Journal of
Low Power Electronics and Applications

MDPI

Article

Towards Low-Power Machine Learning Architectures Inspired by Brain Neuromodulatory Signalling

Taylor Barton [1,†], Hao Yu [2,†], Kyle Rogers [2,*,†], Nancy Fulda [2], Shiuh-hua Wood Chiang [1], Jordan Yorgason [3] and Karl F. Warnick [1]

[1] Electrical & Computer Engineering, Brigham Young University, Provo, UT 84602, USA
[2] Computer Science, Brigham Young University, Provo, UT 84602 , USA
[3] Cellular Biology and Physiology, Center for Neuroscience, Provo, UT 84602, USA
* Correspondence: kroger25@byu.edu
† These authors contributed equally to this work.

Abstract: We present a transfer learning method inspired by modulatory neurotransmitter mechanisms in biological brains and explore applications for neuromorphic hardware. In this method, the pre-trained weights of an artificial neural network are held constant and a new, similar task is learned by manipulating the firing sensitivity of each neuron via a supplemental bias input. We refer to this as neuromodulatory tuning (NT). We demonstrate empirically that neuromodulatory tuning produces results comparable with traditional fine-tuning (TFT) methods in the domain of image recognition in both feed-forward deep learning and spiking neural network architectures. In our tests, NT reduced the number of parameters to be trained by four orders of magnitude as compared with traditional fine-tuning methods. We further demonstrate that neuromodulatory tuning can be implemented in analog hardware as a current source with a variable supply voltage. Our analog neuron design implements the leaky integrate-and-fire model with three bi-directional binary-scaled current sources comprising the synapse. Signals approximating modulatory neurotransmitter mechanisms are applied via adjustable power domains associated with each synapse. We validate the feasibility of the circuit design using high-fidelity simulation tools and propose an efficient implementation of neuromodulatory tuning using integrated analog circuits that consume significantly less power than digital hardware (GPU/CPU).

Keywords: power-constrained devices; low-power analog learning; neural network; spiking neural network; neuromorphic; analog CMOS; life-long learning; machine learning; transfer learning; fine-tuning

Citation: Barton, Taylor; Yu, Hao; Rogers, Kyle; Fulda, Nancy; Chiang, Shiuh-hua Wood; Yorgason, Jordan; Warnick, Karl F. Towards Low-Power Machine Learning Architectures Inspired by Brain Neuromodulatory Signalling. *J. Low Power Electron. Appl.* **2022**, *12*, 59. https://doi.org/10.3390/jlpea12040059

Academic Editors: Huanglong Li and Si Wu

Received: 30 September 2022
Accepted: 1 November 2022
Published: 4 November 2022

1. Introduction

Analog CMOS hardware has the potential to reduce energy consumption of deep neural networks by orders of magnitude, but the in situ training of networks implemented on such hardware is challenging. Once the chip has been programmed with the correct weight values for a task, typically no further learning occurs. We introduce a biologically-inspired knowledge transfer approach for neural networks that offers potential for in situ learning on the physical chip. In our method, the weight matrices of a spiking neural network [1–5] are initialized with values learned via offline (i.e., off-chip) methods, and the system is exposed to an analogous—but distinct—learning task. The bias inputs of the chip's spiking neurons are manipulated such that the network's outputs adapt to the new learning task.

This approach has applications for autonomous, power-constrained devices that must adapt to unanticipated circumstances, including vision and navigation in unmanned aerial vehicles (UAVs) deployed into unpredictable environments; fine-grained haptic controls for robotic manipulators; dynamically adaptive prosthetic devices; and bio-cybernetic

interfaces. In these real-world domains, the system must deploy with initial knowledge relevant to its target environment, then adapt to near-optimal behavior given minimal training examples, a feat beyond the capability of current learning algorithms or hardware platforms. Neuromodulatory tuning offers a path toward implementing such abilities on physical CMOS chips. The key contributions of our work are as follows:

1. We introduce a novel transfer learning variant, called neuromodulatory tuning, that is able to match the performance of traditional fine-tuning approaches with orders of magnitude fewer weight updates. This lends itself naturally to easier, lower power implementation on physical chips, especially because the proposed CMOS implementation of our the fine-tuning method does not involve writing to memory hardware.
2. We provide a biologically-inspired motivation for this tuning method based on recent findings in neuroscience, and discuss additional insights gleaned from modulatory neurotransmitter behaviors in biological brains that may prove valuable for neuromorphic computing hardware.
3. We demonstrate in both traditional (non-analog) feed-forward architectures and spiking neural network simulations that neuromodulatory tuning methods are able to approach or exceed the performance of traditional fine-tuning methods on a number of transfer learning tasks in the domain of image recognition, while overall task performance must still be improved, the trends and potential of the method are encouraging.
4. We outline the mechanisms by which neuromodulatory tuning can feasibly be implemented on CMOS hardware. We present an analog spiking neuron with neuromodulatory tuning capabilities. Post-layout simulations demonstrate energy/spike rates as low as 1.08 pJ.

The remainder of this paper adheres to the following structure: We begin by providing a general background on transfer learning, artificial neural networks, and neuromorphic hardware in Section 2. We then outline the motivating principles and neurobiological foundations of the current work (Section 3.1) and present our biologically inspired tuning method (Section 3.2). A preliminary analysis follows (Section 4), showing performance comparisons of NT versus TFT in digital computation environments across a variety of learning rates and transfer tasks. Lastly, we present our spiking neuron design (Section 5) with confirming evidence that our neuromodulatory tuning method can be used as an acceptable proxy for traditional fine-tuning in analog CMOS environments (Section 6). Conclusions are presented in Section 7.

2. Background

The current study lies at the intersection of three prodigious research fields: Transfer learning (Section 2.1), spiking neural networks (Section 2.2), and neuromorphic computing (Section 2.3). We outline key principles of each below. Our method also draws heavily on recent discoveries in neuroscience, documented alongside the motivating principles of this research in Section 3.1.

2.1. Transfer Learning

Transfer learning allows a network trained for one task to learn a new, similar task with less computational complexity than fully retraining the network. The field includes a broad range of techniques ranging from weighting, importance sampling, and domain adaptation in unsupervised contexts [6–11], to fine-tuning and multi-task learning in supervised settings [12–18]. Recent work in few-shot, one-shot, and zero-shot learning also contributes to this line of research [19–22].

Our approach can be combined with many of these methods, but is most closely related to feature learning from unsupervised data [13], whereby trained parameters from a related task are used to jump-start the learning process. Our method is distinct in that the activation sensitivity of individual neurons, rather than the strengths of their synaptic

connections, are modified. In some sense, this can be viewed as a degenerate form of neural programming interface [23], in that activation patterns are modulated during each forward pass of the network; however, our method adjusts firing sensitivities via supplemental bias inputs rather than by overwriting output signals directly. Our work also has tangential relations to activation function learning [24], although we adjust firing sensitivity only, rather than changing the shape of the activation curve.

Parallel to our work, ref. [25] presented BitFit, which shows bias tuning is an effective sparse fine-tuning method that is competitive with traditional fine-tuning on Transformer-based Masked Language Models. Our work augments and expands upon the insights from this work in two key ways: We apply a bias tuning methodology much like [25] to a convolutional neural network in the domain of computer vision, where we discover that it is not able to match the performance of a traditional fine-tuning method, and we present a novel approach to bias tuning (neuromodulatory tuning) based on multiplicative rather than summative layer modifications, and demonstrate that this method is able to match traditional fine-tuning approaches.

2.2. Spiking Neural Networks

Spiking neural networks (SNNs) [1,3,4,26–28] are artificial neural networks that attempt to mimic temporal and synaptic behaviors of biological brains. Rather than using continuous activation functions, spiking neurons utilize a series of binary pulses, called a spike train [29], to propagate information forward in a brain-like manner. SNNs are particularly well-suited to implementation on analog/mixed-signal hardware, which naturally supports the high parallel sparse activation pathways common in such networks [30].

Despite these potential advantages and their strong parallels with biological brain behavior, SNNs have not gained as much recent prominence as traditional (digital) feed-forward networks, in part because of the difficulty of propagating gradient information backwards through a spike train [31]. One means to compensate for this is by training a traditional (non-spiking) network using back-propagation and then applying a transfer function to convert the learned weights into their SNN equivalents [32]. We leverage this idea in our work, but instead of applying a transfer function, we copy the non-spiking weights directly, then use neuromodulatory tuning to adapt them to a new learning task.

Recent works detailing the conversion of traditional feed-forward networks to SNNs use algorithms which modify weights, biases and activation thresholds of the network to create a SNN from a feed-forward network [33,34]. The difference between our work and others is that we do not train the network to match the behavior with existing feed-forward network. Instead, we seek to train network for different tasks. Therefore, we do not perform layer-wise comparison which is resource consuming. Moreover, our work tunes a single parameter per neuron which is far more implementable on physical chips compared to other more computationally expensive methods.

2.3. Neuromorphic Hardware

Neuromorphic hardware uses dedicated processing units to implement neuronal connections and firing behavior directly on a physical chip, rather than simulating them mathematically. Analog neuromorphic hardware has been shown to be more power efficient than traditional digital computation hardware, and does not suffer from the same bottleneck as Von Neuman computing [35–42]. Some designs take advantage of sub-threshold operation for ultra-low power neurons [43,44]. Further power reductions have been achieved through sparse temporal coding [30].

The temporal nature of spiking neural networks naturally lends itself to on-chip, biologically plausible learning methods. Spike-time-dependent plasticity (STDP) uses analog hardware to directly implement learning rules on chip. Several works have shown impressive learning accuracies using this method [29,35,45–47]. However, direct hardware implementations for learning rules consume large amounts of space and power, limiting its

potential learning capacity. Our work bridges this gap by offering the possibility of on-chip learning with similar performance but reduced space and component requirements.

3. Neuromodulatory Tuning

Neuromodulatory tuning is a novel fine-tuning method based on recent discoveries in neuroscience. Neuronal transmission in biological brains is highly complex in timing and can occur either via rapidly terminating signals that influence only immediately connected cells (synaptic transmission), or via chemical signals that spread further away to simultaneously influence larger groups of neurons (volumetric transmission) [48,49]. Our work is motivated by and takes inspiration from this non-synaptic transmission method. Specifically, we observe that, rather than adjusting connection strengths between neurons directly, modulatory neurotransmitters impact system behavior by affecting the activation threshold of each neuron. Thus, a single trainable parameter, implemented in our case as a supplementary input, can be used in lieu of the large suite of trainable parameters typically employed during a fine-tuning process.

3.1. Biological Foundations

Modulatory neurotransmitters in biological brains use metabotropic g-protein coupled receptors as opposed to strictly ion conducting receptors propagate signals, and can include neurotransmitters such as the cathecholamines dopamine and norepinephrine [50–54]. Interestingly, glutamate is also used by neurons as a modulatory metabotropic signal, though it is largely discussed in the context of ion channel activity [55].

Artificial neural networks principally use neuronal ion channel activity, as represented by classical synapses, to represent synaptic strength. In contrast, metabotropic neuromodulators activate g-protein coupled receptors in neurons, whose downstream effectors can be stimulatory or inhibitory (depending on predefined cellular components) and work through a series of effectors that can amplify signals from traditional synaptic inputs, resulting in multiplicative tuning of the neuron's inputs. This is considered a tuning process since these neurotransmitters often do not directly change the membrane potential, but instead change the activation threshold by modulating the channels receiving inputs. Our neuromodulatory tuning method simulates this increase or decrease in sensitivity by including additional inputs to the incoming signal, as shown in Section 5. In other words, neuromodulatory tuning increases a model's sensitivity to specific pre-learned features, rather than changing the functions represented by those features. To our knowledge, this is the first application of volumetric, as opposed to strictly synaptic, mesolimbic attention modalities within an analog CMOS system.

3.2. Implementation

We simulate increased or decreased resting cell voltage via the introduction of a supplementary bias neuron for each network layer to be fine-tuned, as shown in Figure 1. The weights connecting this bias to neurons within each layer are initialized according to a random uniform distribution, and, if the number of output categories has changed from the original task, a new output layer is appended to the model. These additional bias weights are multiplied to the pre-trained weights in each layer of a network selected for fine tuning. The additional bias weights are then adjusted using standard back-propagation methods while all original weights from the pre-trained model are held fixed. This multiplicative bias method outperformed traditional additive bias, presented by [25], in experiments shown Table 1.

Alternately, neuromodulatory tuning can also be implemented by unfreezing only the existing bias weights of the pre-trained model, leaving all other weights fixed. We denote the additional bias neuron implementation as NT_1, and denote this unfreezing bias weights implementation as NT_2. Although the representational capacity of both methods is equivalent from a theoretical standpoint, we find that, empirically, introducing additional bias neurons (NT_1) functions slightly better in deep feed-forward networks as shown in

Table 2. Consequently, we use NT_1 in our experiments with feed-forward networks in Section 4. In spiking networks, we compare both implementations (NT_1 and NT_2) and we find NT_1 performs better on STL-10 dataset, but has similar performance with NT_2 on Food-11 and BCCD as shown in our experiments with spiking networks in Section 6.1.

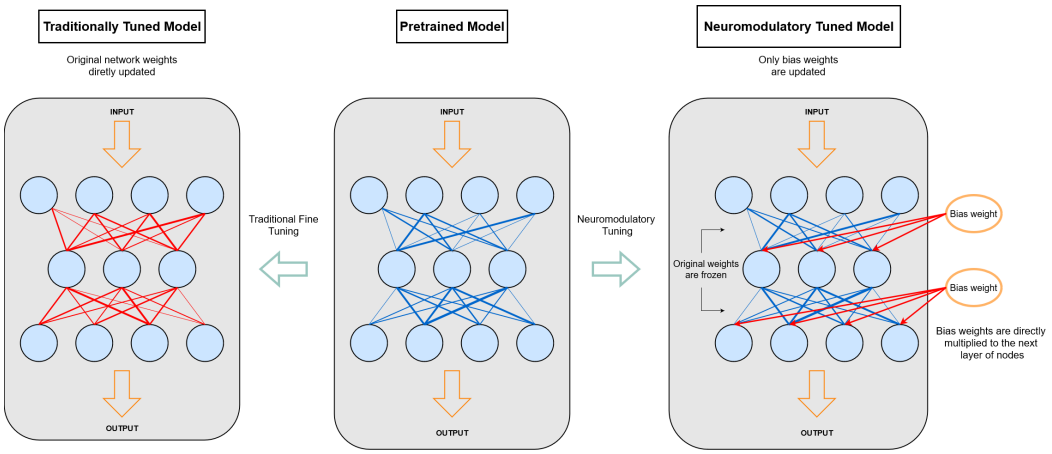

Figure 1. Depiction of neuromodulatory tuning (NT) in contrast with traditional fine-tuning (TFT). In NT, the weights of the pre-trained network are frozen, preserving all learned feature information pertaining to the original training task. A set of auxiliary bias neurons with randomly initialized weights is then inserted into the network, and the auxiliary bias weights are then updated in response to the new learning task. In this diagram, color indicates the weights' update status: red for active, blue for frozen. NT requires far fewer parameter updates than traditional fine-tuning methods, although loss information must still be propagated backward through the entire network.

Table 1. Validation accuracy on STL-10, Food-11, and BCCD datasets after 5 epochs, mean of five training runs using learning rate (lr) = {0.01} and and using the full training set for each dataset after being balanced.

	STL-10 acc	**Food-11 acc**	**BCCD acc**
additive bias tuning	0.1000	0.0863	0.2498
multiplicative neuromodulatory tuning	0.8447	0.7110	0.3966

Table 2. Validation accuracy on the STL-10, Food-11, and BCCD dataset after 10 epochs, mean of five training runs using learning rate (lr) = {0.1, 0.01, 0.001, 0.0001} and using the full training set for each dataset after being balanced. NT_1 = additional bias implementation, NT_2 = modify existing bias implementation. Highest average accuracies are bolded.

	STL-10 acc	**Food-11 acc**	**BCCD acc**
NT_1 (lr = 0.1)	0.8237	0.6819	0.3864
NT_2 (lr = 0.1)	0.7626	0.5309	0.3280
NT_1 (lr = 0.01)	0.8491	0.7184	0.4126
NT_2 (lr = 0.01)	0.8420	0.7030	0.3800
NT_1 (lr = 0.001)	0.8429	0.6929	0.3986
NT_2 (lr = 0.001)	0.8517	0.7173	0.4234
NT_1 (lr = 0.0001)	0.7856	0.5946	0.3939
NT_2 (lr = 0.0001)	0.8333	0.6631	0.4008
NT_1—average	**0.8253**	**0.6720**	**0.3979**
NT_2—average	0.8224	0.6536	0.3831

4. Modeling and Analysis

We first probe the capabilities and weaknesses of neuromodulatory tuning (NT) in a traditional deep learning setting. Using a pre-trained VGG-19 network architecture, we fine-tune the model on three image recognition tasks. VGG-19 was trained on ImageNet [56], an image classification dataset composed of 1000 different image categories. The first dataset we use in our evaluation is STL-10, a subset of ImageNet with only 10 image categories [57]. We expect traditional fine-tuning (TFT) and neuromodulatory tuning (NT) to achieve high accuracies on STL-10 since the data is a subset of the original training data. Next, we evaluate neuromodulatory tuning on a more difficult food classification task, Food-11 [58], which contains images of 11 different types of food none of which match any of ImageNet's classes. Finally, we examine the capability of neuromodulatory tuning to learn blood cell classification (BCCD) [59], which is a task very distinct from ImageNet containing 4 classes of blood cells images. We hypothesize that as the difficulty of the tasks increase, NT will be less effective in tuning the model to solve the given task, but still comparable to TFT.

For simplicity, fine tuning is applied only to the VGG-19 classifier layers, a process which lowers the fine-tuned classification accuracy but facilitates our comparisons to spiking neural network implementations in Section 5.1. Additionally, it is common practice to only fine tune select layers of VGG models in recent literature [60,61]. We then apply neuromodulatory tuning to the same layers that were fine-tuned (i.e., classification layers only) and compare the performance of traditional fine tuning (TFT) to neuromodulatory tuning (NT), as shown in Table 3.

To visualize the comparison between neuromodulatory tuning (NT) and traditional fine-tuning (TFT), we create two model architectures, one with hyper-parameters configured for NT and the other for TFT. We use the existing train and validation partitions in the STL-10, Food-11 and BCCD datasets to train and evaluate the classifier layers of the pre-trained VGG-19 model. We resize the data in each of the datasets to be images of size 256×256 to be compatible with VGG-19. Using an NVIDIA GeForce RTX 2080 Ti GPU, we fine-tune both models for 10 epochs, with various training set sizes and learning rates.

We set the batch size to 64 training instances in all experiments with neural networks. The effect of batch size on model performance has been studied in depth in recent literature. Kandel and Castelli [62] study the effect of varying batch size and learning rate on VGG-16, and also provide a literature review which details several papers concerning the properties of training batch sizes. From these sources, it is clear that batch size and learning rates are dependent, but the measure of dependence often differs depending on the given task, model, and optimizer. Thus, we run a quick experimental analysis of the effect of batch size for a given learning rate on VGG-19 and the Food-11 dataset in Table 4. The learning rate for NT is set to be 0.01 and it is set to 0.0001 for TFT, since these learning rates performed well in preliminary results. As evident from the results in Table 4, we see that batch size does not effect the validation accuracy of NT or TFT models significantly. Therefore, we can fix batch size to 64 in the remainder of our experiments with varying learning rates.

To perform gradient descent we use Cross Entropy Loss and the Adam optimizer. After tuning, we iterate through the entire predefined validation set to find the mean loss and accuracy for a specific model (NT or TFT) and learning rate.

Our results show that algorithm performance between traditional fine-tuning (TFT) and neuromodulatory tuning (NT) is largely on par, a result that remains consistent across a wide variety of learning rates. Table 3 provide our experimental data that highlights the best-performing learning rates for NT (lr = 0.01) and TFT (lr = 0.0001). Interestingly, the optimal learning rate for each tuning algorithm differs, and the average performance of NT across multiple learning rates is higher than that of TFT. TFT achieves the highest validation accuracies overall, but critically, not by much. This is important because it means we can retain much of TFT's learning accuracy while using four orders of magnitude fewer trainable parameters, a circumstance that makes NT far more feasible than TFT to implement on neuromorphic hardware.

Recognizing our success in the results presented above, we further reduced the number of tunable parameters. The reduction in parameters was biologically motivated such that each tunable parameter matches to a single neuron in the classifier layers of VGG-19. Specifically, our initial results as reported in Table 3 include a set of tunable parameters applied after the VGG-19 convolutional layers but before the data was passed into the VGG-19 classifier. Table 5 shows the same experiment repeated with this additional layer of parameters removed, resulting in an even smaller number of trainable parameters—a critical factor for potential implementation of such methods within the space constraints of physical analog chips. We found that this reduction in parameters did decrease the accuracy of the network on each task, but only slightly. As this reduced parameter count is more analogous to biological neuromodulatory transmitters, we use this NT configuration in future experiments in Section 5.1.

Table 3. Validation accuracy on the STL-10, Food-11, and BCCD datasets after 10 epochs, mean of five training runs using learning rate (lr) = {0.1, 0.01, 0.001, 0.0001, 0.00001} and using the full training set for each dataset after being balanced. Highest average accuracies and highest best-performing accuracies are bolded.

	STL-10 acc (n = 500)	**Food-11 acc (n = 280)**	**BCCD acc (n = 2400)**
NT_1 (lr = 0.1)	0.8237	0.6819	0.3864
TFT (lr = 0.1)	0.1031	0.0876	0.2487
NT_1 (lr = 0.01)	0.8491	0.7184	0.4126
TFT (lr = 0.01)	0.1540	0.0987	0.2496
NT_1 (lr = 0.001)	0.8429	0.6929	0.3986
TFT (lr = 0.001)	0.8617	0.7184	0.2509
NT_1 (lr = 0.0001)	0.7856	0.5946	0.3939
TFT (lr = 0.0001)	0.8836	0.8060	0.4291
NT_1 (lr = 0.00001)	0.4969	0.2218	0.3484
TFT (lr = 0.00001)	0.8724	0.7387	0.4209
NT_1—average	**0.7596**	**0.5819**	**0.3880**
TFT—average	0.5750	0.4899	0.3198
NT_1—best	0.8491	0.7184	0.4126
TFT—best	**0.8836**	**0.8060**	**0.4291**
NT_1—tuned parameters	43,290	44,291	37,284
TFT—tuned parameters	123,652,866	123,653,867	123,646,860

Table 4. Validation accuracy on the Food-11 dataset after 10 epochs, mean of ten training runs using bath sizes (bs) = {16, 32, 64, 128}, and using the full training set for the Food-11 dataset after being balanced. The batch size 128 is too large for the TFT setup and is thus omitted. The best learning rates for TFT and NT_1 methods were determined from preliminary results.

	acc (bs = 16)	**acc (bs = 32)**	**acc (bs = 64)**	**acc (bs = 128)**
NT_1 (lr = 0.01)	0.6924	0.6861	0.6933	0.7162
TFT (lr = 0.0001)	0.7900	0.8068	0.8118	-

Table 5. A repeat of the experiments sin Table 3, but with a large subset of neuromodulatory inputs removed. Validation accuracy on the STL-10, Food-11, and BCCD datasets after 10 epochs, mean of five training runs using learning rate (lr) = {0.01, 0.0001} and using the full training set for each dataset after being balanced. The best learning rates for TFT and NT_1 methods were determined from results in Table 3.

	STL-10 acc (n = 500)	Food-11 acc (n = 280)	BCCD acc (n = 2400)
NT_1 (lr = 0.01)	0.8213	0.7056	0.3680
TFT (lr = 0.0001)	0.8836	0.8060	0.4291
NT_1—tuned parameters	18,202	19,203	12,196
TFT—tuned parameters	123,652,866	123,653,867	123,646,860

5. Methods

5.1. Neuromodulatory Tuning on Spike Neural Networks

The VGG-19 architecture is complex and difficult to implement in its entirety on a SNN architecture, in particular due to the large number of convolution and max pooling layers. Since our research goal is to explore the learning effect of neuromodulatory signalling on brain-like architecture, and not to replicate VGG-19, we apply the following simplification in our experiments: The feature layers of VGG-19 are retained in their original (digital) deep format. As illustrated in Figure 2, image inputs are passed through these layers to attain a feature embedding, which would normally be passed through to the VGG-19 classification layers. We replace the VGG-19 classification layers with a spiking neural network having the same number of layers and layer width. The weight matrices of these SNN-VGG classification layers are initialized to the same values as the pre-trained VGG-19 weights.

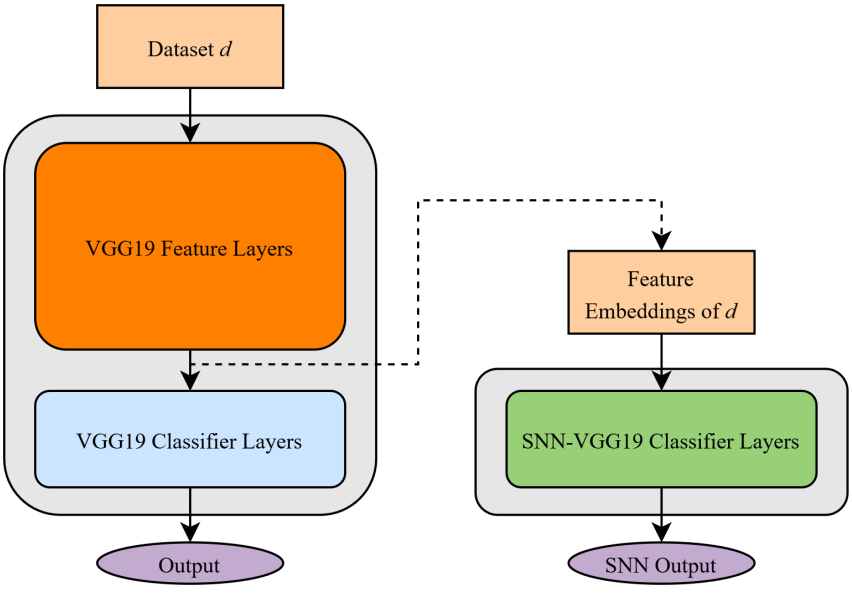

Figure 2. Representation of the Spiking Neural Network (SNN) experimental setup. In these experiments we construct a SNN that mimics the function and purpose of the traditional pretrained VGG-19 classifier layers. To accomplish this, we pass data from a dataset d, where d = {STL-10, Food-11, BCCD}, through the feature layers of VGG-19 to generate a feature embedding for a particular data instance. A traditional usage of VGG-19, like in Section 4, would then pass the feature embedding through the fully-connected classifier layers to produce a model prediction. In these experiments, however, we pass the feature embedding through spiking classifier layers which then in turn produce a spiking model prediction.

We implement our spiking neural network using core algorithm components outlined by leaky integrate-and-fire model [63], with the following adjustments:

- Network update frequency minimization
- Customized simple loss calculation method on network output

5.1.1. Update Frequency Minimization

A typical leaky integrate-and-fire neuron receives input over a set time span. During this time span, neurons must be updated multiple times to simulate temporal connectivity on the actual circuit [29,35,45–47], which greatly increases the computation costs of simulation. Since temporal connections are not a major factor in the VGG-19 image classification tasks, our update frequency for each neuron can be as small as 1 timestep for each task. Therefore, in our simulation for this experiment, we update neurons in each SNN layer exactly once.

Since we update neurons in each SNN layer exactly once, neurons will only fire at most once. As a consequence, argmax is not applicable on our output layer. Argmax chooses the maximum value from the output neurons as the true output, which make sure the output to be exactly one classification. In absent of argmax, network will start to output multiple classifications through activation of multiple neurons, which will be counted as mis-classification. Therefore, the network should not only activate the correct neuron, but it also should avoid the activation of incorrect neurons. Let n be the numbers of neurons which equals to numbers of classes in the tasks. Let p be the actual accuracy of random outputs, then:

$$p = \frac{1}{2} \cdot \frac{1}{2^{n-1}} = \frac{1}{2^n} \tag{1}$$

of which $\frac{1}{2}$ is the possibility of the correct neuron activates and $\frac{1}{2^{n-1}}$ is the possibilities of all incorrect neurons do not fire.

5.1.2. Simple Loss Calculation

For each neuron in our SNN output, one spike indicates an output of 1.0 and no spikes represents an output of 0.0. Therefore, the output of the SNN for each input will be an array consisting exclusive of values in $\{0.0, 1.0\}$. Due to the simplicity of the output as a binary array, we employ a customized simple gradient calculation method on the network output, calculated as follows:

$$\text{loss} = \text{target} - \text{output} \tag{2}$$

This simple method fits our SNN simulation for this experiment, because of the binary output nature of our SNN. A binary output simply indicates whether a neuron fired or not. Losses on the binary output imply whether the neurons on the output layer have fired or not. Therefore, the polarity of the SNN output loss (i.e., whether it is positive or negative) is sufficient for basic training. We believe that other, more complex loss calculation methods have potential to perform better on these tasks, and that will be left to future explorations.

5.1.3. Gradient Calculation

Our network behaves according to the following equations:

$$v_i = \left(\sum_{j=0}^{n} O_j w_{ij} + b_i \right) a_i \tag{3}$$

$$O_i = H(v_i) I \tag{4}$$

$$H(v) = \begin{cases} 1, & \text{if } v \geq \theta \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

where v_i is the voltage of the neuron i, w_{ij} is the weight of the input given by neuron j to neuron i, b_i is the additive bias of the neuron i, a_i is the amplifier bias, O_i is the output of neuron i calculated by our Heaviside function H times I, which represents a neuron's output if fires, and θ is the activation threshold of neuron.

The gradient will then be calculated as:

$$g_{w_{ij}} = \frac{dOutput}{dO_i} \frac{dO_i}{dH_i} \frac{dH_i}{dv_i} \frac{dv_i}{dw_j} \cdot loss \tag{6}$$

$$g_{a_i} = \frac{dOutput}{dO_i} \frac{dO_i}{dH_i} \frac{dH_i}{dv_i} \frac{dv_i}{da_i} \cdot loss \tag{7}$$

Since many researchers implement sigmoidal neurons, with steep sigmoid function, as a replacement for Heaviside step function, we can safely assume:

$$\frac{dH_i}{dv_i} \approx \frac{dSigmoid(dv_i)}{dv_i} \tag{8}$$

The sigmoid method in popular machine learning libraries behaves as follows:

$$\frac{dSigmoid(v)}{dv} = \begin{cases} 1+s, & \text{if } v \geq \theta, s \approx 0 \\ d, & \text{if } v \approx \theta, \text{with } 0 < d < 1 \\ s, & \text{otherwise} \end{cases} \tag{9}$$

where s approaches 0, but never reaches 0. Most of modern day techniques requires sigmoid to be steep, to minimize the window of $v \approx \theta$. Therefore, our method seeks to remove the influence of $v \approx \theta$ by using customized sigmoid derivative σ:

$$\frac{dSigmoid(v)}{dv} \approx \sigma_v = \begin{cases} 1, & \text{if } v \geq \theta \\ s, & \text{otherwise, with } s \approx 0 \end{cases} \tag{10}$$

However, this σ function causes firing neurons to be adjusted $1/s$ times faster than non-firing neurons. Such behavior becomes most problematic on physical chips, due to the fact that weight has its upper limit on physical chips. To make sure the weight adjustment speed on non-firing neurons matches firing neurons we amplified the gradient on non-firing neurons by $1/s$. Then, $\sigma = 1$ for all firing and non-firing neurons.

As a result, our gradient function becomes:

$$g_{w_{ij}} \approx \frac{dOutput}{dO_i} \frac{dO_i}{dH_i} \frac{dv_i}{dw_{ij}} \cdot loss \tag{11}$$

$$g_{a_{ij}} \approx \frac{dOutput}{dO_i} \frac{dO_i}{dH_i} \frac{dv_i}{da_i} \cdot loss \tag{12}$$

Since $\frac{dv_i}{dw_{ij}} = O(v_j) = H(v_j) \cdot I$, and Heaviside step function produces 0 when $v_i < \theta$, the gradient chain will break when the Heaviside function outputs 0.

Therefore, our Heaviside step function on the simulation side is modified as:

$$H_{sim}(v) = \begin{cases} 1, & \text{if } v \geq \theta \\ s, & \text{otherwise, with } s \approx 0 \end{cases} \tag{13}$$

If s is small enough as it approaches 0, s poses no influence on the accuracy of simulation comparing to hardware performance.

5.2. Neuromodulatory Tuning on Analog Hardware

One particularly advantageous aspect of neuromodulatory tuning (NT) is its suitability for implementation on analog neuromorphic hardware. The behavior of fine-tuned bias connections, implemented in digital simulations as additional bias neurons, can also be implemented in analog hardware as a current source with a variable supply voltage. This approach has the following advantages:

- Minimal additional chip area required
- Lower power consumption than digital hardware
- No need to re-load weights to the on-chip memory

To probe this possibility, we use Cadence Virtuoso to explore the feasibility of a NT approach on simulated analog hardware. Our hardware is designed and simulated at the transistor level in TSMC 28-nm CMOS. The analog neuron implements the leaky integrate-and-fire model [63]. Six binary-scaled current sources make up the synapse. A current is driven onto a 50-fF capacitor to produce an integrated membrane voltage that is quantized by a dynamically clocked latched comparator. An adjustable delay line generates a 100-ns spike when the membrane voltage reaches the activation threshold and resets the membrane voltage by connecting the capacitor to ground via a pull-down transistor. A schematic diagram of our proposed neuron is shown in Figure 3.

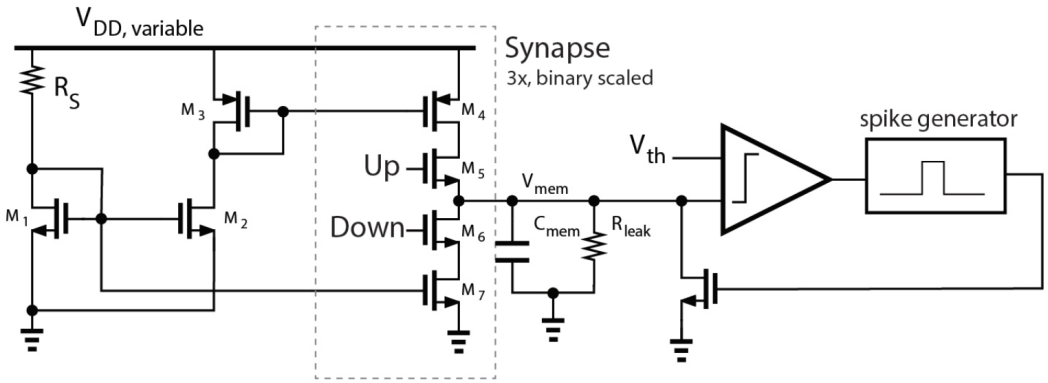

Figure 3. Schematic diagram of the proposed leaky integrate-and-fire neuron with NT ($V_{DD,variable}$) capabilities. The Up and Down signals are generated from the input spike and weight signals.

5.2.1. Synapse Design

Each synapse operates at a supply voltage between 0.5 and 1 V. A higher supply increases the current in the synapse. The neuron core operates at a constant supply of 1 V. Adjusting the supply voltage of individual synapses or groups of synapses effectively changes the weights of the synapse connections. This change in behavior is analogous to the bias neurons in the software implementation and to what is observed biologically [53,54]. To make the synapse current dependent on the supply voltage V_{DD}, we use a current mirror with a resistive load. The current through an N-type MOSFET is given by

$$I_{DS} = \frac{1}{2}\beta(V_{GS} - V_{th})^2 \tag{14}$$

In a current mirror, the gate voltage V_G is related to V_{DD} by

$$V_G = V_{DD} - IR_S \tag{15}$$

Substituting (14) into (15) and solving for I results in

$$I = \frac{\sqrt{(4\beta R_S(V_{DD} - V_{th}) - 1) + 2\beta R_S(V_{DD} - V_{th}) + 1}}{2\beta R_S^2} \tag{16}$$

Equation (16) shows that the synapse current is a function of the supply voltage V_{DD}, which we tune to adjust the weights. Figure 4 shows the neuron behavior when we vary V_{DD} from 550 mV to 750 mV. The higher supply results in a larger current, producing more spikes.

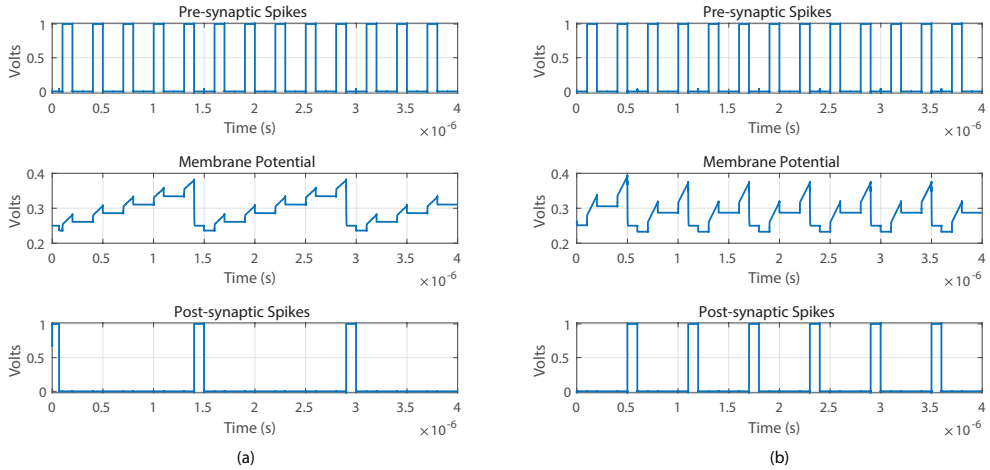

Figure 4. Neuron outputs with the same input spike pattern and synaptic weights, but with varied bias weights implemented as (**a**) $V_{DD} = 550$ mV and (**b**) $V_{DD} = 750$ mV.

The effect of a bias neuron with a weight of W_b on a synapse with weights W_s can be approximated as $I(W_b + W_s)$. The behavior of the analog implementation can be written as kIW where k represents the change in the synapse current due to adjusting V_{DD}. If $IW_b = kW$ then the behavior of the two implementations is identical.

5.2.2. Neuron Core Design

A schematic of the neuron core is shown in Figure 5. The threshold comparator is implemented with the StrongARM topology. We choose a clocked topology to reduce static power, especially when compared to inverter based threshold detectors. Instead of a fixed-period clock, we only clock the comparator after an input spike or after an output spike. We use a 4-input NOR gate to generate the comparator clock. This ensures that power consumption is minimized in a network trained for minimal spiking activity. The membrane capacitance is always reset to $V_{rest} = 250$ mV and the comparator has a fixed threshold of $V_{th,comp} = 350$ mV. We choose V_{rest} to give V_{mem} at least 100 mV of swing without driving the synapse current sources into the triode region, even when the synapse power supply is 0.5 V. Once the membrane potential crosses the preset threshold, the spike generation circuit is triggered. The spike is generated using a self-reset DQ fip-flop with current-starved inverter-based delay cells between Q and reset. The delay cells utilize parasitic capacitance to increase delay so as to decrease the number of stages needed for a certain spike width.

The membrane capacitor is a custom 50-fF finger capacitor which occupies only 27 μm^2. Because the membrane capacitance is only 50 fF, the neuron needs an extremely large resistor for a sufficiently low leakage current. Instead of using a polysilicon resistor which would occupy large area, we implement a CMOS pseudo resistor using a PMOS transistor which occupies only 0.7 μm $\times$ 0.5 μm and achieves approximately 400 MΩ (Figure 6). The pseudo-resistor is implemented as two PMOS transistors connected in a transdiode configuration. The simplest of pseudo-resistors have an asymmetric resistance-voltage characteristic, making them unusable for this neuron because the membrane potential can go both above and below V_{rest}, and must have the same up and down leakage current. To solve this, we use two psuedo-resistors in parallel with opposite connections polarities. This halves the effective resistance, but creates a symmetric resistance-voltage characteristic.

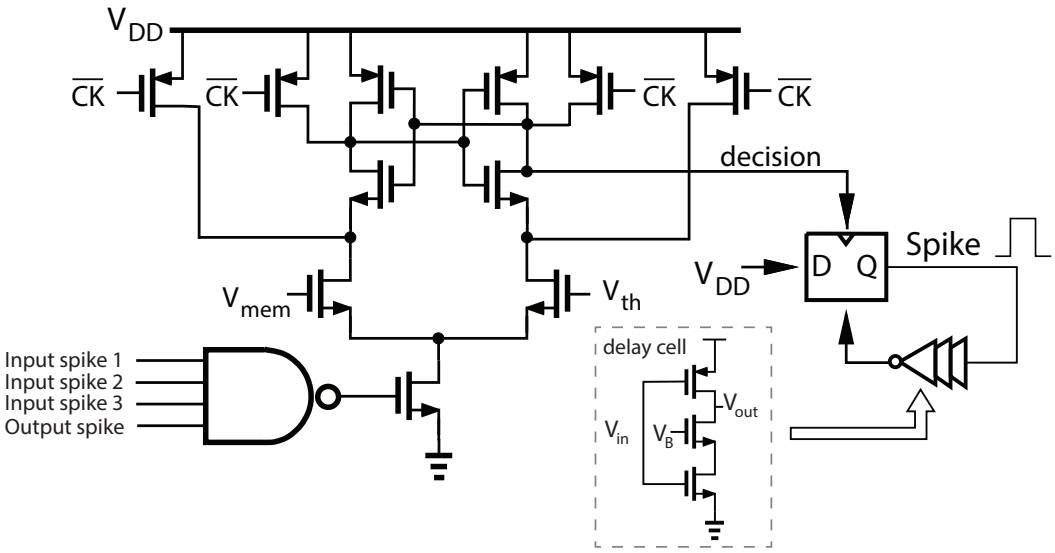

Figure 5. Schematic of the threshold comparator with dynamic clocking, and tunable spike generator circuit.

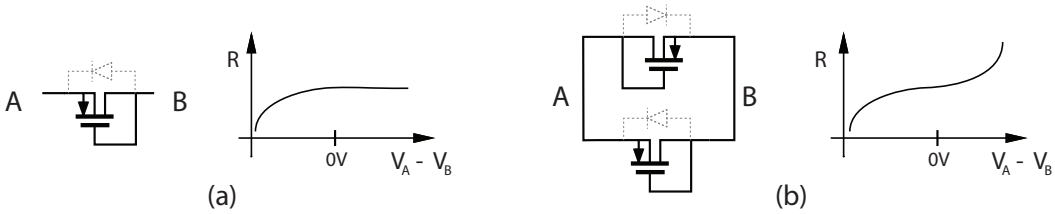

Figure 6. Schematic of (**a**) a one-directional pseudo-resistor and its asymmetric resistance characteristic and (**b**) the proposed pseudo-resistor showing symmetric resistance characteristics.

6. Results

Our long-term objective is to enable low-power analog learning behaviors in situ on physical analog chips. This requires both a viable mechanism for potential in situ learning that does not require large amounts of surface area for gradient calculations and a validated circuit design that can realistically implement that mechanism. We present neuromodulatory tuning as a possible mechanism for this objective, and here provide results showing its performance in simulated (digital) spiking neural networks (Section 6.1) and a full chip design for its eventual implementation on physical CMOS hardware (Section 6.2).

6.1. Neuromodulatory Tuning on Spiking Neural Networks

To validate the performance of neuromodulatory tuning in spiking neural networks (distinct from the traditional feed-foward networks shown in Section 4), we apply neuromodulatory tuning (NT) and traditional fine-tuning (TFT) to the SNN-VGG classification layers using the STL-10, Food-11, and BCCD datasets for comparison. We fix the batch size at 64 for all training, since our experiment with batch sizes (shown in Table 6) reveals that batch size does not impact the model performance dramatically. Both the Food-11 and BCCD datasets are singularly distinct from the ImageNet data [56] which was used to train VGG-19. VGG-19 therefore lacks output classes corresponding to labels from the Food-11 and BCCD datasets. To create the necessary output layer size, we added one extra fully connected layer at the end of each model. This extra layer functions as the output layer for corresponding classes in Food-11 and BCCD. Different from Food-11 and BCCD, STL-10 is a subset of ImageNet. Since VGG-19 is trained on ImageNet, VGG-19 contains classes that are contained within in STL-10 labels. Therefore, we do not add extra layers for the SNN

STL-10 experiments. All SNN models were trained on an AMD Ryzen Threadripper 1920X 12-Core Processor. Results are shown in Tables 7 and 8.

As expected, performance is poor when no tuning is applied. This is partially because SNN architectures, comprised of leaky integrate-and-fire neurons, differ drastically from traditional deep networks in both signal accumulation and signal propagation, resulting in almost 0% accuracy on all three transfer tasks. Tuning improves this accuracy, achieving up to 88% accuracy with TFT and 50% with NT on some tasks with certain learning rates. According to our results shown in Table 7, NT underperforms on the STL-10 dataset comparing to TFT, has equal performance to TFT on BCCD, and outperforms TFT on Food-11, which suggests that neuromodulatory tuning can positively impact learning behaviors on brain-like architectures.

Our performance comparison of the algorithms is influenced by differences between the three datasets. STL-10 is the subset of the dataset used to train VGG-19, so tasks in STL-10 is more native to the network. In contrast, Food11 and BCCD are foreign to the VGG-19 network, so those tasks will require VGG-19 to make adjustments in larger magnitudes or completely re-learn the task. Given that neuromodulatory tuning outperforms TFT on Food11, a foreign dataset, and that TFT requires changes of larger magnitudes, NT is superior for these cases. There are accuracies below random guessing, this might be caused by the low learning rate for NT and the absence of feed-forward to spiking network conversion algorithm for TFT.

Comparing two different types of NT, NT_1 performs better than NT_2 on STL-10 dataset, and has equal performance with NT_2 on Food-11 and BCCD dataset.

According to Table 8, TFT requires over 120 million parameters adjustment to achieve such performance, so the adjustments are impossible to implement on the physical chips. In contrast, NT method only requires 9000–20,000 adjustments, which is implementable on physical chips. Note, the parameter values for NT differ slightly in Table 8 from Table 5 due to the difference in implementing a spiking network versus a feed-forward network.

Table 6. Validation accuracy on the Food-11 dataset on SNN after 10 epochs, mean of 10 training runs using bath sizes (bs) = {16, 32, 64, 128}.

	acc (bs = 16)	**acc (bs = 32)**	**acc (bs = 64)**	**acc (bs = 128)**
NT_1 (lr = 0.1)	0.4568	0.4605	0.4570	0.4647
TFT (lr = 0.1)	0.1304	0.1243	0.1145	0.0770

Table 7. Validation accuracy on STL-10, Food-11, and the BCCD dataset in a spiking neural network (SNN) architecture. Models were trained for 50 epochs for STL-10, Food11, and the BCCD dataset, respectively. Average of five training runs. Best per-task performance of neuromodulatory tuning (NT_2) and traditional fine-tuning (TFT), respectively, is underlined. NT_2 refers to the modify existing bias implementation of NT and NT_1 refers to the additional bias implementation described in Section 3.2.

		lr 0.0001	**lr 0.001**	**lr 0.01**	**lr 0.1**
STL-10	no tuning	0.0007	0.0007	0.0007	0.0007
	TFT	0.8888	0.8014	0.2582	0.1274
	NT_2	0.0000	0.0000	0.3052	0.3062
	NT_1	0.0000	0.0009	0.5428	0.5731
	additive bias	0.0010	0.0008	0.0006	0.0025
Food-11	no tuning	0.0341	0.0341	0.0341	0.0341
	TFT	0.0147	0.0729	0.1017	0.1168
	NT_2	0.0063	0.3645	0.4537	0.4615
	NT_1	0.0020	0.3678	0.4564	0.4665
	additive bias	0.0840	0.1864	0.1404	0.1414

Table 7. *Cont.*

		lr 0.0001	lr 0.001	lr 0.01	lr 0.1
BCCD	no tuning	0.0005	0.0005	0.0005	0.0005
	TFT	0.1003	0.2508	0.2507	0.2508
	NT_2	0.2501	0.2509	0.1371	0.0680
	NT_1	0.2508	0.2509	0.2041	0.0591
	additive bias	0.1848	0.2137	0.2144	0.2505

Table 8. Validation accuracy and parameter on STL-10, Food-11, and the BCCD dataset in a spiking neural network (SNN) architecture. Models were trained for 50 epochs for STL-10, Food11, and the BCCD dataset, respectively. Accuracy from the learning rate with best average accuracy of five training runs. NT_2 refers to the modify existing bias implementation of NT and NT_1 refers to the additional bias implementation described in Section 3.2.

		Best Accuracy	Parameter Amount
STL-10	TFT	0.8888	123,642,856
	NT_2	0.3062	9192
	NT_1	0.5731	9192
	additive bias	0.0025	9192
Food-11	TFT	0.0356	123,653,867
	NT_2	0.4615	20,203
	NT_1	0.4665	20,203
	additive bias	0.1864	20,203
BCCD	TFT	0.2508	123,646,860
	NT_2	0.2509	13,196
	NT_1	0.2509	13,196
	additive bias	0.2505	13,196

6.2. Analog Neuromorphic Hardware Simulation

The goal of this work is to develop a low-power CMOS chip architecture that implements neuromodulatory tuning. In addition to presenting the neuromodulatory tuning algorithm and exploring its performance, we also present a complete neuron design to implements this algorithm on analog CMOS hardware.

Figure 7 shows the layout of the proposed neuron implementing NT fine tuning. The entire neuron, synapse and weight storage occupies only 598 um^2, with the neuron core (including membrane capacitor) occupying only 132 nm^2. We have validated the simulation results from Section 6.1 using post-layout simulations in Cadence Virtuoso to model an XOR task using spiking neurons. Two neurons were chosen to be the inputs to the XOR "gate" and another designated as the output. A train of 10 spikes to an input neuron constituted a "1". No input spikes constituted a "0". The spikes propagated through the network according to the trained weights. The output was "0" if less than three spikes were observed at the output, otherwise the output was a "1". The analog simulation showed 2 spikes at the output for a 0, and 4 for a 1.

The proposed neuron achieves performance competitive with the state-of-the-art in standalone neuron circuits (see Table 9). The total power for the neuron core varies with spike rate. Figure 7 shows the distribution of power for two spike rates and Figure 8 shows the energy/spike vs. spike rate.

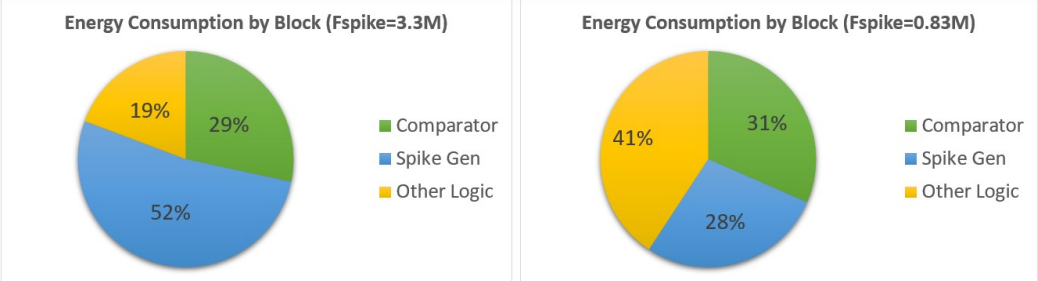

Figure 7. The distribution of power within the neuron core.

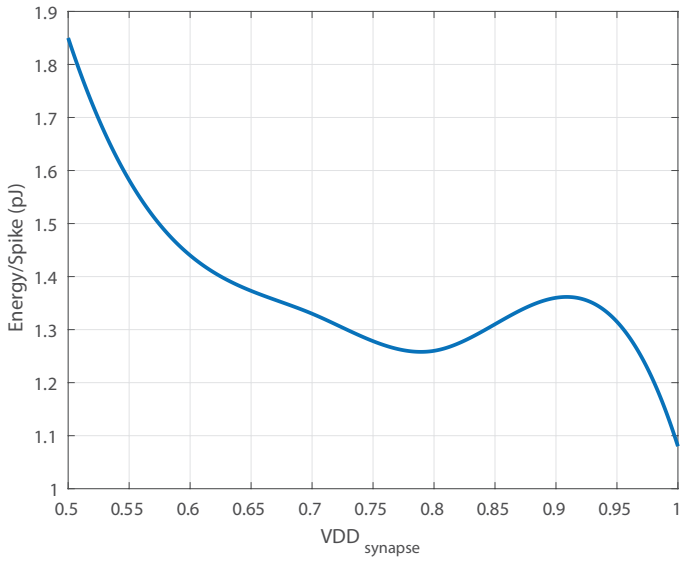

Figure 8. The energy/spike decreases as V_{DD} increases. This is because a higher V_{DD} yields a higher synapse current and therefore more output spikes for the same number of input spikes.

Table 9. Comparison of our proposed neuron implementing neuromodulatory tuning with the state of the art in standalone neurons. * Total area includes neuron core, synapse, and weight storage.

	This Work	Joubert et al., 2012	Cruz-Albrecht et al., 2012	Rangan et al., 2010	Jayawan 2008
Process (nm)	28	65	90	90	350
Area um^2	598 (Total *) 132 (Core)	538	442	897	2800
Max f_{spike} (Hz)	3.3M	1.9M	100	7k	1M
Energy/spike (pJ)	1.08	41	0.4	1	9

7. Conclusions

Low-power analog machine learning has the potential to revolutionize multiple disciplines, but only if novel and physically-implementable learning algorithms are developed that enable in situ behavior modification on physical analog hardware. This paper presents a novel task transfer algorithm, termed neuromodulatory tuning, for machine learning based on biologically-inspired principles. On image recognition tasks, neuromodulatory tuning performs on test cases as well as traditional fine-tuning methods while requiring four orders of magnitude fewer active training parameters (although the total number of weights is comparable between methods). We verify this result using both deep forward networks and spiking neural network architectures. We also present a circuit design for a

neuron that immplements neuromodulatory tuning, a potential layout for the use of such neurons on an analog chip, and a post-layout verification of its capabilities.

Neuromodulatory tuning has the advantage of being well-suited for implementation on neuromorphic hardware, enabling circuit implementations that support life-long learning for applications that require energy-efficient adaptation to constantly changing conditions, such as robotics, unmanned air vehicle guidance, and prosthetic limb controllers. Future research in this area should focus on probing the performance of NT in domains beyond image recognition; exploring the possibility of paired bias links in which multiple neurons connect to a single power domain region; and designing improved SNN update algorithms with stronger convergence properties.

Author Contributions: Neural network studies and writing, H.Y. and K.R.; circuit design, simulation, and writing, T.B.; problem conception, supervision, and writing—review and editing, N.F., S.-h.W.C., J.Y. and K.F.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding

Data Availability Statement: The training data sets used in this study are widely available online.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Pfeiffer, M.; Pfeil, T. Deep learning with spiking neurons: opportunities and challenges. *Front. Neurosci.* **2018**, *12* , 774.
2. Voelker, A.R.; Rasmussen, D.; Eliasmith, C. A spike in performance: Training hybrid-spiking neural networks with quantized activation functions. *arXiv* **2020**, arXiv:2002.03553.
3. Ghosh-Dastidar, S.; Adeli, H. Spiking neural networks. *Int. J. Neural Syst.* **2009**, *19*, 295–308.
4. Tavanaei, A.; Ghodrati, M.; Kheradpisheh, S.R.; Masquelier, T.; Maida, A. Deep learning in spiking neural networks. *Neural Networks* **2019**, *111*, 47–63.
5. Ponulak, F.; Kasinski, A. Introduction to spiking neural networks: Information processing, learning and applications. *Acta Neurobiol. Exp.* **2011**, *71*, 409–433.
6. Daumé III, H. Frustratingly Easy Domain Adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*; Association for Computational Linguistics: Prague, Czech Republic, 2007; pp. 256–263.
7. Sun, B.; Feng, J.; Saenko, K. Return of Frustratingly Easy Domain Adaptation. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*; AAAI Press: Palo Alto, CA, USA, 2016; pp. 2058–2065.
8. Blitzer, J.; McDonald, R.; Pereira, F. Domain Adaptation with Structural Correspondence Learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*; Association for Computational Linguistics: Sydney, Australia, 2006; pp. 120–128.
9. Motiian, S.; Jones, Q.; Iranmanesh, S.; Doretto, G. Few-Shot Adversarial Domain Adaptation. In *Proceedings of the Advances in Neural Information Processing Systems*; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: New York, NY, USA, 2017; Volume 30.
10. Liu, H.; Wang, J.; Long, M. Cycle Self-Training for Domain Adaptation. In *Proceedings of the Advances in Neural Information Processing Systems*; Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W., Eds.; Curran Associates, Inc.: New York, NY, USA, 2021; Volume 34; pp. 22968–22981.
11. Stojanov, P.; Li, Z.; Gong, M.; Cai, R.; Carbonell, J.; Zhang, K. Domain Adaptation with Invariant Representation Learning: What Transformations to Learn? In *Proceedings of the Advances in Neural Information Processing Systems*; Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W., Eds.; Curran Associates, Inc.: New York, NY, USA, 2021; Volume 34, pp. 24791–24803.
12. Lawrence, N.D.; Platt, J.C. Learning to learn with the informative vector machine. In Proceedings of the Ttwenty-First International Conference on Machine Learning, Banff, AB, Canada, 4–8 July 2004; p. 65.
13. Raina, R.; Battle, A.; Lee, H.; Packer, B.; Ng, A.Y. Self-taught learning: transfer learning from unlabeled data. In Proceedings of the 24th International Conference on Machine Learning, New York, NY, USA, 20–24 June 2007; pp. 759–766.
14. Argyriou, A.; Evgeniou, T.; Pontil, M. Multi-task feature learning. *Adv. Neural Inf. Process. Syst.* **2006**, *19;* pp. 1–2.
15. Lee, S.I.; Chatalbashev, V.; Vickrey, D.; Koller, D. Learning a meta-level prior for feature relevance from multiple related tasks. In Proceedings of the 24th International Conference on Machine Learning, Corvalis, OR, USA, 20–24 June 2007; pp. 489–496.
16. Li, D.; Zhang, H. Improved Regularization and Robustness for Fine-tuning in Neural Networks. In *Proceedings of the Advances in Neural Information Processing Systems*; Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W., Eds.; Curran Associates, Inc.: New York, NY, USA, 2021; Volume 34, pp. 27249–27262.
17. Dong, X.; Luu, A.T.; Lin, M.; Yan, S.; Zhang, H. How Should Pre-Trained Language Models Be Fine-Tuned Towards Adversarial Robustness? In *Proceedings of the Advances in Neural Information Processing Systems*; Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W., Eds.; Curran Associates, Inc.: New York, NY, USA, 2021; Volume 34, pp. 4356–4369.

18. Zhang, Y.; Hooi, B.; Hu, D.; Liang, J.; Feng, J. Unleashing the Power of Contrastive Self-Supervised Visual Models via Contrast-Regularized Fine-Tuning. In *Proceedings of the Advances in Neural Information Processing Systems*; Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W., Eds.; Curran Associates, Inc.: New York, NY, USA: 2021; Volume 34, pp. 29848–29860.
19. Vinyals, O.; Blundell, C.; Lillicrap, T.; Wierstra, D. Matching networks for one shot learning. *Adv. Neural Inf. Process. Syst.* **2016**, *29*, 1–2.
20. Snell, J.; Swersky, K.; Zemel, R. Prototypical networks for few-shot learning. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 1–2.
21. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models are Few-Shot Learners. In *Proceedings of the Advances in Neural Information Processing Systems*; Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H., Eds.; Curran Associates, Inc.: New York, NY, USA, 2020; Volume 33, pp. 1877–1901.
22. Yue, Z.; Zhang, H.; Sun, Q.; Hua, X.S. Interventional Few-Shot Learning. In *Proceedings of the Advances in Neural Information Processing Systems*; Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H., Eds.; Curran Associates, Inc.: New York, NY, USA, 2020; Volume 33, pp. 2734–2746.
23. Brown, Z.; Robinson, N.; Wingate, D.; Fulda, N. Towards neural programming interfaces. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 17416–17428.
24. Agostinelli, F.; Hoffman, M.; Sadowski, P.; Baldi, P. Learning activation functions to improve deep neural networks. *arXiv* **2014**, arXiv:1412.6830.
25. Zaken, E.B.; Ravfogel, S.; Goldberg, Y. BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models, 2021. *arXiv* **2021**, arXiv:2106.10199. https://doi.org/10.48550/ARXIV.2106.10199.
26. Kendall, J.; Pantone, R.; Manickavasagam, K.; Bengio, Y.; Scellier, B. Training end-to-end analog neural networks with equilibrium propagation. *arXiv* **2020**, arXiv:2006.01981.
27. Furber, S.B.; Galluppi, F.; Temple, S.; Plana, L.A. The SpiNNaker Project. *Proc. IEEE* **2014**, *102*, 652–665. https://doi.org/10.1109/JPROC.2014.2304638.
28. Voutsas, K.; Adamy, J. A Biologically Inspired Spiking Neural Network for Sound Source Lateralization. *IEEE Trans. Neural Netw.* **2007**, *18*, 1785–1799. https://doi.org/10.1109/TNN.2007.899623.
29. Yang, Z.; Han, Z.; Huang, Y.; Ye, T.T. 55nm CMOS Analog Circuit Implementation of LIF and STDP Functions for Low-Power SNNs. In Proceedings of the 2021 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED), Virtual, 26–28 July 2021; pp. 1–6. https://doi.org/10.1109/ISLPED52811.2021.9502497.
30. Rueckauer, B.; Liu, S.C. Conversion of analog to spiking neural networks using sparse temporal coding. In Proceedings of the 2018 IEEE International Symposium on Circuits and Systems (ISCAS), Florence, Italy, 27–30 May 2018; pp. 1–5. https://doi.org/10.1109/ISCAS.2018.8351295.
31. Zhang, W.; Li, P. Spike-Train Level Backpropagation for Training Deep Recurrent Spiking Neural Networks. In *Proceedings of the Advances in Neural Information Processing Systems*; Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R., Eds.; Curran Associates, Inc.: New York, NY, USA, 2019; Volume 32.
32. Miquel, J.R.; Tolu, S.; Schöller, F.E.T.; Galeazzi, R. RetinaNet Object Detector Based on Analog-to-Spiking Neural Network Conversion. In Proceedings of the 2021 8th International Conference on Soft Computing Machine Intelligence (ISCMI), Cairo, Egypt, 26–27 November 2021; pp. 201–205. https://doi.org/10.1109/ISCMI53840.2021.9654818.
33. Ding, J.; Yu, Z.; Tian, Y.; Huang, T. Optimal ANN-SNN Conversion for Fast and Accurate Inference in Deep Spiking Neural Networks. *arXiv* **2021**, arXiv:2105.11654.
34. Li, Y.; Deng, S.; Dong, X.; Gu, S. Converting Artificial Neural Networks to Spiking Neural Networks via Parameter Calibration, 2022. *arXiv* **2022**, arXiv:2205.10121. https://doi.org/10.48550/ARXIV.2205.10121.
35. Indiveri, G.; Chicca, E.; Douglas, R. A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *IEEE Trans. Neural Netw.* **2006**, *17*, 211–221. https://doi.org/10.1109/TNN.2005.860850.
36. Han, B.; Srinivasan, G.; Roy, K. RMP-SNN: Residual Membrane Potential Neuron for Enabling Deeper High-Accuracy and Low-Latency Spiking Neural Network. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 13555–13564. https://doi.org/10.1109/CVPR42600.2020.01357.
37. Li, J.; Zhao, C.; Hamedani, K.; Yi, Y. Analog hardware implementation of spike-based delayed feedback reservoir computing system. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017; pp. 3439–3446. https://doi.org/10.1109/IJCNN.2017.7966288.
38. Nitundil, S.; Susi, G.; Maestú, F. Design of an Analog Multi-Neuronal Spike-sequence Detector (MNSD) based on a 180nm CMOS Leaky Integrate amp; Fire with Latency Neuron. In Proceedings of the 2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), Bhilai, India, 19–20 February 2021; pp. 1–6. https://doi.org/10.1109/ICAECT49130.2021.9392419.
39. Sun, Q.; Schwartz, F.; Michel, J.; Herve, Y.; Dal Molin, R. Implementation Study of an Analog Spiking Neural Network for Assisting Cardiac Delay Prediction in a Cardiac Resynchronization Therapy Device. *IEEE Trans. Neural Netw.* **2011**, *22*, 858–869. https://doi.org/10.1109/TNN.2011.2125986.
40. Mostafa, H. Supervised Learning Based on Temporal Coding in Spiking Neural Networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 3227–3235. https://doi.org/10.1109/TNNLS.2017.2726060.
41. Hsieh, H.Y.; Tang, K.T. VLSI Implementation of a Bio-Inspired Olfactory Spiking Neural Network. *IEEE Trans. Neural Netw. Learn. Syst.* **2012**, *23*, 1065–1073. https://doi.org/10.1109/TNNLS.2012.2195329.

42. Kim, M.H.; Hwang, S.; Bang, S.; Kim, T.H.; Lee, D.K.; Ansari, M.H.R.; Cho, S.; Park, B.G. A More Hardware-Oriented Spiking Neural Network Based on Leading Memory Technology and Its Application with Reinforcement Learning. *IEEE Trans. Electron. Devices* **2021**, *68*, 4411–4417. https://doi.org/10.1109/TED.2021.3099769.
43. Cincon, V.; Vatajelu, E.I.; Anghel, L.; Galy, P. From 1.8 V to 0.19 V voltage bias on analog spiking neuron in 28 nm UTBB FD-SOI technology. In Proceedings of the 2020 Joint International EUROSOI Workshop and International Conference on Ultimate Integration on Silicon (EUROSOI-ULIS), Caen, France, 1–30 September2020; pp. 1–4. https://doi.org/10.1109/EUROSOI-ULIS49407.2020.9365302.
44. Danneville, F.; Sourikopoulos, I.; Hedayat, S.; Loyez, C.; Hoël, V.; Cappy, A. Ultra low power analog design and technology for artificial neurons. In Proceedings of the 2017 IEEE Bipolar/BiCMOS Circuits and Technology Meeting (BCTM), Miami, FL, USA, 19–21 October 2017; pp. 1–8. https://doi.org/10.1109/BCTM.2017.8112899.
45. Satyaraj, I.; Kailath, B.J. A simple PSTDP circuit for Analog Implementation of Spiking Neural Networks. In Proceedings of the 2020 IEEE 4th Conference on Information Communication Technology (CICT), Chennai, India, 3–5 December 2020; pp. 1–4. https://doi.org/10.1109/CICT51604.2020.9312100.
46. Kim, D.; She, X.; Rahman, N.M.; Chekuri, V.C.K.; Mukhopadhyay, S. Processing-In-Memory-Based On-Chip Learning With Spike-Time-Dependent Plasticity in 65-nm CMOS. *IEEE Solid-State Circuits Lett.* **2020**, *3*, 278–281. https://doi.org/10.1109/LSSC.2020.3013448.
47. Azghadi, M.R.; Al-Sarawi, S.; Iannella, N.; Abbott, D. Efficient design of triplet based Spike-Timing Dependent Plasticity. In Proceedings of the the 2012 International Joint Conference on Neural Networks (IJCNN), Brisbane, Australia, 10–15 June 2012; pp. 1–7. https://doi.org/10.1109/IJCNN.2012.6252820.
48. Clements, J. Transmitter timecourse in the synaptic cleft: its role in central synaptic function. *Trends Neurosci.* **1996**, *19*, 163–171. https://doi.org/https://doi.org/10.1016/S0166-2236(96)10024-2.
49. Agnati, L.F.; Zoli, M.; Strömberg, I.; Fuxe, K. Intercellular communication in the brain: Wiring versus volume transmission. *Neuroscience* **1995**, *69*, 711–726.
50. Yorgason, J.T.; Zeppenfeld, D.M.; Williams, J.T. Cholinergic Interneurons Underlie Spontaneous Dopamine Release in Nucleus Accumbens. *J. Neurosci.* **2017**, *37*, 2086–2096. https://doi.org/10.1523/JNEUROSCI.3064-16.2017.
51. Beaulieu, J..; Gainetdinov, R.R. The physiology, signaling, and pharmacology of dopamine receptors. *Pharmacol. Rev.* **2011**, *63*, 182–217.
52. Depue, R.A.; Collins, P.F. Neurobiology of the structure of personality: Dopamine, facilitation of incentive motivation, and extraversion. *Behav. Brain Sci.* **1999**, *22*, 491–517. Cited By :1391.
53. Frank, M.J. Dynamic dopamine modulation in the basal ganglia: A neurocomputational account of cognitive deficits in medicated and nonmedicated Parkinsonism. *J. Cogn. Neurosci.* **2005**, *17*, 51–72.
54. Stoof, J.C.; Kebabian, J.W. Two dopamine receptors: Biochemistry, physiology and pharmacology. *Life Sci.* **1984**, *35*, 2281–2296.
55. Reiner, A.; Levitz, J. Glutamatergic Signaling in the Central Nervous System: Ionotropic and Metabotropic Receptors in Concert. *Neuron* **2018**, *98*, 1080–1098. https://doi.org/10.1016/j.neuron.2018.05.018.
56. ImageNet Dataset. Available online: https://www.image-net.org/ (accessed on 2 May 2022).
57. Coates, A.; Ng, A.; Lee, H. An analysis of single-layer networks in unsupervised feature learning. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, Fort Lauderdale, FL, USA, 11–13 April 2011; pp. 215–223.
58. Food Image Dataset. Available online: https://www.epfl.ch/labs/mmspg/downloads/food-image-datasets/ (accessed on 2 May 2022).
59. Blood Cell Images. 2018. Available online: https://www.kaggle.com/datasets/paultimothymooney/blood-cells (accessed on 2 May 2022).
60. Liu, X.; Chi, M.; Zhang, Y.; Qin, Y. Classifying High Resolution Remote Sensing Images by Fine-Tuned VGG Deep Networks. In Proceedings of the IGARSS 2018—2018 IEEE International Geoscience and Remote Sensing Symposium, Valencia, Spain, 22–27 July 2018; pp. 7137–7140. https://doi.org/10.1109/IGARSS.2018.8518078.
61. Nagaraju, Y.; Venkatesh.; Swetha, S.; Stalin, S. Apple and Grape Leaf Diseases Classification using Transfer Learning via Fine-tuned Classifier. In Proceedings of the 2020 IEEE International Conference on Machine Learning and Applied Network Technologies (ICMLANT), Hyderabad, India, 20–21 December 2020; pp. 1–6. https://doi.org/10.1109/ICMLANT50963.2020.9355991.
62. Kandel, I.; Castelli, M. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT Express* **2020**, *6*, 312–315. https://doi.org/https://doi.org/10.1016/j.icte.2020.04.010.
63. Aamir, S.A.A.; Stradmann, Y.; Müller, P.; Pehle, C.; Hartel, A.; Grübl, A.; Schemmel, J.; Meier, K. *An Accelerated LIF Neuronal Network Array for a Large-Scale Mixed-Signal Neuromorphic Architecture*; IEEE Transactions on Circuits and Systems I: Regular Papers; IEEE: Piscataway, NJ, USA, 2018.