Working Towards Distributed Inference Compilation at Scale

Frank Wood
fwood@cs.ubc.ca

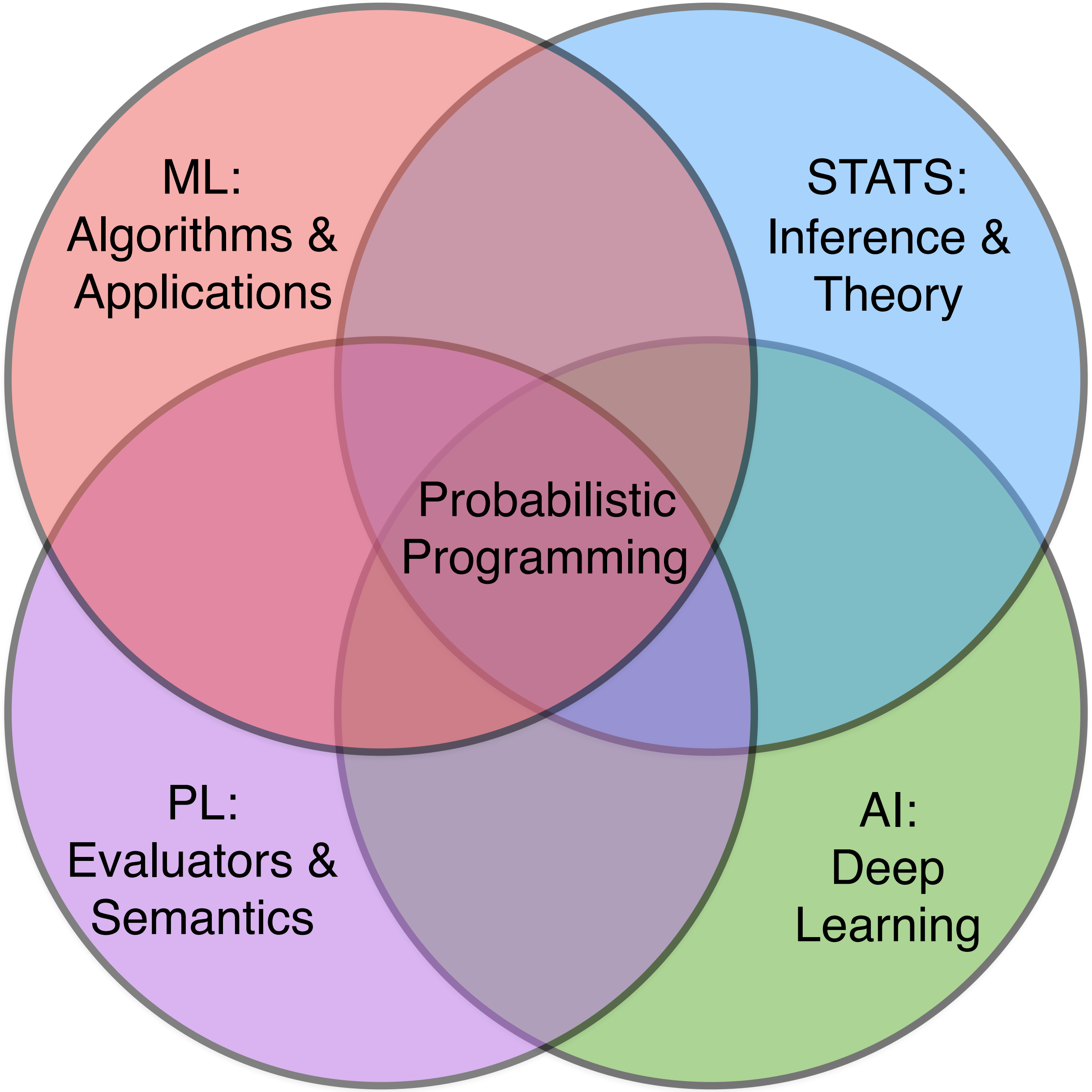
Efficient Probabilistic Inference in the Quest for Physics Beyond the Standard Model

Frank Wood
fwood@cs.ubc.ca

Outline

- Probabilistic Programming
- Model-Based Reasoning & Inference
- Inference Compilation
- The Quest for New Physics
- Challenges
- Massively Distributed Deep Network Training
- The Vision

Probabilistic Programming



Existing Languages

Graphical Models

BUGS STAN

Factor Graphs

Factorie Infer.NET

Infinite Dimensional Parameter Space Models

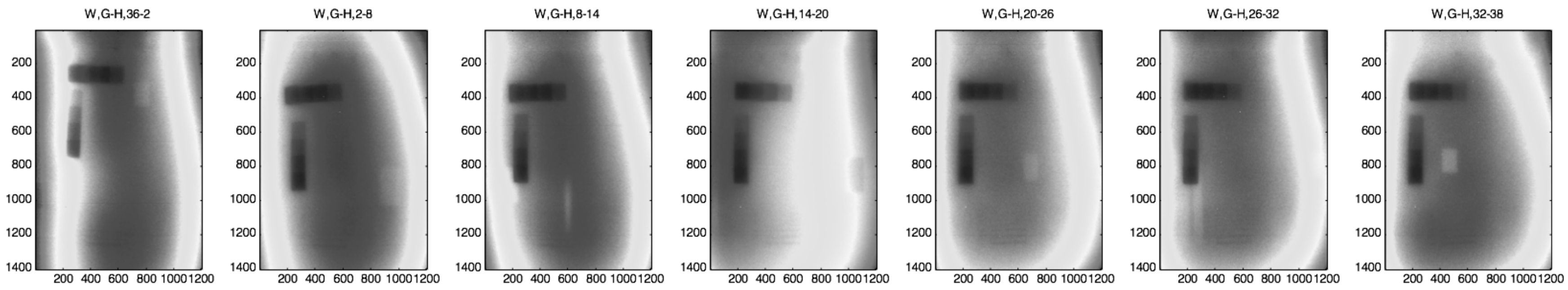
Anglican WebPPL

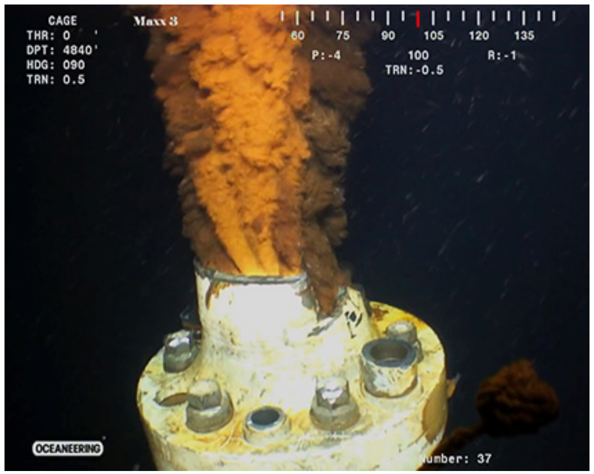
Unsupervised Deep Learning

PYRO ProbTorch

Model-based reasoning and Inference

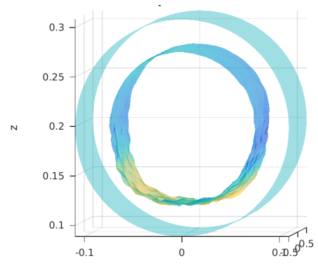
Radiographic Inspection for Oil Spill Prevention



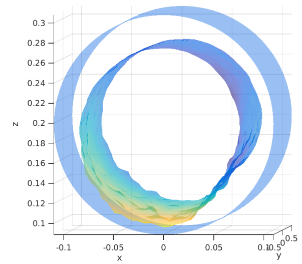


?

Radiographic Inspection for Oil Spill Prevention



Pipe sample 1



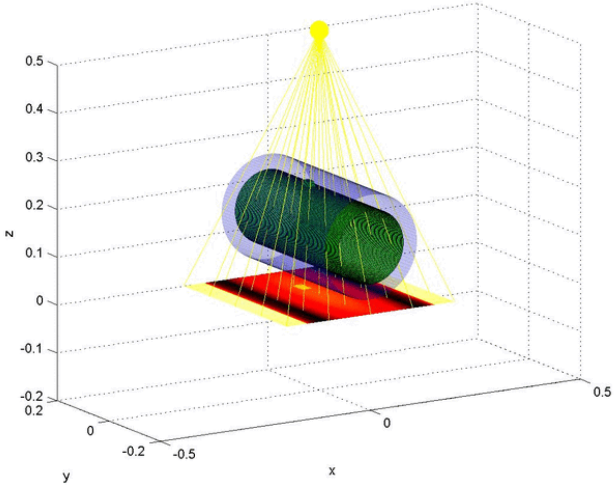
Pipe Sample 2

. . .

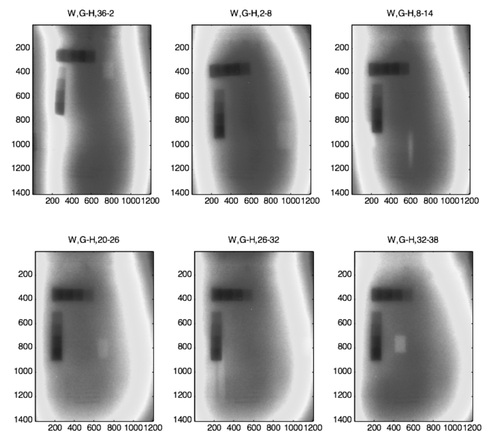
Inference

$$p(X|Y) \boxed{\propto} \boxed{p(X)\, p(Y|X)}$$

Generative Model



+



[**Rainforth**, Mahendran, Osborne, Vedaldi and Wood. Industry sponsored project 2017]

CAPTCHA breaking

SMKBDF

Can you write a program to do this?

$\mathbf{x}$	$\mathbf{y}$
text	image

Mansinghka, Kulkarni, Perov, and Tenenbaum
“Approximate Bayesian image interpretation using generative probabilistic graphics programs." NIPS (2013).

Captcha Generative Model


```
(defm sample-char []
 {:symbol (sample (uniform ascii))
  :x-pos (sample (uniform-cont 0.0 1.0))
  :y-pos (sample (uniform-cont 0.0 1.0))
  :size (sample (beta 1 2))
  :style (sample (uniform-dis styles))
  …})
```


```
(defm sample-captcha []
 (let [n-chars (sample (poisson 4))
       chars (repeatedly n-chars
                         sample-char)
     noise (sample salt-pepper)
     …]
  gen-image))
```

Conditioning

```
(defquery captcha [true-image]
 (let [gen-image (sample-captcha)]
  (observe (similarity-kernel gen-image)
           true-image)
  gen-image))
```

Generative Model

```
(doquery :ipmcmc captcha true-image)
```

Inference

2015 : Probabilistic Programming

- Restricted (i.e. STAN, BUGS, infer.NET)
  - Easier inference problems -> fast
  - Impossible for users to denote some models
  - Fixed computation graph
- Unrestricted (i.e. Anglican, WebPPL)
  - Possible for users to denote all models
  - Harder inference problems -> slow
  - Dynamic computation graph

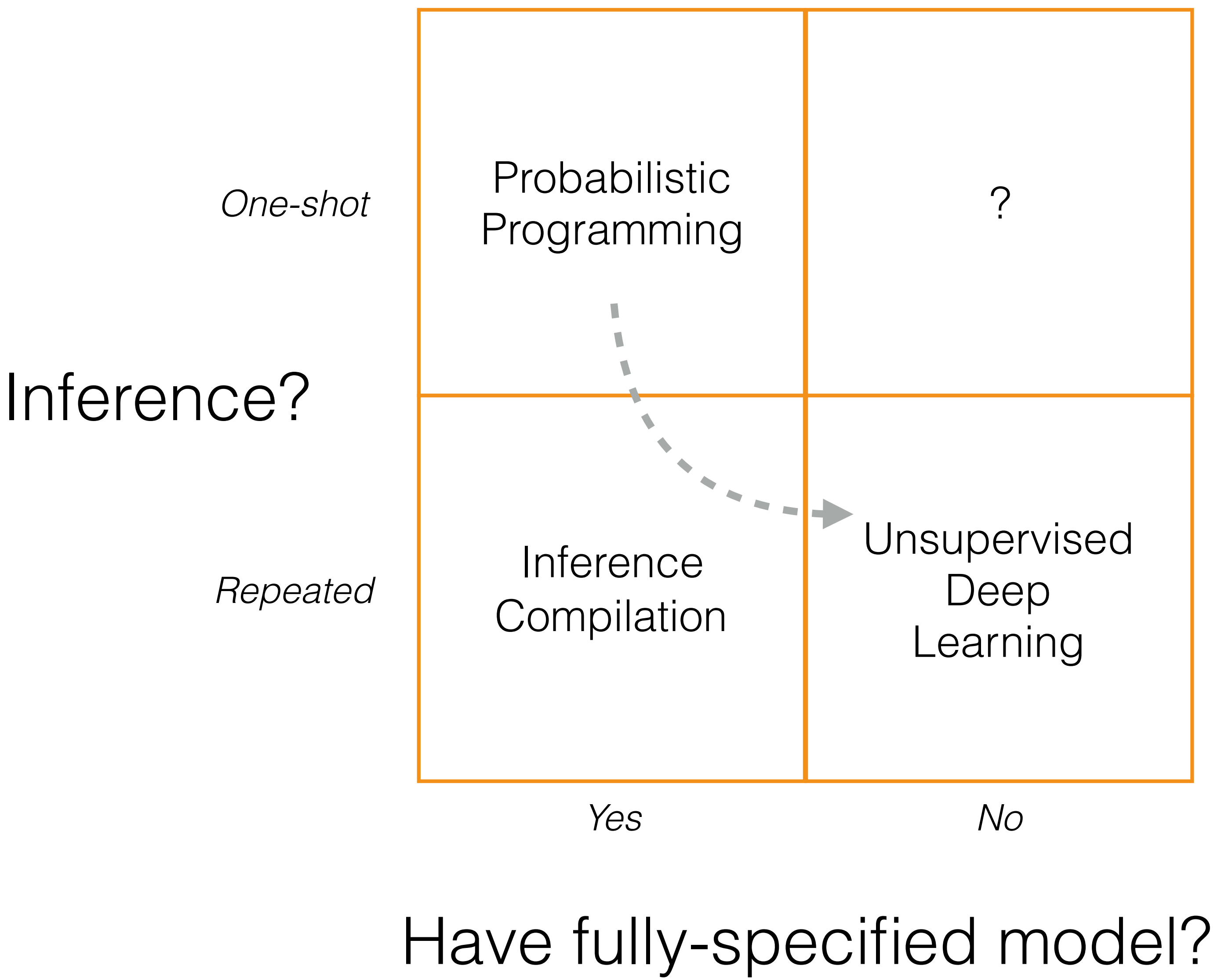
- Fixed, trusted model; one-shot inference

The AI/Repeated-Inference Challenge

“**Bayesian inference** is computationally expensive. Even approximate, sampling-based algorithms tend to take many iterations before they produce reasonable answers. In contrast, human recognition of words, objects, and scenes is extremely rapid, often taking only a few hundred milliseconds—only enough time for a **single pass from perceptual evidence to deeper interpretation**. Yet human perception and cognition are often well-described by **probabilistic inference in complex models**. How can we reconcile the speed of recognition with the expense of coherent probabilistic inference? How can we **build systems**, for applications like robotics and medical diagnosis, **that exhibit similarly rapid performance** at challenging inference tasks?”

Stuhlmüller A, Taylor J, Goodman N. Learning stochastic inverses. In Advances in Neural Information Processing Systems 2013 (pp. 3048-3056).

Resulting Trend In Probabilistic Programming

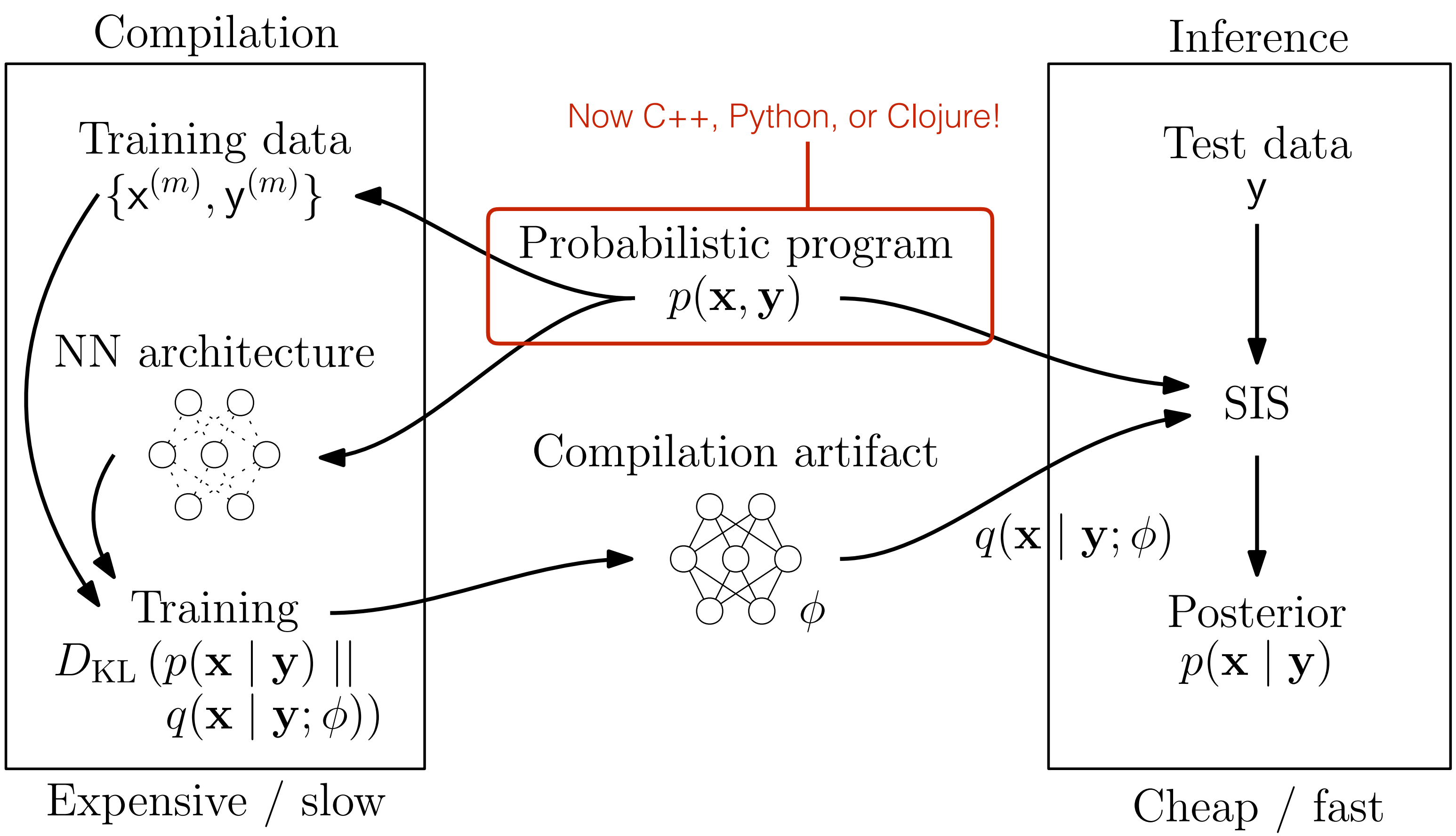


Inference Compilation

Inference Compilation Desiderata

- Denote a model and inference problem as a probabilistic programming language program
- “Compile” for hours or days, depending on the problem, CPU/GPUs at disposal, etc.
- Get a “compilation artifact” controller that enables fast, repeated inference in the original model that is compatible with asymptotically exact inference

Inference Compilation



Input: an inference problem denoted in a probabilistic programming language

Output: a trained inference network (deep neural network "compilation artifact")

Le TA, Baydin AG, **Wood** F. Inference Compilation and Universal Probabilistic Programming. AISTATS. 2017.

Compiling Away Runtime Costs of Inference

Learn to invert the generative model, before seeing data

Objective function:

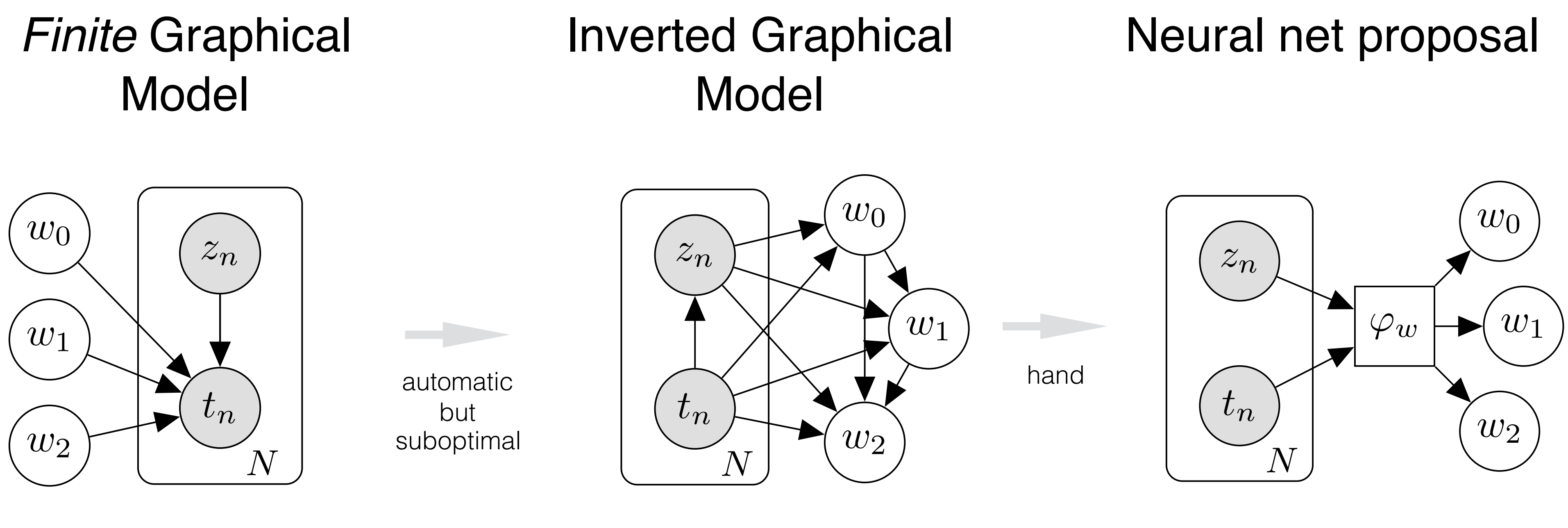
$$
\begin{aligned}
\mathcal{J}(\eta) &= \int D_{KL}(\pi || q_\lambda) p(\mathbf{y}) \mathrm{d}\mathbf{y} \\
&= \int p(\mathbf{y}) \int p(\mathbf{x}|\mathbf{y}) \log \left[\frac{p(\mathbf{x}|\mathbf{y})}{q(\mathbf{x}|\varphi(\eta, \mathbf{y}))} \right] \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y} \\
&= \mathbb{E}_{p(\mathbf{x},\mathbf{y})} \left[-\log q(\mathbf{x}|\varphi(\eta, \mathbf{y})) \right] + const.
\end{aligned}
$$

approximate with samples
from the joint distribution

Fully differentiable;
can train entirely offline: $\nabla_\eta \mathcal{J}(\eta) = \mathbb{E}_{p(\mathbf{x},\mathbf{y})} \left[-\nabla_\eta \log q(\mathbf{x}|\varphi(\eta, \mathbf{y})) \right]$

Paige B, **Wood** F. Inference Networks for Sequential Monte Carlo in Graphical Models. ICML. JMLR W&CP 48: 3040-3049. 2016.

Example : Non-Conjugate Regression Graphical Model



$$w_d \sim \text{Laplace}(0, 10^{1-d}) \qquad \text{for } d = 0, 1, 2;$$
$$t_n \sim \text{t}_\nu(w_0 + w_1 z_n + w_2 z_n^2, \epsilon^2) \qquad \text{for } n = 1, \ldots, N$$

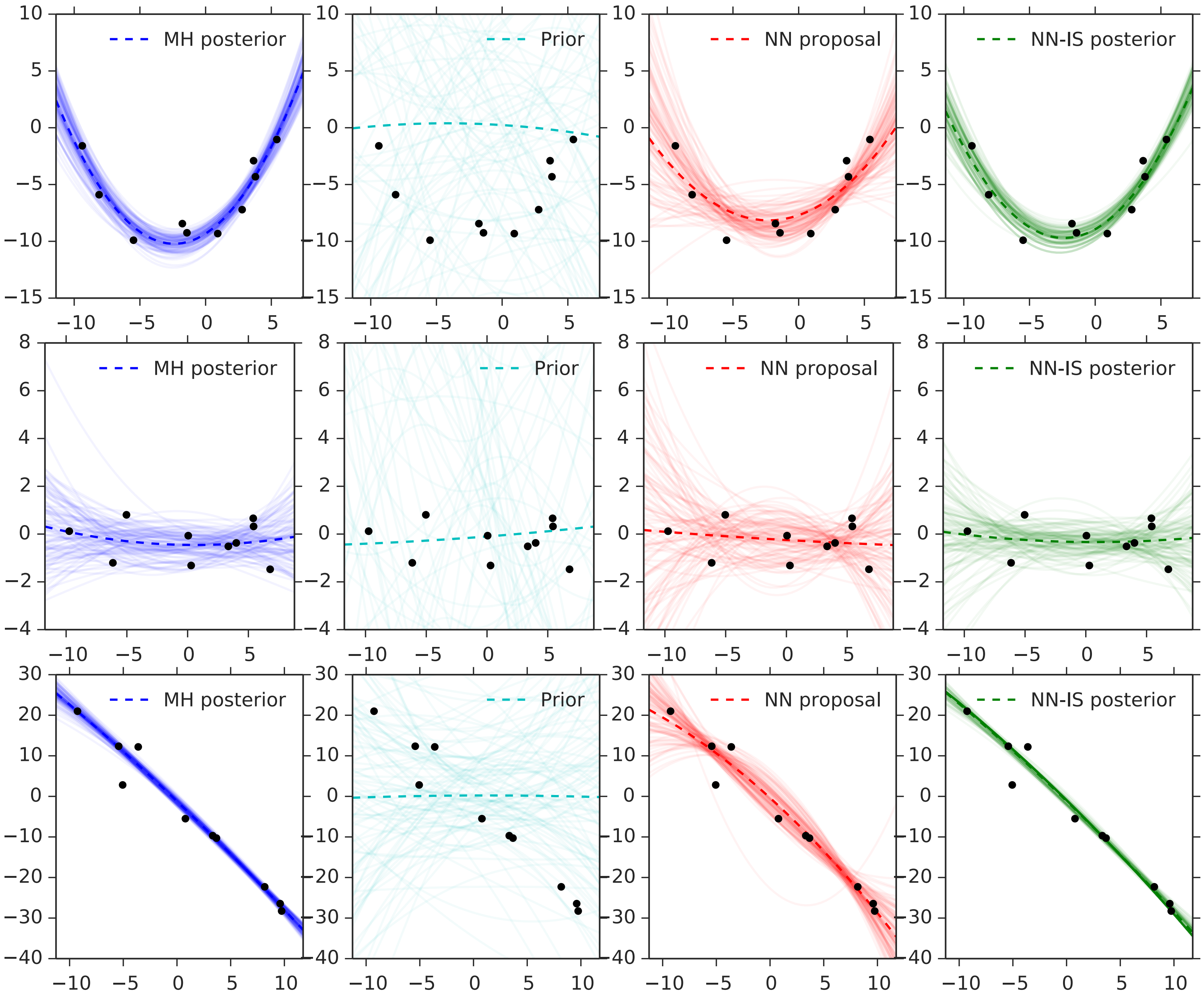
$\nu = 4, \epsilon = 1$, and $z_n \in (-10, 10)$

$$q(w_{0:2}|z_{1:N}, t_{1:N})$$

- Two layer MLP
- 200 units
- 3-component MOG for each output

Paige B, **Wood** F. Inference Networks for Sequential Monte Carlo in Graphical Models. ICML. JMLR W&CP 48: 3040-3049. 2016.

Example Non-Conjugate Regression



Paige B, **Wood** F. Inference Networks for Sequential Monte Carlo in Graphical Models. ICML. JMLR W&CP 48: 3040-3049. 2016.

CSIS : Inf. Comp. for Higher-Order PPL

- Same IS proposal learning objective

$$\nabla_\eta \mathcal{J}(\eta) = \mathbb{E}_{p(\mathbf{x},\mathbf{y})}\left[-\nabla_\eta \log q(\mathbf{x}|\varphi(\eta,\mathbf{y}))\right]$$

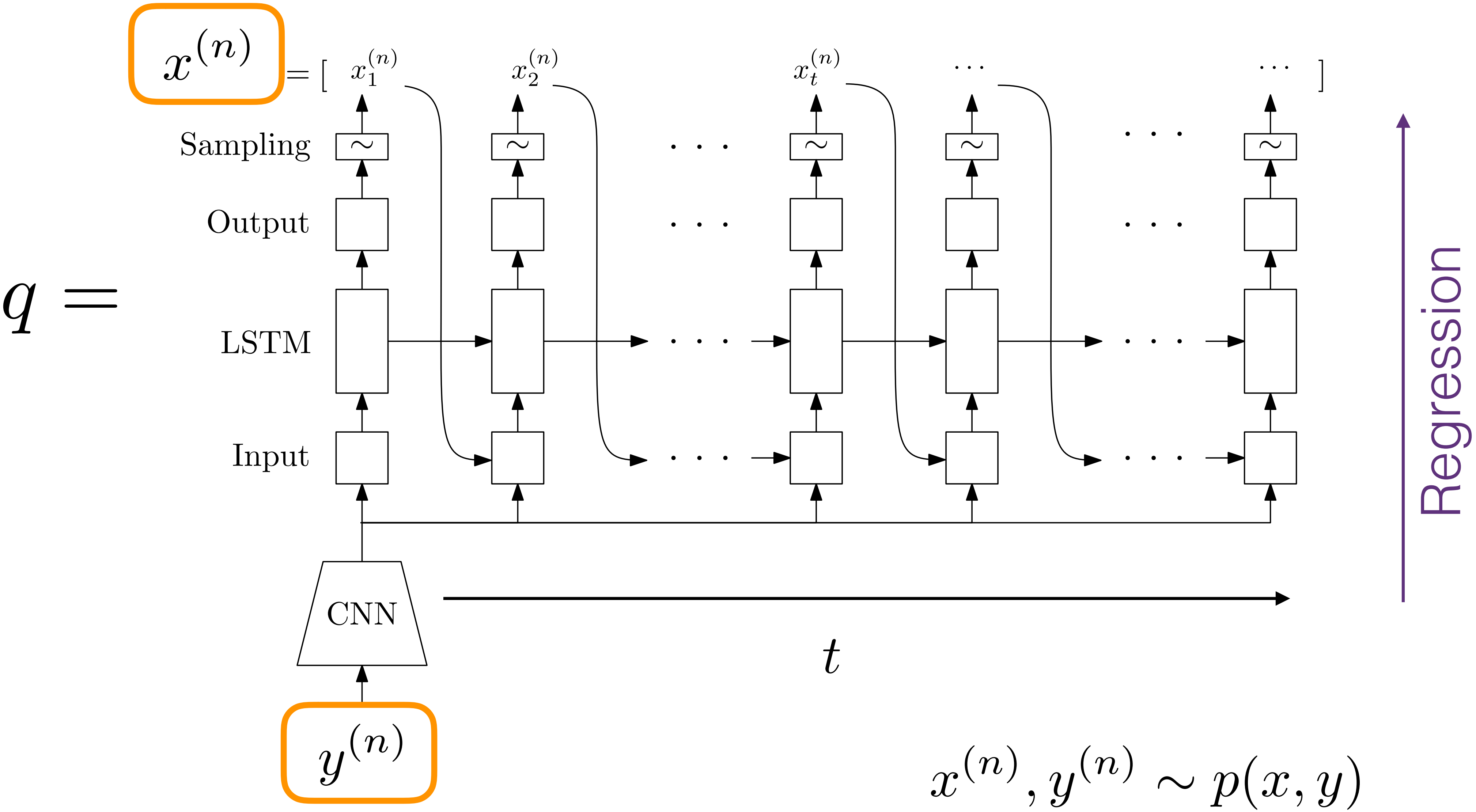
- Inf. dim. graph so only evaluation/“forward” inference methods possible, i.e. SIS with unnormalized weights

$$\frac{\prod_{i=1}^{N} g_i(y_i|\phi_i) \prod_{j=1}^{M} f_j(x_j|\theta_j)}{\prod_{j=1}^{M} q(x_j|\varphi(\eta,\mathbf{y},\mathbf{x}_{1:j-1}))} = w^{(k)}$$

- “Forward”-structured proposal / controller

$$q(\mathbf{x}|\varphi(\eta,\mathbf{y})) \triangleq \prod_{j=1}^{M} q(x_j|\varphi(\eta,\mathbf{y},\mathbf{x}_{1:j-1}))$$

Generic Structured Proposal Architecture



$$x^{(n)}, y^{(n)} \sim p(x, y)$$

synthetic data

Hochreiter S, Schmidhuber J. Long short-term memory. Neural computation. 1997 Nov 15;9(8):1735-80.
Reed S, de Freitas N. Neural programmer-interpreters. arXiv preprint arXiv:1511.06279. 2015 Nov 19.

Captcha Breaking

Type		Baidu (2011)	Baidu (2013)	eBay	Yahoo	reCaptcha	Wikipedia	Facebook
Our method	RR	99.8%	99.9%	99.2%	98.4%	96.4%	93.6%	91.0%
	BT	72 ms	67 ms	122 ms	106 ms	78 ms	90 ms	90 ms
Bursztein et al. [15]	RR	38.68%	55.22%	51.39%	5.33%	22.67%	28.29%	
	BT	3.94 s	1.9 s	2.31 s	7.95 s	4.59 s		
Starostenko et al. [16]	RR				91.5%	54.6%		
	BT					< 0.5 s		
Gao et al. [17]	RR	34%			55%	34%		
Gao et al. [18]	RR		51%		36%			
	BT		7.58 s		14.72 s			
Goodfellow et al. [6]	RR					99.8%		
Stark et al. [8]	RR					90%		

Facebook Captcha

Observed images		(W4kgvQ)	(uV7FeWB)	(MqhnpT)
Inference				
	10^7	W4kgvQ	uV7EeWB	MqhnpT
Training traces	10^6	WA4rjvQ	uV7FeWB	MypppT
	10^5	Woxewd9	mTTEMMm	RIrpES
	10^4	BKvu2Q	C9QDsoN	rS5FP2B

vicarious

$40M raise

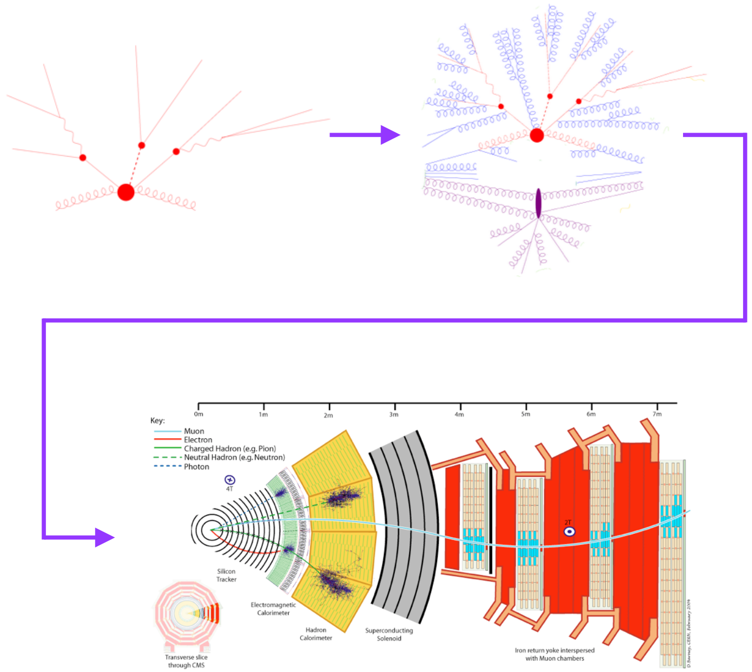
Le TA, Baydin AG, Zinkov R, **Wood** F. Using Synthetic Data to Train Neural Networks is Model-Based Reasoning, IJCNN. 2017.

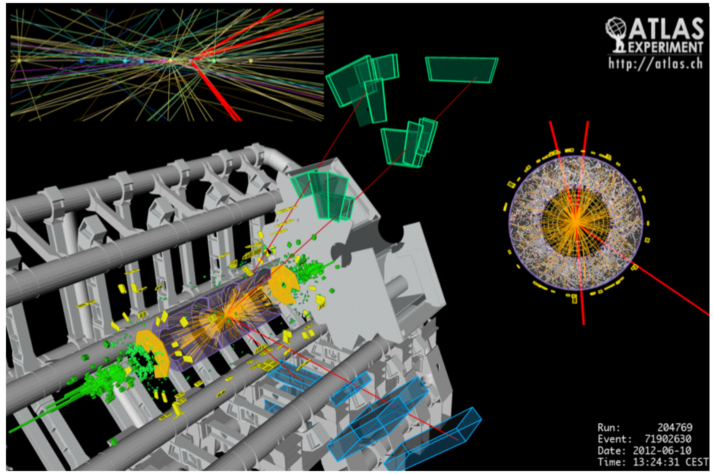
The Quest for New Physics

Inference Compilation : https://github.com/probprog/pyprob

e.g.
Sherpa

e.g.
Geant





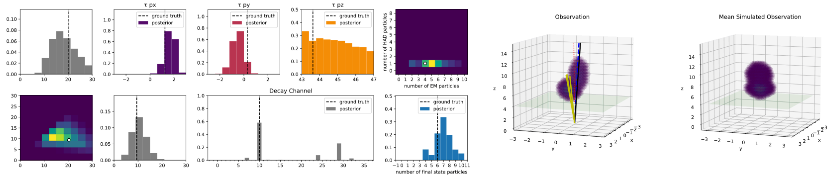
$\mathbf{x}$	$\mathbf{y}$
event & detector simulators	ATLAS detector output

BERKELEY LAB | NERSC National Energy Research Scientific Computing Center | NYU | UBC THE UNIVERSITY OF BRITISH COLUMBIA | intel | ATLAS EXPERIMENT

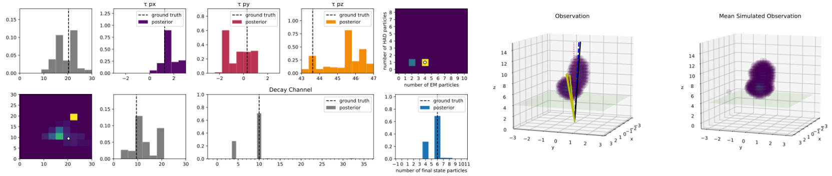
Efficient Probabilistic Inference in the Quest for Physics Beyond the Standard Model. Submitted to ***NIPS.***

High Peaks -- Our Battle to Control a SHERPA

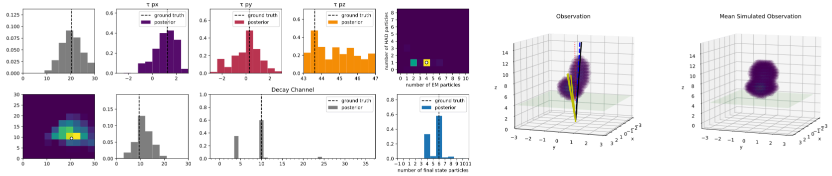
- Controlled by PyProb
  - C++ prior model
    - SHERPA; 1M+ lines
    - Describes standard model
    - Only interface via intercepted U(0,1) RV's
  - Python likelihood
    - ATLAS detector component simulator
- Inf. comp. artifact **first ever** *inference* using LHC generative model software stack
- Neural network SHERPA controller 1000x's more efficient than MH or IS inference; potential for real-time



(a) Inference compilation proposal (9,600 traces).



(b) Inference compilation posterior (9,600 traces).



(c) Ground truth posterior from random-walk Metropolis Hastings (20,000 traces).

Figure 2: The corner plot on the left, shows the particle energies of the two most energetic final state particles and their joint probability. To the right, the distribution of the originating momentum components of the τ lepton and its decay mode is shown. In the middle we show the event composition as characterized by the number of mainly electromagnetically interacting and hadronically interacting final state particles as well as the number of decay products. To the right we show the original observation as well as the mean observation generated during inference.

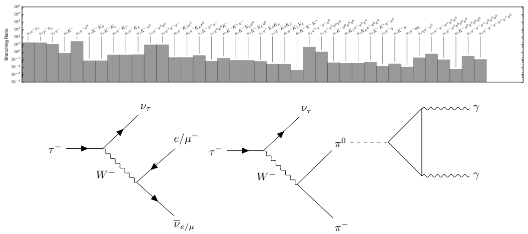
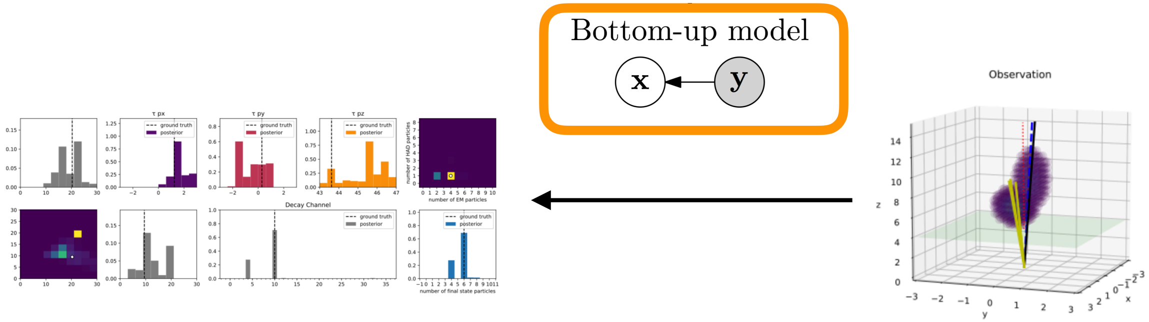


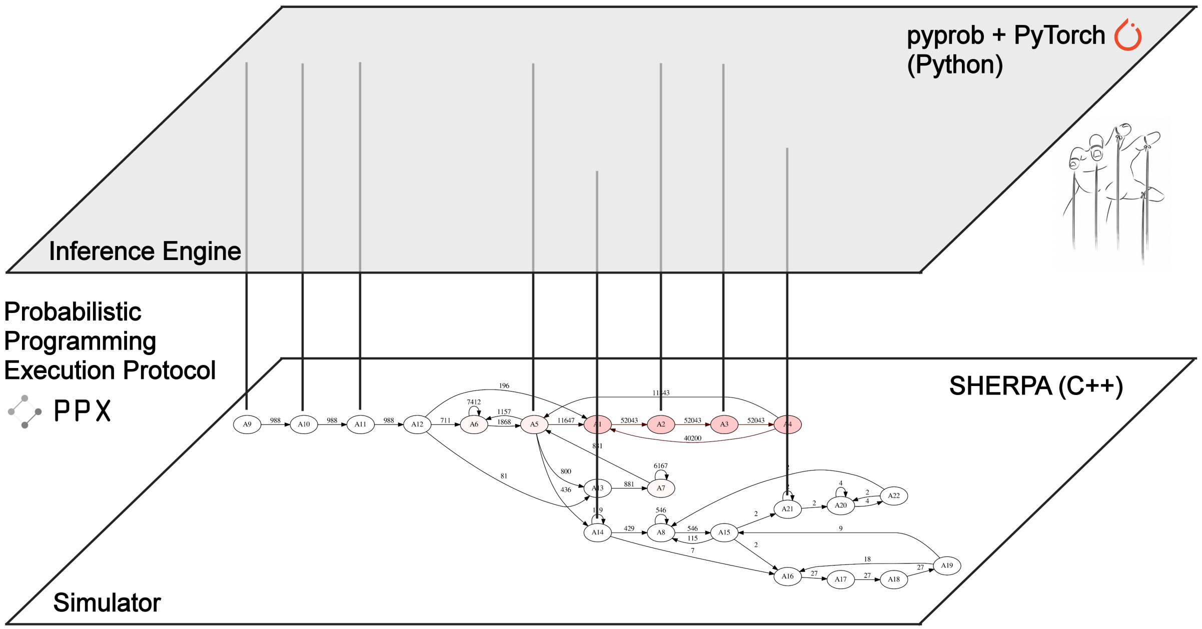
Figure 1: *Top:* branching ratios of the tau lepton, effectively the prior distribution of the decay channels in the SHERPA simulation. Note that the scale is logarithmic. *Bottom:* Feynman diagrams for tau decays illustrating these can produce multiple detected particles.



Efficient Probabilistic Inference in the Quest for Physics Beyond the Standard Model. Submitted to ***NIPS.***

Key Benefit

- Instant interpretability



Code to Try

PySPPL Compiler
https://github.com/Tobias-Kohn/PyPPLCompiler
9692 lines of Python

Compiles a first-order subset of Python to a graphical model

Pyfo
github.com:bradleygramhansen/pyfo
14193 lines of Python

Pure Python STAN replacement + discrete RVs

PyProb_CPP
https://github.com/probprog/pyprob_cpp
762 lines of C++

Lightweight C++ PyProb PPL client

PyProb
github.com:probprog/pyprob.git
4958 lines of Python

Pure Python PPL + Inference Compilation

FOPPLCompiler
git@github.com:probprog/foppl-compiler

Graphical-model inversion

Anglican-lite
https://bitbucket.org/brx/anglican-lite

Clojure FOPPL->graphical model compiler

Anglican
http://anglican.ml

goo.gl/iogdVz

Clojure HOPPL

https://bitbucket.org/probprog/ppaml-summer-school-2016

Challenges

- Sharply peaked likelihoods
  - Adversarial training?
- Efficient forward model execution at test time
  - Surrogates?
- Deep neural network training at scale

Massively Distributed Deep Network Training

Key Ideas

- Asynchronous distributed SGD (aka Hogwild) does not work “at scale”
- Chen et al (Bengio/Google) [2016] suggest “obvious” idea
  - Drop straggling workers in synchronous SGD
  - Provided mini-batch data selection is uncorrelated with worker identity this is completely kosher in expectation

- Our idea
  - Learn a deep nonlinear dynamical system model of cluster performance and use order statistics from said model to drop straggling workers

- TL&DR
  - Higher throughput leads to faster training times despite dropping gradient mini-batch computations
  - Learned model does better than simple heuristics

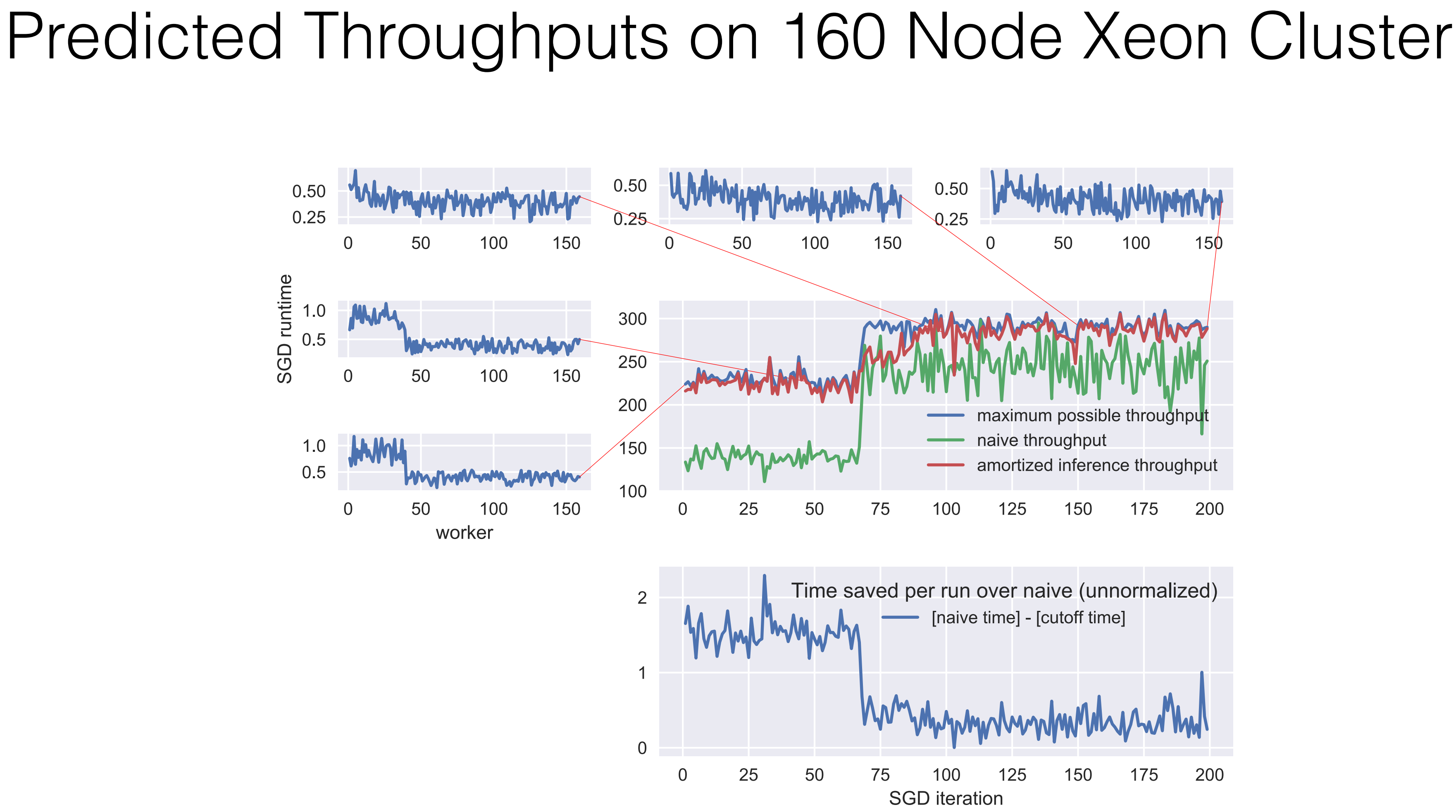


Figure 1: Results of throughputs given by amortized inference. Each runtime plot (5 surrounding the top figure) shows the individual runtimes of the worker (x-axis index) during an iteration of SGD on a 158 node cluster. We highlight SGD iterations 1, 50, 100, 150, and 200 which highlight two significantly different regimes of persistent time-and-machine-identity correlated worker runtimes. The top large figure displays a comparison of throughputs achieved by waiting for all workers to finish (green) and using the inferred cutoff method (red) relative to the ground truth maximum achievable (oracle). The bottom figure displays the reduction in time per iteration when Cutoff SGD is used.

Uniform-Load Cluster-Model Rank Predictions

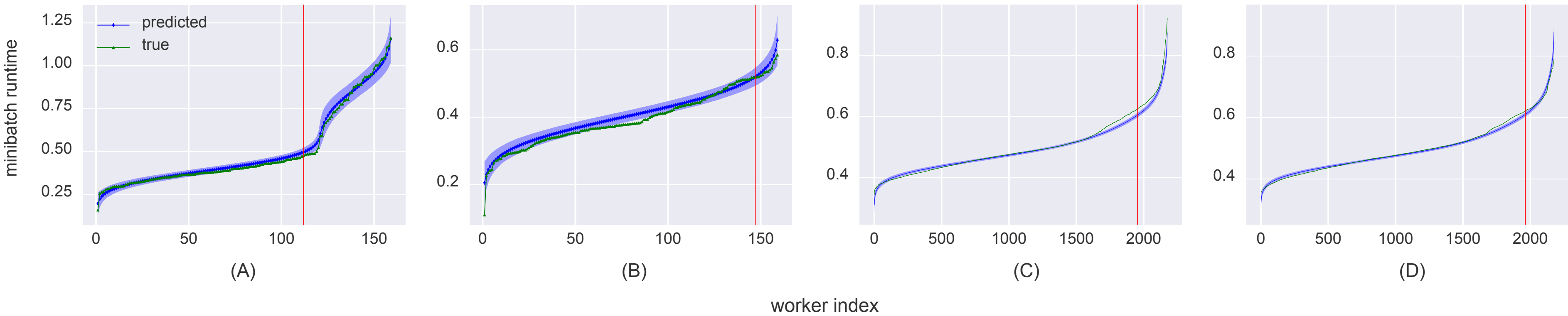


Figure 2: Runtime profiles of various iterations of SGD of the validation set in our training step. The maximum throughput cutoff under the model predictions is shown in red, indicating a large chunk of idle time is reduced as a result of stopping early. (A/B): selected observed runtimes vs predicted runtime order statistics for a 158 node cluster. Notably, when there are exceptionally slow workers present, the cutoff is set to proceed without any of them as seen in figure (A). (C/D): example predicted vs. actual runtimes for 2175 node cluster. All predicted order statistics are shown with ± 2 standard deviations

Improved Training Time on MNIST

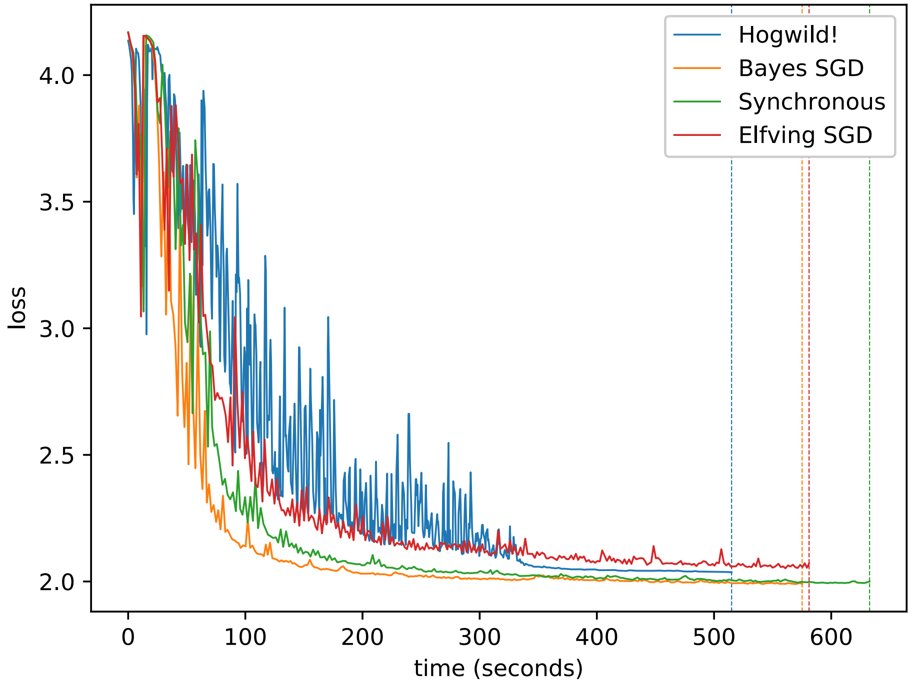


Figure 3: MNIST validation loss convergence for our model based methods, Elfving and Bayesian, and popular approaches. Batch size - 10112, learning rate scaled to 0.64 for sync and 0.004 (0.64 / num workers) for async.

Improved Training Times for Large Neural Networks

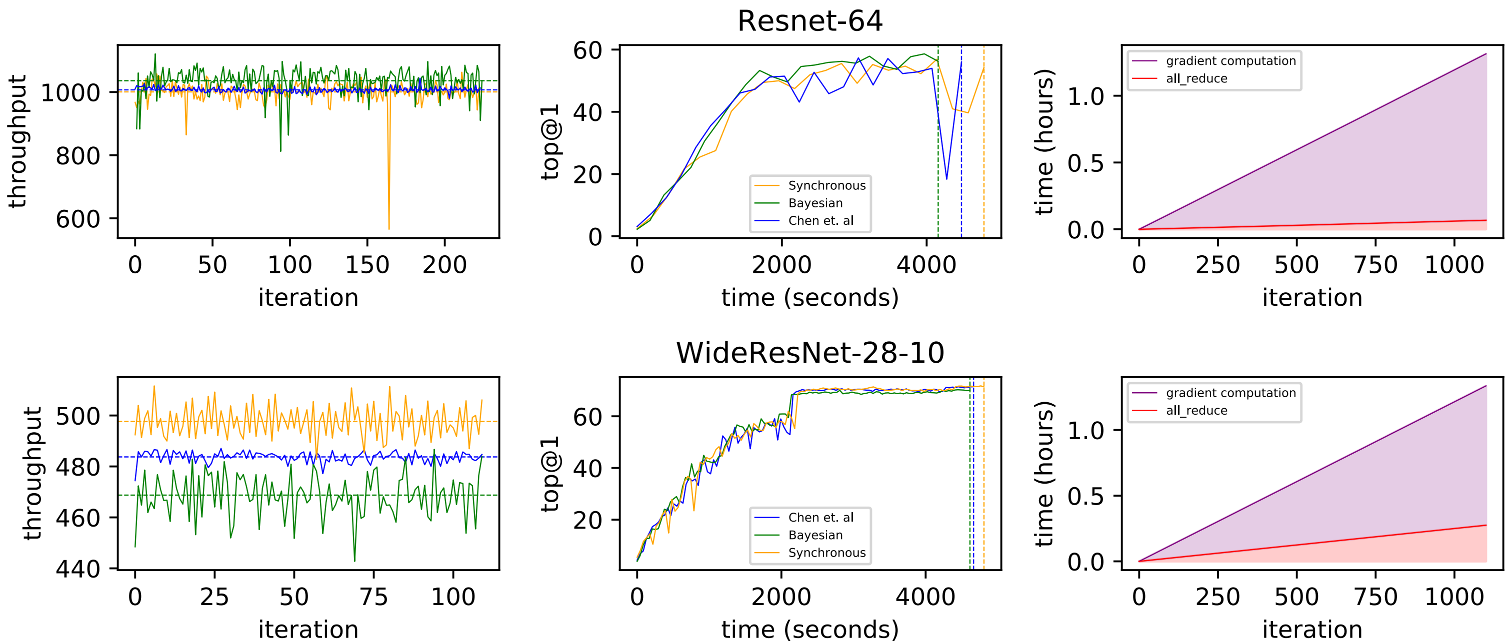


Figure 4: We trained ResNet-64 and the WideResNet with 28 Layers and a width factor of 10. Both networks are trained on CIFAR-100, with a batch-size of 47850 on ResNet-64 and 8700 on WideResNet. All training curves train to 1100 iterations. Initial learning rates for both are set to 0.16, with a 20% decay on WideResNet at t=500 and t=1000. The plots show in order: total throughput over training, wall-clock validation accuracy over training time, and the ratio of training time vs. inter-rank communication cost. On the validation loss curve in the center, dashed vertical lines indicate when the final iteration completed.

Vision

- A learned computational artifact for rapid, even real-time, interpretable LHC event processing
  - Trigger
- A framework for model criticism and new physics using high-quality importance sampling-based evidence estimates
- A framework for efficiently training simulator controllers for various industry applications
  - Leverage existing simulator code
  - Use general purpose compute
  - Useful for realtime anomaly detection, advanced analytics, etc.

Thank You

- People : **Gunes Baydin**, Wahid Bhimji, Lukas Heinrich, Kyle Cranmer, Tuan Anh Le, Jan Willem van de Meent, Hongseok Yang, Brooks Paige, David Tolpin, amongst many others

- Funding : Intel, DARPA, NSERC

Online Learning Rate Adaptation with Hypergradient Descent

Atılım Güneş Baydin[1] **Robert Cornish**[1] **David Martínez Rubio**[2]

Mark Schmidt[3] **Frank Wood**[1]
[1]Department of Engineering Science, University of Oxford, Oxford, United Kingdom
[2]Wadham College, University of Oxford, Oxford, United Kingdom
[3]Department of Computer Science, University of British Columbia, Vancouver, BC, Canada
{gunes,rcornish,fwood}@robots.ox.ac.uk
david.martinez2@wadh.ox.ac.uk schmidtm@cs.ubc.ca

Simple Idea

- Dynamically adjust the learning rate in gradient descent by using automatic differentiation to differentiate wrt the learning rate through the gradient update procedure itself

$$\theta_t = \theta_{t-1} - \alpha \nabla f(\theta_{t-1})$$

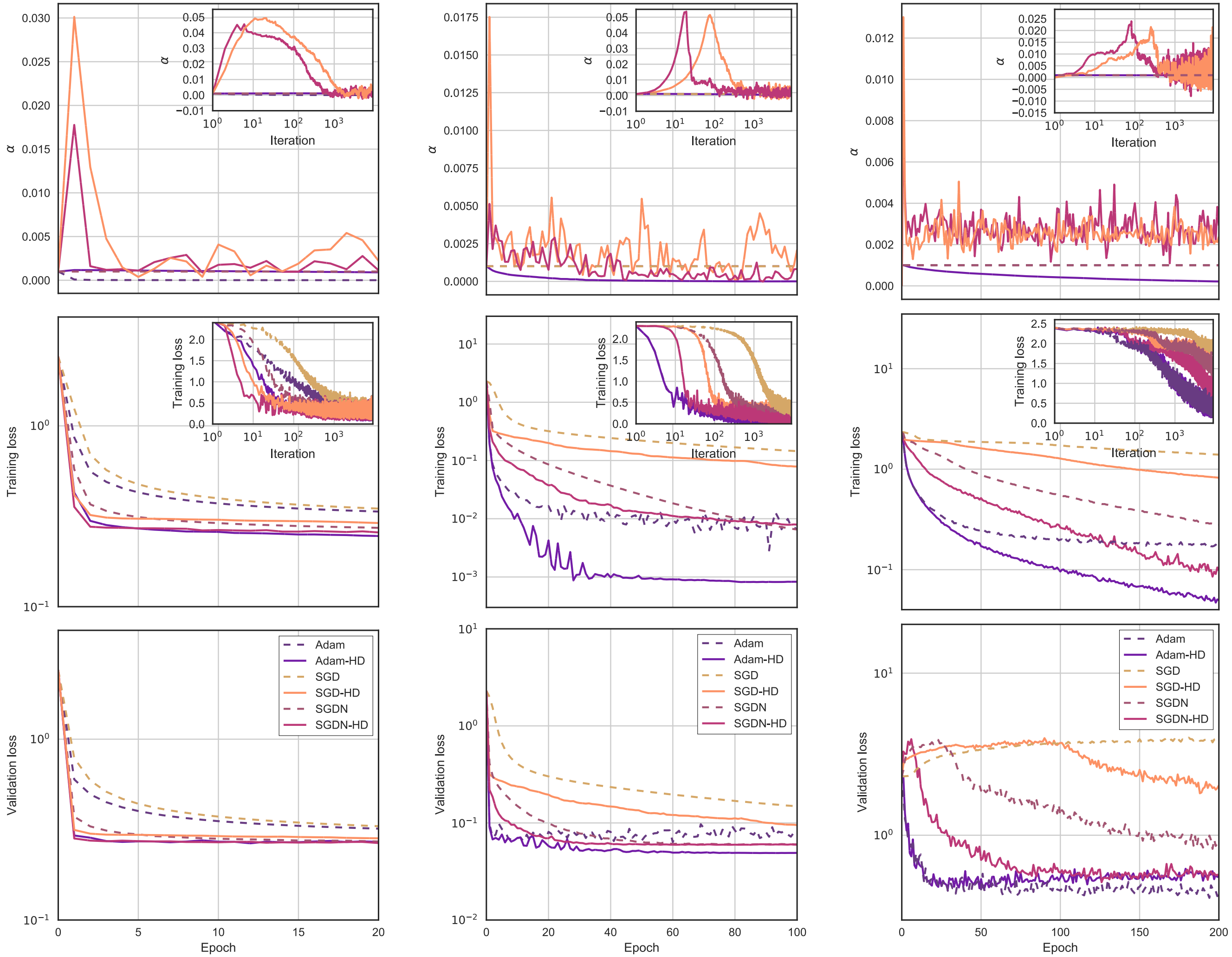
Noting that $\theta_{t-1} = \theta_{t-2} - \alpha \nabla f(\theta_{t-2})$ the chain rule yields

$$\frac{\partial f(\theta_{t-1})}{\partial \alpha} = \nabla f(\theta_{t-1}) \cdot \frac{\partial(\theta_{t-2} - \alpha \nabla f(\theta_{t-2}))}{\partial \alpha} = \nabla f(\theta_{t-1}) \cdot (-\nabla f(\theta_{t-2}))$$

suggesting a simple learning rate update rule

$$\alpha_t = \alpha_{t-1} - \beta \frac{\partial f(\theta_{t-1})}{\partial \alpha} = \alpha_{t-1} + \beta \nabla f(\theta_{t-1}) \cdot \nabla f(\theta_{t-2})$$

. Almeida, Langlois, Amaral, and Plakhov. Parameter adaptation in stochastic optimization. *On-Line Learning in Neural Netwo*



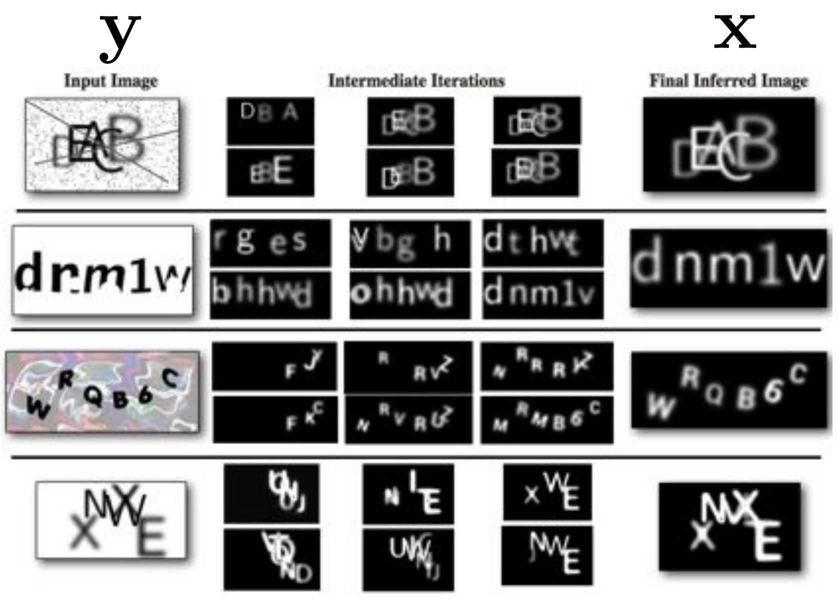
Logistic Regression (MNIST) MLP (MNIST) VGG (CIFAR-10)

Extras

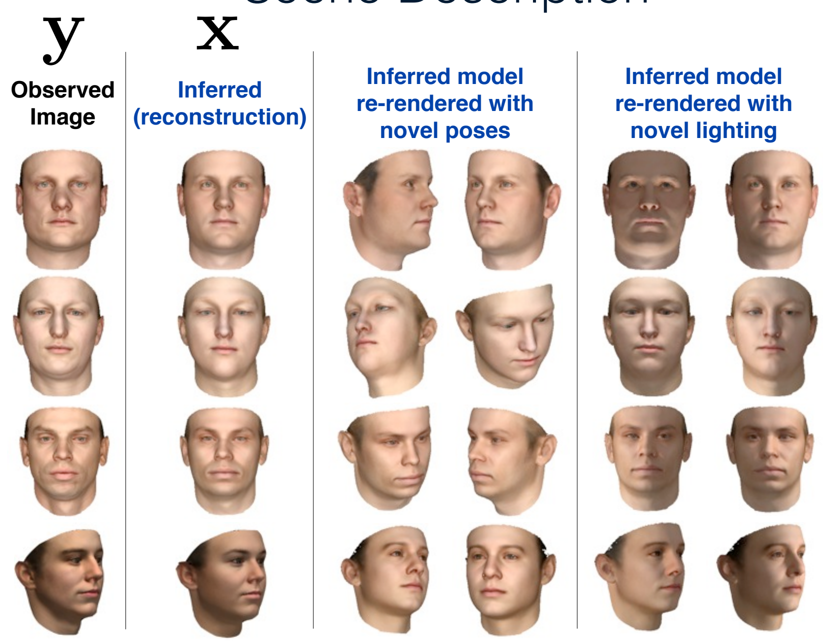
Model-Based Reasoning

Perception / Inverse Graphics

Captcha Solving



Scene Description



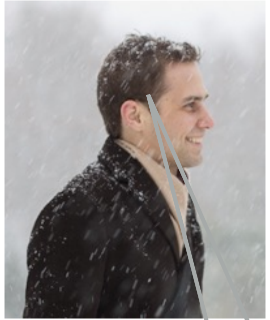
$\mathbf{x}$	$\mathbf{y}$
scene description	image

Mansinghka, Kulkarni, Perov, and Tenenbaum. "Approximate Bayesian image interpretation using generative probabilistic graphics programs." NIPS (2013).

Kulkarni, Kohli, Tenenbaum, Mansinghka "Picture: a probabilistic programming language for scene perception." CVPR (2015).

Reasoning about reasoning

Want to meet up but phones are dead…

I prefer the pub.
Where will Noah go?
Simulate Noah:
Noah prefers pub
but will go wherever Andreas is
Simulate Noah simulating Andreas:
. . .
-> both go to pub

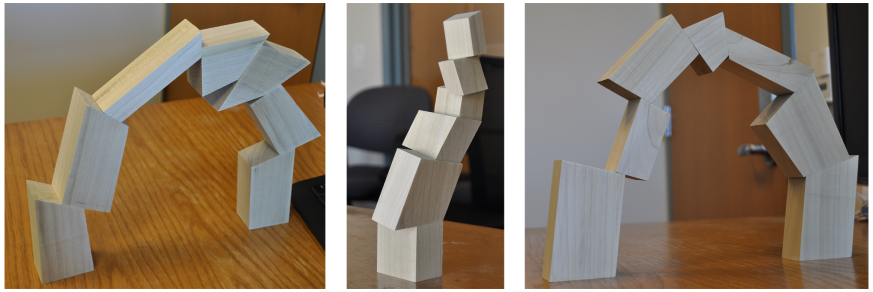
$\mathbf{x}$ …………………………………………………… $\mathbf{y}$

cognitive process — behavior

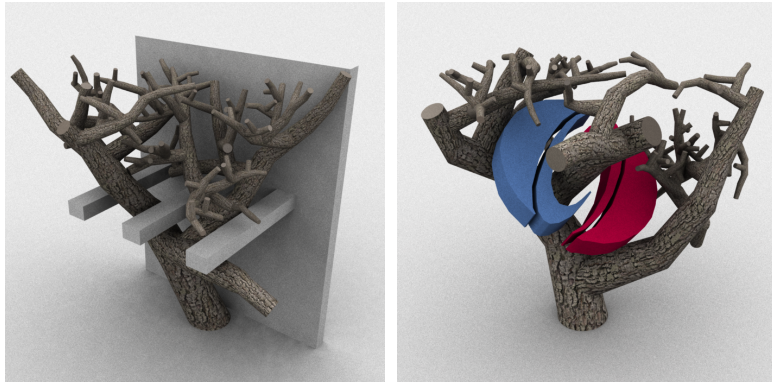
Stuhlmüller, and Goodman.
"Reasoning about reasoning by nested conditioning: Modeling theory of mind with probabilistic programs."
Cognitive Systems Research 28 (2014): 80-99.

Directed Procedural Graphics

Stable Static Structures



Procedural Graphics

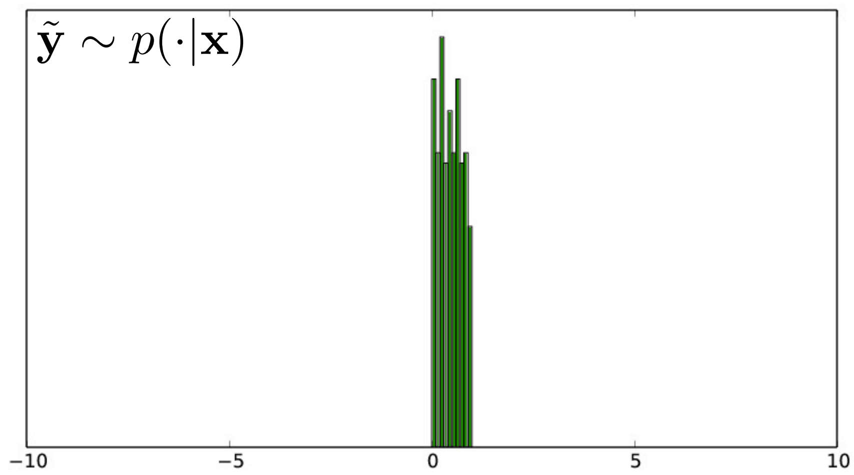


$\mathbf{x}$	$\mathbf{y}$
simulation	constraint

Ritchie, Lin, Goodman, & Hanrahan.
Generating Design Suggestions under Tight Constraints
with Gradient-based Probabilistic Programming.
In Computer Graphics Forum, (2015)

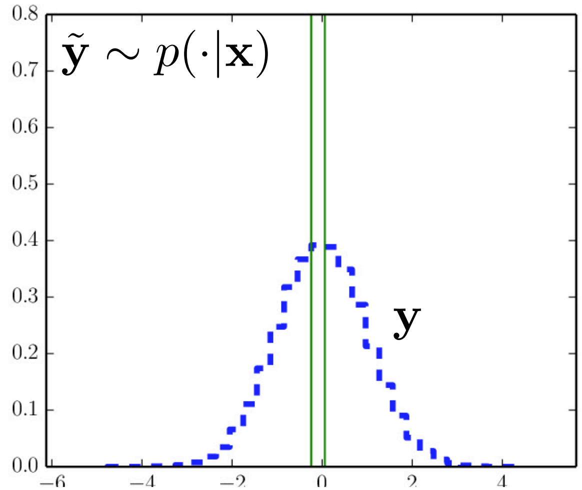
Ritchie, Mildenhall, Goodman, & Hanrahan.
"Controlling Procedural Modeling Programs with
Stochastically-Ordered Sequential Monte Carlo."
SIGGRAPH (2015)

Program Induction




```
(lambda (stack-id) (safe-uc (* (if (< 0.0 (* (* (* -1.0 (begin (define
 G__1147 (safe-uc 1.0 1.0)) 0.0)) (* 0.0 (+ 0.0 (safe-uc (* (* (dec -2
.0) (safe-sqrt (begin (define G__1148 3.14159) (safe-log -1.0)))) 2.0)
 0.0)))) 1.0)) (+ (safe-div (begin (define G__1149 (* (+ 3.14159 -1.0)
 1.0)) 1.0) 0.0) (safe-log 1.0)) (safe-log -1.0)) (begin (define G__11
...
```

$$\mathbf{x} \sim p(\mathbf{x})$$

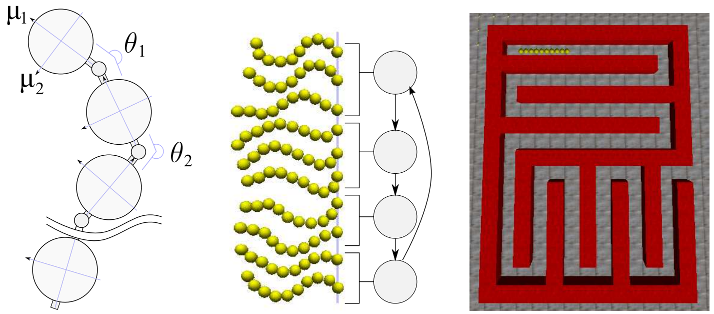


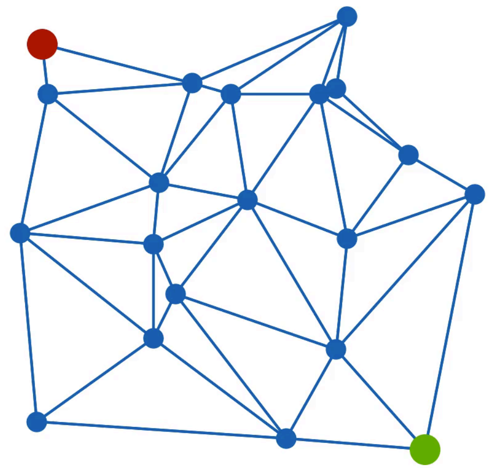
$$\mathbf{x} \sim p(\mathbf{x}|\mathbf{y})$$

$\mathbf{x}$	$\mathbf{y}$
program source code	program output

Perov and Wood.
"Automatic Sampler Discovery via Probabilistic Programming and Approximate Bayesian Computation"
AGI (2016).

Reinforcement Learning





$\mathbf{x}$	$\mathbf{y}$
policy and world	reward

Wingate, Goodman, Roy, Kaelbling, and Tenenbaum.
"Bayesian policy search with policy priors."
(IJCAI), 2011.

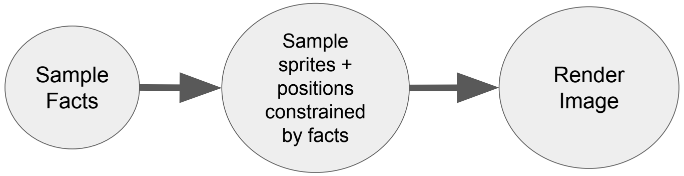
van de Meent, Tolpin, Paige, and Wood.
"Black-Box Policy Search with Probabilistic Programs."
(AISTATS), 2016.

PPAML Week-Long Summer School : 1.5 Days of Student Coding

9 – 10 Intro to Summer School (consent forms, etc.) - **10 - Galois : Overview of PF** **10:30 – 12p** Lecture : Foundations (Galois)	**9 – 10** Lecture Intro to Functional Programming and Clojure **10 – 12p** Hands-On: Functional programming	**9 – 10** Lecture: Introduction to Anglican (Invrea - van de **10 – 12p** Hands-On: Anglican programming	**9 – 10** Lecture: Contributing to Anglican (Invrea - van de **10 – 12p** Hands-On: Project Free Coding	**9 – 12p** Hands-On: Project Free Coding
1:30p – 4p Lecture: Intro to Prob. Prog. (Invrea - Wood) **4p – 5p** Infrastructure Setup (Laptop and VMs)	**1:30p – 2:30p** Lecture: Intro to Generative Modeling **2:30p – 3:30p** Lecture: Intro to Inference (Invrea - Paige) **3:30p – 5p** Hands-On: Probabilistic & Generative Modeling	**1:30p – 2:30p** Project Brainstorming **2:30p – 5p** Hands-On: Anglican Programming	**1:30p – 2:30p** Lecture: Advanced Prob. Prog. (Invrea - Paige) **2:30p – 5p** Hands-On: Project Free Coding	**1:30p – 3p** Hands-On: Project Free Coding **3p – 5p** Project Presentations

Scene Generation

Generative Model for Images

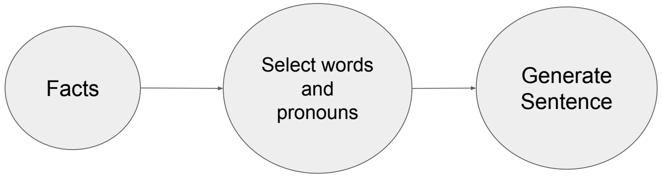


[:faces :bear :boy]
[:kicks :boy :soccer-ball]

{:sprite :boy
:x 100 :y 100
:faces :left}

{:sprite :bear
:x 10 :y 100
:faces :right}

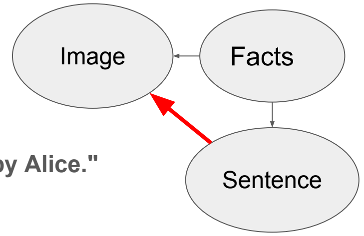
Generative Model for Captions



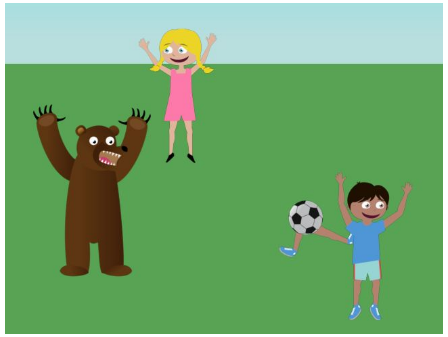
[:faces :bear :girl],
[:close :boy :bear]

"The girl is faced by the bear,
which is close to the boy."

Conditioning



- *Condition image on sentence*
  "The ball kicks Bob while the bear is faced by Alice."

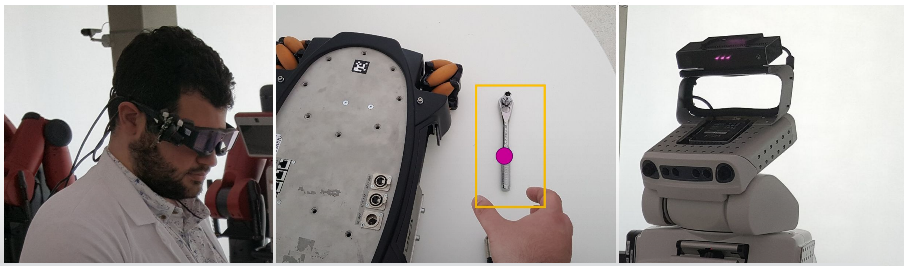


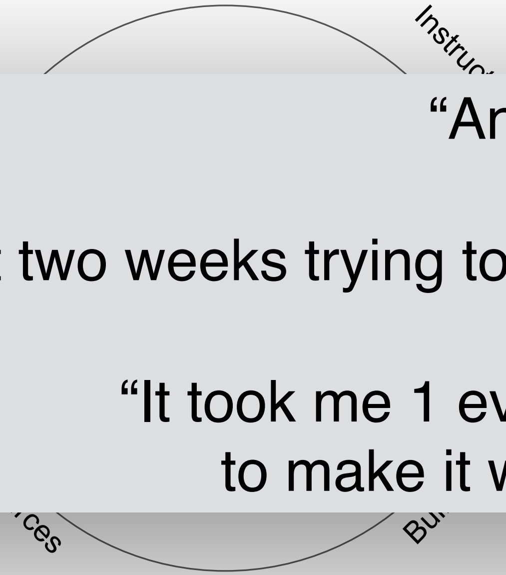
$\mathbf{x}$	$\mathbf{y}$
facts	images and sentences

Alex Collins (acollins@nvidia.com)
Alex Ledger (a.led1027@gmail.com)
Timon Gehr (timon.gehr@gmail.com)

Task&Gaze-Directed Object Localization

Problem: Find the location of objects/regions with an unknown appearance.

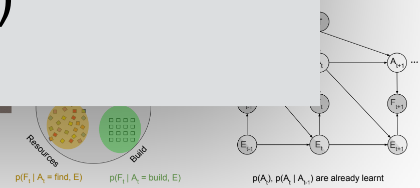




“Anglican = awesome”

“Spent two weeks trying to get the model working with Figaro / Scala”

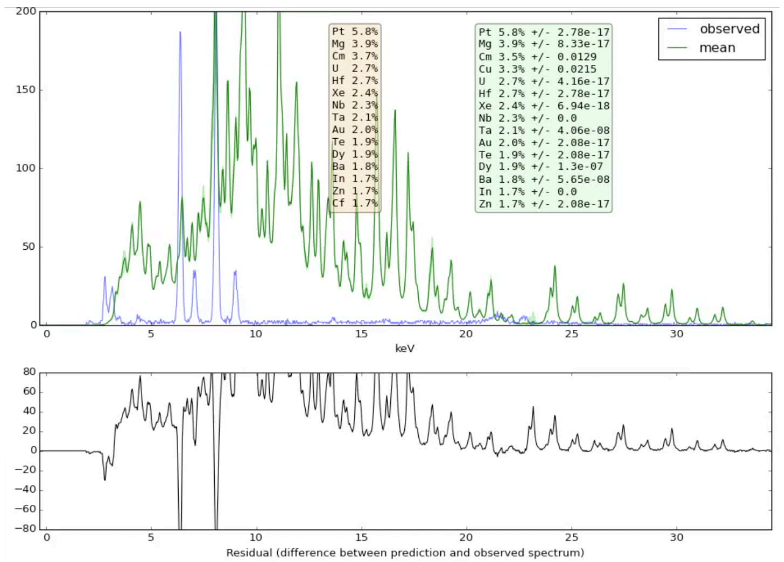
“It took me 1 evening (at the bar with cocktails)
to make it work with Anglican / Clojure”



$\mathbf{x}$	$\mathbf{y}$
unknown locations	fixations

Svetlin Penkov (sv.penkov@ed.ac.uk)

Rock Composition Via X-ray Fluorescence



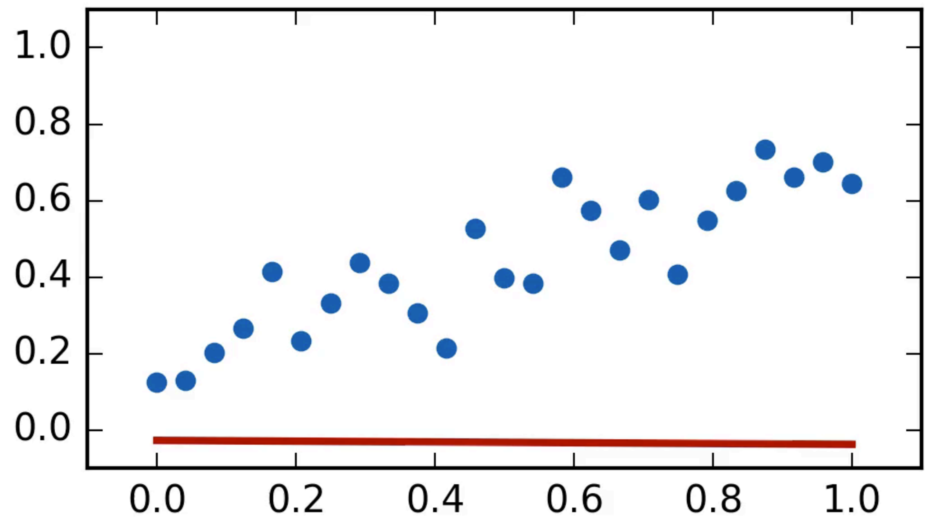
$\mathbf{x}$	$\mathbf{y}$
rock composition	x-ray fluorescence

Matthew Dirks
mcdirks@cs.ubc.ca

minesense

Thinking Generatively about Discriminative Tasks

```
(defquery lin-reg [x-vals y-vals]
 (let [m (sample (normal 0 1))
       c  (sample (normal 0 1))
       f (fn [x] (+ (* m x) c))]
  (map (fn [x y]
          (observe
            (normal (f x) 0.1) y))
         x-vals y-vals))
  [m c])
```




```
(doquery :ipmcmc lin-reg data options)
```

([0.58 -0.05] [0.49 0.1] [0.55 0.05] [0.53 0.04] ….

MD5 Inversion

```
(defquery md5-inverse [L md5str]
    "conditional distribution of strings
     that map to the same MD5 hashed string"
    (let [mesg (sample (string-generative-model L))]
      (observe (dirac md5str) (md5 mesg))
     mesg)))
```

Decision-Making Under Uncertainty

INVREA

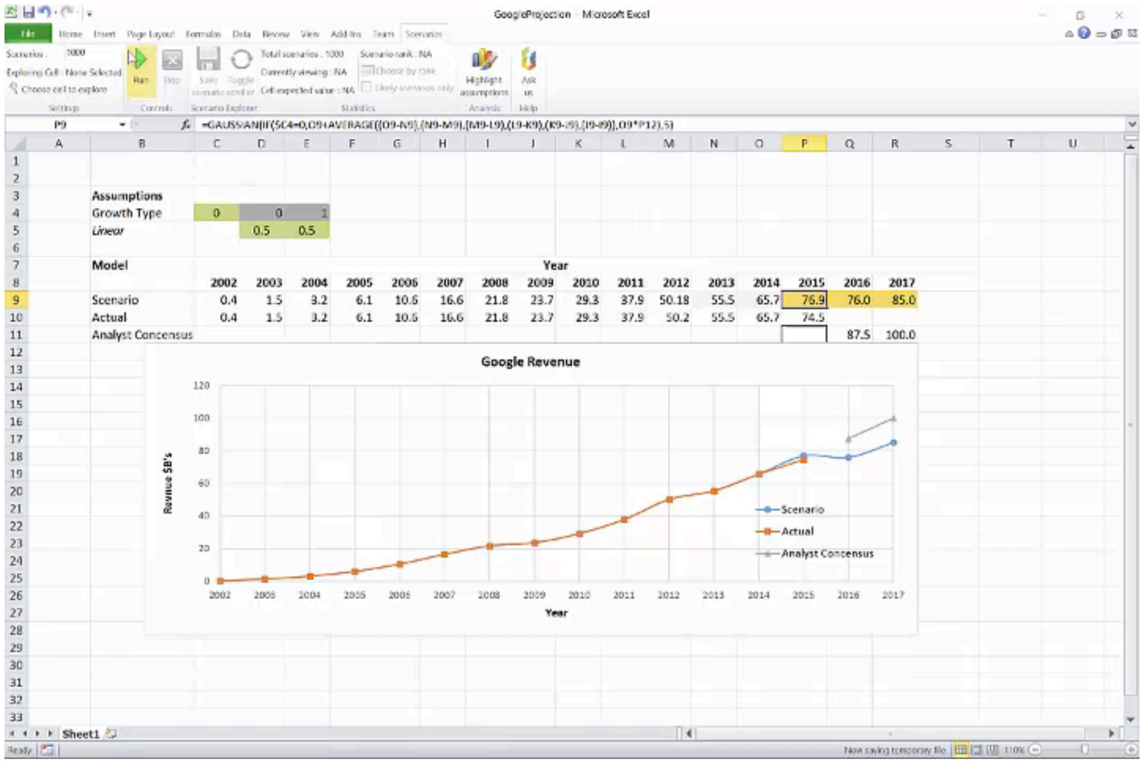
Make Better Decisions

£23B/yr risk analytics market

@RISK

ORACLE

Crystal Ball



x	y
spreadsheet model	actuals

https://invrea.com/plugin/excel/v1/download/

Why Model-Based Reasoning?

- Interpretability
- Source of labeled data
- Regularization
  - Computation structure

- Domain knowledge

Inference Compilation

First: Graphical Model Inference

Goal: efficient posterior inference in generative models with latent variables $\mathbf{x}$ and observed variables $\mathbf{y}$

$$p(\mathbf{x}, \mathbf{y}) \triangleq \prod_{i=1}^{N} p\left(x_i | \mathrm{PA}(x_i)\right) \prod_{j=1}^{M} p\left(y_j | \mathrm{PA}(y_j)\right)$$

Inference

Goal: efficient posterior inference in generative models with latent variables $\mathbf{x}$ and observed variables $\mathbf{y}$

$$p(\mathbf{x}, \mathbf{y}) \triangleq \prod_{i=1}^{N} p\left(x_i \middle| \mathrm{PA}(x_i)\right) \prod_{j=1}^{M} p\left(y_j \middle| \mathrm{PA}(y_j)\right)$$

e.g. importance sampling and SMC approximate the posterior $\pi(\mathbf{x}) \equiv p(\mathbf{x}|\mathbf{y})$ as weighted samples

$$\hat{p}(\mathbf{x}|\mathbf{y}) = \sum_{k=1}^{K} W_k \delta_{\mathbf{x}_k}(\mathbf{x}) \qquad w(\mathbf{x}) = \frac{p(\mathbf{x}, \mathbf{y})}{q(\mathbf{x}|\lambda)} \qquad W_k = \frac{w(\mathbf{x}_k)}{\sum_{j=1}^{K} w(\mathbf{x}_j)}$$

Performance depends on quality of proposal $q(\mathbf{x}|\lambda)$

One Notion of Optimal Proposal

Learning an importance sampling proposal for a single dataset

Target density $\pi(\mathbf{x}) = p(\mathbf{x}|\mathbf{y})$, approximating family $q(\mathbf{x}|\lambda)$

Single dataset $\mathbf{y}$: $\underset{\lambda}{\operatorname{argmin}} D_{KL}(\pi||q_\lambda)$ ← fit λ to learn an importance sampling proposal

$D_{KL}[q_\varphi(\mathbf{x}|\mathbf{y})||p_\theta(\mathbf{x}|\mathbf{y})]$ ← Note: opposite KL to VB/VAE

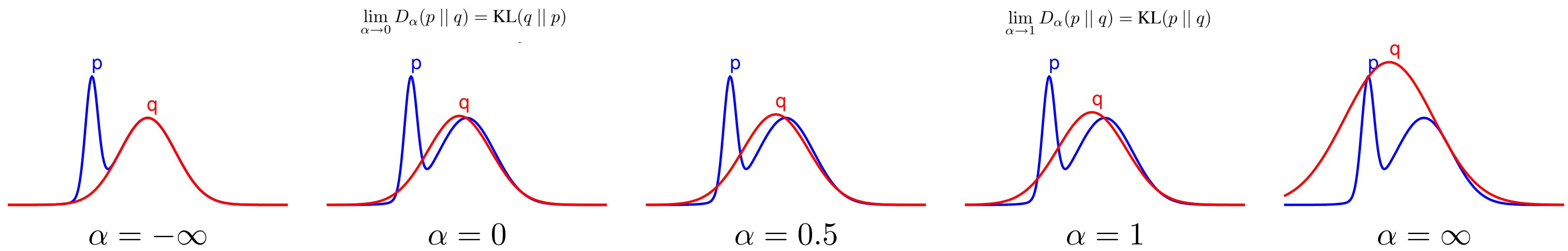


Figure 1: The Gaussian q which minimizes α-divergence to p (a mixture of two Gaussians), for varying α. $\alpha \to -\infty$ prefers matching one mode, while $\alpha \to \infty$ prefers covering the entire distribution.

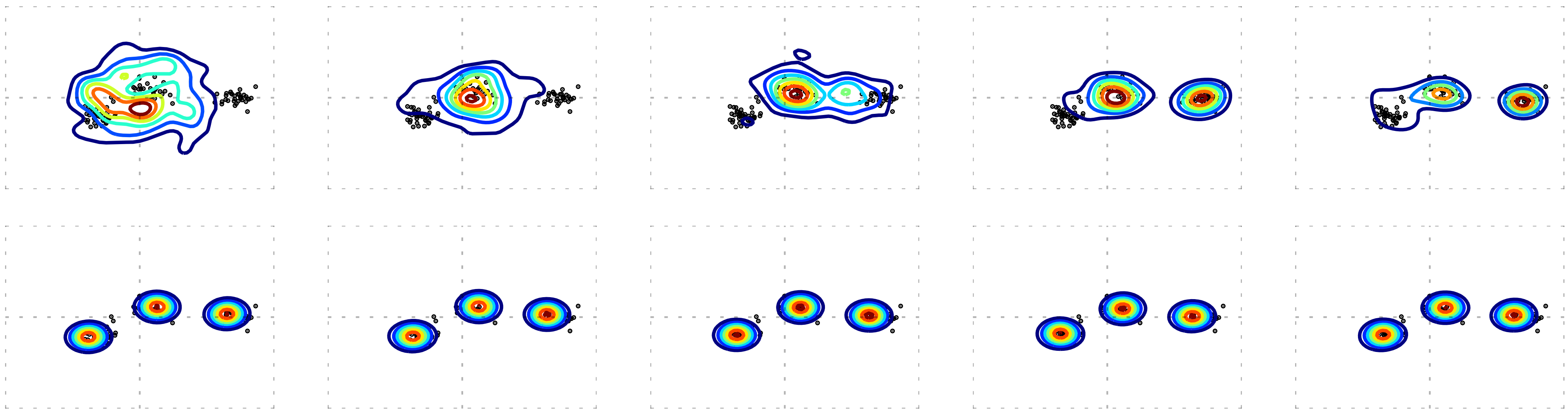
Minka T. Divergence measures and message passing. Technical report, Microsoft Research; 2005 Dec 7.

Cornebise J, Moulines E, Olsson J. Adaptive methods for sequential importance sampling with application to state space models. *Statistics and Computing*, 18:461–480 (2008)

Open-Universe Gaussian Mixture Model

procedure GMM
  $K \sim p(K|\cdot)$ ▷ `sample` number of clusters
  for $k = 1, \ldots, K$ **do**
    $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k \sim p(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k|\cdot)$ ▷ `sample` cluster parameters
  $\boldsymbol{\pi} \leftarrow \text{uniform}(1, K)$
  for $n = 1, \ldots, N$ **do**
    $z_n \sim p(z_n|\boldsymbol{\pi})$ ▷ `sample` class label
    $y_n \sim p(y_n|z_n = k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ ▷ `sample` *or* `observe` data
  return $\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}, K$

GMM Inference



Particles: 1 10 100 1000 10000

Kernel density estimation of the distribution over maximum a-posteriori values of the means $\{\max_{\mu_k} p(\mu_k|\mathbf{y})\}_{k=1}^{3}$ over 50 independent runs

Top: SMC
Bottom: CSIS

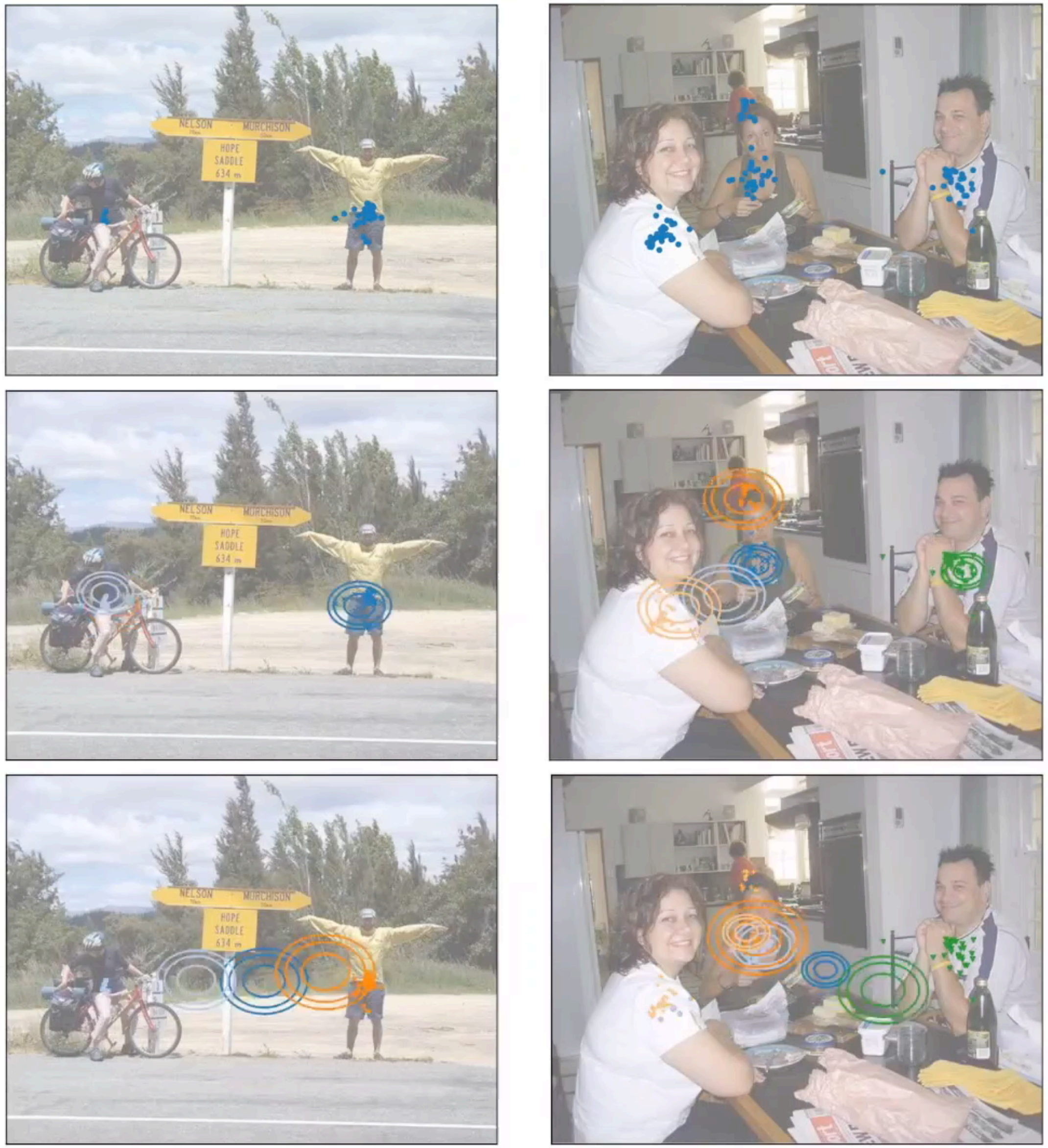
Effect of Training Duration

Observed points

10^7

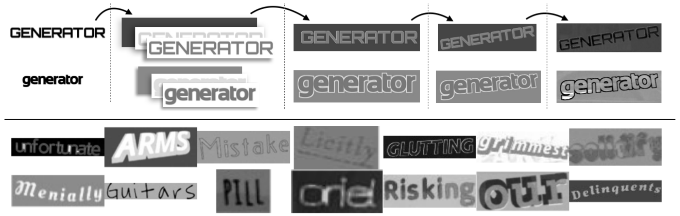
Training traces

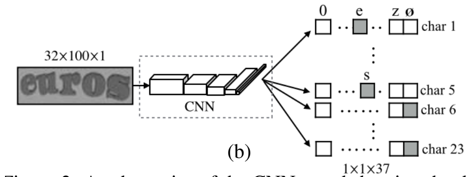
10^4



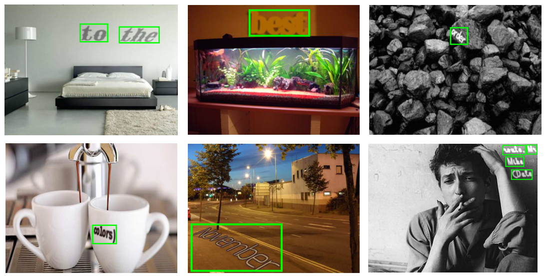
Synthetic Data for Training Deep Networks

Goodfellow, Bulatov, Ibarz, Arnoud, Shet; Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks. 2014.





Jaderberg, Simonyan, Vedaldi, Zisserman; Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition. 2014.



Gupta, Vedaldi, Zisserman; Synthetic Data for Text Localisation in Natural Images. 2016.

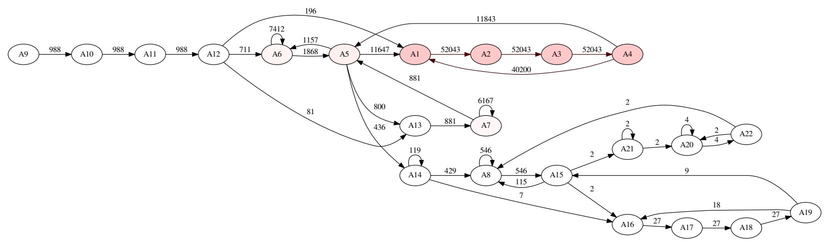
Le, Baydin, Zinkov & **Wood**. ***Using Synthetic Data to Train Neural Networks is Model-Based Reasoning*** IJCNN 2017.

Advanced Topics Take-Homes

- If you have an existing simulator it is, in principle, possible to perform inference in it now (without re-coding it), using it as a prior in a Bayesian sense
- Amortized inference is powerful and works for the same reason that deep neural networks trained on synthetic data work

Challenges

- Probabilistic Programming
  - Nesting/compositionality
  - Automatically factoring inference
  - Finite approximations
- Inference Compilation
  - Inference network structure from generative model or vice versa
  - Forward model surrogates
- Model Learning
  - PLs that support
    - AESMC-style ELBOs
    - Model learning through discrete random primitives



Recent Papers

Published

1. M. Igl, L. Zintgraf, T.A. Le, F. Wood, and S. Whiteson. Deep variational reinforcement learning for POMDPs. In *ICML*, 2018.
2. T. Rainforth, A. Kosiorek, T. A. Le, C Maddison, M Igl, F Wood, and Y.W. Teh. Tighter variational bounds are not necessarily better. In *ICML*, 2018.
3. T. Rainforth, R. Cornish, H. Yang, and F. Wood. On nesting Monte Carlo estimators. In *ICML*, 2018.
4. G. Baydin, R. Cornish, D. Martinez-Rubio, M. Schmidt, and F. Wood. Online learning rate adaptation with hypergradient descent. *ICLR*, 2018.
5. T.A. Le, M. Igl, T. Jin, T Rainforth, and F. Wood. Auto-encoding Sequential Monte Carlo. *ICLR*, 2018.

Submitted

1. Revisiting Reweighted Wake-Sleep. Submitted to *NIPS; on arXiv*
2. **Faithful Inversion of Generative Models for Effective Amortized Inference. Submitted to *NIPS; on arXiv***
3. Bayesian Distributed Stochastic Gradient Descent. Submitted to *NIPS*
4. Efficient Probabilistic Inference in the Quest for Physics Beyond the Standard Model. Submitted to *NIPS.*
5. Hamiltonian Monte Carlo for Probabilistic Programs with Discontinuities. Submitted to *NIPS; on arXiv*
6. **Inference Trees: Adaptive Inference with Exploration. Submitted to *NIPS.***

Ms Theses

1. W. Harvey. Automatic Ingestion of Plot Data, MS Thesis, Oxford, 2018
2. **A. Spencer. Probabilistic Simulation of Neural Dynamics, MS Thesis, Oxford, 2018**

TL&DR

- Programming languages can be used to denote inference problems
- There are at least two families of probabilistic programming languages; one can be compiled to graphical models or factor graphs, the other, corresponding in character to normal, everyday programming languages, cannot
- It is possible to develop generic and reasonably efficient inference algorithms for both families
- There is a rapidly emerging connection between probabilistic programming, variational inference, and differential programming that could give rise to the next generation of AI tools
- There are all kinds of interesting research and engineering challenges remaining