Stochastic Synapses as Resource for Efficient Deep Learning Machines

Emre Neftci[1,2]
[1]Department of Cognitive Sciences, UC Irvine, Irvine, CA, USA [2]Department of Computer Science, UC Irvine, Irvine, CA, USA Email: eneftci@uci.edu

***Abstract*—Synaptic unreliability was shown to be a robust and sufficient mechanism for inducing the stochasticity in biological and artificial neural network models. Previous work demonstrated multiplicative noise (also called dropconnect) as a powerful regularizer during training. Here, we show that always-on stochasticity at networks connections is a sufficient resource for deep learning machines when combined with simple threshold non-linearities. Furthermore, the resulting activity function exhibits a self-normalizing property that reflects a recently proposed "Weight Normalization" technique, itself fulfilling many of the features of batch normalization in an online fashion. Normalization of activities during training can speed up convergence by preventing so-called internal covariate shift caused by changes in the distribution of inputs as the parameters of the previous layers are trained. Collectively, our findings can improve performance of deep learning machines with fixed point representations and argue in favor of stochastic nanodevices as primitives for efficient deep learning machines with online and embedded learning capabilities.**

I. Introduction

Deep learning can confer adaptability in less controlled environments and more fine-grained context awareness to behaving intelligent agents. In several mobile applications such as autonomous vehicles and surveillance, real-time adaptivity to new situations requires dedicated continual or life-long learning machines, where learning is best achieved on-chip to prevent delays and power overhead incurred in communication with data centers.

Stochasticity in neural networks is a valuable resource for computations involved in deep learning, thanks to its regularizing effect and its smoothing effect on the dynamics for gradient descent learning. Furthermore, a large body of work highlight the stochastic properties of emerging nanodevices [1], [2]. Exploiting the physics of nanodevices for generating stochasticity can lead to significant improvements in dedicated deep learning machines and accelerators.

A family of stochastic neural network models called Dropout [3], [4] are inspired by the probabilistic nature of neural activations and their synaptic quantal release. Dropout techniques greatly improved learning and inference in a number of artificial neural networks and spiking neural networks. Dropout and the closely related DropConnect algorithm [5] can be viewed as *multiplicative noise* that plays the dual role of a regularizer during learning and an efficient mechanism for implementing stochasticity in event-based neural networks over a wide dynamic range [6], while being very suitable for implementation in hardware [7], [8]. Stochastic synapses were shown to behave as stochastic counterparts of Hopfield networks [9], but where stochasticity is caused by multiplicative noise at the synapses (rather than logistic noise in Boltzmann machines). These were shown to surpass the performances of equivalent machine learning algorithms [10], [11] on certain benchmark tasks. Such networks have "always-on" stochasticity, which can be used for probabilistic inference [12].

We find that the dynamics induced by multiplicative synapses are sufficient for deep learning when equipped with a threshold function as the only non-linearity. Relating to previous work [13], we call such networks these Synaptic Sampling Machines (S2Ms). We report the novel finding that neurons equipped with such stochasticity inherently perform normalization of the weights. Normalization plays an important role in speeding up modern deep neural networks by mitigating internal covariate shift caused by the distribution of each layers inputs changes during training, as the parameters of the previous layers change [14]. Of particular interest here is "weight normalization" [15], a technique that normalizes the network weights during learning which is suitable for on-line (non-batch) learning. We prove that the self-normalizing effect in the S2M is statistically equivalent to Weight Normalization in the statistical sense, and demonstrate multilayer S2Ms on GPU simulations and observe its convergence properties.

II. Methods

A. Activation Function of Neural Network with Stochastic Synapses

We formulate a framework that coherently links multiplicative (synaptic) noise to probabilistic inference in stochastic neural networks. For mathematical and implementation simplicity, we use threshold (sign) units:

$$z_i = \Theta(u_i) = \begin{cases} 1 & \text{if } u_i \geq 0 \\ -1 & \text{if } u_i < 0 \end{cases} \forall i, \tag{1}$$

equipped with a combination of deterministic and blank-out synapses (Fig. 1):

$$u_i = \sum_{j=1}^{N} \xi_{ij}^p w_{ij} z_j + a_i (\sum_{j=1}^{N} w_{ij} z_j) = \sum_{j=1}^{N} (\xi_{ij}^p + a_i) w_{ij} z_j, \tag{2}$$

where $\xi_{ij}^p \in \{0, 1\}$ is a Bernoulli process with probability p, and a_i is a factor that controls the scale of the weights w_{ij}.

Thanks to the binary nature of z_i, Eq. (2) is multiplication-free except for the term involving a_i. However, the latter is only performed once per neuron and time step since a_i is unique to the neuron.

Since the ξ_{ij}^p are independent, for large fan-in and p not close to 0 or 1, we can approximate this sum with a Gaussian with mean and variance:

$$\mu_i = \sum_j (p + a_i) w_{ij} z_j, \text{ and } \sigma_i^2 = p(1-p) \sum_j (w_{ij} z_j)^2,$$

respectively. Since u_i is Gaussian-distributed, and $P(z_i = 1|\mathbf{z}) = P(u_i \geq 0|\mathbf{z})$, the probability that unit i is active given the network state is equal to one minus the cumulative distribution function of u_i:

$$P(z_i = 1|\mathbf{z}) = \frac{1}{2}\left(1 + \text{erf}\left(\frac{\mu_i}{\sigma_i\sqrt{2}}\right)\right) = \frac{1}{2}\left(1 + \text{erf}\left(\mathbf{v}_i \cdot \mathbf{z}\right)\right),$$
$$\text{with } \beta_i = \frac{a_i + p}{\sqrt{2p(1-p)}}, \text{ and } \mathbf{v}_i = \beta_i \frac{\mathbf{w}_i}{||\mathbf{w}_i||}. \tag{3}$$

To obtain the last term, we have used the identity $\sqrt{\sum_j w_{ij}^2} = \sqrt{\sum_j w_{ij}^2 z_j^2} = ||\mathbf{w}_i||$, owing to the fact that the square of a sign function is always 1, and where $||\cdot||$ is the L2 norm of the weights of neuron i. The S2M neuron's activation function above shows that synaptic weights are effectively normalized. The term a_i is used to scale the magnitude of the weights, as any scaling factor applied to $\mathbf{w}_i$ would be otherwise canceled out by the norm (See Section III B). Several experimental observations point to this normalization as being the key for understanding S2Ms' performance in terms of sparsity and accuracy [13].

B. Gradient Descent in S2M implements Weight Normalization

A recently proposed weight normalization technique sheds some light on why self-normalizing effect in S2M are beneficial to neural networks. The key idea in weight normalization [15] is to normalize the unit activity by normalizing the weight vectors. Interestingly, the normalization used in [15] takes exactly the same form as in Eq. (3), suggesting that S2Ms inherently perform weight normalization in the sense of [15]. The authors argue that the norm of the weight vector from its direction can speed up convergence, and confers many of the features of batch normalization.

To achieve weight normalization effectively, gradient descent is performed with respect to the scalars β in addition to the weights. The gradients of a loss function $\mathcal{L}$ with respect to the weights $\mathbf{w}$ and scaling factors β become:

$$\partial_{\beta_i}\mathcal{L} = \frac{\sum_j w_{ij}\partial_{v_{ij}}\mathcal{L}}{||\mathbf{w}_i||}$$
$$\partial_{w_{ij}}\mathcal{L} = \frac{\beta_i}{||\mathbf{w}_i||}\partial_{v_{ij}}\mathcal{L} - \frac{\beta_i}{||\mathbf{w}_i||^2}\partial_{\beta_i}\mathcal{L}$$

For all experiments, we used cross-entropy loss $\mathcal{L}^n = -\sum_i t_i^n \log p_i^n$, where n indexes the data sample and p_i is the Softmax output.

III. Results

A. S2Ms Outperform Standard Stochastic Neural Networks in Speed and Accuracy

In order to characterize the classification abilities of the S2M and validate its hardware implementation, we trained a fully connected network on the MNIST hand written digit image database for digit classification[1]. The network consisted of three fully-connected layers of size 300, and a softmax layer for 10-way classification and all Bernouilli process parameters were set to $p = .5$. Inputs were added with Gaussian noise and discretized to -1, 1 using Eq. (1). The S2M was trained using standard root-mean-square gradient back-propagation (rms-prop) using a negative log-likelihood loss and mini-batches of size 100. As a baseline for comparison, we used the stochastic neural network (SNN) presented in [16] without biases, with and sigmoid activation probability $P_{sig}(z_i = 1|\mathbf{z}) = \text{sigmoid}(\mathbf{w}_i \cdot \mathbf{z})$.

The results of this experiment are shown in Table I and convergence is shown in Fig. 3. The 15th, 50th and 85th percentiles of the input distributions to the last hidden layer during training is shown in Fig. 2. The evolution of the distribution is much smoother in the S2M, suggesting that S2M prevents internal covariate shift.

Both speed of convergence and accuracy within 1000 epochs are higher in the S2M compared to the SNN. Attractively, the higher performance in S2M is achieved using inference dynamics that are simpler than the SNN (sign activation function compared to a sigmoid activation function) and using binary random variables.

B. Robustness to Weight Fluctuations

The decoupling of the weight matrix as in $\mathbf{v}_i = \beta_i \frac{\mathbf{w}_i}{||\mathbf{w}_i||}$ introduces several other advantages. During learning, the distribution of the weights for layer tend to remain more stable in S2M compared to SNN (Fig. 4). This feature can be exploited to mitigate saturation at the boundaries of fixed range weight representations (*e.g.* in fixed point representations or memristors). Another subtle advantage from an implementation point of view is that the probabilities are invariant to positive scaling of the weights, *i.e.* $\frac{\alpha\mathbf{w}_i}{||\alpha\mathbf{w}_i||} = \frac{\mathbf{w}_i}{||\mathbf{w}_i||}$. Fig. 3 shows that S2M with weights multiplied by a constant factor .1 during inference did not significantly affect the classification accuracy. This confers S2Ms inherent robustness against certain spurious fluctuations affecting the rows of the weight matrix. Note that this property does not hold for SNNs, where classification is largely impeded by such scaling.

IV. Conclusions

Our results demonstrate that S2Ms can outperform standard stochastic networks on standard machine learning benchmarks on convergence speed and accuracy. This is achieved using strictly simpler inference dynamics, that are well suited to

[1]Simulations were performed using Theano, based on code gracefully provided by [16]. All scripts used to generate the data and figures are available online (https://gitlab.com/NMI-lab/s2mwnorm)

emerging nanodevices, and argue strongly in favor of *exploiting* stochasticity in the devices for deep learning. Several implementation advantages would accrue from this approach: S2Ms are self-normalizing, rendering them robust alternatives to batch normalization and dropout normalizing. The self-normalizing feature can further mitigate saturation at the boundaries of fixed range weight representations and confer robustness against certain spurious fluctuations affecting the rows of the weight matrix.

Compared to SNNs, the weight update rule is slightly more involved as it requires calculating the row-wise L2-norms of the weight matrices, and the derivatives of the erf function. However, these terms are shared for all connections fanning in to a neuron, such that the overhead in computing them is small. Furthermore, based on existing work, we speculate that approximating the learning rule either by hand [17] or automatically [18] can lead to near optimal learning performances, while being implemented with simple primitives.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1652159 "CAREER: Scalable Neuromorphic Learning Machines" and by the Intel Corporation.

REFERENCES

[1] D. Querlioz, O. Bichler, A. F. Vincent, and C. Gamrat, "Bioinspired programming of memory devices for implementing an inference engine," *Proceedings of the IEEE*, vol. 103, no. 8, pp. 1398–1416, 2015.

[2] R. Naous, M. Al-Shedivat, and K. N. Salama, "Stochasticity modeling in memristors," *IEEE Transactions on Nanotechnology*, vol. 15, pp. 15–28, Jan 2016.

[3] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[4] P. Baldi and P. J. Sadowski, "Understanding dropout," in *Advances in Neural Information Processing Systems*, pp. 2814–2822, 2013.

[5] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp. 1058–1066, 2013.

[6] R. Moreno-Bote, "Poisson-like spiking in circuits with probabilistic synapses," *PLoS computational biology*, vol. 10, no. 7, p. e1003522, 2014.

[7] D. Goldberg, G. Cauwenberghs, and A. Andreou, "Probabilistic synaptic weighting in a reconfigurable network of VLSI integrate-and-fire neurons," *Neural Networks*, vol. 14, pp. 781–793, Sep 2001.

[8] S. Sheik, S. Paul, C. Augustine, C. Kothapalli, G. Cauwenberghs, and E. Neftci, "Stochastic synaptic sampling in neuromorphic spiking systems," in *International Symposium on Circuits and Systems, (ISCAS), 2015*, IEEE, May 2016. accepted.

[9] J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[10] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[11] E. Neftci, S. Das, B. Pedroni, K. Kreutz-Delgado, and G. Cauwenberghs, "Event-driven contrastive divergence for spiking neuromorphic systems," *Frontiers in Neuroscience*, vol. 7, Jan. 2014.

[12] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," *arXiv preprint arXiv:1506.02142*, 2015.

[13] E. O. Neftci, B. U. Pedroni, S. Joshi, M. Al-Shedivat, and G. Cauwenberghs, "Stochastic synapses enable efficient brain-inspired learning machines," *Frontiers in Neuroscience*, vol. 10, no. 241, 2016.

[14] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[15] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," *arXiv preprint arXiv:1602.07868*, 2016.

[16] D.-H. Lee, S. Zhang, A. Biard, and Y. Bengio, "Target propagation," *arXiv preprint arXiv:1412.7525*, 2014.

[17] E. O. Neftci, C. Augustine, S. Paul, and G. Detorakis, "Event-driven random back-propagation: Enabling neuromorphic deep learning machines," *Frontiers in Neuroscience*, vol. 11, 2017.

[18] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas, "Learning to learn by gradient descent by gradient descent," in *Advances in Neural Information Processing Systems*, pp. 3981–3989, 2016.

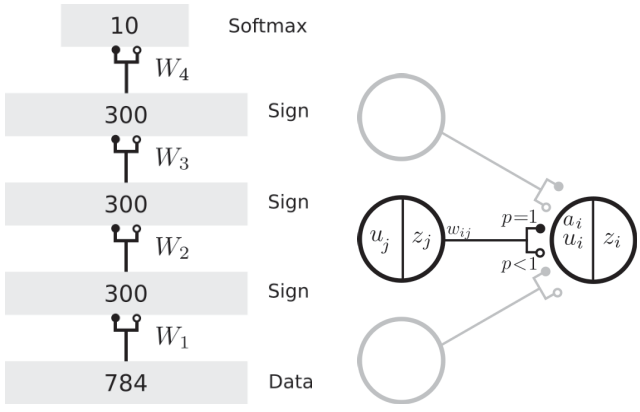
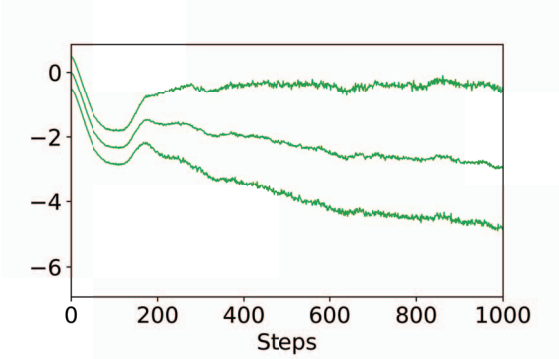


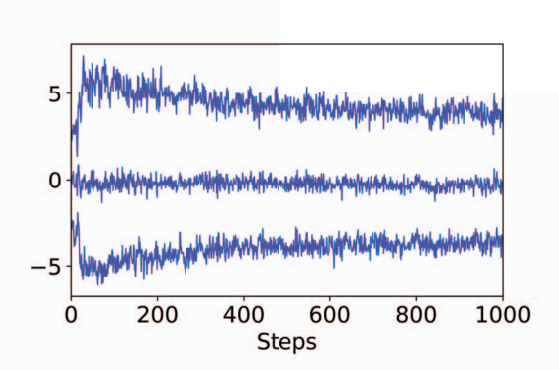
Fig. 1: Network architecture (left) and stochastic connection model (right). Weights are accumulated on u_i as a sum of a deterministic term scaled by a_i (full circles) and a stochastic term with fixed p blank-out probability (empty circles).

Dataset	Network	S2M
PI MNIST	S2M 784-300-300-300-10	1.36 %
PI MNIST	SNN 784-300-300-300-10	1.47 %
PI MNIST	S2M scaled 784-300-300-300-10	1.38 %

TABLE I: Classification error on the permutation invariant MNIST task (test set). Error is estimated by averaging test errors over 100 samples and over the 50 last epochs.



(a) SNN



(b) S2M

Fig. 2: *S2M mitigates internal covariate shift*. 15th, 50th and 85th percentiles of the input distribution to the last hidden layer (similarly to Fig. 1 in [14]). The internal covariate shift is visible in the SNN as the input distributions change significantly during the learning (see epochs 0-200). The self normalizing effect in S2M performs weight normalization, which is known to mitigate this shift and speed up learning. Each step corresponds to one mini-batch update (100 data samples per mini-batch, 50000 data samples total)

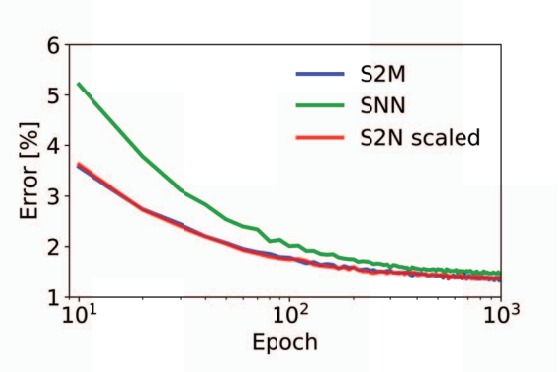
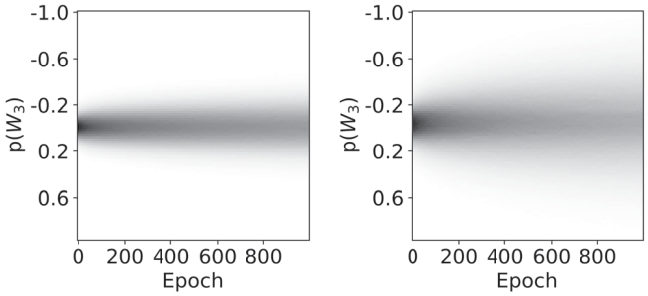
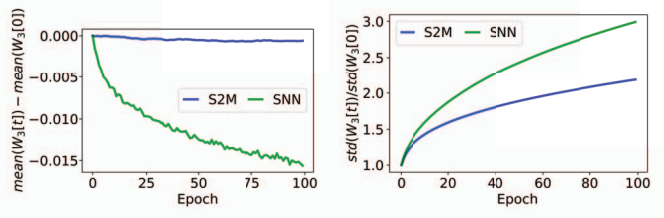


Fig. 3: *Test error on Permutation Invariant (PI) MNIST during training*. S2M scaled is similar to S2M during training, but weights were altered during inference as described in Section III, B.



(a) S2M (b) SNN



(c) Mean of (W_3) (d) Standard deviations of (W_3)

Fig. 4: Evolution of W_3 weight distributions during learning, normalized to initial values. In the S2M, the scale of the weights is controlled by the factors β_i. This renders the weights during learning more stable compared to the SNN, which tends to grow at a faster rate.