Thinking Compositionally about Inference

Jan-Willem van de Meent

University of Amsterdam

Northeastern University

Heiko Zimmermann  Babak Esmaeili  Sam Stites  Hao Wu  Eli Sennesh
Deep Learning Success Stories

PaLM

540 billion parameters
### Explaining a joke

**Prompt**

Explain this joke:

Joke: Did you see that Google just hired an eloquent whale for their TPU team? It showed them how to communicate between two different pods!

**Model Response**

Prediction: TPUs are a type of computer chip that Google uses for deep learning. A "pod" is a group of TPUs. A "pod" is also a group of whales. The joke is that the whale is able to communicate between two groups of whales, but the speaker is pretending that the whale is able to communicate between two groups of TPUs.

PaLM
Deep Learning Success Stories

Painting of the orange cat Otto von Garfield, Count of Bismarck-Schönhausen, Duke of Lauenburg, Minister-President of Prussia. Depicted wearing a Prussian Pickelhaube and eating his favorite meal - lasagna.

A photo of the back of a wombat wearing a backpack and holding a walking stick. It is next to a waterfall and is staring at a distant mountain.

Dall-E 2, Imagen, Parti, Stable Diffusion
Is Scale All We Need?

The Bitter Lesson
Rich Sutton, March 13, 2019
The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin.

- Scale can lead to abstractions and generalization across tasks
- Still difficult to know when a model will succeed or fail.
- How can we scale up to more diverse application domains?

http://www.incompleteideas.net/InclIdeas/BitterLesson.html
Is Scale All We Need?

- Scale can lead to abstractions and generalization across tasks
- Still difficult to know when a model will succeed or fail.
- How can we scale up to more diverse application domains?

Lessons from deep learning

1. Gradient descent scales *really* well
2. Model engineering scales *pretty* well
Horizons of AI Research

Science & Engineering
- Deep domain knowledge
- but limited data

Autonomous Vehicles
- Generalization to
- long tail events

Healthcare
- Many prediction tasks,
- imbalanced data

Challenges in emerging domains
1. Incorporating (enough) domain knowledge
2. Reliable generalization across related tasks
3. Avoiding overconfident predictions
What Models are Useful?

**Simulation-based Modeling**

[Smedemark-Margulies et al., 2021]

**Planning and Robotics**

[Biza et al., 2021]

**Vision & Language**

[McInerney et al., 2020]

**Stronger assumptions**

- Known dynamics (e.g. PDEs) for system

**Weaker assumptions**

- More knowledge (and edge cases)

- Some domain knowledge (e.g. structure)
What Models are Useful?

Simulation-based Modeling

Planning and Robotics

Vision & Language

**Stronger assumptions**
- Known dynamics (e.g. PDEs) for system

**Weaker assumptions**
- More knowledge (and edge cases)
- Some domain knowledge (e.g. structure)

[Smedemark-Margulies et al., 2021]
[Biza et al., 2021]
[McInerney et al., 2020]

[McInerney et al., 2020]
[Biza et al., 2021]
[Smedemark-Margulies et al., 2021]
The Next 700 AI Domains
Two Ingredients for a Language

1. Core operations / abstractions
2. Mechanisms for composition into program
Differentiable Programming

1. **Abstractions:** Differentiation, Tensor Calculus, Layers
2. **Composition:** Networks, Objectives, Optimization
1. **Abstractions**: Distributions, Conditioning, Inference
2. **Composition**: Programs as Probabilistic Models
Programs as Models

Forward Simulation

\[ \eta \rightarrow f \rightarrow \hat{x} \]

- **inputs** (parameters)
- **model** (program)
- **outputs** (predictions)
Programs as Models

Forward Simulation

\( \eta \rightarrow f \rightarrow x \)

prior model (program) likelihood

Generative Model

\[ p(x, \eta) = p(\eta) p(x \mid f(\eta)) \]
Programs as Models

Forward Simulation

\( \eta \rightarrow f \rightarrow x \)

prior \hspace{1cm} model \hspace{1cm} likelihood

(program)

Inference (Inverse Reasoning)

\[
p(\eta \mid x) = \frac{p(\eta) p(x \mid f(\eta))}{p(x)}
\]
Programs as Models

Forward Simulation

\[ \eta \xrightarrow{f} x \]

prior \quad \text{model (program)} \quad \text{likelihood}

Inference (Inverse Reasoning)

\[ p(\eta | x) = \frac{p(\eta) p(x | f(\eta))}{p(x)} \]

Intractable integral
Can have intractable likelihood (ABC/SBI)

Inference (Inverse Reasoning)

\[ p(\eta \mid x) = \frac{p(\eta) p(x \mid f(\eta))}{p(x)} \]
Programs as Models

Probabilistic Program (User Defined)

prior \rightarrow f \rightarrow x

\eta \quad \text{model (deterministic)} \quad x

likelihood
Programs as Models

Probabilistic Program (User Defined)

Prior \( \eta \) \( \xrightarrow{\text{model}} \) Likelihood \( f \) \( \xrightarrow{\text{likelihood}} \) Observation \( x \)

Inference (Implemented by System)

\[ \eta \sim p(\eta \mid x) \]
Programs as Probabilistic Models

Probabilistic Program

```python
def regression(y_vals, x_vals):
    a = sample(normal(0, 10))
    b = sample(normal(0, 10))
    f = lambda x: a * x + b
    for y, x in zip(y_vals, x_vals):
        observe(normal(f(x), 0.1), y)
    return f, a, b
```

Probabilistic Special Forms

- **sample** random value
- **observe** condition on a value

Inference (MCMC)

\[
p(a, b \mid y) = \frac{p(y, a, b)}{p(y)}
\]

Infer **sample** values that are in agreement with **observe** values (using Bayesian statistics)
Programs as Probabilistic Models (~2013)

Probabilistic Program

```python
def render(chars, shape):
    ...

def char():
    x = sample(uniform(0.0, 1.0))
    y = sample(uniform(0.0, 1.0))
    size = sample(uniform(0.5, 1.5))
    weight = sample(uniform(0.8, 1.2))
    return x, y, size, weight

def captcha(image):
    K = sample(uniform(range(1, 6)))
    chars = [char() for k in range(K)]
    captcha = render(chars, image.shape)
    observe(normal(captcha, 0.1), image)
    return captcha, chars
```

Data

Inference (MCMC)

[Mansinghka, Kulkarni, Perov, Tenenbaum, NeurIPS 2013]
Programs as Probabilistic Models (~2013)

Probabilistic Program

```python
def render(chars, shape):
    ...

def char():
    x = sample(uniform(0.0, 1.0))
    y = sample(uniform(0.0, 1.0))
    size = sample(uniform(0.5, 1.5))
    weight = sample(uniform(0.8, 1.2))
    return x, y, size, weight

def captcha(image):
    K = sample(uniform(range(1, 6)))
    chars = [char() for k in range(K)]
    captcha = render(chars, image.shape)
    observe(normal(captcha, 0.1), image)
    return captcha, chars
```

Data

Inference (MCMC)

[Mansinghka, Kulkarni, Perov, Tenenbaum, NeurIPS 2013]
Probabilistic Program

• Writing realistic simulators can be difficult (might need source code for CAPTCHA generator)
• Inference can be prohibitively slow (MCMC takes ~15 mins for a single image)

Actual CAPTCHAs

Programs as Probabilistic Models (~2013)
Deep Probabilistic Programming

1. **Abstractions:** Differentiable Programming, Densities, Sampling, Variational Objectives

2. **Composition:** Deep Generative Models, Amortized Importance Samplers

Edward2

https://github.com/google/edward2

Probabilistic Torch

https://github.com/probtorch/probtorch

Pyro / NumPyro

https://pyro.ai
Deep Probabilistic Programming

- Deep Learning
- Amortized Inference
- Language Design & Semantics
- Importance Sampling
- Inference Programming

AI

PL

Stats
Deep Probabilistic Programming

- Differentiable Programming
- Language Design & Semantics
- Inference Programming
- Deep Learning
- Amortized Inference
- Importance Sampling
- Stats
- PL

AI
Amortized Inference

Generative Model
(stochastic simulator)

\[ \eta \rightarrow f \rightarrow x \]

prior

model (program)

likelihood

Inference Model
(approximate inverse)

\[ \eta \leftarrow g \leftarrow x \]

approx. posterior

network (program)

data distribution
Amortized Inference

Generative Model
(stochastic simulator)

\[ \eta \xrightarrow{f} x \]

- prior
- model (program)
- likelihood

Inference Model
(approximate inverse)

\[ \eta \xleftarrow{g} x \]

- approx. posterior
- network (program)
- data distribution

Model Learning

\[ \min_{\theta} D(q(x) \parallel p_\theta(x)) \]

Amortized Inference

\[ \min_{\phi} \mathbb{E}_{x \sim q} \left[ D(q_\phi(\eta \mid x) \parallel p_\theta(\eta \mid x)) \right] \]
Reparameterized Variational Inference

Variational Lower Bound

$$\max_{\theta, \phi} \mathcal{L} = \max_{\theta, \phi} \mathbb{E}_{\eta, x \sim q} [\log w]$$

$$= \min_{\theta, \phi} \left\{ D_{KL}(q_{\phi}(x) \parallel p_{\theta}(x)) \right\}$$

$$+ \mathbb{E}_{x \sim q} \left[ D_{KL}(q_{\phi}(\eta | x) \parallel p_{\theta}(\eta | x)) \right]$$

Importance Weight

$$w = \frac{p_{\theta}(x, \eta)}{q_{\phi}(\eta | x)}$$

Combines model learning and amortized inference

[Kingma, Welling, ICLR 2014; Rezende, Mohamed, Wierstra, ICML 2014]
Reparameterized Variational Inference

Variational Lower Bound

$$\max_{\theta, \phi} \mathcal{L} = \max_{\theta, \phi} \mathbb{E}_{\eta, x \sim q} \left[ \log w \right]$$

Importance Weight

$$w = \frac{p_\theta(x, \eta)}{q_\phi(\eta \mid x)}$$

Main Requirement: Fully differentiable model

$$\nabla_\phi \mathbb{E}_{\eta \sim q_\phi} \left[ \log w \right] = \mathbb{E}_{\epsilon \sim q_0} \left[ (\nabla_\eta \log w)^\top \nabla_\phi \eta(\epsilon) \right]$$

$$+ \nabla_\phi \log w$$

[Kingma, Welling, ICLR 2014; Rezende, Mohamed, Wierstra, ICML 2014]
Deep Probabilistic Programming

AI

Deep Learning

Amortized Inference

Differentiable Programming

Language Design & Semantics

Importance Sampling

Inference Programming

PL

Stats
Minimizing the Inclusive KL divergence

**Idea 1:** Minimize inclusive KL (rather than exclusive KL)

\[
\min_{\phi} \mathbb{E}_{x \sim q} \left[ D_{KL}(q_{\phi}(\eta \mid x) \mid\mid p_{\theta}(\eta \mid x)) \right] \rightarrow \min_{\phi} \mathbb{E}_{x \sim q} \left[ D_{KL}(p_{\theta}(\eta \mid x) \mid\mid q_{\phi}(\eta \mid x)) \right]
\]

**Idea 2:** Use importance sampling to approximate gradient

\[
-\nabla_{\phi} D_{KL}(p_{\theta}(\eta \mid x) \mid\mid q_{\phi}(\eta \mid x)) = \mathbb{E}_{\eta \sim p_{\theta}(\cdot \mid x)} \left[ \nabla_{\phi} \log q_{\phi}(\eta \mid x) \right]
\]

Use importance sampling

\[
w^l = \frac{p_{\theta}(x, \eta^l)}{q_{\phi}(\eta^l \mid x)} \quad \eta^l \sim q_{\phi}(\eta \mid x)
\]

[Bornschein and Bengio, ICLR 2015] [Le, Kosiorek, Siddarth, Teh, Wood, UAI 2019]
Minimizing the Inclusive KL divergence

**Idea 1:** Minimize inclusive KL (rather than exclusive KL)

\[
\begin{align*}
\min_{\phi} \mathbb{E}_{x \sim q} \left[ D_{KL}(q_\phi(\eta \mid x) \mid\mid p_\theta(\eta \mid x)) \right] &\rightarrow \min_{\phi} \mathbb{E}_{x \sim q} \left[ D_{KL}(p_\theta(\eta \mid x) \mid\mid q_\phi(\eta \mid x)) \right]
\end{align*}
\]

**Idea 2:** Use importance sampling to approximate gradient

\[
-\nabla_{\phi} D_{KL}(p_\theta(\eta \mid x) \mid\mid q_\phi(\eta \mid x)) = \mathbb{E}_{\eta \sim p_\theta(\cdot \mid x)} \left[ \nabla_{\phi} \log q_\phi(\eta \mid x) \right]
\]

Use importance sampling

\[
w^l = \frac{p_\theta(x, \eta^l)}{q_\phi(\eta^l \mid x)} \quad \eta^l \sim q_\phi(\eta \mid x)
\]

\[
\approx \sum_{l=1}^{L} \frac{w^l}{\sum_{l'} w^{l'}} \nabla_{\phi} \log q_\phi(\eta^l \mid x)
\]

[Bornschein and Bengio, ICLR 2015] [Le, Kosiorek, Siddarth, Teh, Wood, UAI 2019]
Amortized Importance Samplers

Better gradient estimates

Importance Sampling
Use proposals \( q_\phi(\eta | x) \) to sample from \( p_\theta(\eta | x) \)

Variational Inference
Learn proposals \( q_\phi(\eta | x) \) using samples from \( p_\theta(\eta | x) \)

Better proposals

• Does not rely on differentiable models / reparameterization
• Often works as well as, or better than, maximizing a lower bound

[Bornschein and Bengio, ICLR 2015]  [Le, Kosiorek, Siddarth, Teh, Wood, UAI 2019]
Amortized Importance Samplers

Importance Sampling
Use proposals $q_\phi(\eta \mid x)$ to sample from $p_\theta(\eta \mid x)$

Variational Inference
Learn proposals $q_\phi(\eta \mid x)$ using samples from $p_\theta(\eta \mid x)$

Better gradient estimates
Better proposals

Opportunity: New VI methods based on SMC samplers, nested importance samplers, etc

[Bornschein and Bengio, ICLR 2015] [Le, Kosiorek, Siddarth, Teh, Wood, UAI 2019]
Example (~2019): Amortized Population Gibbs

Task: Unsupervised Tracking

- Corpus level (*many videos*)
  Digit shapes
  Transition dynamics

- Instances (*single videos*)
  Object representations

- Data-points (*single frames*)
  Object positions

[Wu, Zimmermann, Sennesh, Le, van de Meent, ICML 2020]
Example (~2019): Amortized Population Gibbs

Classic Chicken-and-Egg Problem

- **Easy**: Infer object representations given object positions

- **Also Easy**: Infer positions given object representations

- **Not Easy**: Joint inference of positions and representations

[Wu, Zimmermann, Sennesh, Le, van de Meent, ICML 2020]
Example (~2019): Amortized Population Gibbs

Classic Solution: Iterate

- **Step 0**: Initialize representations and positions.

- **Update 1**: Infer object representations given object positions
  \[ \eta \sim p(\eta \mid x, z) \]

- **Update 2**: Infer object representations given object positions
  \[ z \sim p(z \mid x, \eta) \]

Problem: Only computable in conjugate exponential family models

[Wu, Zimmermann, Sennesh, Le, van de Meent, ICML 2020]
Modern solution: Learn Updates

- **Step 0**: Initialize representations and positions.
- **Update 1**: Infer object representations given object positions
  \[ \eta \sim q_{\phi}(\eta | x, z) \]
- **Update 2**: Infer object representations given object positions
  \[ z \sim q_{\phi}(z | x, \eta) \]

Example (~2019): Amortized Population Gibbs

[Wu, Zimmermann, Sennesh, Le, van de Meent, ICML 2020]
Example (~2019): Amortized Population Gibbs

- Completely unsupervised
- Computationally efficient (~5 updates, ~10 particles)

[Wu, Zimmermann, Sennesh, Le, van de Meent, ICML 2020]
Deep Probabilistic Programming

- Differentiable Programming
- Amortized Inference
- Language Design & Semantics
- Importance Sampling
- Inference Programming
Reasoning Compositionally About Inference

What (inference) DSL could define this sampler / variational method?

<table>
<thead>
<tr>
<th>Algorithm 1 Amortized Population Gibbs Sampling</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: for $n$ in 1, ..., $N$ do</td>
</tr>
<tr>
<td>2: $G_{\phi} = 0$</td>
</tr>
<tr>
<td>3: $x^n \sim p^{\text{DATA}}(x)$</td>
</tr>
<tr>
<td>4: for $l$ in 1, ..., $L$ do</td>
</tr>
<tr>
<td>5: $z^{n,1,l} \sim q_\phi(z \mid x^n)$</td>
</tr>
<tr>
<td>6: $w^{n,1,l} \leftarrow p_\theta(x^n, z^{n,1,l}) / q_\phi(z^{n,1,l})$</td>
</tr>
<tr>
<td>7: for $k$ in 2, ..., $K$ do</td>
</tr>
<tr>
<td>8: $\bar{z}, \bar{w} = z^{n,k-1}, w^{n,k-1}$</td>
</tr>
<tr>
<td>9: for $b$ in 1, ..., $B$ do</td>
</tr>
<tr>
<td>10: $\bar{z}, \bar{w} = \text{RESAMPLE}(\bar{z}, \bar{w})$</td>
</tr>
<tr>
<td>11: for $l$ in 1, ..., $L$ do</td>
</tr>
<tr>
<td>12: $\bar{z}<em>b^{l} \sim q</em>\phi(\cdot \mid x^n, \bar{z}_b^{l-1})$</td>
</tr>
<tr>
<td>13: $\bar{w}<em>b^{l} = p</em>\theta(x^n, \bar{z}_b^{l-1}, \bar{z}<em>b^{l}) q</em>\phi(\bar{z}_b^{l} \mid x^n, \bar{z}_b^{l-1}) \bar{w}_b^{l}$</td>
</tr>
<tr>
<td>14: $\bar{z}_b^{l} = \bar{z}_b^{l}$</td>
</tr>
<tr>
<td>15: $G_\phi = G_\phi + \sum_{l=1}^{L} \frac{d}{d\phi} \log q_\phi(\bar{z}_b^{l} \mid x^n, \bar{z}_b^{l})$</td>
</tr>
<tr>
<td>16: $z^{n,k}, w^{n,k} = \bar{z}, \bar{w}$</td>
</tr>
<tr>
<td>17: return $G_\phi, z, w$</td>
</tr>
</tbody>
</table>

- **APG** is an example of a amortized SMC sampler
- **Known building blocks, but not trivial to combine correctly**
- **Can we define compositional methods for importance sampling and gradient estimation?**
Combinators: A DSL for Inference

\[
f ::= \text{A primitive program} \\
p ::= f \mid \text{extend}(p, f) \\
q ::= p \mid \text{resample}(q) \mid \text{compose}(q', q) \mid \text{propose}(p, q)
\]

[Stites, Zimmerman, Wu, Sennesh, van de Meent, UAI 2021]
Combinators: A DSL for Inference

\[
f ::= \text{A primitive program} \\
p ::= f \mid \text{extend}(p, f) \\
q ::= p \mid \text{resample}(q) \mid \text{compose}(q', q) \mid \text{propose}(p, q)
\]

[Stites, Zimmerman, Wu, Sennesh, van de Meent, UAI 2021]
Combinators: A DSL for Inference

\[ f ::= \text{A primitive program} \]
\[ p ::= f \mid \text{extend}(p, f) \]
\[ q ::= p \mid \text{resample}(q) \mid \text{compose}(q', q) \mid \text{propose}(p, q) \]

[Stites, Zimmerman, Wu, Sennesh, van de Meent, UAI 2021]
Combinators: A DSL for Inference

\[
f ::= \text{A primitive program} \\
p ::= f \mid \text{extend}(p, f) \\
q ::= p \mid \text{resample}(q) \mid \text{compose}(q', q) \mid \text{propose}(p, q)
\]

[Stites, Zimmerman, Wu, Sennesh, van de Meent, UAI 2021]
Combinators: A DSL for Inference

\[ f ::= \text{A primitive program} \]
\[ p ::= f \mid \text{extend}(p, f) \]
\[ q ::= p \mid \text{resample}(q) \mid \text{compose}(q', q) \mid \text{propose}(p, q) \]

[Stites, Zimmerman, Wu, Sennesh, van de Meent, UAI 2021]
Core Property: Proper Weighting

Definition: A pair $w, z$ is properly weighted with respect to a density $\pi(z)$ when, for all measurable $h(z)$,

$$
E_{w,z \sim \Pi} [w h(z)] = \mathcal{Z} E_{z \sim \pi} [h(z)]
$$

- **Sampler** (can be a black box)
- **Constant of proportionality** (marginal likelihood)
- **Quantity of interest** (return value of program)
- **Density of interest** (program posterior)

[Naesseth, Lindsten, Schön, Foundations and Trends in Machine Learning, 2019]
Combinators: A DSL for Inference

Semantics: Composition preserves proper weighting

https://github.com/probtorch/combinators
(Pyro implementation forthcoming)

[Stites, Zimmerman, Wu, Sennesh, van de Meent, UAI 2021]
def pop_gibbs(target, proposal, kernels, sweeps):
    q = propose(partial(target, suffix=0),
                partial(proposal, suffix=0))
    for s in range(sweeps):
        for k in kernels:
            q = propose(
                        extend(partial(target, suffix=s+1),
                                partial(k, suffix=s)),
                        compose(partial(k, suffix=s+1),
                                resample(q, dim=0)))
    return q

Example: Amortized Gibbs Samplers

High-level algorithm description
(transition operators, resampling)

Low-level algorithm description
(weight and gradient computations)

[Stites, Zimmerman, Wu, Sennesh, van de Meent, UAI, 2021]
Differentiable + Probabilistic + Inference Programming

Deep Generative Model
program $p_{\theta}(x, z)$

Inference Model
program $q_{\phi}(z \mid x)$

Importance Sampling
Use proposals $q_{\phi}(z \mid x)$ to sample from $p_{\theta}(z \mid x)$

Variational Inference
Learn proposals $q_{\phi}(z \mid x)$ using samples from $p_{\theta}(z \mid x)$

Better gradient estimates

Better proposals

User-specified importance sampler (inference combinators)

User-specified variational objectives (nested variational inference)
Action priors

The Dirichlet prior... instead, we learn the action prior as a convolutional... rewards for the current task. Rosman and Ramamoorthy... policy parameters.

Note that action priors refer to a di... learned policies are valid over di... in robotics with a large action space.

In contrast to pre-de...

This paper makes two main contributions. First, we show that a... effective way to transfer knowl...

In concurrent work, Ajay and Agrawal... distill policies from training tasks into a single student,... has been studied extensively both in classi-... targeted exploration compared to the notion of surprise alone.

---

Abstractions for Emerging Problems:

- Learning surrogate models
- Modeling search spaces
- Inferring differential equations
Thank You!

Nested Variational Inference
H. Zimmermann, H. Wu, Babak Esmaeili, J.-W. van de Meent

Learning Proposals for Probabilistic Programs with Inference Combinators
S. Stites*, H. Zimmermann*, H. Wu, E. Sennesh, J.-W. van de Meent
UAI 2021 [https://arxiv.org/abs/2103.00668]

Amortized Population Gibbs Samplers with Neural Sufficient Statistics
H. Wu, H. Zimmermann, E. Sennesh, Tuan Anh Le, J.-W. van de Meent

An Introduction to Probabilistic Programming
J.-W. van de Meent, B. Paige, H. Yang, F. Wood
ArXiv 2018 [https://arxiv.org/abs/1809.10756]