# Neuromorphic Acceleration for Approximate Bayesian Inference on Neural Networks via Permanent Dropout

Nathan Wycoff*†
nathw95@vt.edu
Virginia Tech
Blacksburg, Virginia

Prasanna Balaprakash
Fangfang Xia
pbalapra@anl.gov
fangfang@anl.gov
Argonne National Laboratory
Lemont, Illinois



## ABSTRACT

As neural networks have begun performing increasingly critical tasks for society, ranging from driving cars to identifying candidates for drug development, the value of their ability to perform uncertainty quantification (UQ) in their predictions has risen commensurately. Permanent dropout, a popular method for neural network UQ, involves injecting stochasticity into the inference phase of the model and creating many predictions for each of the test data. This shifts the computational and energy burden of deep neural networks from the training phase to the inference phase. Recent work has demonstrated near-lossless conversion of classical deep neural networks to their spiking counterparts. We use these results to demonstrate the feasibility of conducting the inference phase with permanent dropout on spiking neural networks, mitigating the technique's computational and energy burden, which is essential for its use at scale or on edge platforms. We demonstrate the proposed approach via the Nengo spiking neural simulator on a combination drug therapy dataset for cancer treatment, where UQ is critical. Our results indicate that the spiking approximation gives a predictive distribution practically indistinguishable from that given by the classical network.

## KEYWORDS

Neuromorphic computing, Bayesian inference, Uncertainty quantification

## 1 INTRODUCTION

Deep neural networks (DNNs) are the arguable flagship of the machine learning (ML) revolution, having captured the imagination of the academic research community, industry, and to some extent the public at large because of their widespread empirical successes and captivating connection to human information processing. Historically sporting a black-box, predictive-error-driven approach, ML culture is increasingly interested in quantifying the uncertainty of its predictions. Standard, off-the-shelf tools from classical and Bayesian statistics to this end are often too computationally expensive to be of use in problems of even modest scale, a challenge the ML community has risen to meet.

DNNs are increasingly being used for tasks that require quantification of prediction uncertainty. For instance, many autonomous vehicle frameworks are built on convolutional networks [16]. Also, in the context of reinforcement learning with a DNN value function approximator, understanding model uncertainty is important in order to determine where the agent should next explore [19]. For camera relocalization, Kendall and Cipolla [17] avail themselves of the uncertainty obtained from permadrop to obtain improvements in challenging indoor and outdoor problems. Recently, Thulasidasan et al. [23] developed a neural net with abstention, where the DNN may decide not to classify an instance if sufficient uncertainty exists. Furthermore, uncertainty quantification (UQ) is critical to many scientific ML applications as well [2].

Dropout [22], an approach wherein individual neurons are randomly turned off (or otherwise perturbed), has been shown to be an effective approach for regularizing DNNs. The same approach applied during inference can approximate a Bayesian treatment of model uncertainty [11]. In particular, it was shown that permanent dropout (called Monte Carlo dropout in the initial article and referred to as permadrop here) networks approximate a form of deep Gaussian processes [8, 20]. Traditionally, the cost of training dominates that of inference [13]; however, the permadrop strategy reverses this paradigm, since the inference phase must be executed many times, with increasing iterations giving increasing Monte Carlo accuracy. With the utility of UQ in DNNs having been carried out via permadrop, the challenge of reducing the concomitant computational and energy costs has become critical and nontrival.

Spiking neural networks (SNNs) that run on neuromorphic hardware are a promising approach to address the computational and energy concerns of DNNs running on CPUs and GPUs for a class of applications. A recent study [4] using Intel Loihi [9] found that it used 23.1 times fewer joules than a CPU (Xeon E5-2630) and 109.1 times fewer joules than a GPU (Quadro K4000) on an audio-processing problem with a two-layer neural net during the inference phase. In this paper, we explore the prospect of offsetting the energy expense of the permadrop procedure in DNNs by converting them to SNNs during the inference phase. To do so, we expand upon the

*To whom correspondence should be addressed.
†Some of this work was conducted during an internship at the Mathematics and Computer Science Division of Argonne National Laboratory.


*Knoxville '19, July 23–25, 2019, Knoxville, TN*

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

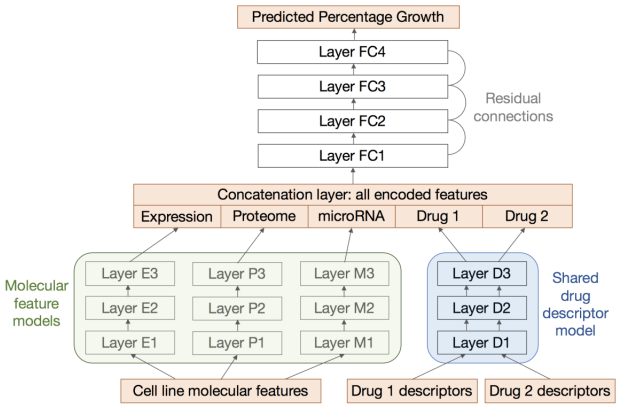


**Figure 1: Architecture of the Combo neural network. Given two drugs, the aim is to predict the percent growth in human-derived cancer cell lines where these two drugs to be applied in a combination therapy.**

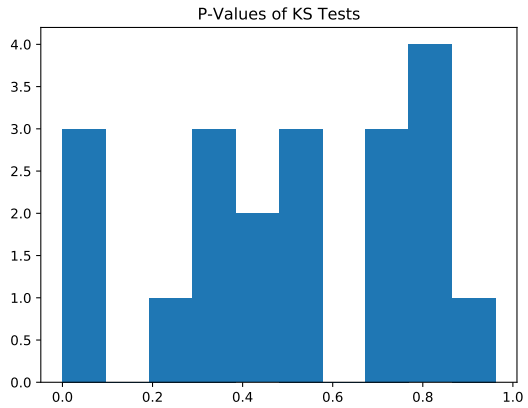


**Figure 2: P-values for KS tests comparing samples of size 100 output by the DNN and the SNN for the first 20 observations. Were the distributions equal, we would expect the p-values to be approximately uniformly distributed.**

nengo and `nengo_extras` [3] packages, which allow conversion of simple DNNs to SNNs, implementing permadrop layers in the Nengo framework and demonstrating the feasibility of the process using the simulator therein.

## 2 BACKGROUND AND RELATED WORK

This article addresses a topic at the confluence of two threads of research: UQ on DNNs, and spiking conversion of DNNs.

### 2.1 Permanent Dropout

Dropout [22] is a method for regularization in DNNs. In its simplest form, it involves randomly turning off neurons during each minibatch of training independently with some probability $p$. As originally proposed, the inference phase is unmodified aside from a scaling of the weights of each layer (as there are now more units present than during training). The intuition behind the method is that nodes cannot rely on a particular upstream or downstream neuron to modify their output and must instead pass on information that is more generally useful, as well as being forced to learn redundant representations. As outlined in [22], dropout may be viewed as approximate model averaging over all networks formed by subsets of the full network architecture. Gal and Ghahramani [11] showed that keeping dropout active during prediction (permadropout) is an approximation to a fully Bayesian treatment using a connection between neural networks and Gaussian processes. Each forward evaluation gives a random output; many forward evaluations build up an approximate predictive distribution.

### 2.2 Spiking Conversion of Classical Neural Networks

While SNNs are more powerful than DNNs in terms of theoretical computational ability [18], their often-discontinuous and computationally expensive nature means that training SNNs has been more challenging in practice than has been training DNNs, an already daunting task and the subject of major research. For this reason, the idea of conducting the training phase on a DNN and finding an SNN with similar behavior is an appealing one. Several approaches have been suggested for converting a DNN to an SNN while minimizing performance loss. Diehl et al. [10] focused on converting DNNs with standard nonlinearities such as the softmax or ReLU functions, which was expanded upon by Rueckauer et al. [21] to enable conversion of much more general neural architectures.

Other work [5] requires tailoring the DNN to optimize the SNN's performance, the approach we take in this paper. In particular, we follow the technique outlined in [15], which simply requires using a specific activation function, termed the SoftLIF function.

Given a sufficiently large constant input to trigger an action potential, the firing rate of a linear Leaky Integrate and Fire (LIF) neuron with input current $\lambda$ is given by [12]

$$\frac{1}{\tau_{ref} + \tau_{RC} \log(1 + \frac{\nu}{\rho(\lambda - \nu)})}, \tag{1}$$

where $\rho(x) = \max[0, x]$. Unfortunately, this function is not continuously differentiable, complicating gradient-based optimization methods. To resolve this issue, Hunsberger and Eliasmith [15] suggest replacing $\rho$ with a smooth approximation given by

$$\sigma(x) = \gamma \log(1 + e^{\frac{x}{\gamma}}), \tag{2}$$

which matches $\rho$ exactly as $\gamma \to 0$. Having trained a DNN with the SoftLIF activation, weights need simply to be transferred to an SNN of identical structure.

An SNN may be imbued with permadrop in a manner analogous to DNNs. We used the Nengo framework in simulations; since it did not previously have support for permadrop, we modified the `nengo_extras` package for our purposes. This task involved simply sampling a drop-mask for each layer during each simulation, that is, a binary vector of length equal to the number of neurons in a particular layer, in which 1's represent "on" neurons, which will

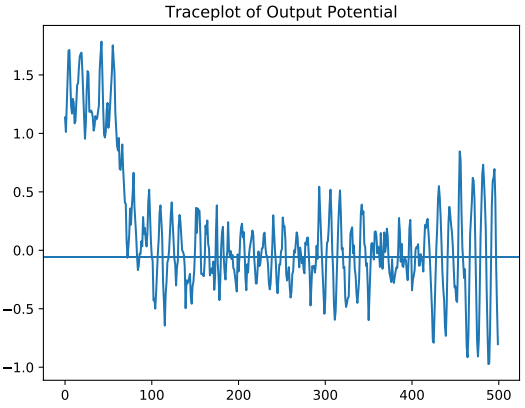


**Figure 3: Output potential for the SNN on one observation. The transition time between states is removed using a 0.2 ms (or 200 tick) burn-in. Horizontal line gives DNN output.**

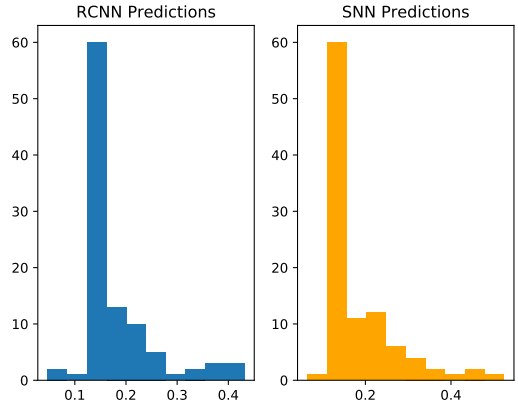


**Figure 4: Histograms representing 100 draws from the predictive distributions for each neural network for the training example with the largest KS statistic (i.e., that with the most different distribution). Since they are similar visually, we conclude that the SNN is a good approximator.**

contribute to this simulation normally, and 0's represent "off" neurons, which will not contribute at all. These vectors were sampled independently from a Bernoulli measure with some probability of success (i.e., neuron is active) $\rho$. A new drop-mask was sampled during each simulation, ultimately giving a distribution of outputs.

## 3 EXPERIMENTATION

We executed our proposed method on the Combo benchmark of CANDLE, a U.S. Department of Energy Exascale Computing Project activity. The Combo deep neural network aims to predict the effectiveness of two drugs used in combination given tumor cell features (942 dimensions) as well as the description of each drug (3,820 dimensions), containing 248,656 observations. The data were obtained from the National Cancer Institute's ALMANAC resource [14]. Network weights were shared for processing each drug of the pair; see Figure 1 for details. In decision-making for cancer treatment, a complete accounting of uncertainty is critical, motivating the need for permadrop. On this benchmark, however, inference is expected to be 7 times more computationally expensive than training, because of UQ, underlining the potential gain from neuromorphic acceleration.

To implement our SNN, we used the Nengo framework [3], a Python-based spiking neuron simulator. Nengo allows conversion of feedforward neural networks implemented in, for instance, Keras [7], into spiking Nengo objects, which may subsequently be simulated on a standard computer or in specialized hardware, such as Loihi. In our experiments, we trained a permadrop DNN using Keras with TensorFlow [1] as a backend.

While the DNN's output is a scalar quantity giving predicted cell growth in percent, the output of its SNN analogue will be a time-valued quantity. We summarize the output potential over the time period by simply averaging the results, treating the first 0.2 ms as a "burn in" period and omitting the potential during this time from the average. Figure 3 illustrates that the output potential of the SNN hovers around the output value of the DNN for most of the period on the first record of the Combo dataset. This same behavior is exhibited for all other observations.

We demonstrate that the distributions of outputs from the permadrop DNN and SNN are indistinguishable after averaging SNN output as described above after each dropout sampling. To quantitatively verify this claim, we got distributions of predictions for 100 observations containing 20 model forward steps each and ran a statistical hypothesis test that the two samples come from the same distribution. We used the Kolmogorov-Smirnov (KS) test, which involves measuring the infinity-norm difference (that is, maximum absolute discrepancy) between the empirical cumulative distribution functions of each sample. Figure 2 gives a histogram of p-values from each pairwise comparison, corresponding to the output distributions of each neural net for a particular observation. In general hypothesis testing, under the null distribution, the p-value is uniformly distributed on the unit interval [6]; however, since the KS test is asymptotic, we should expect this to hold only approximately in this case. We are satisfied that the KS test p-values generally seem to follow a uniform distribution,[1] indicating that we could not detect a statistical difference between the two samples, and implying functional equivalence of the SNN and DNN. The histogram of the two predictive distributions corresponding to a single observation is shown in Figure 4 for illustration purposes.

## 4 CONCLUSIONS AND PERSPECTIVES

We showed that permanent dropout for the purpose of approximate Bayesian predictive distribution computation on classical neural networks can be carried out on an SNN without any noticeable loss

[1] A common criticism of KS tests (and general point-null hypothesis testing) is that for large sample sizes, even the smallest discrepancy will cause the test to reject the null hypothesis [24]. It is likely that we could consistently get results indicating that the two predictive distributions are different if we were willing to use a much larger sample size, though this would not mean that the distributions are, practically speaking, significantly different.

in distribution quality, opening the door for low-energy UQ via permadrop. We used the open source Nengo framework for simulation, which allows easy transfer of these models to neuromorphic hardware.

In our experiments, we first sampled a dropout mask, then ran an SNN with that mask, repeating this process many times to achieve a distribution of outputs. However, each of these outputs represents an aggregation of SNN potentials over some period of time. It may be possible to conduct the dropout sampling *during* SNN simulation, such that the network connections are constantly changing in the SNN, and only one forward evaluation is required, even further reducing the computational burden. It is not *a priori* clear whether the naive approach of simply sampling a different dropout mask at each iteration would match permadrop exactly or what modifications may be necessary. We leave investigations of such an approach to future work.

All this work was conducted on a simulator. A complete proof of concept would involve actual neuromorphic hardware and energy comparisons with standard DNNs run on standard hardware such as CPUs, GPUs, or TPUs.

## ACKNOWLEDGMENTS

N. Wycoff acknowledges funding from DOE LAB 17-1697 via a subaward from Argonne National Laboratory for SciDAC/DOE Office of Science ASCR and High Energy Physics. This material is based upon work supported by the U.S. Department of Energy (DOE), Office of Science, Office of Advanced Scientific Computing Research, under Contract DE-AC02-06CH11357.

## REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.

[2] Nathan Baker, Frank Alexander, Timo Bremer, Aric Hagberg, Yannis Kevrekidis, Habib Najm, Manish Parashar, Abani Patra, James Sethian, Stefan Wild, and Karen Willcox. 2018. Brochure on Basic Research Needs for Scientific Machine Learning: Core Technologies for Artificial Intelligence. (12 2018). https://doi.org/10.2172/1484362

[3] Trevor Bekolay, James Bergstra, Eric Hunsberger, Travis DeWolf, Terrence Stewart, Daniel Rasmussen, Xuan Choo, Aaron Voelker, and Chris Eliasmith. 2014. Nengo: a Python tool for building large-scale functional brain models. *Frontiers in Neuroinformatics* 7 (2014), 48. https://doi.org/10.3389/fninf.2013.00048

[4] Peter Blouw, Xuan Choo, Eric Hunsberger, and Chris Eliasmith. 2018. Benchmarking Keyword Spotting Efficiency on Neuromorphic Hardware. *CoRR* abs/1812.01739 (2018). arXiv:1812.01739 http://arxiv.org/abs/1812.01739

[5] Yongqiang Cao, Yang Chen, and Deepak Khosla. 2015. Spiking Deep Convolutional Neural Networks for Energy-Efficient Object Recognition. *International Journal of Computer Vision* 113 (05 2015), 54–66. https://doi.org/10.1007/s11263-014-0788-3

[6] G. Casella and R.L. Berger. 2002. *Statistical Inference*. Thomson Learning. https://books.google.com/books?id=0x_vAAAAMAAJ

[7] François Chollet et al. 2015. Keras. https://keras.io.

[8] Andreas Damianou and Neil Lawrence. 2013. Deep Gaussian Processes. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research)*, Carlos M. Carvalho and Pradeep Ravikumar (Eds.), Vol. 31. PMLR, Scottsdale, Arizona, USA, 207–215. http://proceedings.mlr.press/v31/damianou13a.html

[9] M. Davies, N. Srinivasa, T. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y. Weng, A. Wild, Y. Yang, and H. Wang. 2018. Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. *IEEE Micro* 38, 1 (January 2018), 82–99. https://doi.org/10.1109/MM.2018.112130359

[10] Peter U. Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer. 2015. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. *2015 International Joint Conference on Neural Networks (IJCNN)* (2015), 1–8.

[11] Yarin Gal and Zoubin Ghahramani. 2016. Dropout As a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48 (ICML'16)*. JMLR.org, 1050–1059. http://dl.acm.org/citation.cfm?id=3045390.3045502

[12] Wulfram Gerstner and Werner M. Kistler. 2002. *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press. https://doi.org/10.1017/CBO9780511815706

[13] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. The MIT Press.

[14] Susan L. Holbeck, Richard Camalier, James A. Crowell, Jeevan Prasaad Govindharajulu, Melinda Hollingshead, Lawrence W. Anderson, Eric Polley, Larry Rubinstein, Apurva Srivastava, Deborah Wilsker, Jerry M. Collins, and James H. Doroshow. 2017. The National Cancer Institute ALMANAC: A Comprehensive Screening Resource for the Detection of Anticancer Drug Pairs with Enhanced Therapeutic Activity. *Cancer Research* 77, 13 (2017), 3564–3576. https://doi.org/10.1158/0008-5472.CAN-17-0489 arXiv:http://cancerres.aacrjournals.org/content/77/13/3564.full.pdf

[15] Eric Hunsberger and Chris Eliasmith. 2015. Spiking Deep Networks with LIF Neurons. *CoRR* abs/1510.08829 (2015).

[16] J. Janai, F. Güney, A. Behl, and A. Geiger. 2017. Computer Vision for Autonomous Vehicles: Problems, Datasets and State-of-the-Art. *arXiv e-prints* (April 2017). arXiv:cs.CV/1704.05519

[17] A. Kendall and R. Cipolla. 2016. Modelling uncertainty in deep learning for camera relocalization. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 4762–4769. https://doi.org/10.1109/ICRA.2016.7487679

[18] Wolfgang Maass. 1997. Networks of spiking neurons: The third generation of neural network models. *Neural Networks* 10, 9 (1997), 1659 – 1671. https://doi.org/10.1016/S0893-6080(97)00011-7

[19] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. 2016. Deep Exploration via Bootstrapped DQN. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., 4026–4034. http://papers.nips.cc/paper/6501-deep-exploration-via-bootstrapped-dqn.pdf

[20] Carl Edward Rasmussen and Christopher K. I. Williams. 2005. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

[21] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. 2017. Conversion of Continuous-Valued Deep Networks to Efficient Event-Driven Networks for Image Classification. *Front Neurosci* 11 (07 Dec 2017), 682–682. https://doi.org/10.3389/fnins.2017.00682 29375284[pmid].

[22] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15 (2014), 1929–1958. http://jmlr.org/papers/v15/srivastava14a.html

[23] Sunil Thulasidasan, Tanmoy Bhattacharya, Jeffrey Bilmes, Gopinath Chennupati, and Jamal Mohd-Yusof. 2019. Knows When it DoesnâĂŹt Know: Deep Abstaining Classifiers. https://openreview.net/forum?id=rJxF73R9tX

[24] Ronald L. Wasserstein and Nicole A. Lazar. 2016. The ASA's Statement on p-Values: Context, Process, and Purpose. *The American Statistician* 70, 2 (2016), 129–133. https://doi.org/10.1080/00031305.2016.1154108 arXiv:https://doi.org/10.1080/00031305.2016.1154108