AFRL-RI-RS-TR-2019-108

HASTY – A GENERATIVE MODEL COMPILER

UNIVERSITY OF OXFORD

MAY 2019

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2019-108 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
MICHAEL MANNO
Work Unit Manager

/ S /
TIMOTHY A. FARRELL
Deputy Chief, Information Intelligence
Systems and Analysis Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
MAY 2019	FINAL TECHNICAL REPORT	SEP 2017 – JAN 2019

4. TITLE AND SUBTITLE

HASTY – A GENERATIVE MODEL COMPILER

5a. CONTRACT NUMBER
FA8750-17-2-0093

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
62702E

6. AUTHOR(S)

Frank Wood, Michael Teng, and Rob Zinkov

5d. PROJECT NUMBER
D3ME

5e. TASK NUMBER
00

5f. WORK UNIT NUMBER
05

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Oxford
Department of Engineering Science
South Parks Rd
Oxford, OX1 2JD, UK

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RIED
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSOR/MONITOR'S REPORT NUMBER
AFRL-RI-RS-TR-2019-108

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This work describes our contribution of proof of concept primitives to the D3M program and research progress made towards an initial version of Hasty. Although we were unable to complete the initial version of Hasty, or contribute to the D3M primitive library the types of primitives that Hasty will enable we did train a number of Highly Qualified Personnel (HQP) and have interacted with the AutoML, probabilistic programming languages, neural networking, and other communities which our work is expected to impact.

15. SUBJECT TERMS
Data modeling primitives, automated selection of data modeling primitives, automated data model composition, machine learning for data model composition, human data model interaction, model composition user interfaces.

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**MICHAEL MANNO**
U	U	U	UU	16	19b. TELEPHONE NUMBER (*Include area code*) **N/A**

TABLE OF CONTENTS

Section **Page**

1.0 SUMMARY

This final technical report summarizes the conducted research and completed software output by the University of Oxford in the course of the DARPA Data Driven Discovery (D3M) program.

The main completed deliverables can be summarized as the contribution of proof of concept primitives to the D3M program and research progress made towards an initial version of Hasty. As of the completion of the Oxford contract we have not yet completed an initial version of Hasty, nor have we contributed to the D3M primitive library the types of primitives that Hasty will enable.

We did, however, train a number of Highly Qualified Personnel (HQP) and have interacted with the AutoML, probabilistic programming languages, neural networking, and other communities which our work is expected to impact. This is summarized in section 2 of this document.

Finally, we have produced several peer-reviewed research papers which were accepted to high-impact conferences such as NIPS, ICML, ICLR, UAI, and AISTATS. These papers and their research relation to the D3M program is summarized in section 3. In section 4, we summarize the software projects whose development has been supported in preparation for developing the Hasty compiler.

2.0 INTRODUCTION

We are building an extensive set of model primitives and contributing them to the D3M library of discoverable models. We are developing a software toolchain that makes it realistic to vastly extend the set of model primitives by making it possible for not only our team but all other teams to efficiently develop and deploy new model primitives. Our method of choice for extending the set of model primitives will be to use our own toolchain while it is in development. The toolchain we are developing is a compiler that makes bottom-up, discriminative models from top-down, generative models specified in the form of probabilistic programs. We call this compiler and runtime Hasty.

3.0 METHODS, ASSUMPTIONS, AND PROCEDURES

3.1.1 Inference Trees: Adaptive Inference with Exploration

T. Rainforth, Y. Zhou, X. Lu, Y. W. Teh, F. Wood, H. Yang, and J.-W.van de Meent. Inference trees: Adaptive inference with exploration. NIPS Workshop: Advances in Approximate Bayesian Inference, 2017. [8]

We introduce inference trees (ITs), a new adaptive Monte Carlo inference method building on ideas from Monte Carlo tree search. Unlike most existing methods which are implicitly based on pure exploitation, ITs explicitly aim to balance exploration and exploitation in the inference process, alleviating common pathologies and ensuring consistency. More specifically, ITs use bandit strategies to adaptively sample from hierarchical partitions of the parameter space, while simultaneously learning these partitions in an online manner. This allows ITs to “hone-down" on regions of interest, but also maintain uncertainty estimates on whether regions of significant posterior mass have been missed. Though they can be used more generally, we show that ITs are particularly effective when combined with sequential Monte Carlo (SMC), allowing long-range dependencies to be captured and potentially improving beyond what can be achieved by proposal adaptation alone.

3.1.2 Auto-Encoding Sequential Monte Carlo

T.A. Le, M. Igl, T. Jin, T Rainforth, and F. Wood. Auto-encoding Sequential Monte Carlo. ICLR, 2018. [4]

We build on auto-encoding sequential Monte Carlo (AESMC): a method for model and proposal learning based on maximizing the lower bound to the log marginal likelihood in a broad family of structured probabilistic models. Our approach relies on the efficiency of sequential Monte Carlo (SMC) for performing inference in structured probabilistic models and the flexibility of deep neural networks to model complex conditional probability distributions. We develop additional theoretical insights and introduce a new training procedure which improves both model and proposal learning. We demonstrate that our approach provides a fast, easy-to-implement and scalable means for simultaneous model learning and proposal adaptation in deep generative models.

3.1.3 Faithful Inversion of Generative Models for Effective Amortized Inference

S. Webb, A. Golinski, R. Zinkov, S. Narayanaswamy, T. Rainforth, Y. W.Teh, and F. Wood. Faithful inversion of generative models for effective amortized inference. NIPS, 2018. [10]

Inference amortization methods share information across multiple posterior inference problems, allowing each to be carried out more efficiently. Generally, they require the inversion of the dependency structure in the generative model, as the modeler must learn a mapping from observations to distributions approximating the posterior. Previous approaches have involved inverting the dependency structure in a heuristic way that fails to capture these dependencies correctly, thereby limiting the achievable accuracy of the resulting approximations. We introduce an algorithm for faithfully, and minimally, inverting the graphical model structure of any generative model. Such inverses have two crucial properties: (a) they do not encode any independence assertions that are absent from the model and; (b) they are local maxima for the number of true independencies encoded. We prove the correctness of our approach and empirically show that the resulting minimally faithful inverses lead to better inference amortization than existing heuristic approaches.

3.2 AutoML

3.2.1 Training of Differentiable Pipelines Across Machine Learning Frameworks

Mitar Milutinovic, At_l_m G□une_s Baydin, Robert Zinkov, William Harvey, Dawn Song, Frank Wood, Wade Shen. End-to-end Training of Di_erentiable Pipelines Across Machine Learning Frameworks. NIPS, 2017. [5]

In this work we present a unifed interface and methodology for performing end-to-end gradient-based refinement of pipelines of differentiable machine-learning primitives. This is distinguished from recent interoperability efforts such as the Open Neural Network Exchange (ONNX) format and other language centric cross-compilation approaches in that the final pipeline does not need to be implemented nor trained in the same language nor cross-compiled into any single language; in other words, primitives may be written and pre-trained in PyTorch, TensorFlow, Ca_e, scikit-learn or any of the other popular machine learning frameworks and _ne-tuned end-to-end while being executed directly in their host frameworks. Provided primitives expose our proposed interface, it is possible to automatically compose all such primitives and refine them based on an end-to-end loss.

3.2.2 Online learning rate adaptation with hypergradient descent

A. G. Baydin, R. Cornish, D. M. Rubio, M. Schmidt, and F. Wood. On-line learning rate adaptation with hypergradient descent. ICLR, 2018. [1]

We introduce a general method for improving the convergence rate of gradient based optimizers that is easy to implement and works well in practice. We demonstrate the effectiveness of the method in a range of optimization problems by applying it to stochastic gradient descent, stochastic gradient descent with Nesterov momentum, and Adam, showing that it significantly reduces the need for the manual tuning of the initial learning rate for these commonly used algorithms. Our method works by dynamically updating the learning rate during optimization using the gradient with respect to the learning rate of the update rule itself. Computing this "hypergradient" needs little additional computation, requires only one extra copy of the original gradient to be stored in memory, and relies upon nothing more than what is provided by reverse-mode automatic differentiation.

3.3 Probabilistic Programming Languages

3.3.1 LF-PPL: A Low-Level First Order Probabilistic Programming Language for Non-Differentiable Models

Y. Zhou, B. J. Gram-Hansen, T. Kohn, T. Rainforth, H. Yang, and F. Wood. Lf-ppl: A Low-level First Order Probabilistic Programming Language For Non differentiable Models. AISTATS, 2019. [11]

We develop a new Low-level, First-order Probabilistic Programming Language (LF-PPL) suited for models containing a mix of continuous, discrete, and/or piecewise-continuous variables. The key success of this language and its compilation scheme is in its ability to automatically distinguish parameters the density function is discontinuous with respect to, while further providing runtime checks for boundary crossings. This enables the introduction of new inference engines that are able to exploit gradient information, while remaining efficient for models which are not everywhere differentiable. We demonstrate this ability by incorporating a discontinuous Hamiltonian Monte Carlo (DHMC) inference engine that is able to deliver automated and efficient inference for non-differentiable models. Our system is backed up by a mathematical formalism that ensures that any model expressed in this language has a density with measure zero discontinuities to maintain the validity of the inference engine.

3.3.2 Inference Compilation and Universal Probabilistic Programming

T. Le, A. Baydin, and F. Wood. Inference Compilation and Universal Probabilistic Programming. AISTATS, 2017. [3]

We introduce a method for using deep neural networks to amortize the cost of inference in models from the family induced by universal probabilistic programming languages, establishing a framework that combines the strengths of probabilistic programming and deep learning methods. We call what we do "compilation of inference" because our

method transforms a denotational specification of an inference problem in the form of a probabilistic program written in a universal programming language into a trained neural network denoted in a neural network specification language. When at test time this neural network is fed observational data and executed, it performs approximate inference in the original model specified by the probabilistic program. Our training objective and learning procedure are designed to allow the trained neural network to be used as a proposal distribution in a sequential importance sampling inference engine. We illustrate our method on mixture models and Captcha solving and show significant speedups in the efficiency of inference.

3.4 Deep Generative Models

3.4.1 Bayesian Distributed Stochastic Gradient Descent

M. Teng and F. Wood. Bayesian Distributed Stochastic Gradient Descent. NeurIPS, 2018. [9]

We introduce Bayesian distributed stochastic gradient descent (BDSGD), a high-throughput algorithm for training deep neural networks on parallel clusters. This algorithm uses amortized inference in a deep generative model to perform joint posterior predictive inference of mini-batch gradient computation times in a compute cluster specific manner. Specifically, our algorithm mitigates the straggler effect in synchronous, gradient-based optimization by choosing an optimal cutoff beyond which mini-batch gradient messages from slow workers are ignored. In our experiments, we show that eagerly discarding the mini-batch gradient computations of stragglers not only increases throughput but actually increases the overall rate of convergence as a function of wall-clock time by virtue of eliminating idleness. The principal novel contribution and finding of this work goes beyond this by demonstrating that using the predicted run-times from a generative model of cluster worker performance improves substantially over the static-cutoff prior art, leading to reduced deep neural net training times on large computer clusters.

3.4.2 Tighter variational bounds are not necessarily better.

T. Rainforth, A. R. Kosiorek, T. A. Le, C. J. Maddison, M. Igl, F. Wood,and Y. W. Teh. Tighter variational bounds are not necessarily better. ICML, 2018. [7]

We provide theoretical and empirical evidence that using tighter evidence lower bounds (ELBOs) can be detrimental to the process of learning an inference network by reducing the signal-to-noise ratio of the gradient estimator. Our results call into question common implicit assumptions that tighter ELBOs are better variational objectives for simultaneous model learning and inference amortization schemes. Based on our insights, we introduce three new algorithms: the partially importance weighted auto-encoder (PIWAE), the

multiply importance weighted auto-encoder (MIWAE), and the combination importance weighted auto-encoder (CIWAE), each of which includes the standard importance weighted auto-encoder (IWAE) as a special case. We show that each can deliver improvements over IWAE, even when performance is measured by the IWAE target itself. Furthermore, our results suggest that PIWAE may be able to deliver simultaneous improvements in the training of both the inference and generative networks.

3.4.3 Learning Disentangled Representations with Semi-Supervised Deep Generative Models

N. Siddarth, B. Paige, A. Desmaison, J.W. van de Meent, F. Wood, N. Goodman, P. Kohli, and P.H.S Torr.

Learning disentangled representations with semi-supervised deep generative models. In NIPS, 2017. [6]

Variational auto encoders (VAEs) learn representations of data by jointly training a probabilistic encoder and decoder network. Typically these models encode all features of the data into a 4 single variable. Here we are interested in learning disentangled representations that encode distinct aspects of the data into separate variables. We propose to learn such representations using model architectures that generalize from standard VAEs, employing a general graphical model structure in the encoder and decoder. This allows us to train partially-specified models that make relatively strong assumptions about a subset of interpretable variables and rely on the exit of neural networks to learn representations for the remaining variables. We further define a general objective for semi-supervised learning in this model class, which can be approximated using an importance sampling procedure. We evaluate our framework's ability to learn disentangled representations, both by qualitative exploration of its generative capacity, and quantitative evaluation of its discriminative ability on a variety of models and datasets.

3.4.4 Deep Variational Reinforcement Learning for POMDPs

M. Igl, L. Zintgraf, T. A. Le, F. Wood, and S. Whiteson. Deep variational reinforcement learning for pomdps. ICML, 2018. [2]

Many real-world sequential decision making problems are partially observable by nature, and the environment model is typically unknown. Consequently, there is great need for reinforcement learning methods that can tackle such problems given only a stream of incomplete and noisy observations. In this paper, we propose deep variational reinforcement learning (DVRL), which introduces an inductive bias that allows an agent to learn a generative model of the environment and perform inference in that model to

effectively aggregate the available information. We develop an n-step approximation to the evidence lower bound (ELBO), allowing the model to be trained jointly with the policy. This ensures that the latent state representation is suitable for the control task.

4.0 RESULTS AND DISCUSSION

The majority of Oxford's D3M effort supported the training of and research production of a number of graduate students and postdoctoral fellows. In addition to research activities, all those listed below have contributed to the D3M framework software project by way of primitives for automated machine learning and/or development of the D3M runtime. Their main contributions to the project are listed below.

4.1 PyFOPPL

https://github.com/Tobias-Kohn/PyFOPPL-2
PyFOPPL is a first-order probabilistic programming library for Python.

This library focuses on the set of probabilistic models which are amendable to be being solved with Hamiltonian Monte Carlo algorithm as well as graphical model methods. Models are compiled into a graphical representation which is portable and for which specialized solvers have been written.

4.2 Pyprob

https://github.com/probprog/pyprob

pyprob is a PyTorch-based library for probabilistic programming and inference compilation. The main focus of this library is on coupling existing simulation codebases with probabilistic inference with minimal intervention. Support for multiple languages. We support front ends in multiple languages through the PPX interface that allows execution of models and inference engines in separate programming languages, processes, and machines connected over a network. The currently supported languages are Python and C++.

- Python: pyprob is implemented and directly usable from Python
- C++: A lightweight C++ front end is available through the pyprob cpp Library

Inference engines pyprob currently provides the following inference engines:

- Markov chain Monte Carlo
  - Lightweight Metropolis Hastings (LMH)
  - Random-walk Metropolis Hastings (RMH)
- Importance sampling

- Regular sequential importance sampling (proposals from prior)
- Inference compilation

Inference compilation is an amortized inference technique for performing fast repeated inference using deep neural networks to parameterize proposal distributions in the importance sampling family of inference engines. We are planning to add other inference engines, e.g., from the variational inference family.

4.3 Pyprob cpp

https://github.com/probprog/pyprob_cpp

pyprob cpp is a C++ library providing a lightweight interface to the pyprob probabilistic programming library implemented in Python. The two components communicate through the PPX interface that allows execution of models and inference engines in separate programming languages, processes, and machines connected over a network.

4.4 PPX

https://github.com/probprog/ppx

PPX is a cross-platform Probabilistic Programming eXecution protocol and API based on at- buffers. It is intended as an open interoperability protocol between models and inference engines implemented in different probabilistic programming languages. Probabilistic programming is about the execution probabilistic models under the control of inference engines, and PPX allows the model and the inference engine to be:

- Implemented in different programming languages and
- Executed in separate processes and on separate machines across networks.

PPX is inspired by ONNX, the Open Neural Network Exchange project allowing interoperability between major deep learning frameworks.

4.5 Hypergradient Descent

https://github.com/gbaydin/hypergradient-descent

In gradient-based optimization, one optimizes an objective function by using its derivatives (gradient) with respect to model parameters. In addition to this basic gradient, a hypergradient is the derivative of the same objective function with respect to the optimization procedure's hyperparameters (such as the learning rate, momentum, or regularization parameters). There can be many types of hypergradients, and in this work we're interested in the hypergradient with respect to a scalar learning rate. We are providing ready-to-use implementations of the hypergradient versions of SGD (with or without momentum) and Adam optimizers for PyTorch. These comply with the torch.optim API and can be used as drop-in replacements in your code.

4.6 Common Primitives

https://gitlab.com/datadrivendiscovery/common-primitives

When performing an AutoML task there are a set of tasks that always come up. We developed a set of primitives to be used in the creation of nearly all AutoML problems. Among these primitives includes:

- Logistic regression, a probabilistic classification model
- Data loaders for images, csv _les, and text documents
- Feedforward neural networks for regression or classification
- Convolutional neural networks for working with image data
- Kmeans clustering for unsupervised learning
- Probabilistic PCA for unsupervised learning
- Random Forests for classification
- Linear Regression with regularization for regression

5.0 CONCLUSIONS

This work described our contribution of proof of concept primitives to the D3M program and research progress made towards an initial version of Hasty. Although we were unable to complete the initial version of Hasty, or contribute to the D3M primitive library the types of primitives that Hasty will enable we did train a number of Highly Qualified Personnel (HQP) and have interacted with the AutoML, probabilistic programming languages, neural networking, and other communities which our work is expected to impact.

6.0 Postdocs

Gunes Baydin Wrote inference compilation based PPL, PyProb and trace interpolation protocol, PPX. Currently a postdoc at Oxford. Attended D3M integration events.

Tobias Kohn Wrote _rst-order Python-based probabilistic programming compiler, PyLFPPL. Currently a postdoc at Cambridge.

Adam Scibior Currently a post-doc at UBC. Attended D3M integration events.

6.1 Graduate Students

William Harvey Wrote and contributed primitives to the D3M primitives library. Currently a graduate student at UBC. Attended D3M integration events.

Michael Teng Wrote and contributed primitives to the D3M primitives library. Currently a graduate student at Oxford. Currently a graduate student at Oxford and visiting UBC. Attended D3M integration events.

Robert Zinkov Wrote and contributed primitives to the D3M primitives library. Currently a graduate student at Oxford and visiting UBC. Attended D3M integration events.

7.0 References

[1] A. G. Baydin, R. Cornish, D. M. Rubio, M. Schmidt, and F. Wood. Online learning rate adaptation with hypergradient descent. arXiv preprint arXiv:1703.04782, 2017.

[2] M. Igl, L. Zintgraf, T. A. Le, F. Wood, and S. Whiteson. Deep variational reinforcement learning for pomdps. arXiv preprint arXiv:1806.02426, 2018.

[3] T. Le, A. Baydin, and F. Wood. Inference Compilation and Universal Probabilistic Programming. In 20th International Conference on Artificial Intelligence and Statistics, 2017.

[4] T. A. Le, M. Igl, T. Rainforth, T. Jin, and F. Wood. Auto-encoding sequential monte carlo. arXiv preprint arXiv:1705.10306, 2017.

[5] M. Milutinovic, A. G. Baydin, R. Zinkov, W. Harvey, D. Song, F. Wood, and W. Shen. End-to-end training of differentiable pipelines across machine learning frameworks. In NIPS Workshop on Autodiff, 2017.

[6] S. Narayanaswamy, T. B. Paige, J.-W. Van de Meent, A. Desmaison, N. Goodman, P. Kohli, F. Wood, and P. Torr. Learning disentangled representations with semi-supervised deep generative models. In Advances in Neural Information Processing Systems, pages 5925{5935, 2017.

[7] T. Rainforth, A. R. Kosiorek, T. A. Le, C. J. Maddison, M. Igl, F. Wood, and Y. W. Teh. Tighter variational bounds are not necessarily better. arXiv preprint arXiv:1802.04537, 2018.

[8] T. Rainforth, Y. Zhou, X. Lu, Y. W. Teh, F. Wood, H. Yang, and J.-W. van de Meent. Inference trees: Adaptive inference with exploration. arXiv preprint arXiv:1806.09550, 2018.

[9] M. Teng and F. Wood. Bayesian distributed stochastic gradient descent. In Advances in Neural Information Processing Systems, pages 6378{6388, 2018.

[10] S. Webb, A. Golinski, R. Zinkov, S. Narayanaswamy, T. Rainforth, Y. W. Teh, and F. Wood. Faithful inversion of generative models for e_ective amortized inference. In Advances in Neural Information Processing Systems 31. 2018.

[11] Y. Zhou, B. J. Gram-Hansen, T. Kohn, T. Rainforth, H. Yang, and F. Wood. Lf-ppl: A low-level first order probabilistic programming language for non-differentiable models. arXiv preprint arXiv:1903.02482, 2019.

8.0 LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

AutoML	Auto Machine Learning
BDSGD	Bayesian Distributed Stochastic Gradient Descent
CIWAE	Combination Importance Weighted Auto-Encoder
cpp	Certified Professional Programmer
DHMC	Discontinuous Hamiltonian Monte Carlo
ELBO	Evidence Lower Bounds
IT	Inference Trees
IWAE	Importance Weighted Auto-Encoder
LF-PPL	First-order Probabilistic Programming Language
MIWAE	Multiply Importance Weighted Auto-Encoder
PIWAE	Partially Importance Weighted Auto-Encoder
PPX	Probabilistic Programming eXecution
PyFOPPL	Python First Order Probabilistic Programming Language
SMC	Sequential Monte Carlo
VAE	Variational Auto Encoder