Deep Density Destructors
(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.

"Black box" deep model

End-to-end learning

- Real NVP
- MAF
- GLOW
- Etc.

"Gray box" deep model
Modular deep learning would allow *local* learning within each component.

End-to-end learning

- Real NVP
- MAF
- GLOW
- Etc.

“Black box” deep model

Modular learning

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

“Gray box” deep model
**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” data

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

Mainstream Deep Learning

Weak Learners
Why use modular weak learning for deep models?

Algorithmic Interpretability

Increasing or decreasing model complexity is straightforward

Resource Constraints

Layer-wise training (memory bottleneck)

Pipelined training (computation bottleneck)

Shrink model if problem

Grow if more data

versus

Distributed on different processors or devices
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 $\Omega$, which can be challenging

- Often requires more layers to achieve similar result because of optimization
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 $D_X(X)$ is maximum entropy.
- A *density destructor* is an invertible transformation such that
  \[
  X \sim \mathbb{P}_X \quad \quad \quad \quad D_X(X) \sim \text{Uniform}([0, 1]^d)
  \]
  - $\Omega^{-1}(D_X) = |\det J_{D_X}| = \mathbb{P}_X \quad \leftarrow \text{Closed-form density!}$
  - Different from multivariate CDF function: $F(x): \mathbb{R}^d \to [0,1]$
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.
# Example Destructors

<table>
<thead>
<tr>
<th>Description</th>
<th>Density</th>
<th>Transformation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Autoregressive Density</strong></td>
<td>$\prod_{s=1}^{d} \mathbb{P}(x_s \mid x_{1:s-1})$</td>
<td>$[F_1(x_1), F_2(x_2 \mid x_1), \ldots, F_d(x_d \mid x_{1:s-1})]$</td>
</tr>
<tr>
<td>Mixture of Gaussians Conditionals (e.g. MADE, MAF)</td>
<td>$\prod_{s=1}^{d} \left[ \sum_{t=1}^{m} \pi_t(x_{1:s-1}) \times \mathbb{P}<em>{\mathcal{N}}(x_s \mid \mu</em>{st}(x_{1:s-1}), \sigma^2_{st}(x_{1:s-1})) \right]$</td>
<td>$[F_1(x_1), F_2(x_2 \mid x_1), \ldots, F_d(x_d \mid x_1, \ldots, x_{s-1})]$</td>
</tr>
<tr>
<td>Block Gaussian Conditionals (e.g. Real NVP, NICE)</td>
<td>$\mathbb{P}<em>{\mathcal{N}}(x</em>{1:t} \mid 0, I) \times \mathbb{P}<em>{\mathcal{N}}(x</em>{t+1:d} \mid \mu(x_{1:t}), \sigma^2(x_{1:t}))$</td>
<td>$[\Phi(x_{1:t}), \Phi\left(\frac{x_{t+1} - \mu_{t+1}(x_{1:t})}{\sigma_{t+1}(x_{1:t})}\right), \ldots, \Phi\left(\frac{x_{d} - \mu_{d}(x_{1:t})}{\sigma_{d}(x_{1:t})}\right)]$</td>
</tr>
<tr>
<td><strong>Linear Projection Density</strong></td>
<td>$\mathbb{P}_{\phi}(W x)$</td>
<td>$D_\theta(W x)$</td>
</tr>
<tr>
<td>Independent Components (e.g. Gaussianization via ICA)</td>
<td>$\prod_{s=1}^{d} \mathbb{P}(w^T_s x)$</td>
<td>$F(W x)$</td>
</tr>
<tr>
<td>Gaussian (e.g. via PCA)</td>
<td>$\mathbb{P}_{\mathcal{N}}(x \mid \mu, \Sigma)$</td>
<td>$\Phi(\Sigma^{-1/2}(x - \mu))$</td>
</tr>
<tr>
<td><strong>Copula-based Density</strong></td>
<td>$\mathbb{P}<em>{\mathcal{N}}^{\cop}(F(x)) \prod</em>{s=1}^{d} \mathbb{P}(x_s)$</td>
<td>$D_\theta(F(x))$</td>
</tr>
<tr>
<td>Gaussian Copula</td>
<td>$\mathbb{P}<em>{\mathcal{N}}^{\cop}(F(x)) \prod</em>{s=1}^{d} \mathbb{P}(x_s)$</td>
<td>$\Phi(R_{1/2}^{-1}(F(x)))$</td>
</tr>
<tr>
<td><strong>Gaussian Mixture</strong> (note that $F_s(x_s \mid x_{-s})$ is computable)</td>
<td>$\sum_{t=1}^{m} \pi_t \mathbb{P}_{\mathcal{N}}(x)$</td>
<td>$[F_1(x_1), F_2(x_2 \mid x_1), \ldots, F_d(x_d \mid x_1, \ldots, x_{s-1})]$</td>
</tr>
</tbody>
</table>

Examples of new destructors enabled by our unified destructor framework

<table>
<thead>
<tr>
<th>Description</th>
<th>Transformation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Piecewise Density (or Tree Density)</strong></td>
<td>${D_{\theta_t}(x), \text{ if } x \in \mathcal{L}_t}$, where $\mathcal{L}_t$ are the disjoint subspaces of the leaves.</td>
</tr>
<tr>
<td><strong>Piecewise Uniform (e.g. DET)</strong></td>
<td>$c_{\ell}, \text{ if } x \in \mathcal{L}_\ell$</td>
</tr>
<tr>
<td><strong>Image-Specific Feature Pairs</strong></td>
<td>${\text{Diag}(a_{\ell})x + b_{\ell}, \text{ if } x \in \mathcal{L}_\ell}$</td>
</tr>
<tr>
<td><strong>Image-Specific Feature Pairs</strong></td>
<td>$\prod_{P \in \mathcal{P}} \mathbb{P}<em>{\mathcal{P}}(x</em>{P(1)}, x_{P(2)})$, where feature pairs $\mathcal{P}$ are based on pixel locality.</td>
</tr>
</tbody>
</table>
Deep density destructors via sequence of weak destructors

Weak density estimation

Data

$D_1$

$P_1$

$\Omega$

$D_2$

$P_2$

$\Omega$

$D_3$

$P_3$

Train Data

2\textsuperscript{nd} Layer

8\textsuperscript{th} Layer

53\textsuperscript{rd} Layer

Implicit Model

David I. Inouye  Destructive Deep Learning
Density computation and sample generation

\[ P_{\hat{X}}(x) = \prod_{i=1}^{3} |\text{det} J_{D_i}| \]

Destructor

\[ \approx \text{Uniform} \]

Generator

\[ \approx \text{Data} \]

\[ D_1 \]

\[ D_2 \]

\[ D_3 \]
Reuse: 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

<table>
<thead>
<tr>
<th>Models from MAF paper computed on Titan X GPU</th>
<th>MNIST</th>
<th>CIFAR-10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian</td>
<td>LL</td>
<td>D</td>
</tr>
<tr>
<td>MADE</td>
<td>-1367</td>
<td>1</td>
</tr>
<tr>
<td>MADE MoG</td>
<td>-1385</td>
<td>1</td>
</tr>
<tr>
<td>Real NVP</td>
<td>-1329</td>
<td>5</td>
</tr>
<tr>
<td>Real NVP</td>
<td>-1765</td>
<td>10</td>
</tr>
<tr>
<td>MAF</td>
<td>-1300</td>
<td>5</td>
</tr>
<tr>
<td>MAF</td>
<td>-1314</td>
<td>10</td>
</tr>
<tr>
<td>MAF MoG</td>
<td>-1100</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Our proposed destructors computed on 10 CPUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copula</td>
</tr>
<tr>
<td>LL = Log Likelihood (higher is better)</td>
</tr>
<tr>
<td>D = # of layers, T = Time</td>
</tr>
</tbody>
</table>
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.
Density destructor algorithm performs greedy layer-wise construction of deep destructor

1. Simple density estimation (GMM, Gaussian, tree density, etc.)
   \[
   Q^t \leftarrow \arg \min_{Q \in \mathcal{Q}} KL(P(x^{t-1}), Q(x^{t-1}))
   \]

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
   \[
   D^t = d^t \circ D^{t-1}
   \]
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 $U_z(z)$ is the uniform density function
  \[
  \arg\min_D KL\left(P_z(z; D), U(z)\right)
  \]

- KL equivalence lemma, let $z = D(x)$ for invertible $D$
  \[
  KL\left(P_x(x), Q_x(x)\right) = KL\left(P_z(z), Q_z(z)\right)
  \]

- Simple corollary is that objective above is MLE:
  \[
  KL\left(P_z(z; D), U(z)\right)
  \]
  \[
  = KL\left(P_x(x), Q_x(x; D)\right) \quad \text{(KL equivalence, MLE objective)}
  \]
  \[
  = KL \left(P_x(x), |J_D(x)|U(D(x))\right) \quad \text{(In terms of $D$)}
  \]
  \[
  = KL(P_x(x), |J_D(x)|) \quad \text{($U(z) = 1$)}
  \]
Deep density destructors via sequence of weak destructors

Weak density estimation

Data

$D_1$

$D_2$

$D_3$

Train Data

Implicit Model

$P_1$

$P_2$

$P_3$

$\Omega$
Destructive 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
  \[
  \arg\min_D KL(P_z(z), U(z))
  \]

- Want: Every iteration decreases objective:
  \[
  KL(P_z(d^t(x)), U(d^t(x))) \leq KL(P_x(d^{t-1}(x)), U(d^{t-1}(x)))
  \]

- Let $x = D^{t-1}(x^{(0)})$ and $z = d^t(x)$

- $KL(P_z(z; D), U(z))$

- $= KL(P_x(x), Q_x(x; D))$
  (KL equivalence lemma)

- $\leq KL(P_x(x), Q_x(x; D = Id))$
  (minimization is better than one particular)

- $= KL\left(P_x(x), |J_D(x)|U(D(x))\right)$ (Expand in terms of $D$)

- $= KL(P_x(x), U(x))$