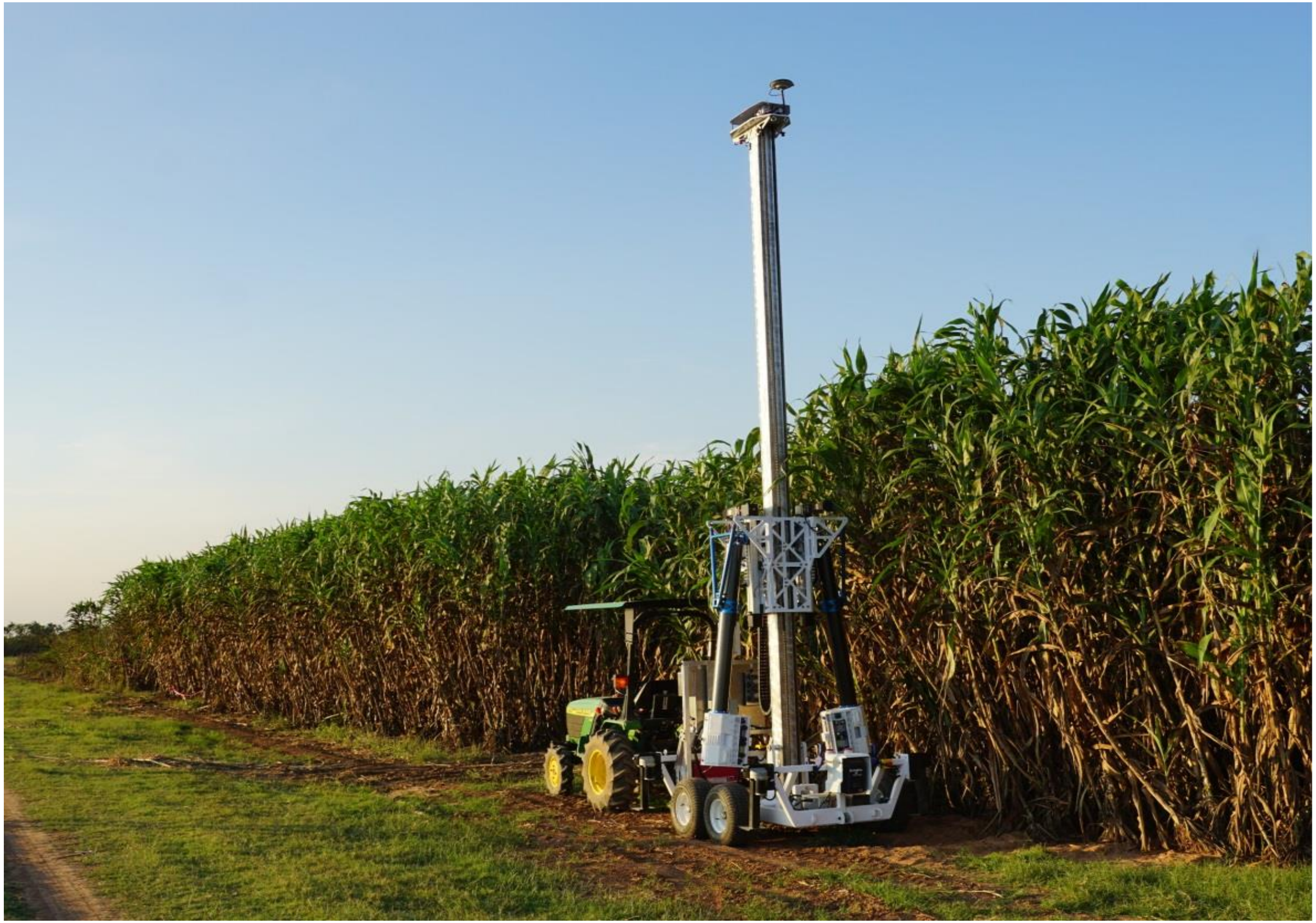


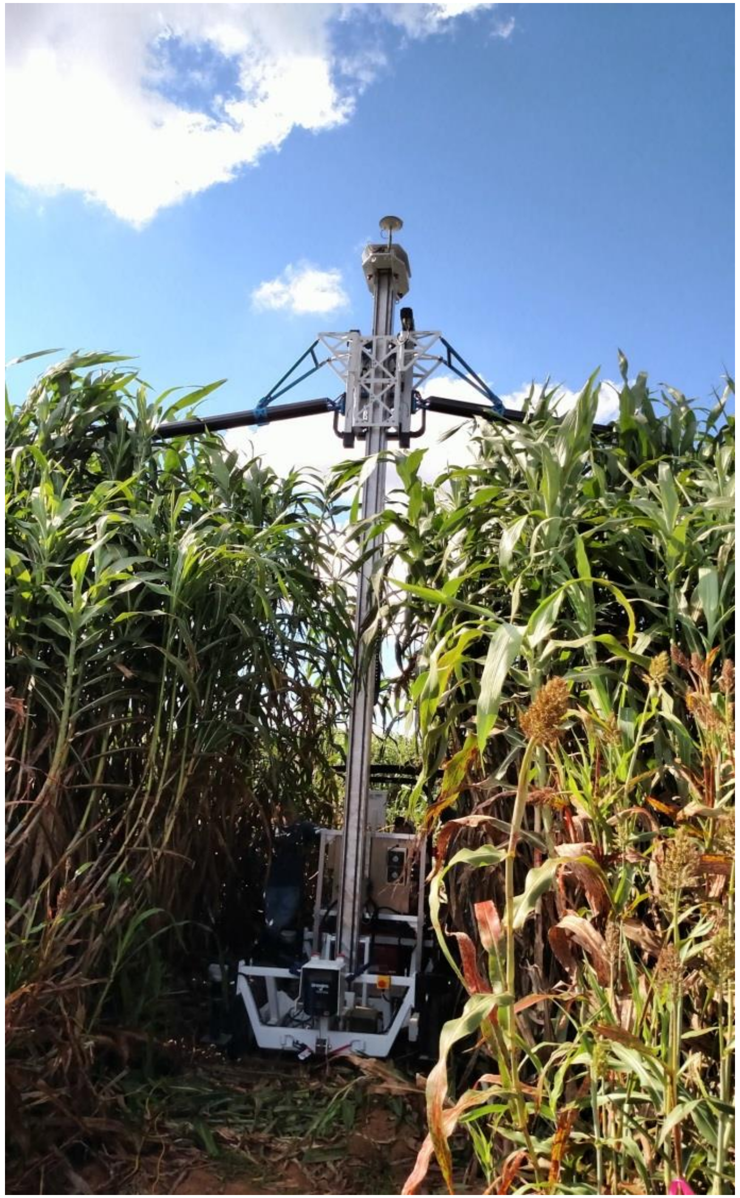
Corn Evolution

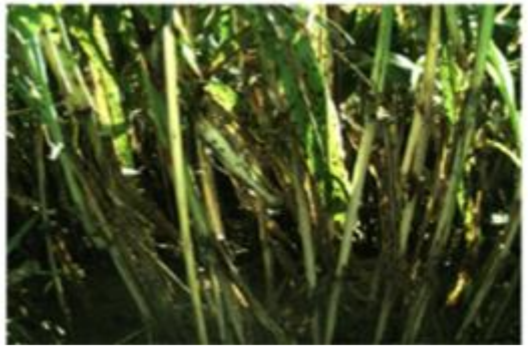


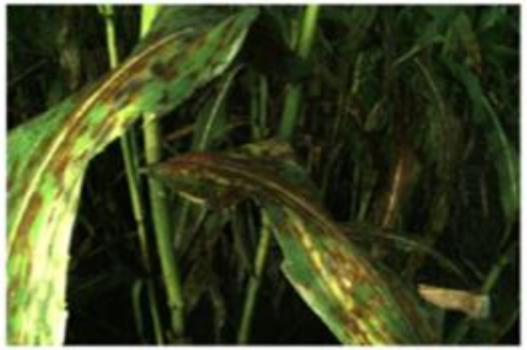

U.S. Average Corn Grain Yields, 1863-2002
Grain yield (bushels per acre)
160
140
120
100
80
60
40
20
0
1860
1880
1900
1920
1940
1960
1980
2000
2020
Year

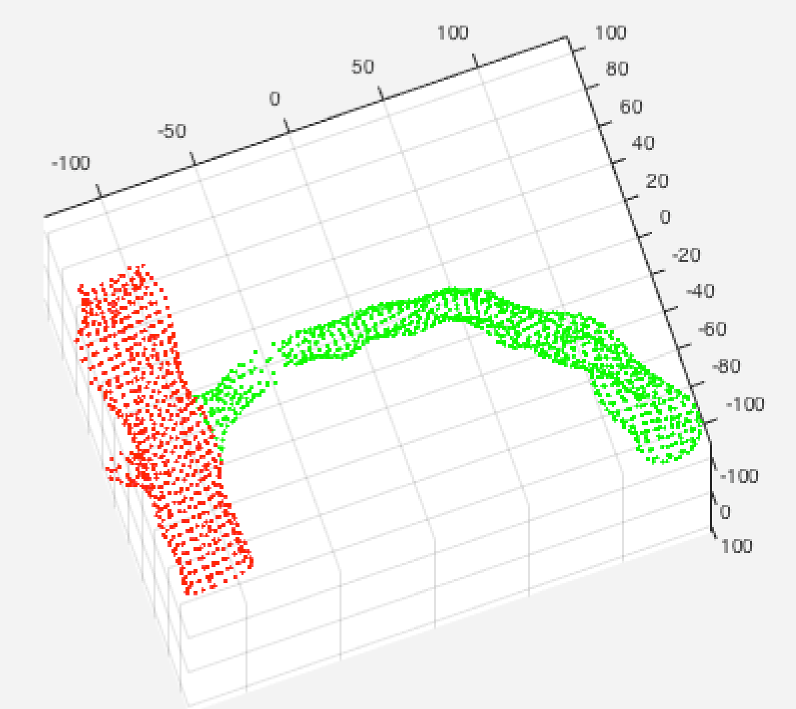
Surrogate robotic system in the field

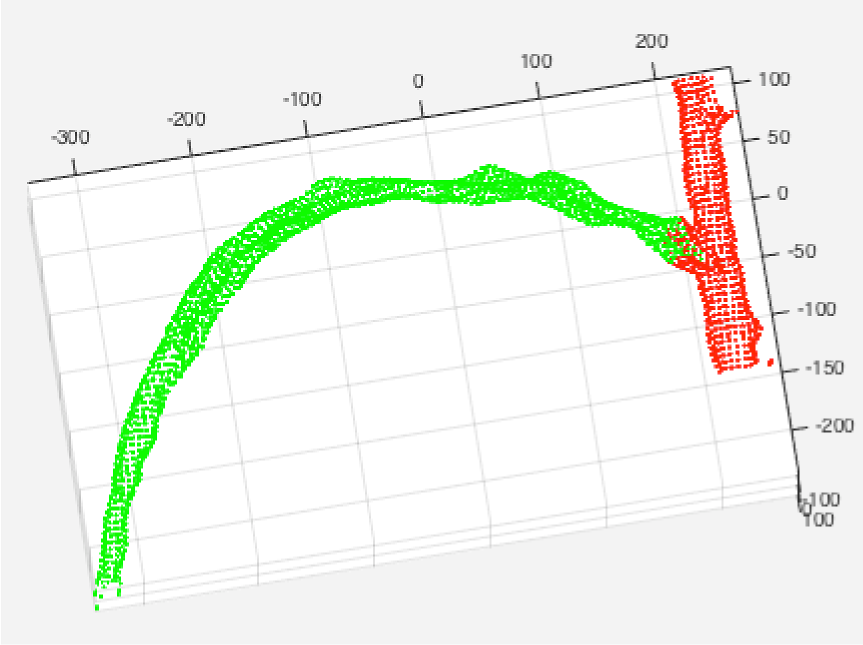








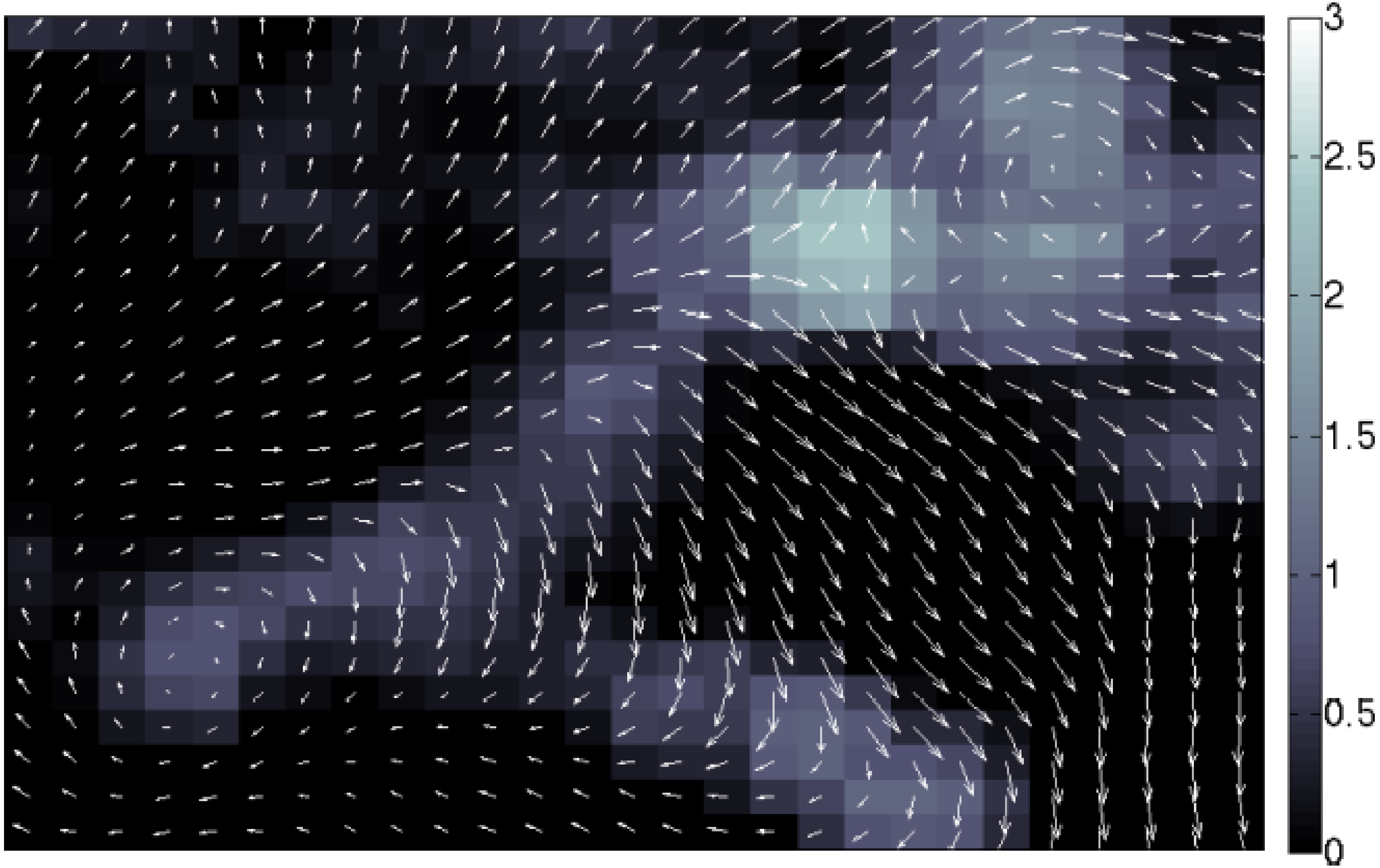




Name	**Range**	**RMSE error**
Leaf angle*	75.94	3.30 (4.35%)
Leaf radiation angle*	120.66	4.34 (3.60%)
Leaf length*	35.00	0.87 (2.49%)
Leaf width [max]	3.61	0.27 (7.48%)
Leaf width [average]	2.99	0.21 (7.02%)
Leaf area*	133.45	8.11 (6.08%)

Find Interesting Phenomena in Turbulence Data

Anomaly detection

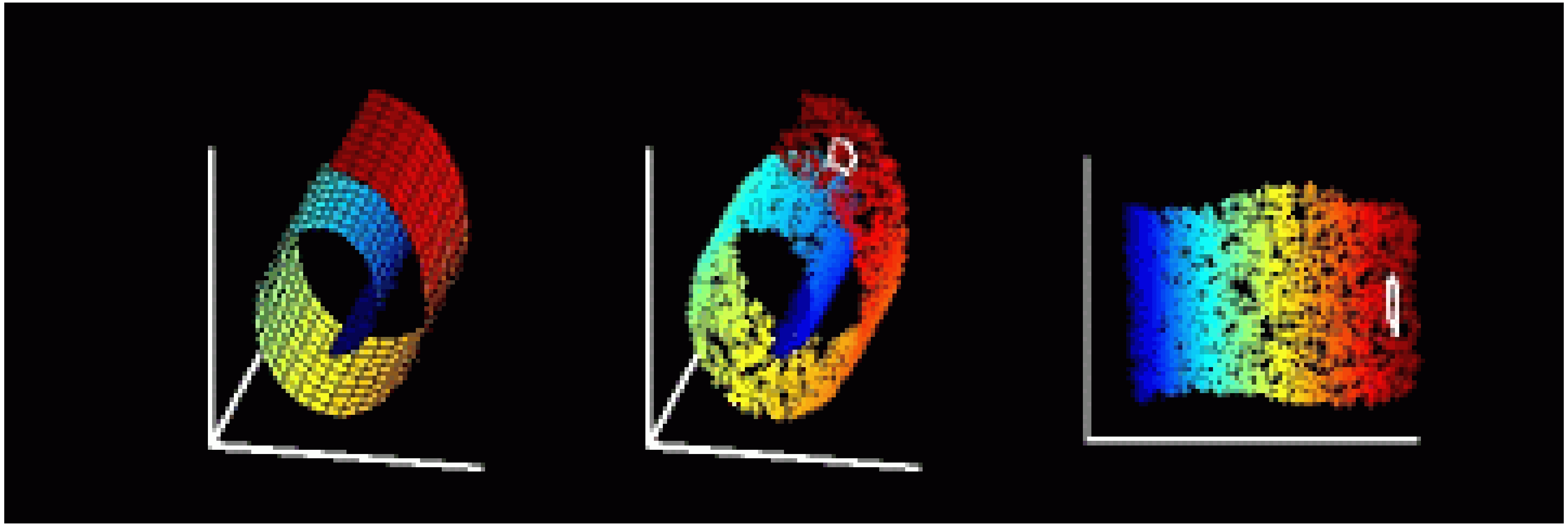


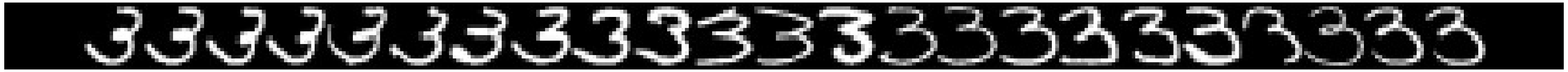
Anomaly scores

Find Interesting Phenomena in Turbulence Data

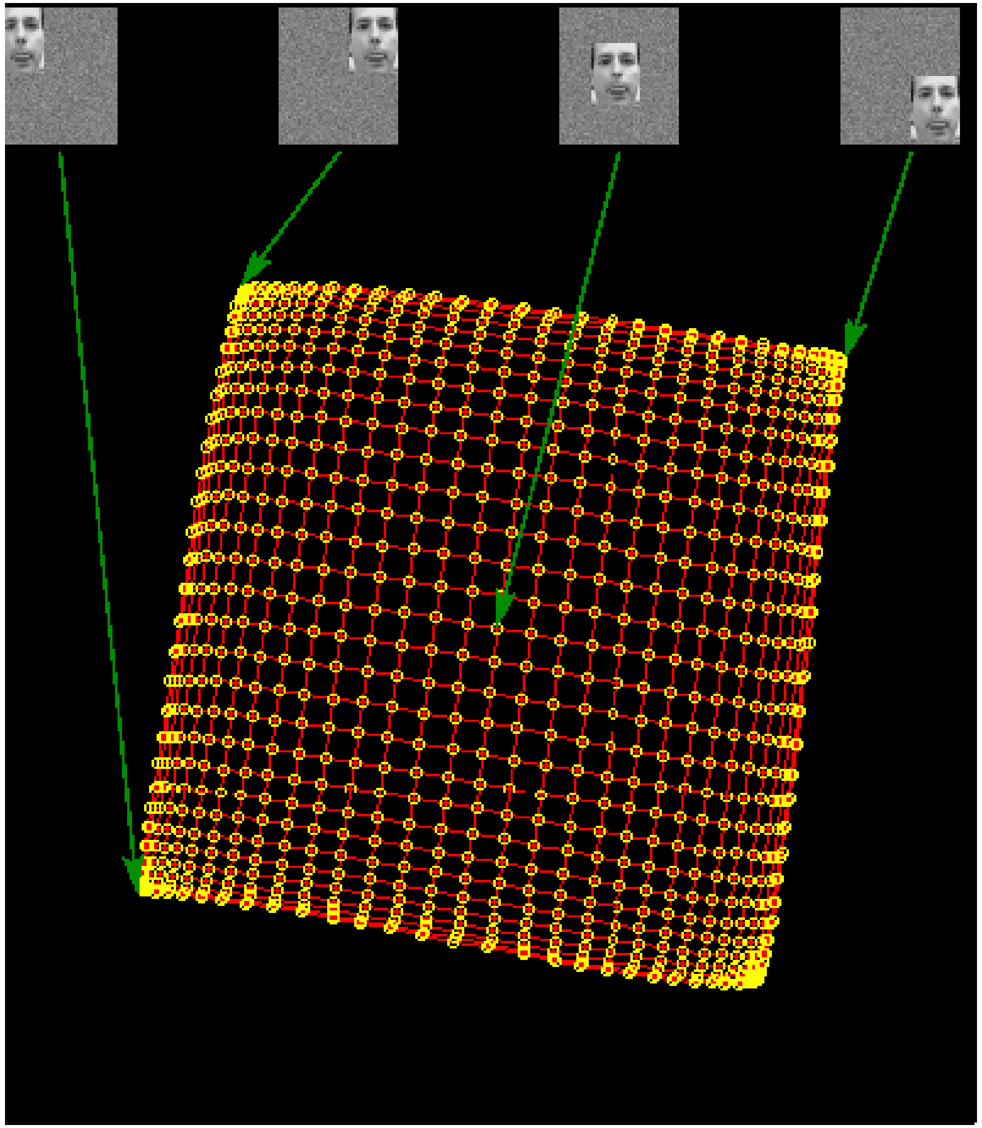
Locally Linear Embedding

Locally Linear Embedding

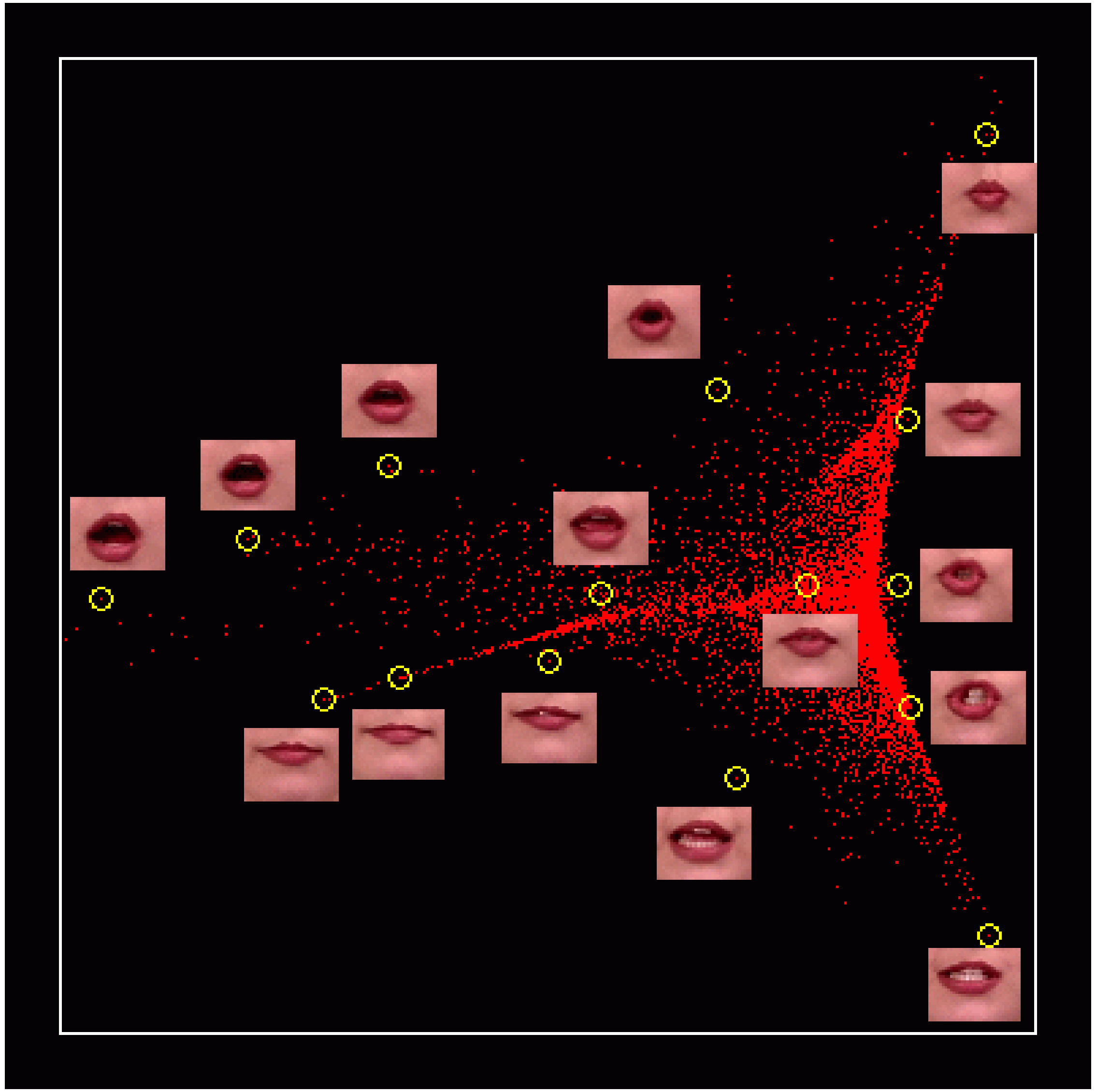




Locally Linear Embedding

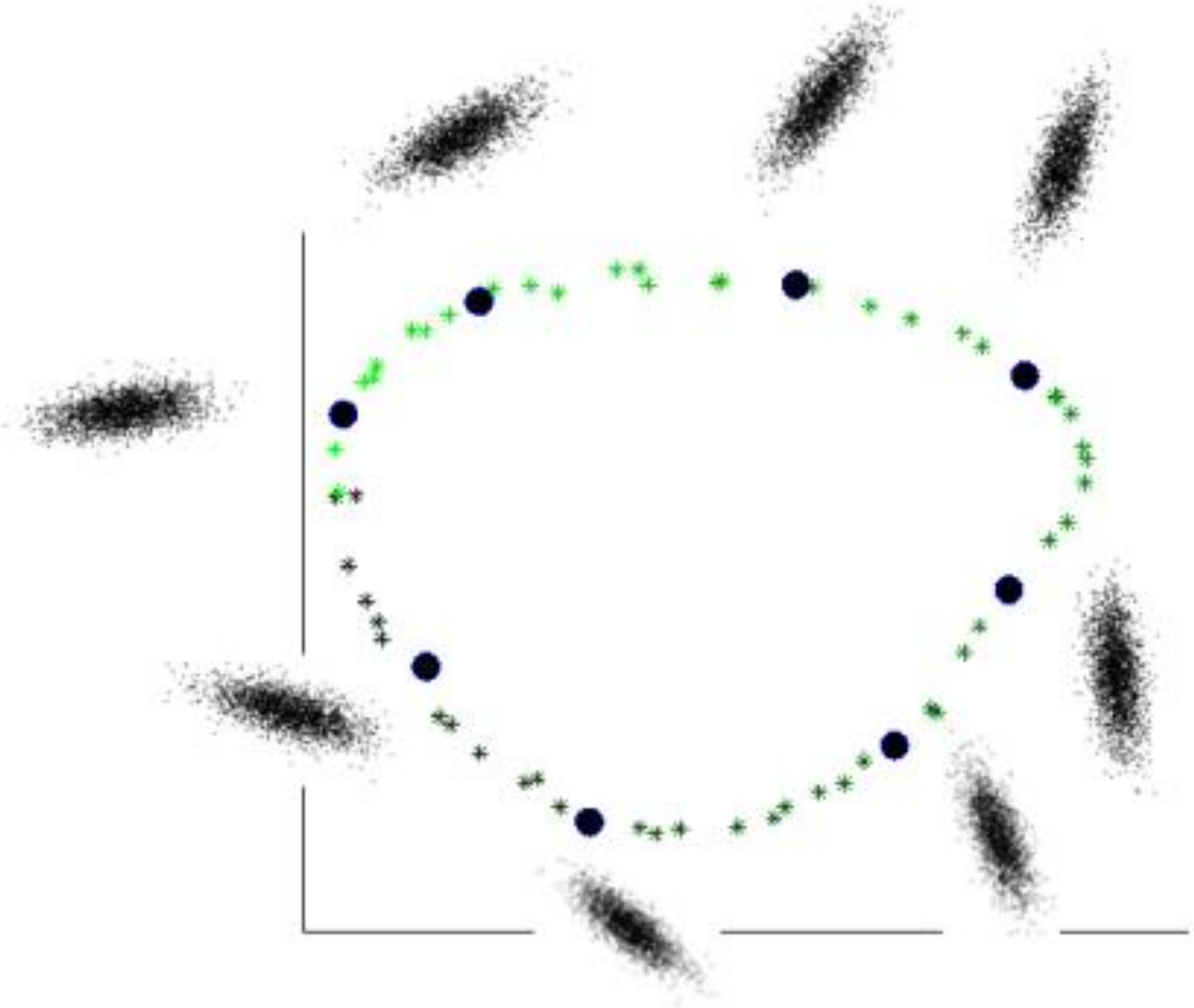


Locally Linear Embedding



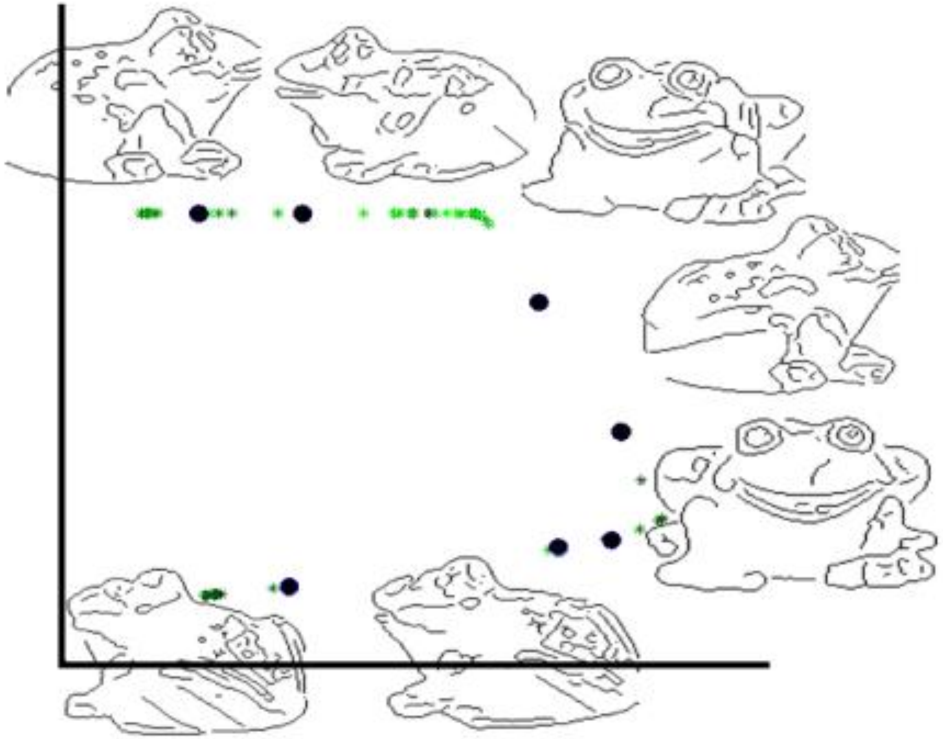
Local Linear Embedding of Sets and Distributions

Embedding rotated Gaussian distributions into 2D

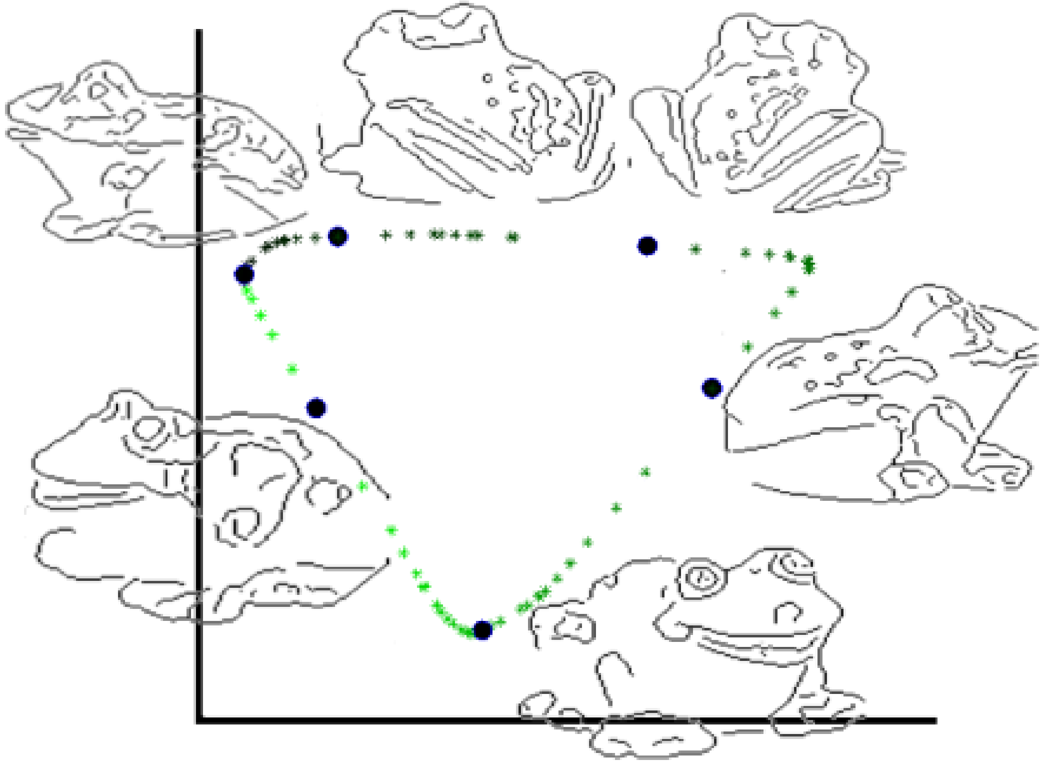


Local Linear Embedding of Sets and Distributions

Embedding rotated frog images into 2D



LLE with Euclidean distances fails

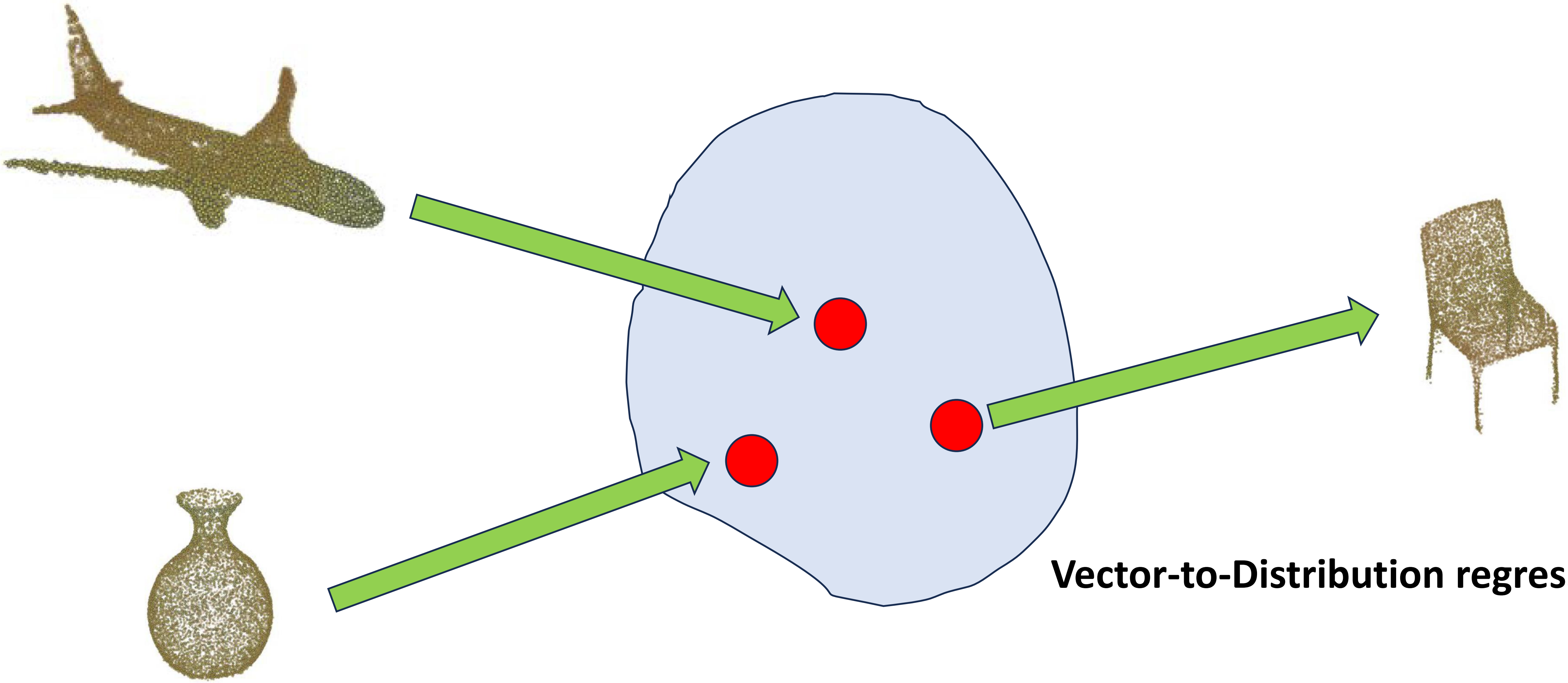


LLE with estimated Renyi divergence is successful

Take me Home!

**There are machine learning algorithms
that can operate
on sets/distributions as instances**

Generative Methods without Deep Learning



Vector-to-Distribution regression

Distribution embedding with LLE

Convergence Rate of GANs

Adversarial Losses

Our goal is to study minimax convergence rates
for **density estimation** under **adversarial losses**

Special cases of adversarial losses as distances between distributions:

- Kolmogorov-Smirnov distance
- Lp loss
- Maximum mean discrepancy (MMD)
- Energy distance
- Wasserstein distance

- Kantorivich – Rubinstein distance
- Total variation distance
- Sobolev distance
- Dudley metric
- Neural network distance
- …

We will study how

- choice of loss (encoded by the discriminator)
- smoothness of density (encoded by the generator)

affects the convergence rate of density estimation.

Adversarial Losses

Definition [Adversarial loss / Integral Probabilistic Metric]

$$d_{\mathcal{F}_d}(P,Q) \doteq \sup_{f \in \mathcal{F}_d} \left| \mathbb{E}_{X \sim P}\left[f(X)\right] - \mathbb{E}_{X \sim Q}\left[f(X)\right] \right|$$

$\mathcal{F}_d$: **Discriminator class**

- $\star$ Bounded Borel measurable functions $\{f : \mathcal{X} \to \mathbb{R}\}$

$\mathcal{F}_g$: **Generator class**

- $\star$ Borel probability measures on $\mathcal{X}$
- $\star$ $P, Q \in \mathcal{F}_g$

Problem Statement

- ⋆ Let $P \in \mathcal{F}_g$ be an unknown probability measure on $\mathcal{X}$.

  [P: the true distribution that we want to learn]

- ⋆ $X_{1:n} = X_1, \dots, X_n \overset{IID}{\sim} P$ observations. [Training data]

- ⋆ We are interested in constructing an estimator $\hat{P}(X_{1:n})$ where $\hat{P} : \mathcal{X}^n \to \mathcal{F}_g$

[GAN estimate]

$$\hat{P}_n^* \doteq \arg\min_{\hat{P}\in\mathcal{F}_g} \underbrace{\sup_{f\in\mathcal{F}_d} \left|\mathbb{E}_{X\sim P}\left[f(X)\right] - \mathbb{E}_{X\sim\hat{P}(X_1,\dots,X_n)}\left[f(X)\right]\right|}_{\text{Adversarial loss } d_{\mathcal{F}_d}(P,\hat{P})}$$

best optimized estimator in $\mathcal{F}_g$

Question: When does $\lim_{n\to\infty} d_{\mathcal{F}_d}(\hat{P}_n^*, P) = 0$?

Question: What is the rate of the convergence?

Notation

Let $\mathcal{Z} \subset \mathbb{Z}^d$ be a countable family [d-dim grid].

Let $\mathcal{B} \doteq \{\phi_z : \mathcal{X} \to \mathbb{R}, \sup_u \phi_z(u) < \infty, z \in \mathcal{Z}\}$ be an orthonormal basis in $\mathcal{L}^2$.

Let $\tilde{P}_z \doteq \mathbb{E}_{X\sim P}[\phi_z(X)] = \int_{\mathcal{X}} p(x)\phi_z(x)dx$ [The z^{th} coefficient of P (or p) in $\mathcal{B}$]

We say $\{a_z\}_{z\in\mathcal{Z}}$ is a real-valued net if $a_z \in \mathbb{R}$, $\forall z \in \mathcal{Z}$

Definition [Generalized Ellipse]:

$$\mathcal{H}_{p,a}(L) = \left\{ f \in \mathcal{L}^1(\mathcal{X}) \middle| \sum_{z\in\mathcal{Z}} \left(a_z^p |\tilde{f}_z|^p\right)^{1/p} \leq L \right\}$$

a real-valued net

$p \in [1, \infty]$

$\mathbb{E}_{X\sim f}[\phi_z(X)] = \int_{\mathcal{X}} f(x)\phi_z(x)dx$

The z^{th} coefficient of f in $\mathcal{B}$

Special case: Sobolev Space

Definition [Sobolev Ball]:

$$\mathcal{W}^{s,p}(L) = \left\{ f \in \mathcal{L}^1(\mathcal{X}) \,\middle|\, \sum_{z\in\mathcal{Z}} \left(|z|^{sp}\,|\tilde{f}_z|^p\right)^{1/p} \le L \right\}$$

$$a_z = \|z\|^s$$

$$p \in [1, \infty]$$

$$\mathbb{E}_{X\sim f}[\phi_z(X)] = \int_{\mathcal{X}} f(x)\phi_z(x)dx$$

The z^{th} coefficient of f in $\mathcal{B}$

For example, when $\mathcal{B}$ is the standard Fourier basis and s is an integer, for a constant factor c depending only on s and the dimension d:

$$\mathcal{W}^{s,p}(cL) = \left\{ f \in \mathcal{L}^p(\mathcal{X}) \,\middle|\, \|f^{(s)}\|_{\mathcal{L}^p} \le L \right\}$$

Consequences for GANs

Theorem:

Let $\epsilon > 0$ be a desired accuracy. Let $s, t > 0$.

Then there exists a GAN architecture, in which

$\star$ The discriminator $\mathcal{F}_d$ has at most $O(\log(1/\epsilon))$ layers.
and $O(\epsilon^{-d/s}\log(1/\epsilon))$ parameters.

$\star$ The generator $\mathcal{F}_g$ has at most $O(\log(1/\epsilon))$ layers.
and $O(\epsilon^{-d/t}\log(1/\epsilon))$ parameters.

such that if $\widehat{P}_*(X_{1:n}) \doteq \arg\min_{\widehat{P}\in\mathcal{F}_g} d_{\mathcal{F}_d}(\widehat{P}, P)$ is the optimized GAN estimate of P,

$$\Rightarrow \sup_{P\in\mathcal{W}^{t,2}} \mathbb{E}_{X_{1:n}}\left[d_{\mathcal{W}^{s,2}(P,\widehat{P}_*(X_{1:n}))}\right] \leq C\left(\epsilon + n^{-\min\{\frac{1}{2},\frac{s+t}{2t+d}\}}\right)$$

... and the GAN is consistent and minimax optimal!

Take me Home!

Under some conditions, GANs are consistent and their convergence rate is minimax optimal

Open Problems

- Statistical properties under less restrictive conditions?
- Results for convolutional neural nets?
- Best way for training GANs (i.e best way for solving the minmax optimization)?
- GANs on manifolds?
- Rare event generation?
- Generate uniform distribution on the support of the data?
- Maybe *minimax* rates are too pessimistic designed for the worst-case scenarios and we need to study different framework?
- Physics informed generative methods? Adding inductive bias to the generation?

- Similar question for diffusion based generative models …

Thanks for your Attention!