TUAN ANH LE—RESEARCH STATEMENT

Updated in December 2020
Email: tuananh@mit.edu
Website: https://www.tuananhle.co.uk
Google Scholar: https://scholar.google.com/citations?user=tkceMM0AAAAJ

My research goal is to develop interpretable machine learning systems that robustly generalize from sparse data, handle uncertainty and support fast inference. My approach combines **probabilistic programming** for writing structured probabilistic models using **symbolic** latent variables and complex forward generative functions like graphics renderers; and **neural networks** for fast inference and accurate modeling of high-dimensional perceptual data. Structured probabilistic modeling addresses the main issues of current machine learning systems: generalization, interpretability, and principled modeling of uncertainty. Neural networks, on the other hand, address the shortcomings of the probabilistic approaches by serving as recognition models that perform fast approximate Bayesian inference and as learnable parts of generative models that are hard to model. Centered upon the framework of probabilistic programming, my work focuses on inference, learning and modeling which I will illustrate using the following research projects.

Amortized inference for probabilistic programs

In collaboration with Atılım Güneş Baydin and Frank Wood, I led a project on amortized inference which is a way of using neural networks to aid inference in probabilistic programs [10]. The core idea is that training a neural network, or any conditional distribution for that matter, on synthetic data sampled from a generative model is actually equivalent to minimizing the inclusive Kullback-Leibler divergence [9]. This simple idea allowed us to connect the (at the time) seemingly disparate worlds of probabilistic programming and neural networks. To make it work with universal probabilistic programming systems, we used a recurrent neural network core which, together with dynamically added neural network submodules, could handle the variable length of program execution traces. Our first implementation was for a Clojure-based probabilistic programming system Anglican that we developed and this allowed us to solve Captchas in seconds, and with posterior uncertainties that resembled human uncertainties. This system was subsequently reimplemented several times and is currently a fully-fledged probabilistic programming system called pyprob based on Pytorch which is being picked up by the community ($>$ 300 stars and $>$ 40 forks on Github) and has recently been used for inverting large-scale particle-collision simulators [1].

Wake-sleep methods for learning probabilistic programs

One of the main issues in applying deep generative modeling approaches to learning probabilistic programs is their poor performance on models with discrete, symbolic latent variables which are often used for modeling compositional generalization in humans [4, 5]. To address this, I took inspiration from the wake-sleep literature [3, 7]. The idea is to train the inference model and the generative model separately using two different objectives which removes issues with learning discrete latent variables. We revisited reweighted wake-sleep (RWS) [11], which is a method for training deep generative models that is underexplored within the community, and argued that it is actually more effective in the kind of models where discrete latent variables are meaningful, and not just fed into neural nets, like probabilistic context-free grammars. These are exactly the kind of discrete latent variables desirable for modeling concept learning and compositional generalization. In collaboration with Siddharth Narayanaswamy, we implemented RWS in pyro [2], another fully-fledged universal probabilistic programming system with first-class deep learning support. Following this work, I've worked on projects that connect wake-sleep ideas with Gibbs sampling and particle methods [14]; semi-supervised learning [13]; variational inference and thermodynamic integration [12]; and memoization [6].

Concept learning using neuro-symbolic generative models

Can these algorithms actually be used for human-like concept learning? One of the clearest examples of using probabilistic programs for concept learning is the work of Lake et al. [8]. Given a novel character from an unfamiliar alphabet, we imagine how it could have been drawn, we can draw an instance of such a character ourselves, and we can reliably recognize another instance of this character. We can also imagine and draw a completely new character, either completely unconstrained or following a style of a particular alphabet. Humans can reliably do these tasks but current deep generative models fail [8].

Lake et al. [8]'s model is learned in a supervised fashion, and inference requires custom segmentation, in addition to MCMC steps. In collaboration with Luke Hewitt and Josh Tenenbaum, I developed a model that learns directly from unlabeled image data and inference is done using a recognition model to perform fast, neural-net based inference [6]. The model composes strokes one after another where each stroke is drawn from a finite bank of learnable strokes. This is an instance of a neuro-symbolic generative model. The "neuro" part serves two purposes, one for speeding up

inference, but importantly also for learning the prior over stroke sequences, the stroke bank parameters and renderer parameters. Our model can be used for sampling from the prior, reconstruction and alphabet-conditional generation which allows us to solve nearly the entire range of tasks proposed by Lake et al. [8], but with substantially less hand engineering and much faster evaluation through neurally-guided inference.

Looking ahead

Despite a lot of the recent success of supervised learning, our systems are still brittle and data hungry. How can we learn more from less data? I believe that anchoring the design of our systems on probabilistic modeling and inference is the right way to go. Probabilistic programming provides an expressive modeling language and neural networks provide fast inference and the ability to fit more complex models. I will continue working on inference and learning algorithms that combine the strengths neural networks and probabilistic programming. To maximize the impact of methodological work beyond my intended applications, I will work closely with research groups developing probabilistic programming systems like Turing.jl, pyro, Gen, and pyprob to implement these algorithms. For modeling, I will focus on structured probabilistic models like hybrid discrete-continuous latent-variable models to improve compositional generalization of current models. In such models, the discrete latent variables represent the underlying structure in the data and the continuous latent variables represent the remaining lower-level noise. I plan to collaborate with researchers in computer vision in order to tackle more direct applications like scene understanding and agent intention understanding which are the main bottlenecks of current self-driving car systems. A model-based approach to these problems will make our systems more robust and generalizable.

References

[1] Atilim Gunes Baydin, Lei Shao, Wahid Bhimji, Lukas Heinrich, Saeid Naderiparizi, Andreas Munk, Jialin Liu, Bradley Gram-Hansen, Gilles Louppe, Lawrence Meadows, et al. Efficient probabilistic inference in the quest for physics beyond the standard model. In *Advances in neural information processing systems*, pages 5459–5472, 2019.

[2] Eli Bingham, Jonathan P Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D Goodman. Pyro: Deep universal probabilistic programming. *The Journal of Machine Learning Research*, 20(1):973–978, 2019.

[3] Peter Dayan, Geoffrey E Hinton, Radford M Neal, and Richard S Zemel. The Helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

[4] Jerry A Fodor. *The language of thought*, volume 5. Harvard university press, 1975.

[5] Noah D Goodman, Joshua B Tenenbaum, and Tobias Gerstenberg. Concepts in a probabilistic language of thought. *The Conceptual Mind: New Directions in the Study of Concepts*, 2014.

[6] Luke B Hewitt, Tuan Anh Le, and Joshua B Tenenbaum. Learning to learn generative programs with memoised wake-sleep. In *Uncertainty in Artificial Intelligence*, 2020.

[7] Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 1995.

[8] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[9] Tuan Anh Le, Atilim Giineş Baydin, Robert Zinkov, and Frank Wood. Using synthetic data to train neural networks is model-based reasoning. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 3514–3521. IEEE, 2017.

[10] Tuan Anh Le, Atilim Gunes Baydin, and Frank Wood. Inference compilation and universal probabilistic programming. In *Artificial Intelligence and Statistics*, pages 1338–1348, 2017.

[11] Tuan Anh Le, Adam R. Kosiorek, N. Siddharth, Yee Whye Teh, and Frank Wood. Revisiting reweighted wake-sleep for models with stochastic control flow. In *Uncertainty in Artificial Intelligence*, 2019.

[12] Vaden Masrani, Tuan Anh Le, and Frank Wood. The thermodynamic variational objective. In *Advances in Neural Information Processing Systems*, pages 11525–11534, 2019.

[13] Michael Teng, Tuan Anh Le, Adam Scibior, and Frank Wood. Semi-supervised sequential generative models. In *Uncertainty in Artificial Intelligence*, 2020.

[14] Hao Wu, Heiko Zimmermann, Eli Sennesh, Tuan Anh Le, and Jan-Willem van de Meent. Amortized population gibbs samplers with neural sufficient statistics. In *International Conference on Machine Learning*, 2020.