Iterative Normalizing Flows via Destructive Learning (from a biased viewpoint)

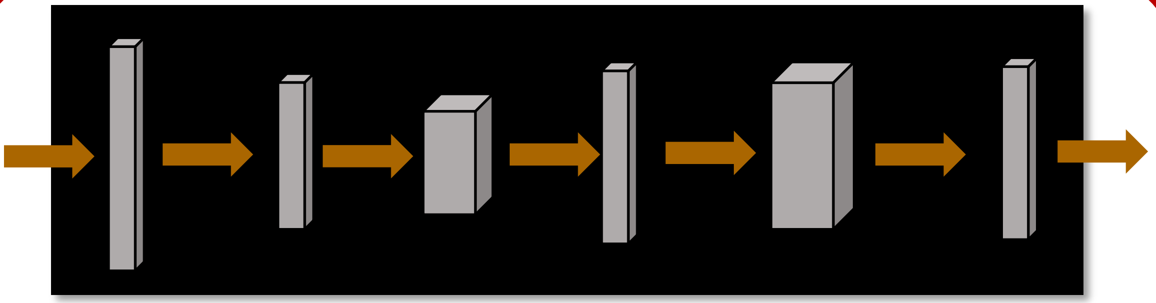
David I. Inouye

Electrical and Computer Engineering

Purdue University

Previous deep normalizing flows are trained end-to-end where all components are optimized simultaneously

End-to-end learning

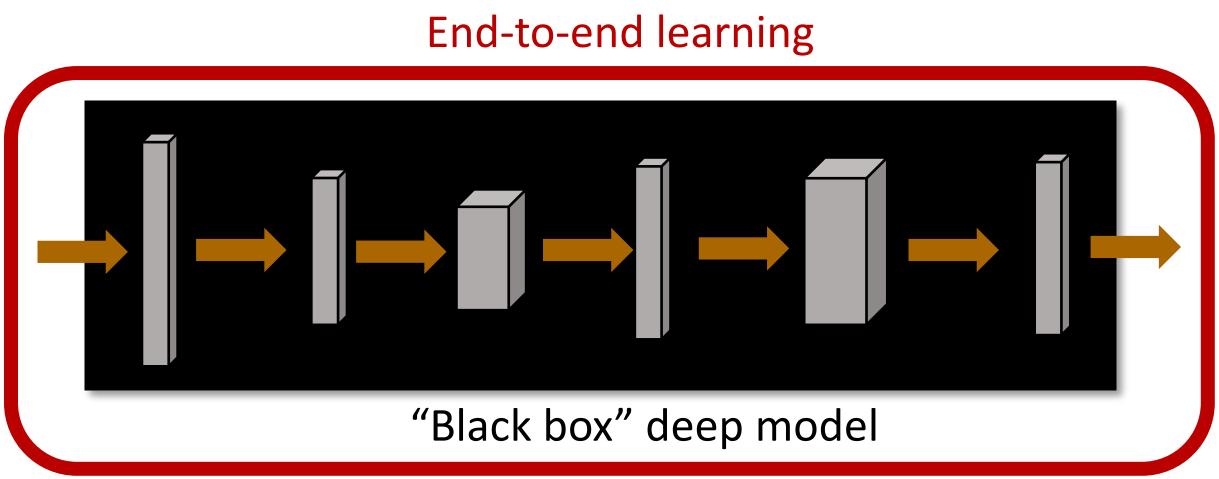


"Black box" deep model

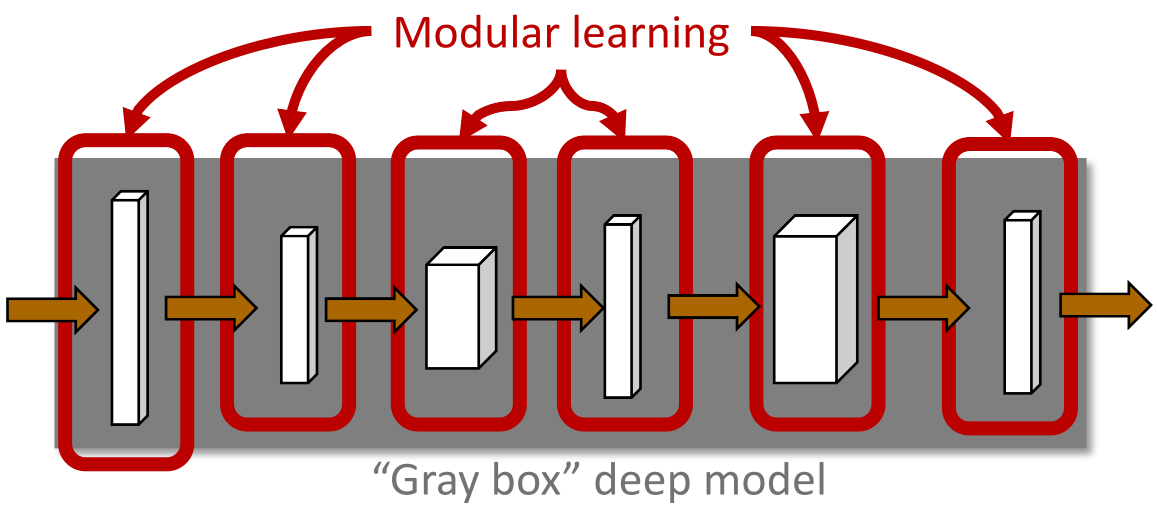
- Real NVP
- MAF
- GLOW
- Etc.

"Gray box" deep model

Modular deep learning would allow *local* learning within each component



- Real NVP
- MAF
- GLOW
- Etc.



- Density destructors
- Each weak/shallow learning algorithm is independent
- Learning algorithms could be heterogeneous (e.g., SGD and decision trees)

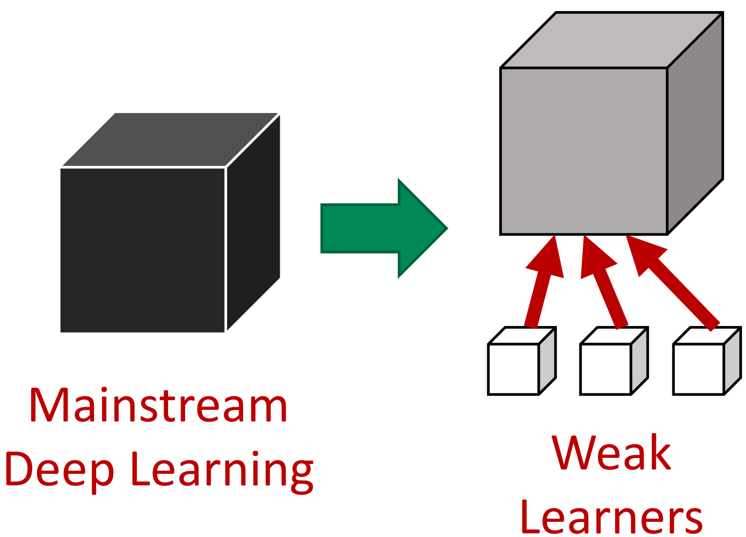
Why use modular weak learning for deep models?

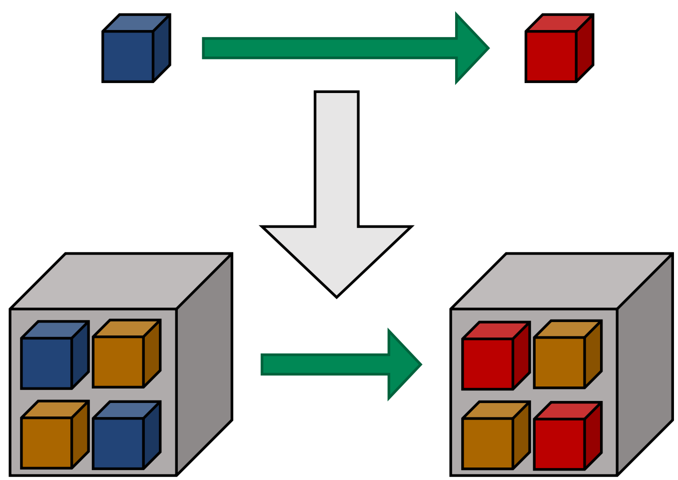
Reuse

The algorithms, insights and intuitions of shallow learning can be lifted into the deep context

Decoupling

Components can be debugged, tested and improved separate from the system

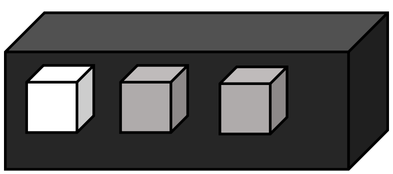


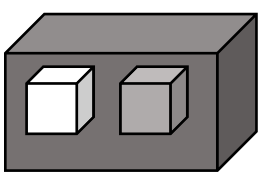


Why use modular weak learning for deep models?

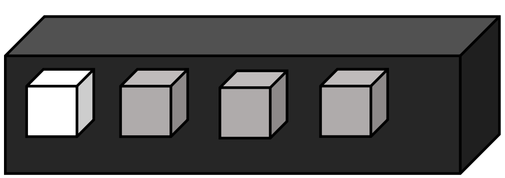
Algorithmic Interpretability

Increasing or decreasing model complexity is straightforward





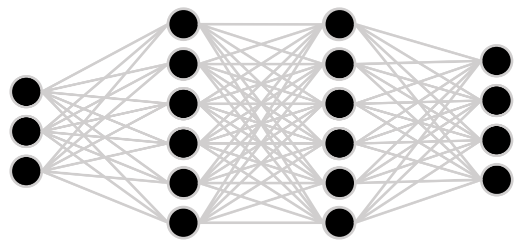
Shrink model if problem



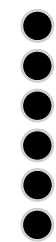
Grow if more data

Resource Constraints

Layer-wise training (memory bottleneck)

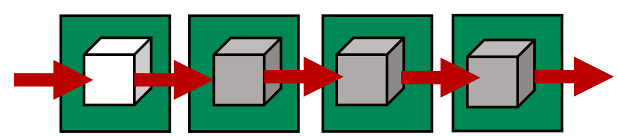


versus



Pipelined training (computation bottleneck)

Shallow/weak online learners



Distributed on different processors or devices

Destructive learning enables modular deep learning via “reverse engineering” **data**

Reverse engineering phone

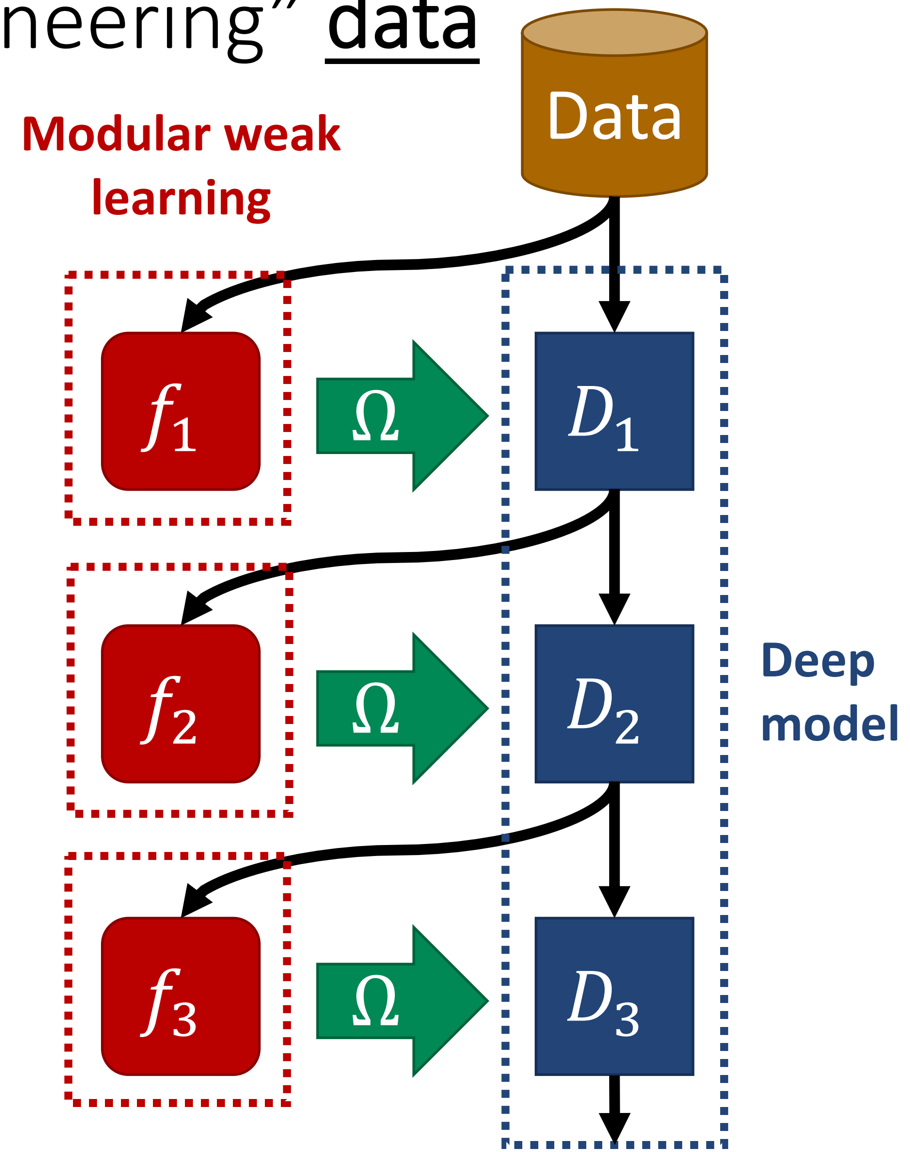
1. **Find** *part* to take off using **understanding and expertise**
2. **Determine how** to take off part in a *reversible* way (e.g., unscrewing bolts)
3. **Remove part**
4. **Repeat**

Reverse engineering data

1. **Find** *patterns* in data via **shallow/weak learning**
2. **Map model** to destructive but *invertible* transformation
3. **Destroy the patterns** via transformation
4. **Repeat**

Destructive learning enables modular deep learning via "reverse engineering" <u>**data**</u>

1. **Find** patterns in data via **shallow/weak learning**
2. **Map model** to destructive transformation
3. **Destroy the patterns** via transformation
4. **Repeat**



Overview of iterative destructive learning

Motivation and intuition for modular destructive learning

Density destructors objective function

Modular and greedy deep destructive algorithm

- Simple density destructors
- Deep density destructors
- Theory: Monotonic decrease of objective

Results, Limitations and Open problems

Background for objective: KL equivalence lemma

- KL equivalence lemma: If $z = D(x)$ for invertible D, then
$$KL\big(P_x(x), Q_x(x)\big) = KL\big(P_z(z), Q_z(z)\big)$$
- $KL\big(P_x(x), Q_x(x)\big)$
- $= E_{P_x}\left[\log \frac{P_x(x)}{Q_x(x)}\right]$
- $= E_{P_x}\left[\log \frac{P_z(D(x))|J_D(x)|}{Q_z(D(x))|J_D(x)|}\right]$ (Change of variables formula)
- $= E_{P_x}\left[\log \frac{P_z(D(x))}{Q_z(D(x))}\right]$
- $= E_{P_z}\left[\log \frac{P_z\left(D\left(D^{-1}(z)\right)\right)}{Q_z\left(D\left(D^{-1}(z)\right)\right)}\right]$ (Expectation change of variables LOTUS)
- $= E_{P_z}\left[\log \frac{P_z(z)}{Q_z(z)}\right]$
- $= KL\big(P_z(z), Q_z(z)\big)$

Destructive learning objective is equivalent to MLE

- The destructive learning objective, where $z = D(x)$, and $U_z(z)$ is the uniform density function
$$\arg\min_D KL\left(P_z(z; D), U(z)\right)$$
- Simple corollary is that objective above is MLE:
- $KL\left(P_z(z; D), U(z)\right)$
- $= KL\left(P_x(x), Q_x(x; D)\right)$ (KL equivalence, MLE objective)
- $= KL\left(P_x(x), |J_D(x)| U\left(D(x)\right)\right)$ (In terms of D)
- $= KL(P_x(x), |J_D(x)|)$ $(U(z) = 1)$

Overview of iterative destructive learning

Motivation and intuition for modular destructive learning

Density destructors objective function

Modular and greedy deep destructive algorithm

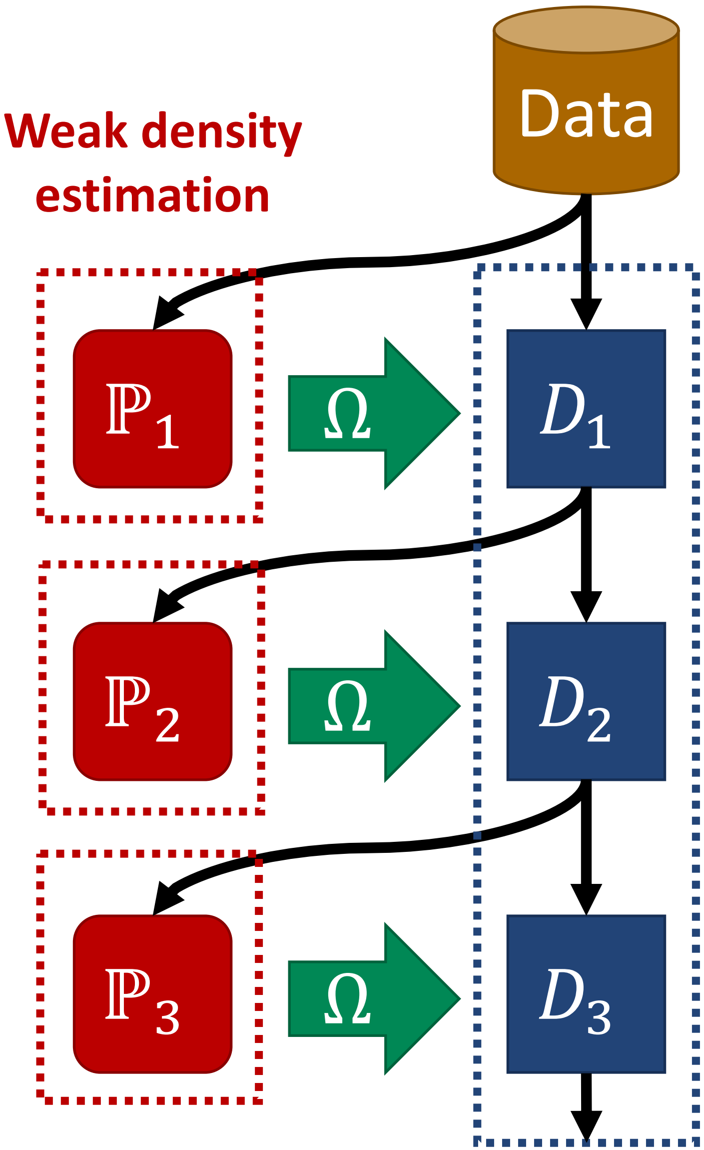
- Simple density destructors
- Deep density destructors
- Theory: Monotonic decrease of objective

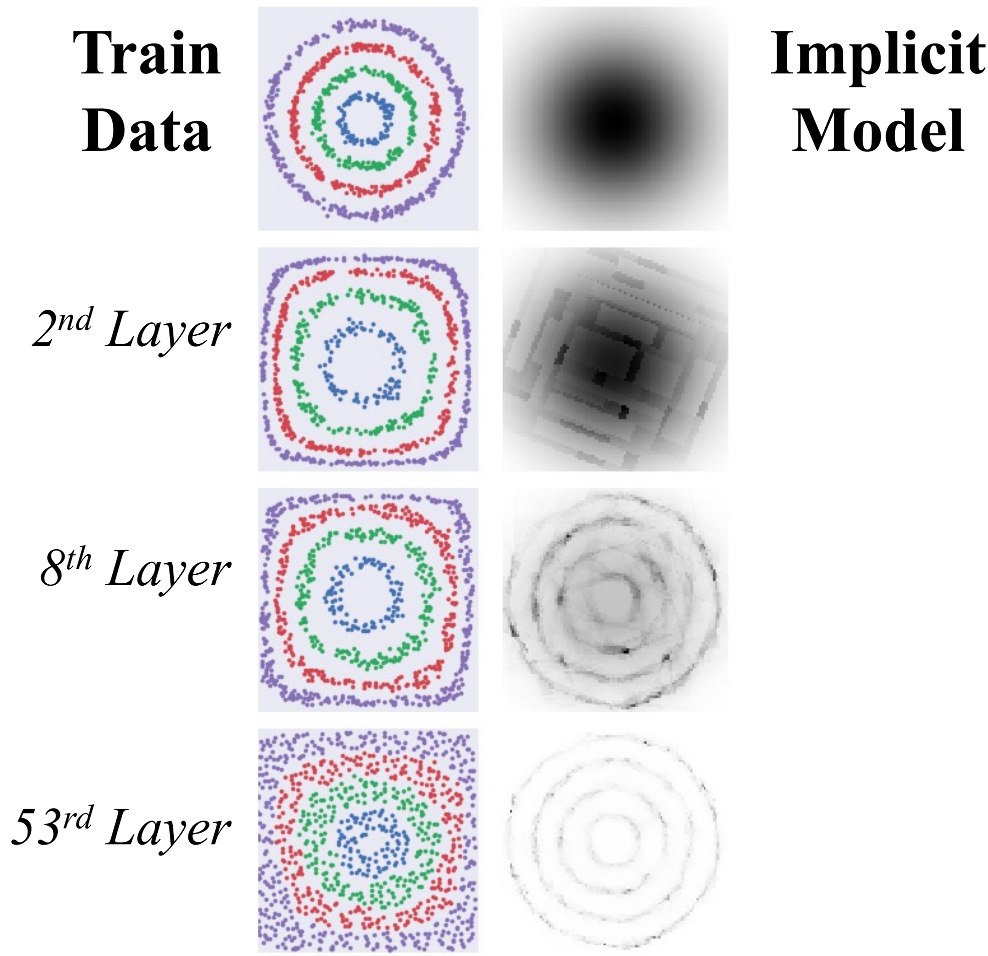
Results, Limitations and Open problems

Density destructor algorithm performs greedy layer-wise construction of deep destructor

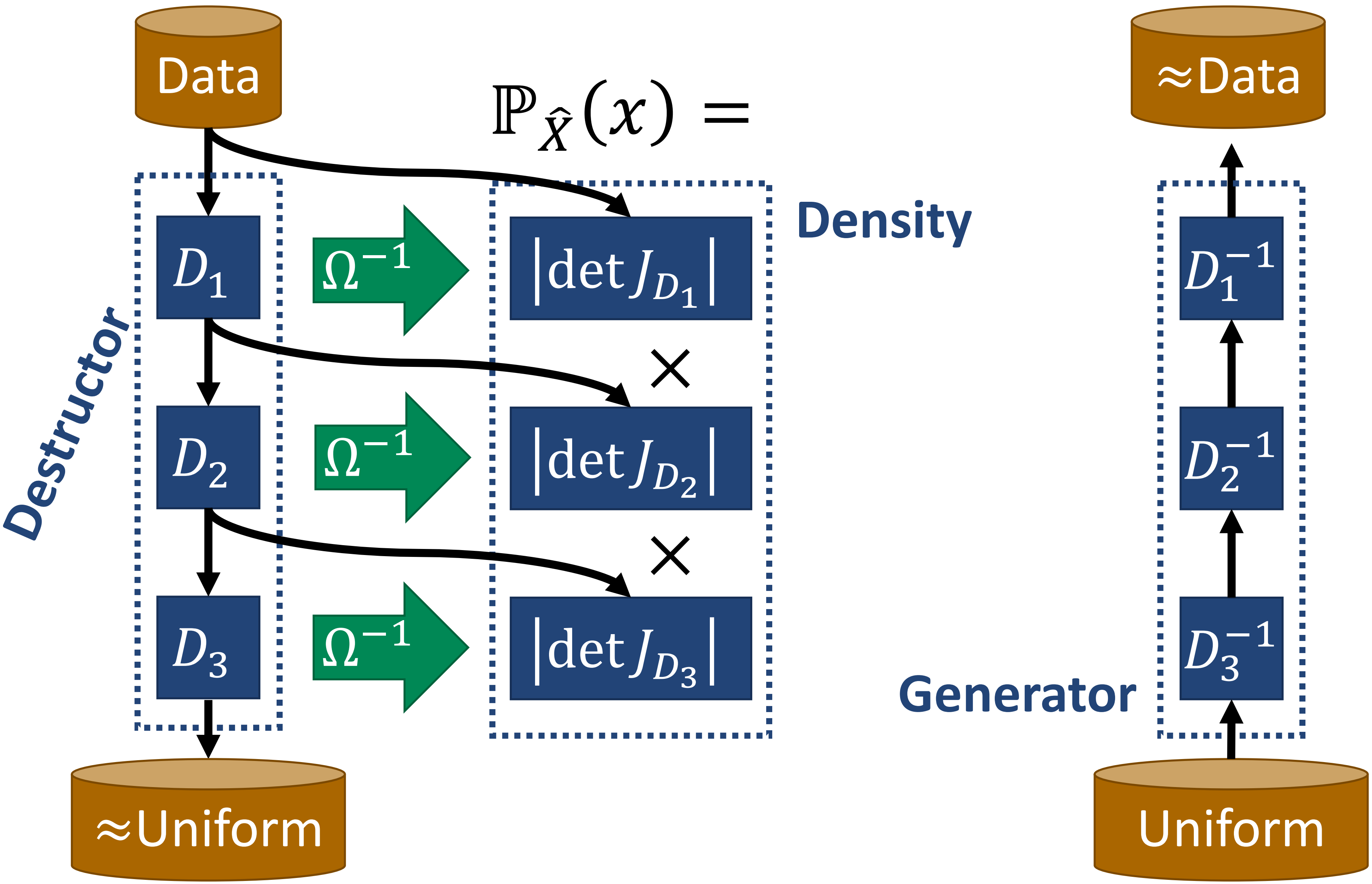
1. Simple density estimation (GMM, Gaussian, tree density, etc.)
   $$Q^t \leftarrow \arg\min_{Q \in \mathcal{Q}} KL\big(P(x^{t-1}), Q(x^{t-1})\big)$$
2. Map density to simple destructor layer
   $$d^t = \Omega(Q^t)$$
3. Transform data for next layer
   $$x^t = d^t(x^{t-1})$$
4. Update deep destructor
   $$D^t = d^t \circ D^{t-1}$$

Algorithm: ***Deep*** density destructors via sequence of *weak* destructors



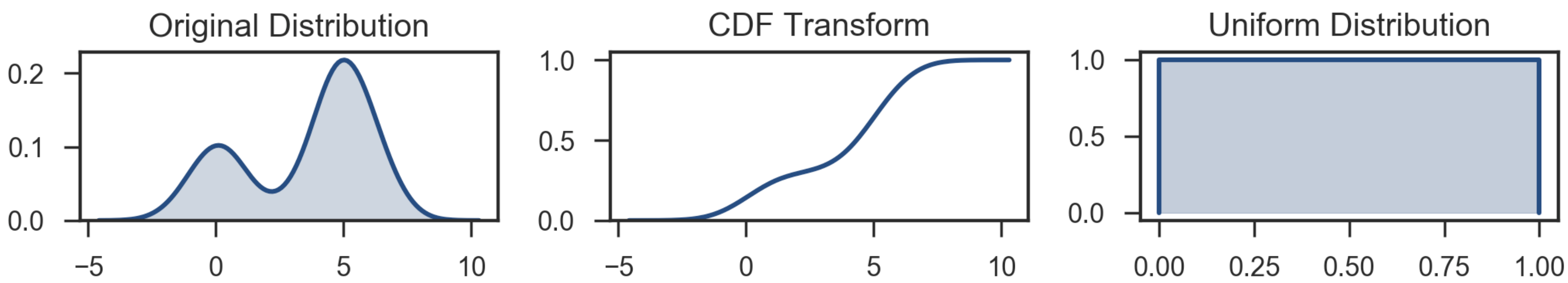


Density computation and sample generation



Definition: *Density destructors* generalize the univariate CDF transformation

- Univariate: CDF transformation

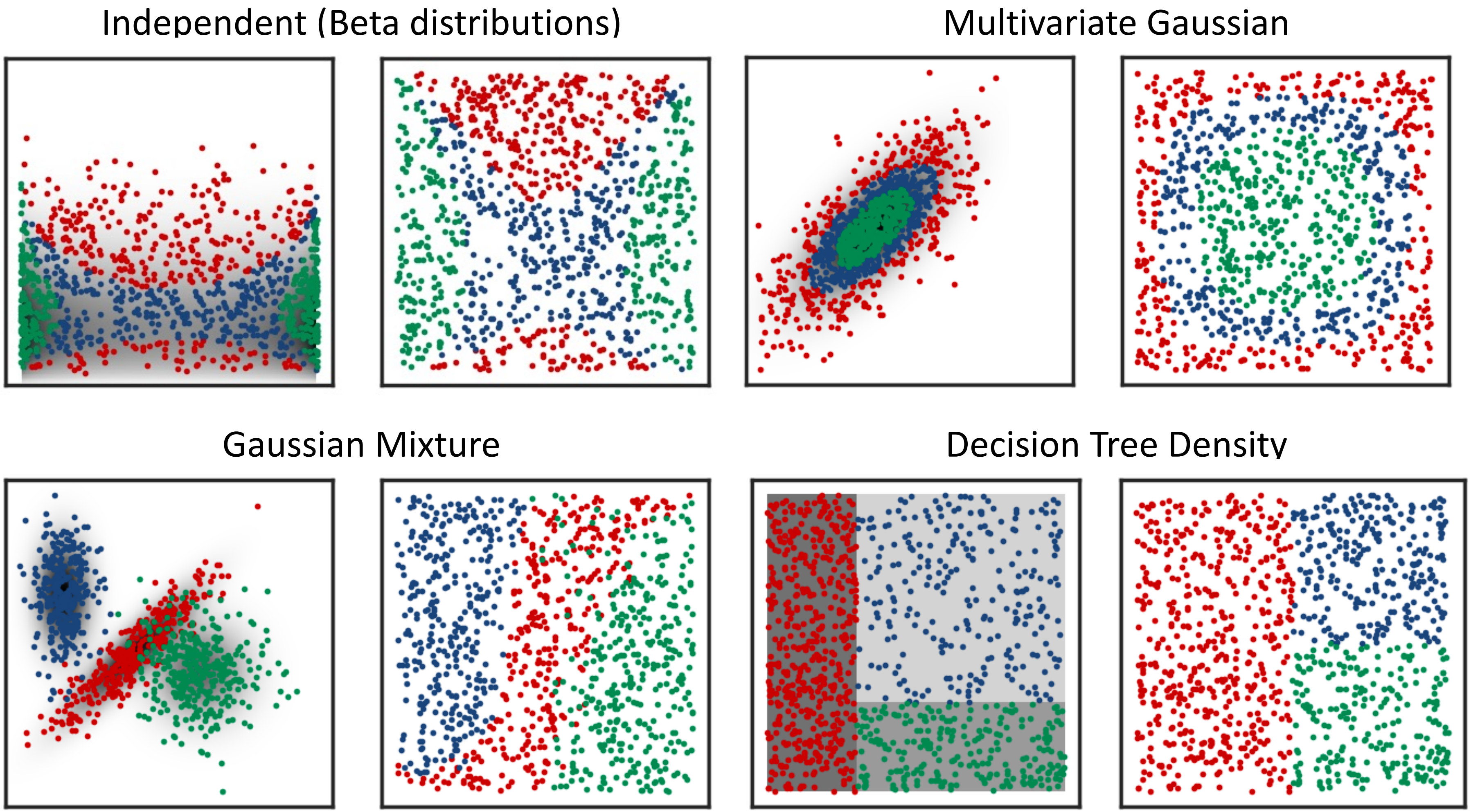


- The map $\Omega(\mathbb{P}) = D$ should:
  1. Encode the density $\mathbb{P}$ into D, i.e. $\exists\ \Omega^{-1}$.
  2. Ensure D destroys all patterns in $\mathbb{P}$ when applied to the random variable, i.e. the distribution of $\mathrm{D_X}(X)$ is maximum entropy.
- A *density destructor* is an invertible transformation such that

$$X \sim \mathbb{P}_X$$
$$D_X(X) \sim \mathrm{Uniform}\left([0,1]^d\right)$$

  - $\Omega^{-1}(D_X) = \left|\det J_{D_X}\right| = \mathbb{P}_X$ ← Closed-form density!
  - Different from multivariate CDF function: $F(x)\colon \mathbb{R}^d \to [0,1]$

Many shallow densities can be mapped to destructors

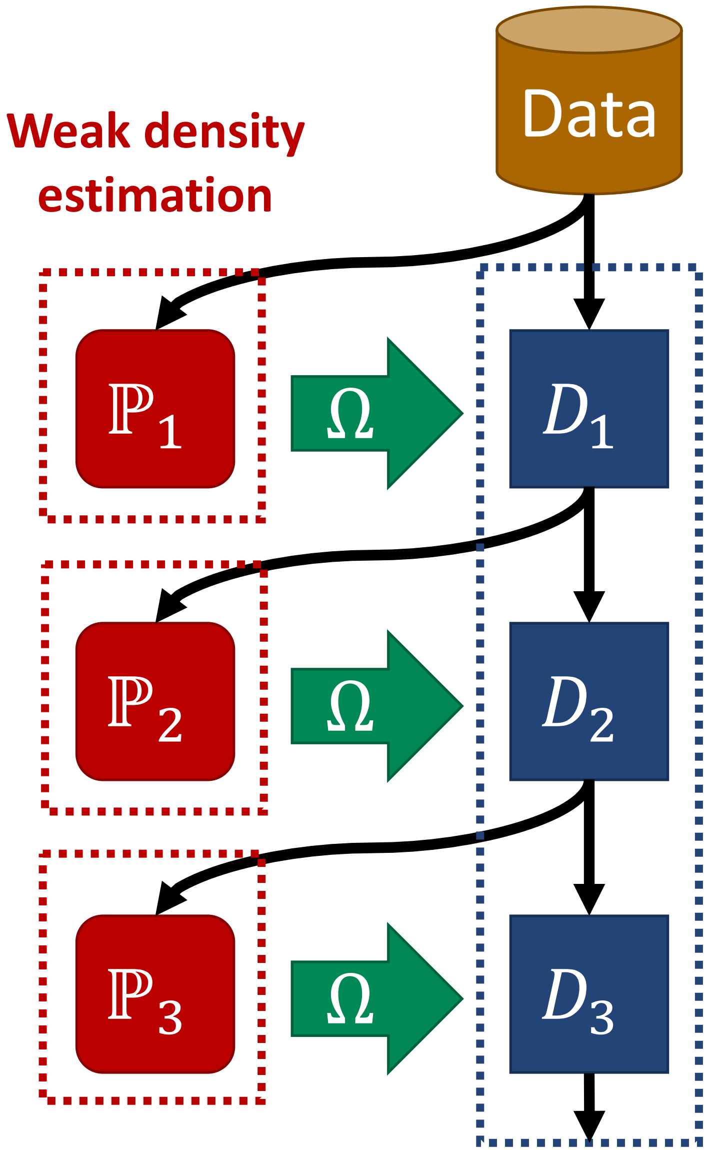


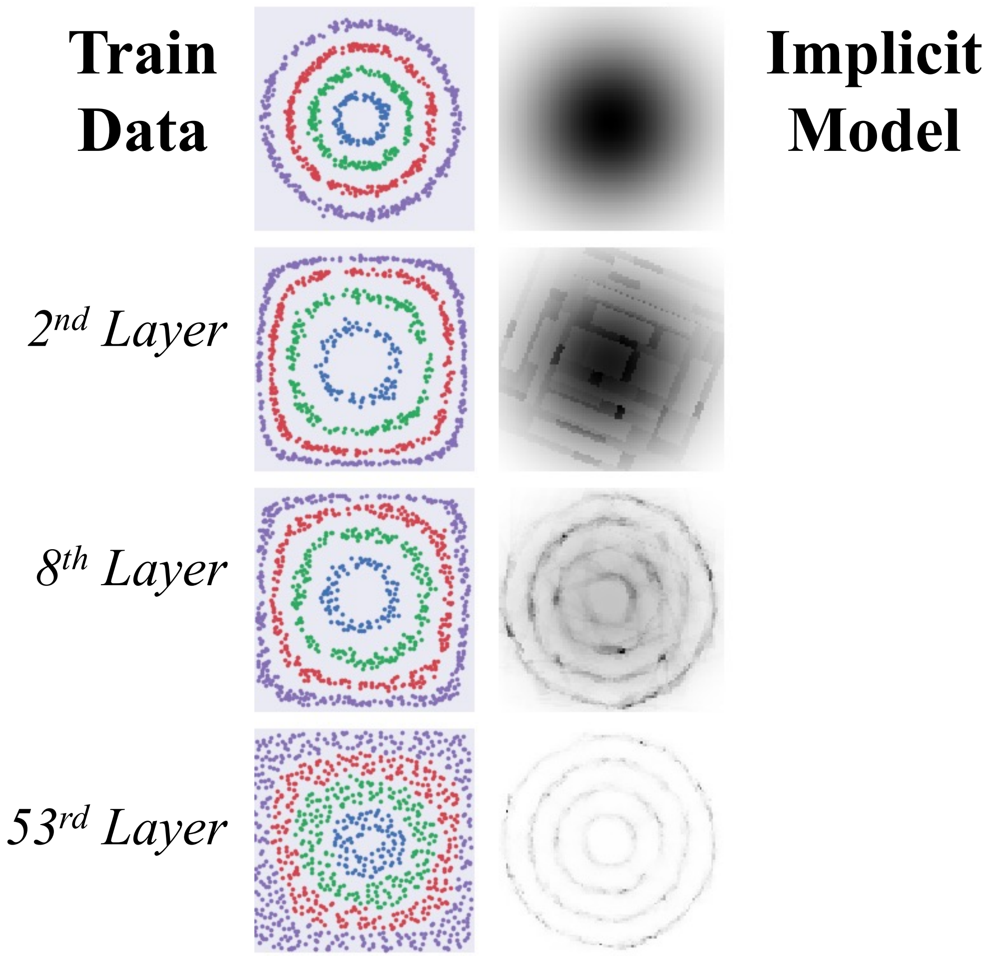
Data before (left) and after (right) transformation via corresponding density destructor.
Note: Color is just to show correspondence between areas before and after transformation.

Density destructor algorithm performs greedy layer-wise construction of deep destructor

1. Simple density estimation (GMM, Gaussian, tree density, etc.)
   $$Q^t \leftarrow \arg\min_{Q \in \mathcal{Q}} KL\left(P(x^{t-1}), Q(x^{t-1})\right)$$
2. Map density to simple destructor layer
   $$d^t = \Omega(Q^t)$$
3. Transform data for next layer
   $$x^t = d^t(x^{t-1})$$
4. Update deep destructor
   $$D^t = d^t \circ D^{t-1}$$

Deep density destructors via sequence of *weak* destructors





Overview of iterative destructive learning

Motivation and intuition for modular destructive learning

Density destructors objective function

Modular and greedy deep destructive algorithm

- Simple density destructors
- Deep density destructors
- Theory: Monotonic decrease of objective

Results, Limitations and Open problems

Destructor algorithm can be shown to monotonically decrease the negative log likelihood after every iteration/layer

- The destructive learning objective, where $z = D(x)$, and $U(z)$ is the uniform density function
$$\arg\min_{D} KL\big(P_z(z), U(z)\big)$$
- Want: Every iteration decreases objective:
$$KL\big(P_{d^t \circ D^{t-1}(x)}, U\big) \le KL\big(P_{D^{t-1}(x)}, U\big)$$
- Let $x = D^{t-1}\big(x^{(0)}\big)$ and $z = d^t(x) = d^t\left(D^{t-1}\big(x^{(0)}\big)\right)$
- $KL\big(P_z, (z; d^t) U(z)\big)$
- $= KL\big(P_x(x), Q_x(x; d^T)\big)$
(KL equivalence lemma)
- $\le KL\big(P_x(x), Q_x(x; d^T = Id)\big)$
(minimization is better than one particular)
- $= KL\left(P_x(x), \left|J_{d^t}(x)\right| U\big(d^t(x)\big)\right)$ (Expand in terms of *D*)
- $= KL\big(P_x(x), U(x)\big)$

Overview of iterative destructive learning

Motivation and intuition for modular destructive learning

Density destructors objective function

Modular and greedy deep destructive algorithm

- Simple density destructors
- Deep density destructors
- Theory: Monotonic decrease of objective

Results, Limitations and Open problems

Reuse results: Deep density destructors can be built from simple and well-understood components

- MNIST $d = 784$
- CIFAR-10 $d = 3072$

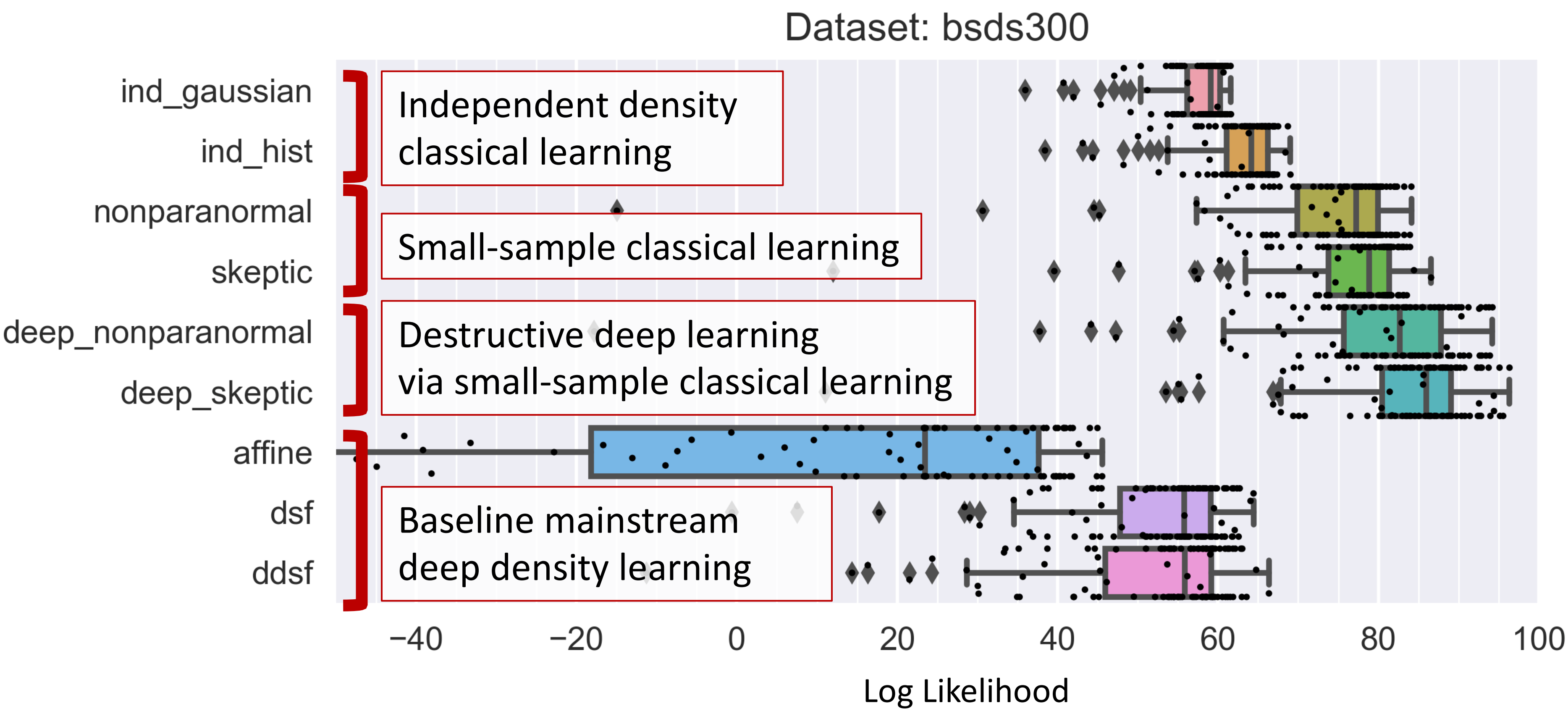
- Autoregressive flow baselines (DNN-based)
  - MADE [Germain et al., 2015]
  - Real NVP [Dinh, et al. 2017]
  - MAF [Papamakarios et al. 2017]

- Our deep copula method
  - PCA + histograms

	MNIST			CIFAR-10		
	LL	D	T	LL	D	T
Models from MAF paper computed on Titan X GPU						
Gaussian	-1367	1	0.0	2367	1	0.0
MADE	-1385	1	0.0	448	1	0.2
MADE MoG	-1042	1	0.1	-53	1	0.3
Real NVP	-1329	5	0.2	2600	5	1.4
Real NVP	-1765	10	0.2	2469	10	1.0
MAF	-1300	5	0.1	2941	5	3.7
MAF	-1314	10	0.2	**3054**	10	7.5
MAF MoG	-1100	5	0.2	2822	5	3.9
Our proposed destructors computed on 10 CPUs						
Copula	-1028	5	0.2	2626	17	10.1

LL = Log Likelihood (higher is better)
D = # of layers, T = Time

Modularity enables classical learning improvements to carry over to deep learning



Small-sample experiment where number of dimensions is 63 and number of training samples is 30. Notice how mainstream deep learning fails in this setting.

Limitations of destructive modular learning

- Unlikely to perform as well as joint learning
  - Greedy vs joint optimization
  - Local vs global optimization

- Must create destructor mapping Ω, which can be challenging

- Often requires more layers to achieve similar result because of optimization

- Normalizing flows transform to simple known distribution
  - What about transforming between any two distributions?