From Neural NLP to Neural SBI

Will Wolf
williamabrwolf@gmail.com

Abstract

Recent work has demonstrated the efficacy of using neural networks as estimators within simulation-based inference (SBI) routines; broadly, their use has been shown to improve the accuracy, sample efficiency and ability to scale to high-dimensional data of the SBI routine itself. In parallel, in the Natural Language Processing (NLP) community, significant progress has been made in using neural networks to improve performance on canonical tasks—supervised classification, question answering, machine translation, etc.—along similar lines. In this proposal, we ask the following question: how can techniques from Neural NLP be applied to Neural SBI? Specifically, we detail three ideas–inductive bias, data-augmentation via back-translation, and transfer learning via pretrained language models–and discuss their potential application.

1 Introduction

1.1 Neural SBI

Bayesian inference is the task of quantifying a posterior belief over parameters $\boldsymbol{\theta}$ given observed data $\mathbf{x}$—where $\mathbf{x}$ was generated from a model $p(\mathbf{x}|\boldsymbol{\theta})$—via Bayes' Theorem:

$$p(\boldsymbol{\theta}|\mathbf{x}) = \frac{p(\mathbf{x}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathbf{x})}$$

In numerous applications of scientific interest, e.g. cosmological, climatic or urban-mobility phenomena, the likelihood of the data $\mathbf{x}$ under the data-generating function $p(\mathbf{x}|\boldsymbol{\theta})$ is intractable to compute, precluding classical inference approaches. Notwithstanding, *simulating* new data $\mathbf{x}$ from this function is often possible, motivating the study of *simulation-based* Bayesian *inference* methods, termed SBI.

Recent work has explored the use of neural networks to perform key density estimation tasks, i.e. subroutines, of the SBI routine itself. We refer to this work as Neural SBI. In *neural likelihood estimation* (NLE) techniques [20, 15], a neural network $q_\phi(\mathbf{x}|\boldsymbol{\theta})$ is trained to estimate the (intractable) true likelihood $p(\mathbf{x}|\boldsymbol{\theta})$. Similarly, in *neural posterior estimation* (NPE) techniques [18, 14, 7], a neural network $q_\phi(\boldsymbol{\theta}|\mathbf{x})$ is trained to directly estimate the target posterior density $p(\boldsymbol{\theta}|\mathbf{x})$. Finally, in *neural ratio estimation* (NRE) techniques [10], a neural network is trained to approximate likelihood ratios which are later used as part of an MCMC subroutine to obtain posterior samples.

In each class of algorithm, and as compared to baseline approaches, neural networks have been shown to improve key performance metrics of the SBI routine itself [16]: accuracy (of the inferred target parameters), sample efficiency (of the often-costly simulator) and the ability to scale to high-dimensional (observed and simulated) data. At the same time, we observe that while much focus has been placed on *how to use neural networks* in SBI, comparatively little attention is paid to *the neural network itself.* In this vein, an obvious question becomes: in pursuing the latter, **what further performance improvements lie in wait?**

1.2 Neural NLP

In recent years, the NLP community (among others) has successfully used neural networks to similarly improve performance in canonical learning routines. Often, these gains result directly from its sustained focus on the workings of the neural network itself. On this note, we detail three (of many) key ideas from Neural NLP, and discuss their potential application to Neural SBI.

2 Proposal

2.1 Inductive Bias

2.1.1 Modeling Natural Language

"Inductive bias" refers to knowledge possessed by a machine learning model that does not come from data [1]. For example, this bias may derive from the model's architecture, optimization routine or optimization target. Namely, it is this bias that "informs" the model's behavior when predicting on novel data.

Nature language is sequential in nature. (Typically, natural language is encoded as a sequence of "one-hot" basis vectors in a high-dimensional space.) As such, an ideal statistical learner of natural language would consider important properties of sequential data, e.g. positional information, long-range dependencies, and repeating patterns of elements in input and output sequences. On this note, in recent years, the NLP community has effectively converged on a small set of neural network architectures which capture these inductive biases to varying degrees—Recurrent Neural Networks (RNNs) [6], Long Short-Term Memory Networks (LSTMs) [11], Convolutional Neural Networks (CNNs) [12] and most recently Transformers [26]—all of which continue to demonstrate impressive performance on difficult NLP tasks.

2.1.2 Modeling Complex Scientific Phenomena

Recent NLE, NPE and NRE techniques all employ neural network models as density estimation subroutines to great effect. In NLE and NPE, the neural network is often chosen to be a *normalizing flow* or Mixture Density Network, as they are flexible enough to model complex densities, offer exact density computation and are fast to train and evaluate on GPUs [20, 15, 18, 14, 7, 19]. In NRE, [9] employ a variety of *multilayer perceptron* and ResNet architectures.

In the sum of these works, experiments are performed on tasks as diverse as: the M/G/1 queue model (a stochastic arrival process), the Lotka-Volterra population model (evolution of two adversarial populations), the Hodgkin-Huxley cortical pyramidal neuron model (a model of spiking neurons), and more. In the SBI community at large, practitioners model more diverse phenomena still, often using data with rich structure (e.g. graphs, spatio-temporal rollouts, etc.) and high-dimensionality.

Motivated by (i) the apparent lack of existing work exploring inductive bias for Neural SBI components and (ii) the resounding success of such efforts in Neural NLP, we ask: do the neural networks typically used in Neural SBI sufficiently encode an inductive bias consistent with the complex structure of the data they seek to model? If not, what performance gains do we currently miss?

2.2 Data-augmentation via Back-translation

2.2.1 Back-translation in Sequence Modeling

Back-translation is a data-augmentation strategy that has been shown to improve sample efficiency in supervised NLP tasks [24, 27]. Given a dataset mapping sentences $\mathbf{x}_A$ in language A to sentences $\mathbf{x}_B$ in language B, we can first train translation models $p_{A\to B}(\mathbf{x}_B|\mathbf{x}_A)$ and $p_{B\to A}(\mathbf{x}_A|\mathbf{x}_B)$; then, for each training pair $(\mathbf{x}, y)$ in our *target* dataset, we can create (additional) synthetic examples $(\tilde{\mathbf{x}} \sim p_{B\to A}(\mathbf{x}_B \sim p_{A\to B}(\mathbf{x})), y)$ on which to train. Back-translation techniques have been used to create synthetic data from both labeled and unlabeled datasets, and have enabled state-of-the-art results on numerous supervised benchmarks with orders-of-magnitude less labeled data [27].

2.2.2 Improving Sample Efficiency via Back-translation

In SBI, the simulator $p(\mathbf{x}|\boldsymbol{\theta})$ is typically expensive to run. As such, numerous works have explored training neural conditional density estimators q_ϕ in a *sequential* manner, where in each training round, they aim to train q_ϕ on the *specific* simulated training data tuples $\{\boldsymbol{\theta}_n, \mathbf{x}_n\}_{1:N}$ that most efficiently facilitate (i.e. imply the lowest number of required simulations overall) the downstream inference task [18, 20, 14, 7, 9]. This approach is an instance of *active learning* [8].

For example, in [18], the authors train an estimator $q_\phi(\boldsymbol{\theta}|\mathbf{x})$ of the target posterior density. They then use this estimator to compute $q_\phi(\boldsymbol{\theta}|\mathbf{x} = \mathbf{x_o})$, where $\mathbf{x_o}$ is the original, observed data. Intuitively, they note that the *sample efficiency* of this estimate is directly related to the number of diverse data $\{\boldsymbol{\theta}_n, \mathbf{x}_n\}_{1:N}$—where $\mathbf{x}_n$ is *close* to $\mathbf{x_o}$—on which q_ϕ is trained. In this vein, in each round of estimation, they simulate data $\mathbf{x}_n \sim p(\mathbf{x}|\boldsymbol{\theta}_n)$ using $\boldsymbol{\theta}_n$ of high (approximate) posterior density $\hat{p}(\boldsymbol{\theta}|\mathbf{x} = \mathbf{x_o})$; in this way, they *augment* their training set for q_ϕ with a diverse set of synthetic inputs $\mathbf{x}_n$ near the target input $\mathbf{x_o}$—improving the sample efficiency of q_ϕ itself, and the SBI routine at large. Similarly, in [20], the authors augment the training set of a neural estimator of the data likelihood in related fashion.

In fact, we can reinterpret these active learning approaches as back-translation: $\hat{p}(\boldsymbol{\theta}|\mathbf{x} = \mathbf{x_o})$ gives $p_{A\to B}(\mathbf{x}_B|\mathbf{x}_A)$—mapping "labeled" inputs $\mathbf{x_o}$ to samples $\boldsymbol{\theta}_n$ in a complementary space Θ—which are then "translated back" through some stochastic input generator $\mathbf{x}_n \sim p(\mathbf{x}|\boldsymbol{\theta}_n) = p_{B\to A}(\mathbf{x}_A|\mathbf{x}_B)$—mapping $\boldsymbol{\theta}_n \in \Theta$ to synthetic tuples $\{\boldsymbol{\theta}_n, \mathbf{x}_n\}_{1:N}$ used to further train q_ϕ in the input region ($\mathbf{x_o}$) of interest.

In sum, we propose reconsidering the back-translation literature as a set of potential techniques for active learning in Neural SBI—where the high marginal cost of annotating NLP training data is analogous to that of simulating data on which an estimator q_ϕ is trained. In this vein, we aim to leverage its many insights regarding auxiliary datasets (both supervised and unsupervised), models, training strategies and more to further improve the sample efficiency of Neural SBI routines.

2.3 Transfer Learning via Pretrained Language Models

2.3.1 Pretrain, Fine-tune

Language modeling refers to a statistical model of language defined as:

$$p(\mathbf{x}) = p(x_0, ..., x_N) = \prod_i^N p(x_i|\mathbf{x}_{<i})$$

which computes the probability density at a given text sequence (e.g. sentence) $\mathbf{x}$. As $p(\mathbf{x})$ factorizes into factors $p(x_i|\mathbf{x}_{<i})$, we can construct a language model by simply training a "next-token-prediction" classifier. Finally, as unlabeled text data (e.g. all text on the Internet) implicitly forms a gargantuan, *labeled* training set for this task, we can train powerful language models without any (expensive) human annotation at all.

Transfer learning, on the other hand, refers to the process of injecting knowledge from one task and/or dataset into another [17, 25, 4]. On this note, in recent years, it has become standard in NLP to (i) "pretrain" a language model (e.g. BERT, RoBERTa, GPT-3) on a large corpus of text, then (ii) "fine-tune" this model given a *labeled* dataset for the target task (e.g. question answering). In general, this "pipeline" has led to significant improvements in the sample efficiency and accuracy of downstream NLP models [13, 5, 23, 2].

2.3.2 Improving Sample Efficiency and Accuracy via Transfer Learning

NLE, NPE and NRE techniques (implicitly) train neural networks from scratch on (costly) simulated data. Instead, how might transfer learning improve the sample efficiency and accuracy of these models? On what upstream tasks might we "pretrain" neural estimators q_ϕ so as to reduce—ideally, *significantly*—the number of simulations required for downstream inference?

3 Initial Research Direction

To begin, I would target **Improving Sample Efficiency and Accuracy via Transfer Learning** as it offers:

1. A wide variety of **"free" baseline approaches**—initializing a neural estimator q_ϕ as any number of pretrained models (whose API fits the estimation task in question) from pytorch.org/video, pytorch.org/audio, huggingface.co/models, and other such repositories.

2. A **clear evaluation metric**—the number of simulations s required to achieve a fixed Classifier 2-Sample Test (C2ST; a classifier to discriminate samples as having been drawn from the true or inferred posterior) performance p [16].

3.1 Proposed Research Plan

Week 1-3 Reproduce benchmarking results from [16], isolating their comparison of s and p for the models and tasks they study.

Week 3-6 Select a single task, e.g. the Lotka-Volterra population model, and a single SBI routine, e.g. NLE [20]. In this domain, $q_\phi(\mathbf{x}|\boldsymbol{\theta})$ is trained to estimate the true likelihood $p(\mathbf{x}|\boldsymbol{\theta})$ of a *sequence* of 20 summary statistics $\mathbf{x}$ given parameters $\boldsymbol{\theta}$, where $\mathbf{x} \in \mathbb{R}^{20}$ and $\boldsymbol{\theta} \in \mathbb{R}^4$. As such, in this step, we initialize q_ϕ with a variety of pretrained sequence models of language, vision and audio, then fine-tune as per the prescribed NLE routine. For each model, we note how its basic characteristics—number of parameters, amount of training data, training task, general architecture—impact p. From this study, we select a single pretraining training task and architecture, denoting this set of candidate models as $\mathcal{C}$.

Week 6-10 Obtain a dataset of historical Lotka-Volterra simulations, e.g. [22]. For each task and architecture in $\mathcal{C}$, pretrain q_ϕ (from scratch) on this data. Then, fine-tune on the downstream task and measure p. Additionally, vary the amount of pretraining data and repeat this step.

Week 10-14 Should more data help, train a Neural ODE model [3] to learn the dynamical system itself given the datasets above. Then, simulate *pseudo-observations*, append to our pretraining dataset, and repeat **Week 6-10**.

Week 14-20 For each architecture in $\mathcal{C}$, craft and test novel pretraining tasks specific to the Lotka-Volterra problem domain. For example, given a random subset of the values in $\mathbf{x}$, predict the timestep t of each? Or, given a random pair of values from $\mathbf{x}$, predict their temporal order?

Week 20-24 Experiment with novel tasks and architectures outright. For example, given a *randomly shuffled* $\mathbf{x}$, predict its sorting permutation via an architecture like SoftSort [21]?

4 Conclusion

Neural SBI routines have demonstrated substantial promise in performing statistical inference in meaningful problems of scientific interest. In this proposal, we've presented three directions of research—inspired by developments in Neural NLP—for improving these algorithms further.

References

[1] Samira Abnar, Mostafa Dehghani, and Willem Zuidema. Transferring inductive biases through knowledge distillation. *arXiv*, 2020.

[2] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. *arXiv*, 2020.

[3] Ricky T Q Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural Ordinary Differential Equations. 2018.

[4] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. *arXiv*, 2017.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

[6] Jeffrey L. Elman. Finding structure in time. *COGNITIVE SCIENCE*, 14(2):179–211, 1990.

[7] David S Greenberg, Marcel Nonnenmacher, and Jakob H Macke. Automatic Posterior Transformation for Likelihood-Free Inference. *arXiv*, 2019.

[8] Corander Gutmann. Bayesian Optimization for Likelihood-Free Inference of Simulator-Based Statistical Models. *jmlr*, 2016.

[9] Joeri Hermans, Volodimir Begy, and Gilles Louppe. Likelihood-free MCMC with Amortized Approximate Ratio Estimators. *arXiv*, 2019.

[10] Joeri Hermans, Volodimir Begy, and Gilles Louppe. Likelihood-free MCMC with amortized approximate ratio estimators. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4239–4248. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/hermans20a.html.

[11] Sepp Hochreiter and Jrgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9 (8):1735–1780, 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735.

[12] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, R. Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1990. URL https://proceedings.neurips.cc/paper/1989/file/53c3bce66e43be4f209556518c2fcb54-Paper.pdf.

[13] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv*, 2019.

[14] Jan-Matthis Lueckmann, Pedro J Goncalves, Giacomo Bassetto, Kaan Öcal, Marcel Nonnenmacher, and Jakob H Macke. Flexible statistical inference for mechanistic models of neural dynamics. *arXiv*, 2017.

[15] Jan-Matthis Lueckmann, Giacomo Bassetto, Theofanis Karaletsos, and Jakob H. Macke. Likelihood-free inference with emulator networks. In Francisco Ruiz, Cheng Zhang, Dawen Liang, and Thang Bui, editors, *Proceedings of The 1st Symposium on Advances in Approximate Bayesian Inference*, volume 96 of *Proceedings of Machine Learning Research*, pages 32–53. PMLR, 02 Dec 2019. URL http://proceedings.mlr.press/v96/lueckmann19a.html.

[16] Jan-Matthis Lueckmann, Jan Boelts, David S Greenberg, Pedro J Gonçalves, and Jakob H Macke. Benchmarking Simulation-Based Inference. *arXiv*, 2021.

[17] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in Translation: Contextualized Word Vectors. *arXiv*, 2017.

[18] George Papamakarios and Iain Murray. Fast ϵ-free inference of simulation models with bayesian conditional density estimation. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/6aca97005c68f1206823815f66102863-Paper.pdf.

[19] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked Autoregressive Flow for Density Estimation. *arXiv*, 2017.

[20] George Papamakarios, David Sterratt, and Iain Murray. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 837–848. PMLR, 16–18 Apr 2019. URL http://proceedings.mlr.press/v89/papamakarios19a.html.

[21] Sebastian Prillo and Julian Martin Eisenschlos. SoftSort: A Continuous Relaxation for the argsort Operator. *arXiv*, 2020.

[22] Juste Raimbault. Simulation data of Lotka-Volterra systems. 2018. doi: 10.7910/DVN/P0OMNM. URL https://doi.org/10.7910/DVN/P0OMNM.

[23] Sebastian Ruder. Recent Advances in Language Model Fine-tuning. http://ruder.io/recent-advances-lm-fine-tuning, 2021.

[24] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving Neural Machine Translation Models with Monolingual Data. *arXiv*, 2015.

[25] Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. Learning General Purpose Distributed Sentence Representations via Large Scale Multi-task Learning. *arXiv*, 2018.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[27] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. Unsupervised Data Augmentation. *arXiv*, 2019.