The camera-ready version of the manuscript will be modified with all the changes and new results we describe below. Thank you all reviewers for your in-depth comments.

Reviewer 1 We fixed the typos and abbreviations indicated by reviewer 1 in the text. S2M are Stochastic Sampling Machines, corrected. We added results on CIFAR100 using the same CIFAR10 network (test error rates for BNSM: 34.85%, dCNN: 34.37%). Regarding the comments about "smart" initialization on CIFAR10 experiment, we ran experiments without that and we achieved similar results (CIFAR10 with initialization: Test error rate10.2%, without: 9.94%). We restructured the text such that the NSM is better and earlier introduced in the main text, add derivation details and provide the experimental results. Previously unexplained terms (*e.g.*, the offset parameter to the noise) are now elaborated. We have not attempted NSMs in a (deep) RL framework as we are not aware of convincing theoretical/practical work on successful deep RL with binary neural networks. On the other hand, using the RL framework to de-bias learning rules in StNN is an active field of study and explored in MuProp [18] and closely related REBAR. With MuProp, the number of passes in the network for each gradient step scales with the number of layers, which does not scale to the larger networks used here. Backpropagating through the probability function yields a good result while being straightforward to implement in software and hardware. See response to reviewer 3 concerning the learning rule bias. We improved the SI by elaborating on Figure 3, fixing missing sections (4.3) and detailing the DVS gestures dataset and preprocessing.

Reviewer 2 Abbreviation SNN (Stochastic Neural Network) was changed to StNN. We suggest "Inherent Weight Normalization in Stochastic Neural Networks" as a less confusing title. Klambauer et al. in 2017 indeed introduced self-normalization. Our work is different in terms of objective and results. Klambauer et al. construct *mathematically* an activation function with which outputs are normalized in non-binary, deterministic networks. *Weight* Normalization emerges in the NSM unit from the multiplicative noise. That the central limit theorem in such units leads to weight normalization in the Salimans & Kingma sense is original, and establishes a one-of-a-kind connection between exploiting the physics of hardware systems and recent deep learning techniques, while achieving reasonable accuracy on classification tasks. This is highly significant for the devices community, as it implies a simple circuit (threshold operations and crossbars) that can exploit (rather than mitigate) device non-idealities such as stochasticity. Many challenges remain to achieve hardware NSMs, but we believe this is a seminal step. Reviewer suggested three more experiments. So, we tested a binary (sign non-linearity) deterministic network (BD), the same with weight normalization (wBD), a stochastic network (noisy rectifier) (SN) on MNIST (all these networks were trained using a Straight-Through Estimator (STE)) and a deterministic Binary Network (BN) trained using erf in the backward pass giving these results: BD: 3.11%, wBD: 2.72%, SN: 2.05%, BN:1.10%, NSM: 0.70%. Regarding NSM's initialization process, we ran a more experiments using the same allconvnet implementation without pre-trained weight initialization, which achieved similar results (CIFAR10 test error rate 9.95%). We corrected the probability of a neuron firing (line 110, line 108 was correct). We removed the bias term from equations in lines 136 and 142 since it was a mistake (pointed out by the reviewer 2). Regarding Eq. (7), our calculations confirm the derivative w.r.t. β_i. See also Eq. (3) in [47]. The details of the derivation will be added to the SI. We agree with the expression "where β models the effects due to the noise, ξ". a_i (line 129) is critical for controlling β_i. We did initially investigate whether β_i can be controlled by tweaking σ (Gaussian noise) or p (Bernouilli noise) instead. The former case would work theoretically, but in the latter case, the free parameter is $0 < p < 1$. The bounds translate to bounds on a_i that are too restrictive for training the networks. Note that β_i is a trainable parameter and it (or a_i) must be stored anyway, and using a_i is process-independent and so preferable. The gradients (line 170) are biased because they are estimated using switching probability rather than the realization Eq (3). The NSM is compatible with the binary case of Concrete relaxations [31] with fixed temperature, which are unbiased w.r.t the continuous network. To assess the bias, we trained the NSM using BinConcrete units [31] on MNIST (test error rate: 0.78%). We observed that the angles between two gradients (current and BinConcrete) are close. The training indeed relies on automatic differentiation (line 174) but probabilities are only computed for the gradients. NSMs propagate forward binary states only. A computational graph will detail this in the SI. We removed the confusing line 187 regarding the hardware implementation, as the NSM non-volatile memory (NVM) crossbar implementation is not documented here (see subsection 1.3 in the main text). The expression in line 191 was changed to "NSM was trained using back-propagation and a softmax layer with cross-entropy loss and mini-batches of size 100". The dataset splitting in training/testing is standard (50K/10K for MNIST, and 50K/10K for CIFAR). By "error was computed over 100 samples" we meant that for testing, each mini-batch is run 100 times (MC samples) with different seeds. Classification is made on the average over the MC samples. We'll change the wording.

Reviewer 3: Our original statement about continual learning was wrong. We replaced it by "online learning". NSM theoretically supports online learning since it implements weight normalization, which is not based on batches [47] (online learning capability is relevant to physical neural networks since batching is not possible). NSM training on hardware is out of scope here, but it is in principle compatible with analog NVMs: The gradient of the erf is a Gaussian, and so gradients w.r.t. v (see main text) can be sampled with noise in the device. The gradient w.r.t. β are more challenging, but only one per neuron needs to be computed, which can be done in peripheral CMOS. We agree with the reviewer that the present paper would benefit from a quantitative comparison with STE. To this end we, trained an NSM ($\sim 1\%$) and an NSM ($\sim 3\%$) using a STE on MNIST. The proposed NSM was more accurate.