Leveraging Biological Mechanisms in Machine Learning

Kyle J. Rogers
Brigham Young University

Leveraging Biological Mechanisms in Machine Learning

Kyle J. Rogers

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science

Brigham Young University

ABSTRACT

Leveraging Biological Mechanisms in Machine Learning

Kyle J. Rogers
Department of Computer Science, BYU
Master of Science

This thesis integrates biologically-inspired mechanisms into machine learning to develop novel tuning algorithms, gradient abstractions for depth-wise parallelism, and an original bias neuron design. We introduce neuromodulatory tuning, which uses neurotransmitter-inspired bias adjustments to enhance transfer learning in spiking and non-spiking neural networks, significantly reducing parameter usage while maintaining performance. Additionally, we propose a novel approach that decouples the backward pass of backpropagation using layer abstractions, inspired by feedback loops in biological systems, enabling depth-wise training parallelization.

We further extend neuromodulatory tuning by designing spiking bias neurons that mimic dopamine neuron mechanisms, leading to the development of volumetric tuning. This method enhances the fine-tuning of a small spiking neural network for EEG emotion classification, outperforming previous bias tuning methods. Overall, this thesis demonstrates the potential of leveraging neuroscience discoveries to improve machine learning.

Keywords: spiking neural networks, neuromorphic computing, dopamine-inspired learning structures, layer abstractions

ACKNOWLEDGMENTS

I would like to thank my advisor, Nancy Fulda, for her consistent support and guidance. Additionally, I would like to thank members of our interdisciplinary Neuromorphic Computing group who contributed significantly to the first two chapters of this thesis. Lastly, I would like to express tremendous gratitude to my wife, Talbot, who has supported me during my academic career.

Table of Contents

Chapter 1

Introduction

On the quest to develop human-like artificial intelligence, researchers often draw inspiration from observations of biological intelligence. Perhaps the first example of this pattern as it relates to machine learning is the Perceptron model published by Frank Rosenblatt[2]. The Perception is a computational model of a biological neuron that still has relevance in modern deep neural networks today. Modern deep learning algorithms, while originally inspired by biology, learn very differently than our biological brains. For example, backpropagation is not biologically plausible. This does not mean that backpropagation should be disregarded, but rather that biological structures and learning methods can be considered to address some of the shortcomings of backpropagation such as its struggle to learn from small datasets and the computational expense of computing the gradient. Our biological brain, however, does not have these shortcomings [1]. Perhaps by mimicking different biological structures and processes to some level of abstraction we can resolve problems in machine learning that we have yet to solve using conventional methods.

In this thesis, we follow a well established pattern of applying recent findings in neuroscience and biology to develop biologically-inspired learning algorithms for both spiking and non-spiking neural networks. First, in Chapter 2, we examine the the application of neurotransmitter mechanisms in biological brains to solve a transfer learning objective. We present a method, called neuromodulatory tuning, which manipulates the firing sensitivity of each neuron using a supplemental bias input. We evaluate our method on both feed forward and spiking networks to fine tune a pretrained general image classification model on three separate image classification tasks. We compare our method of tuning just bias vectors, with the traditional form of fine tuning which involves adjusting the weight matrices and bias vectors of a pretrained network. We find that neuromodulatory tuning approaches the performance of traditional fine tuning, but uses orders of magnitude less parameters during fine tuning.

Next, we present an original, biologically-inspired learning approach in Chapter 3. Specifically, we decouple the backward pass of backpropagation using layer abstractions. We motivate this approach from observations in neuroscience where distant, upstream layers of neurons in biological systems are connected and tuned via feedback loops. We develop a mathematically accurate abstraction for piece-wise linear networks and a more general biological abstraction allowing the gradient the be passed around layers. This construction yields theoretical depth-wise training parallelization.

Finally, in Chapter 4, we extend our previous work with neuromodulatory tuning to design biologically-inspired spiking bias neurons. We mimic the stimulus and output mechanisms of dopamine neurons in an effort to create more capable bias neurons. We apply our previous neuromodulatory tuning method on a small spiking neural network (SNN) to solve a complex EEG emotion classification task where we compare traditional biases with our spiking biases. We conduct additional experiments and ultimately present a variant of the neuromodulatory tuning method, named volumetric tuning, which performs significantly better at fine tuning the pretrained emotion classification SNN.

This collection of work has demonstrated the power in leveraging discoveries in neuroscience to solve problems in the broader machine learning community; namely, parameter-efficient fine tuning, and depth-wise parallelization. Not only does this research address these issues directly, but it also provides fascinating avenues for future work. Our work with abstracted gradients introduces a novel learning paradigm where we show that the sequentiality of backpropagation is no longer strictly required. This discovery raises questions regarding what other mechanisms of backpropagation are necessary.

Our work developing neuromodulatory tuning not only reduces the parameter count needed for fine tuning deep learning models, but we also show how such an algorithm could effectively operate and be implemented on CMOS hardware. This allows manufactured chips programmed with pretrained, offline weights to be effectively tuned on new data. Tunable chips such as these have immediate applications for autonomous, power-constrained devices. We extend this work, to develop spiking bias neurons and volumetric tuning which still feature parameter-efficient tuning, but do so with increased performance on a challenging BCI task. This method has direct applications on physical chips, specifically in the domain of decoding EEG signals. One can imagine the eventual development of a chip pretrained on EEG data that is calibrated or fine tuned for the specific person using a emotion classification device. Such advances and aspirations come as a result of adapting observed biological processes to existing machine learning practices, and provide a strong argument for continuing to apply biological inspiration to innovate in the machine learning community.

Chapter 2

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

Taylor Barton, Hao Yu and I are primary authors of this paper. We have agreed that each of us will include this paper in our theses.

The following list summarizes our respective contributions to the research and writing contained in this this chapter. This list also appears in Hao's thesis.

- Kyle developed the neuromodulatory tuning method in feed-forward fully-connected networks in the domain of image recognition.
- Kyle made the decision to utilize VGG-19 and fine tune it on three tasks.
- Hao created a spiking classifier for the VGG-19 model.
- Hao developed the method for a spiking version of the VGG-19 classifier.
- Taylor outlined the mechanisms by which neuromodulatory tuning can feasibly be implemented on CMOS hardware.
- Taylor presented an analog spiking neuron with neuromodulatory tuning capabilities.
- We worked together to create experiments and gather experimental results.
- We worked together to write and revise the text to create the final draft.

I hereby confirm that the use of this article is compliant with all publishing agreements.

Barton, T.; Yu, H.; Rogers, K.; Fulda, N.; Chiang, S.-h.W.; Yorgason, J.; Warnick, K.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

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

check for updates

Citation: Barton, T.; Yu, H.; Rogers, K.; Fulda, N.; Chiang, S.-h.W.; Yorgason, J.; Warnick, K.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.

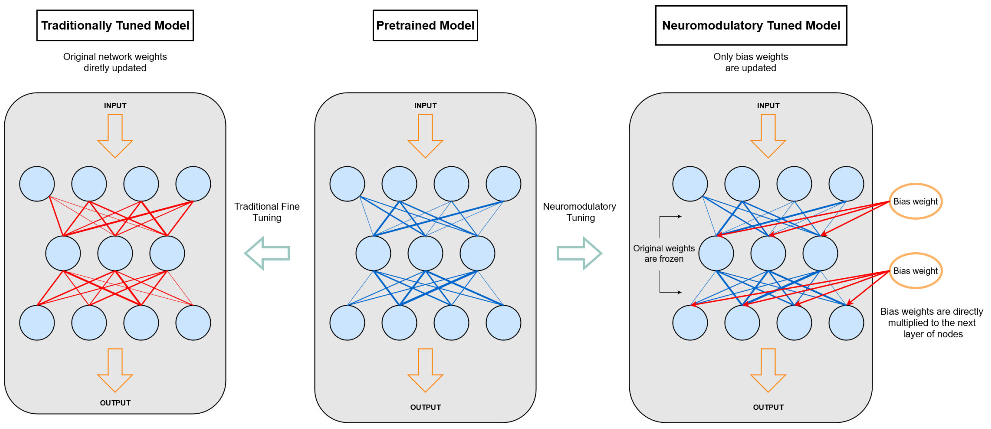
Traditionally Tuned Model
Original network weights diretly updated
INPUT
OUTPUT
Traditional Fine Tuning
Pretrained Model
INPUT
OUTPUT
Neuromodulatory Tuning
Neuromodulatory Tuned Model
Only bias weights are updated
INPUT
Original weights are frozen
Bias weight
Bias weight
Bias weights are directly multiplied to the next layer of nodes
OUTPUT

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.

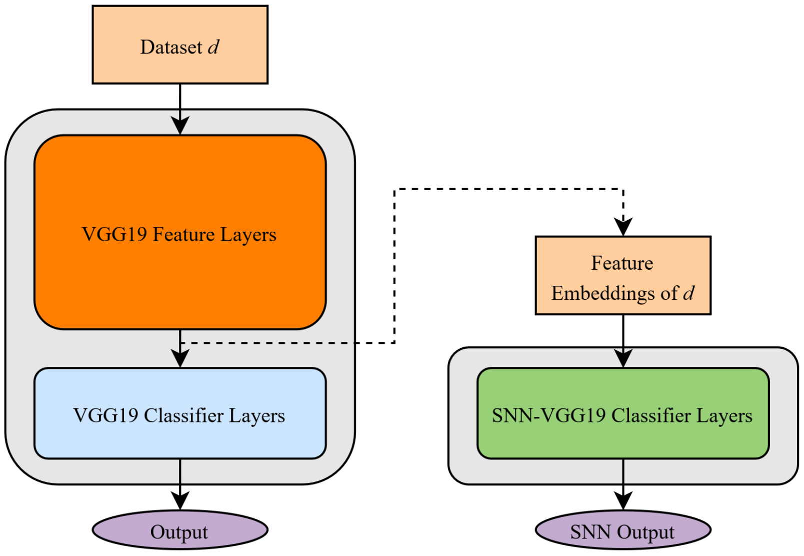

Figure 2. Representation of the Spiking Neural Network (SNN) experimental setup. In these experiments we construct a SNN that mimics the function and purpose of the traditional pretrained VGG-19 classifier layers. To accomplish this, we pass data from a dataset d, where d = {STL-10, Food-11, BCCD}, through 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.

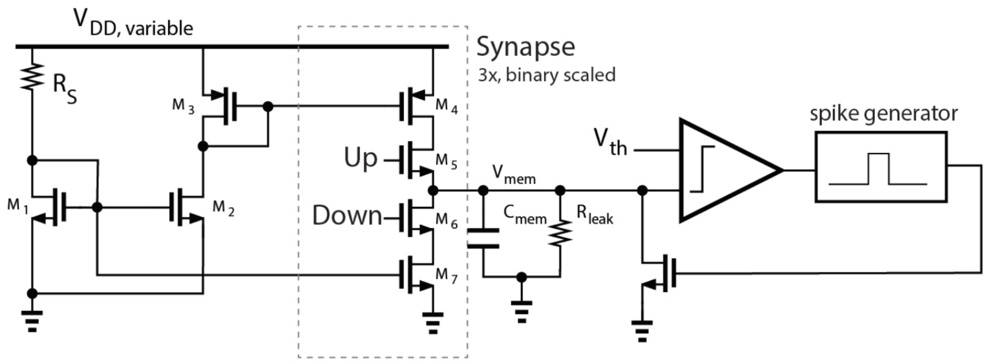

Figure 3. Schematic diagram of the proposed leaky integrate-and-fire neuron with NT ($V_{DD,variable}$) capabilities. The Up and Down signals are generated from the input spike and weight signals.

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.

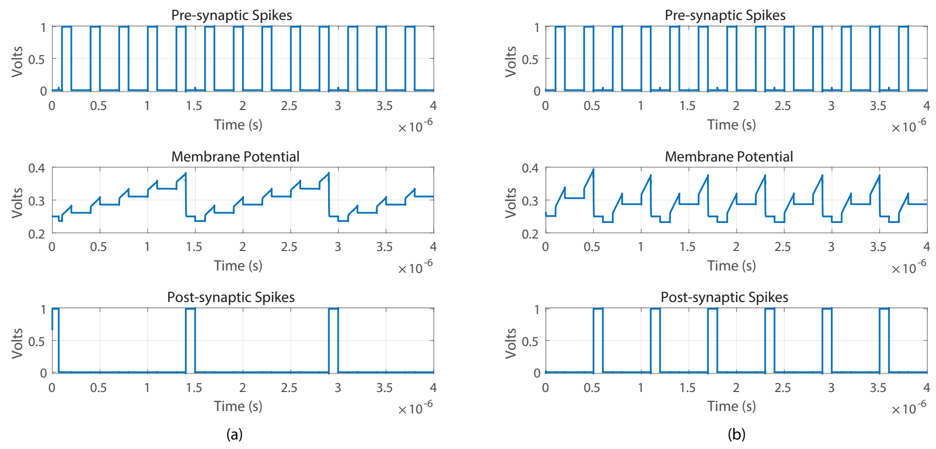

Figure 4. Neuron outputs with the same input spike pattern and synaptic weights, but with varied bias weights implemented as (**a**) $V_{DD} = 550$ mV and (**b**) $V_{DD} = 750$ mV.

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 $\mu m \times 0.5$ 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.

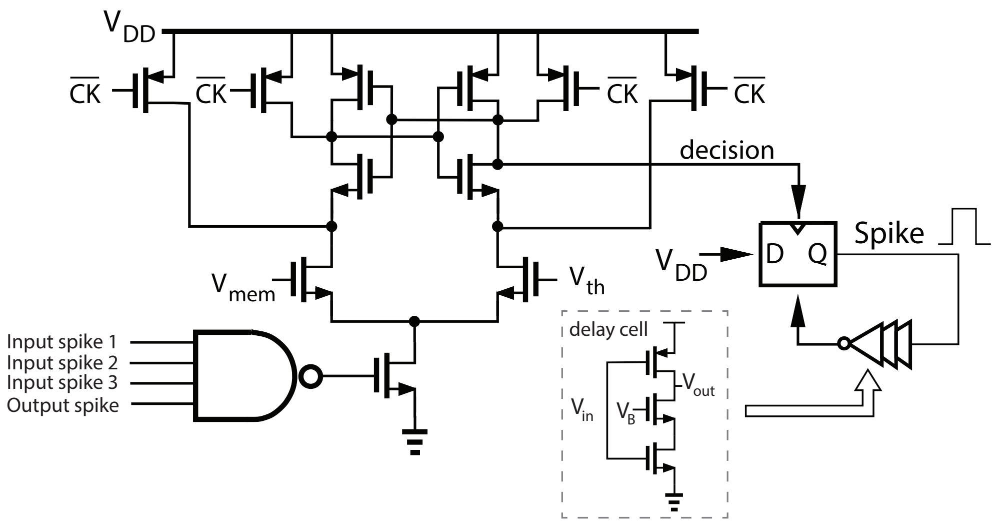

Figure 5. Schematic of the threshold comparator with dynamic clocking, and tunable spike generator circuit.

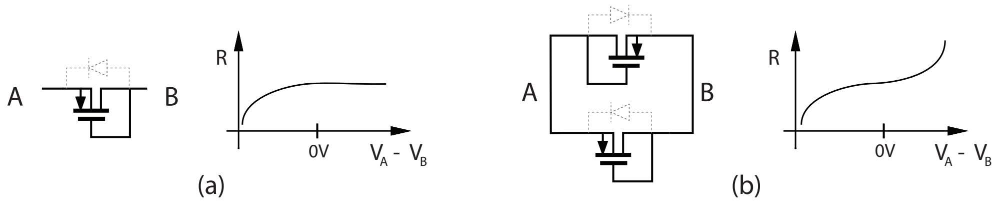

Figure 6. Schematic of (**a**) a one-directional pseudo-resistor and its asymmetric resistance characteristic and (**b**) the proposed pseudo-resistor showing symmetric resistance characteristics.

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.

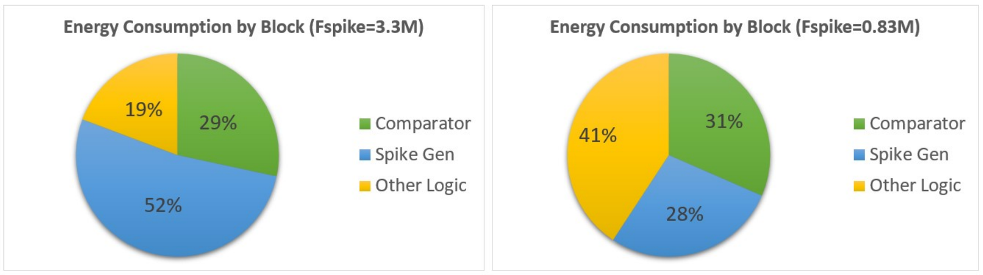

Figure 7. The distribution of power within the neuron core.

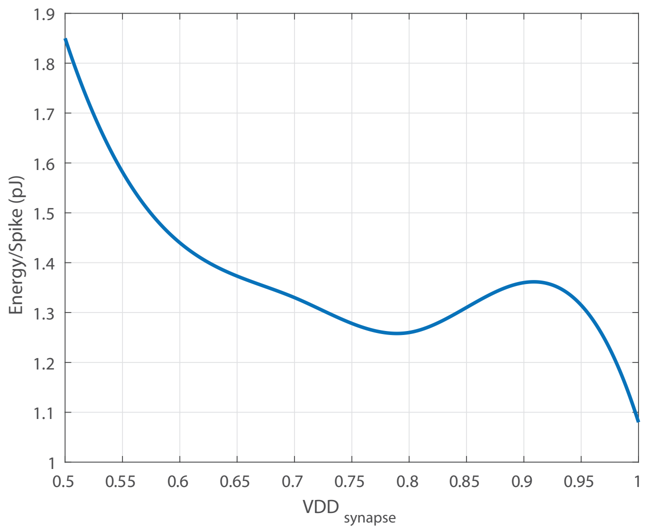

Figure 8. The energy/spike decreases as V_{DD} increases. This is because a higher V_{DD} yields a higher synapse current and therefore more output spikes for the same number of input spikes.

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 µm^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. [CrossRef] [PubMed]
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. [CrossRef] [PubMed]
4. Tavanaei, A.; Ghodrati, M.; Kheradpisheh, S.R.; Masquelier, T.; Maida, A. Deep learning in spiking neural networks. *Neural Networks* **2019**, *111*, 47–63. [CrossRef] [PubMed]
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.
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. [CrossRef]
28. Voutsas, K.; Adamy, J. A Biologically Inspired Spiking Neural Network for Sound Source Lateralization. *IEEE Trans. Neural Netw.* **2007**, *18*, 1785–1799. [CrossRef] [PubMed]
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. [CrossRef]
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. [CrossRef]
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. [CrossRef]
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.
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. [CrossRef]
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. [CrossRef]
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. [CrossRef]
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. [CrossRef]
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. [CrossRef]
40. Mostafa, H. Supervised Learning Based on Temporal Coding in Spiking Neural Networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 3227–3235. [CrossRef]
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. [CrossRef]
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. [CrossRef]

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 September 2020; pp. 1–4. [CrossRef]
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. [CrossRef]
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. [CrossRef]
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. [CrossRef]
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. [CrossRef]
48. Clements, J. Transmitter timecourse in the synaptic cleft: its role in central synaptic function. *Trends Neurosci.* **1996**, *19*, 163–171. [CrossRef]
49. Agnati, L.F.; Zoli, M.; Strömberg, I.; Fuxe, K. Intercellular communication in the brain: Wiring versus volume transmission. *Neuroscience* **1995**, *69*, 711–726. [CrossRef]
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. [CrossRef] [PubMed]
51. Beaulieu, J..; Gainetdinov, R.R. The physiology, signaling, and pharmacology of dopamine receptors. *Pharmacol. Rev.* **2011**, *63*, 182–217. [CrossRef] [PubMed]
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. [CrossRef]
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. [CrossRef]
54. Stoof, J.C.; Kebabian, J.W. Two dopamine receptors: Biochemistry, physiology and pharmacology. *Life Sci.* **1984**, *35*, 2281–2296. [CrossRef]
55. Reiner, A.; Levitz, J. Glutamatergic Signaling in the Central Nervous System: Ionotropic and Metabotropic Receptors in Concert. *Neuron* **2018**, *98*, 1080–1098. [CrossRef]
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. [CrossRef]
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. [CrossRef]
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. [CrossRef]
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.

Chapter 3

Decoupling the Backward Pass Using Abstracted Gradients

Hao Yu and I are primary authors of this paper. We have agreed that each of us will include this paper in our theses.

The following list summarizes our respective contributions to the research and writing contained in this this chapter. This list also appears in Hao's thesis.

- Hao presented the original concept drawn from biology.
- Kyle developed the original mathematical layer abstraction for fully connected linear networks.
- Hao developed the biological abstraction.
- We worked together to create experiments and gather experimental results.
- We worked together to explore theoretical parallelization and network modularization as applications of our method.
- Kyle led the writing.
- We worked together to revise the text to create the final draft.

I hereby confirm that the use of this article is compliant with all publishing agreements.

Rogers K., Yu H., Cho S., Fulda N., Yorgason J. and Jarvis T. (2024). Decoupling the Backward Pass Using Abstracted Gradients. In Proceedings of the 16th International Conference on Agents and Artificial Intelligence - Volume 3: ICAART; ISBN 978-989-758-680-4, SciTePress, pages 507-518. DOI: 10.5220/0012362800003636

Decoupling the Backward Pass Using Abstracted Gradients

Kyle Rogers[1,*][a], Hao Yu[1,*][b], Seong-Eun Cho[2][c], Nancy Fulda[1][d], Jordan Yorgason[3][e]
and Tyler J. Jarvis[2][f]

[1]*Department of Computer Science, Brigham Young University, Provo, Utah, U.S.A.*
[2]*Department of Mathematics, Brigham Young University, Provo, Utah, U.S.A.*
[3]*Cellular Biology and Physiology, Center for Neuroscience, Brigham Young University, Provo, Utah, U.S.A.*

Keywords: Machine Learning, Matrix Abstraction, Biologically Inspired Learning Algorithm, Model Parallelization, Network Modularization, Backpropagation, Skip Connections, Neuromorphic.

Abstract: In this work we introduce a novel method for decoupling the backward pass of backpropagation using mathematical and biological abstractions to approximate the error gradient. Inspired by recent findings in neuroscience, our algorithm allows gradient information to skip groups of layers during the backward pass, such that weight updates at multiple depth levels can be calculated independently. We explore both gradient abstractions using the identity matrix as well as an abstraction that we derive mathematically for network regions that consist of piecewise-linear layers (including layers with ReLU and leaky ReLU activations). We validate the derived abstraction calculation method on a fully connected network with ReLU activations. We then test both the derived and identity methods on the transformer architecture and show the capabilities of each method on larger model architectures. We demonstrate empirically that a network trained using an appropriately chosen abstraction matrix can match the loss and test accuracy of an unmodified network, and we provide a roadmap for the application of this method toward depth-wise parallelized models and discuss the potential of network modularization by this method.

1 INTRODUCTION

There are numerous types of deep neural networks which excel on various tasks, but they heavily rely on a rigid error backpropagation procedure. From multilayer perceptrons to convolutional neural nets to transformer-based architectures, these models compute the gradient of the loss with respect to each model parameter to find a local minimum on the model's loss surface. Gradient computation is an expensive process and requires the gradient to be calculated layerwise backward through the neural network. This learning paradigm is incredibly successful, but also inflexible as gradient computation requires differentiable operations and sequential processing of data through the network.

In this work we present a new tool, termed abstraction matrices, which enable gradient information to be passed backward to multiple locations in the network in a decoupled fashion. We show that breaking up the backward pass in this way does not hinder model performance and allows more flexibility during backpropagation. Given this result, we explore several implications of our method: 1) theoretical depth-wise model parallelization, 2) network modularization, and 3) algorithm innovation.

Our method introduces a set of matrices $\{M^1,\ldots,M^n\}$ which correspond to the abstracted network regions. These matrices are calculated during each forward pass in such a way that, when M^k is multiplied by gradient information from the network layer immediately following the kth abstracted region, the result is a reasonable approximation of the gradient information which would have been passed to the preceding layer via traditional backpropagation methods. Said another way, during the backward pass, the abstraction matrices $\{M^1,\ldots,M^n\}$ are used to quickly transmit error information backward across multiple layer blocks via a simple matrix multiplica-

[a] https://orcid.org/0000-0001-8494-5121
[b] https://orcid.org/0009-0003-5106-6042
[c] https://orcid.org/0009-0003-3416-461X
[d] https://orcid.org/0000-0001-9391-8301
[e] https://orcid.org/0000-0002-5687-0676
[f] https://orcid.org/0000-0002-3767-029X
*These authors contributed equally

Rogers, K., Yu, H., Cho, S., Fulda, N., Yorgason, J. and Jarvis, T.
Decoupling the Backward Pass Using Abstracted Gradients.
DOI: 10.5220/0012362800003636
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 16th International Conference on Agents and Artificial Intelligence (ICAART 2024) - Volume 3*, pages 507-518
ISBN: 978-989-758-680-4; ISSN: 2184-433X
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

tion rather than via more complex backpropagation calculations.

The biological inspiration for this method, which both motivates and, to an extent, justifies the use of imperfect gradient approximation in lieu of rigorously calculated gradients, lies in observed findings from foundational neuroscience studies that identified feedback signals in biological brains are backpropagated both through localized synaptic retrograde signalling and through shortened feedback loops to distant layers (Seger and Miller, 2010; Sesack and Grace, 2010). The synaptic updates are more precise, whereas the shortened feedback loops are less accurate but facilitate a faster training response because they bypass many of the intervening neurons (Gerdeman et al., 2002; Alger, 2002). These biological foundations provide both the inspiration and a motivating precedent for our study of approximate signal mediation via abstraction matrices.

The contributions of this paper are as follows: (a) We present a biologically inspired paradigm for neural network training based on abstracted gradient information mediated via simple matrix multiplications (Sections 3.1 and 3.2); (b) We present a justification for a least-squares method for computing abstraction matrices $\{M^1,...,M^k\}$ in the case that the corresponding layers are comprised of piecewise-linear functions; (c) We introduce a simplification paradigm $M^k = I \; \forall k$ that reduces calculation overhead and is rooted in biological precedents (Section 3.4); (d) We validate the effectiveness of abstraction matrices in both multilayer perception and transformer architectures, and show that the abstraction of layer blocks via M^k can be achieved without a drop in training accuracy (Sections 4.1 and 4.2); (e) We examine anticipated speedups that could be obtained by implementing our abstracted architecture in a fully parallelized environment (Section 5.1): and (d) we discuss the application of our method in algorithm innovation and network modularization (Section 5.2).

2 RELATED WORK

Incomplete Gradient-Based Learning: Computing the error gradient with respect to model parameters is, in its pure form, a prohibitively intensive process. True gradient descent involves iterating through the entire dataset, computing each weight's gradient with respect to calculated error, and then updating the parameters in proportion to the learning rate. This is highly impractical, and thus gradients are usually computed for only a subset of the data at a time (Amari, 1993). Despite using only an approximation of the true gradient, SGD methods have proved to be quite effective in training neural networks in a supervised manner. Our work builds on this precedent by using abstraction matrices to quickly transmit approximations of the calculated error gradient.

Further approximations of the error gradient have been utilized to implement layer-parallelization (Günther et al., 2019; Song et al., 2021). Both works use optimized approximations of the forward pass and (Song et al., 2021) requires additional external compute power. Our method does not interfere with the forward pass, which does exclude the possibility of a parallelized forward pass, but addresses the more expensive backward pass. Additionally, our method is lightweight and only requires additional computation for solving the least squares problem.

One unexpected aspect of our work (see Section 4.2) is the superior effectiveness of a simplified approximation of the gradients over a more theoretically sound abstraction on certain models. While this result appears highly unintuitive, it is similar to prior work by (Neftci et al., 2017), who have shown that in neuromorphic contexts a neural network can be trained using random feedback weights multiplied by the error gradient. More generally, feedback alignment utilizes randomly initialized backward weight matrices which still facilitate learning as presented by (Lillicrap et al., 2016). (Lillicrap et al., 2016) also provide some justification as to why feedback alignment is effective which we also rely on partially in motivating our use of the identity matrix. Despite connections to these works, we draw our inspiration for, and to some extent justify, use of the identity matrix from biology as described in Section 3.4.

Residual Connections: Our work also is thematically related to residual connections as originally presented in (He et al., 2015; Srivastava et al., 2015). Conceptually, our work can be viewed as an extension of this concept to multilayer blocks, with the residual connection taking the form of an abstraction matrix M that delivers an approximation of what the calculated gradients would have been.

3 METHODOLOGY

3.1 Biological Foundations

In order for supervised machine learning or biological learning to occur, there must be an update in synaptic weights based on some error and resulting adjustment. In traditional machine learning this adjustment is often performed using an error signal that is backpropagated through the same pathway as the forward propagating signal, a method which is very effective and in some ways analogous to the neurobiological mechanisms of backpropagating action potentials (Stuart and Häusser, 2001; Letzkus et al., 2006) and release of retrograde neurotransmitters (e.g., cannabinoids and nitric oxide) (Wilson and Nicoll, 2001; Hardingham et al., 2013). These are effective mechanisms for transmitting learning signals across local connections (i.e., one layer of neurons). However, such signaling mechanisms do not typically propagate across multiple layers in biology due to interference from ongoing activity such as ion channel activation refractory periods (Burke et al., 2001). Instead, biological systems seem to prefer a combined approach where local tuning is performed by backpropagating action potentials and retrograde transmitters, while more distant upstream layers are connected and tuned via long indirect and short direct feedback loops that bypass the initial layers (Sesack and Grace, 2010). These feedback loops provide a faster method for tuning upstream neurons and are used throughout the brain, including, for example, the cortico-basal ganglia network for reward learning (Sesack and Grace, 2010).

Our work utilizes an abstraction matrix M which is computed to allow the gradient to flow around certain groups of layers of a neural network, a function analogous to the role feedback loops play in biological brains. In traditional backpropagation, the gradient is computed from the output layer sequentially up through the rest of the network. Using the matrix M, however, the gradient calculation can be divided such that the gradient in different regions of the network does not have to be computed sequentially.

3.2 Layer Abstraction to Compute the Gradient

In an effort to design a learning scheme more analogous to the human brain in deep neural networks (DNNs), we design a method to abstract the gradient computation process of several sequential layers of a DNN using a single matrix we denote M. The layers abstracted by M thus become a localized learning region with neurons whose gradient propagation process is detached from that of upstream layers. A visualization of this abstraction using M is shown in Figure 1. In some cases the identity matrix will be used in lieu of M, as depicted in Figure 1.(3-1) and described in Section 3.4. In all cases, we assume that the default regions of the network are trained using backpropagation. As such, during the backward pass of training the error gradient with respect to the model weights is computed sequentially backward through the network until reaching the region of layers abstracted by M. Then, instead of continuing the standard backpropagation procedure, the gradient is approximated for the abstracted layers using M, and the gradient computation continues around these layers according to Equation 1. G_i represents the gradient of the layer after (from the backward pass perspective) the layers abstracted by M, and G_j is the gradient of the layer immediately before the layers abstracted by M. Layer j is among the ancestor layers of layer i.

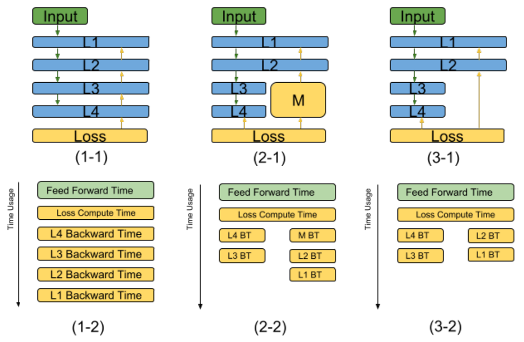

Figure 1: (1-1) shows a model composed of 4 layers, labeled as L1 to L4. Input is given through L1, and the forward data flow is indicated by green arrows. Loss is introduced through L4, and the backward data flow is indicated by yellow arrows. (2-1) and (3-1) show the same model implementing our method, with M abstracting the backward processes. (3-1) uses identity matrix as M. As illustrated in (1-2), backward processes of the traditional model are sequential. In comparison,shown by (2-2) and (3-2), backward processes using our method can become parallelized, since L2 obtains loss values through M, instead of L3.

$$G_i M = G_j \tag{1}$$

The layers abstracted by M can then either be trained according to standard backpropagation or a more simple learning rule which more closely mimics biological learning behavior. Assume, however, that the layers abstracted are also trained using backpropagation. In this case G_i is used to continue the backward pass through the layers abstracted by M, but, critically, there now exist two gradient computation paths after layer i. These two paths can be computed in parallel, which thus introduces a potential new type of parallelism in which model training can be distributed depth-wise.

3.3 Derivation of the Abstraction Matrix

Consider a neural network $\mathcal{N}$ defining a function $F_{\mathcal{N}}(\mathbf{x}) = a_k(W_k a_{k-1}(W_{k-1} \ldots a_1(W_1\mathbf{x})))$, where a_i is the activation function for the ith layer and the matrix W_i consists of the weights for layer i. (Note that if the input vector is extended with an additional 1, the bias term can be included as an additional column in the weight matrix). For a given $i < j \leq k$ and input $\mathbf{x}$ let $L_i = a_i(W_i a_{i-1}(W_{i-1} \dots a_1(W_1\mathbf{x})))$ be the output of the ith layer, let $L_j = a_j(W_j a_{j-1}(W_{j-1} \dots W_{i+1}L_i)))$ be the output of the jth layer (thought of as a function of L_i), and let $L_k = a_k(W_k a_{k-1}(W_{k-1} \dots W_{j+1}L_j)))$ be the output of the kth layer.

As a fundamental part of backpropagation we must compute gradients $G_j = G_i \left(\frac{\partial L_j}{\partial L_i}\right)^{\mathsf{T}}$, where

$$\frac{\partial L_j}{\partial L_i} = \frac{\partial L_j}{\partial L_{j-1}} \frac{\partial L_{j-1}}{\partial L_{j-2}} \cdots \frac{\partial L_{i+1}}{\partial L_i} \qquad (2)$$

is the derivative of the layer L_j as a function of L_i. It's relatively expensive to compute these derivatives by computing the corresponding matrix products in (2). Moreover, each of these matrix derivatives depends on the value of the input $\mathbf{x}$, so the product must be recomputed for each $\mathbf{x}_\ell$ in a given batch. To emphasize this dependence, we use a superscript $\mathbf{x}_\ell$ on the layers: $\frac{\partial L_j^{\mathbf{x}_\ell}}{\partial L_i^{\mathbf{x}_\ell}}$.

Expressed mathematically, the main idea of this paper is to approximate all the different transposed derivatives $\left(\frac{\partial L_j^{\mathbf{x}_\ell}}{\partial L_i^{\mathbf{x}_\ell}}\right)^{\mathsf{T}}$ with a single abstraction matrix M, which depends on the batch, but is the same for all choices of $\mathbf{x}_\ell$.

Our choice of M is motivated by the observation that any piecewise-linear function f satisfies the differential relation $f(\mathbf{x}) = D_{\mathbf{x}} f(\mathbf{x}) \cdot \mathbf{x}$, where $D_{\mathbf{x}} f(\mathbf{x})$ is the derivative of f with respect to $\mathbf{x}$. Specifically, if the activation functions in the neural network $\mathscr{N}$ are all piecewise linear (e.g., ReLU or leaky ReLU), then for any input $\mathbf{x}$ we have

$$L_j^{\mathbf{x}_\ell} = \frac{\partial L_j^{\mathbf{x}_\ell}}{\partial L_i^{\mathbf{x}_\ell}} L_i^{\mathbf{x}_\ell}.$$

A matrix M^{T} that approximates every derivative $\frac{\partial L_j^{\mathbf{x}_\ell}}{\partial L_i^{\mathbf{x}_\ell}}$ should, therefore, give a good approximate solution to the system of equations

$$M^{\mathsf{T}} L_i^{\mathbf{x}_\ell} = L_j^{\mathbf{x}_\ell} \qquad \forall \ell \in B, \qquad (3)$$

where B is the set of all indices in the batch. Assembling the various columns $L_i^{\mathbf{x}_\ell}$ together into a single matrix L_i^B and the columns $L_j^{\mathbf{x}_\ell}$ together into a single matrix L_j^B, we can write the system (3) as

$$M^{\mathsf{T}} L_i^B = L_j^B. \qquad (4)$$

The natural choice for an approximate solution to any (potentially non-square) linear system is the least-squares solution of (4), which can be written as

$$M = \left(L_i^B\right)^{\mathsf{T}+} \left(L_j^B\right)^{\mathsf{T}}, \qquad (5)$$

where $\left(L_i^B\right)^{\mathsf{T}+}$ is the Moore–Penrose pseudoinverse of $(L_i^B)^{\mathsf{T}}$. This motivates our choice of the abstraction matrix M to be defined by (5).

3.4 A Simplified Abstraction Matrix

In the cortico-basal ganglia brain region from which we take our inspiration, feedback loops that bypass initial layers do not use an estimation of those layers' gradients, but instead pass the error signal directly to the more distant neurons (Sesack and Grace, 2010; Seger and Miller, 2010). To mimic this behavior, we also ran a number of experiments with M equal to the identity matrix rather than the derived value given in Eq. (5). This simplification ($\forall k, M^k$=I) reduces calculation overhead and is better aligned with biological precedents; however, it is a less accurate way of estimating the abstracted gradients. Our expectation was that it would result in reduced neural network performance as compared to the more rigorously calculated M; however, as described in Section 4.2, this was not the case.

4 EXPERIMENTS

We explore the effectiveness of the layer abstraction M on a variety of models and training tasks, with the goal of establishing (a) the performance of models trained using abstraction matrices as compared to unmodified models, and (b) the theoretical speedup which might be gained if the model were parallelized along the layer blocks approximated by abstraction matrices. We further consider two distinct methods for calculating M: The theoretical derivation described in Sec. 3.3, and a biologically motivated simplification using the identity matrix (M=I).

4.1 Multilayer Perceptron

While small multilayer perceptron (MLP) (Block et al., 1962) models are not the most suitable candidates for the downstream implications of our method, we chose them as an initial testbed due to their simplicity and conformity with the constraints of Section 3.3. Our aim in this experiment is two-fold: (1) to ensure that training using M (which creates a decoupled backward pass) does not decrease final model accuracy and (2) to establish an algorithm that enables the separation of the backward pass into multiple procedures.

To validate the mathematical theory behind layer abstraction we verify that we can compute and use

M on a five-layer multilayer perceptron (MLP) comprised of fully connected layers with ReLU activations. We train this MLP model on three simple image recognition datasets—MNIST (Deng, 2012), EMNIST (Cohen et al., 2017), and FashionMNIST (FMNIST) (Xiao et al., 2017)—and compare the abstracted model's performance to a baseline MLP model trained without any abstractions. In this experiment different models are used for different datasets, with some base code from (Koehler, 2020). In the model, the computed matrix *M* spanned three of the five MLP layers, leaving the input and output layers unmodified. All models contained hidden layers ranging from dimension (392,196) to (49, 10) or (49,26) for EMNIST, with larger layers on input side and smaller layers closer to output layers, as defined in (Gregor Koehler and Markovics, 2020). Models were trained for 10 epochs, using the Adam optimizer and negative log likelihood loss, on batches of size 64 and an initial learning rate of 0.0001.

For this experiment, we also studied the simple $M = I$ abstraction. One limitation of using such a simple abstraction, however, is that the gradient vectors, G_i, G_j, must be of the same size since I is a square matrix. Thus, to apply the $M = I$ abstraction to this MLP model we instead utilize a block identity matrix, $I_{block} = [I \ 0]$. This effectively adds zero padding to maintain the proper gradient chain. Observe that using I_{block} is essentially dropping gradient information in order to project the gradient to a different dimension size.

Results are shown in Table 1. We see that the derived matrix *M* matches the performance of the non-abstracted model on the MNIST and FMNIST dataset and nearly matches on EMNIST. As the model architecture is the same in both cases, this suggests that like many other aspects of neural architecture design, the effectiveness of the abstracted gradient calculation technique is partially dependent on the specific task being solved. The $M = I_{block}$ abstraction performs measurably worse on all three datasets, validating the worth of deriving the matrix *M* as presented in Section 3.3.

4.2 Seq2Seq Transformer

Our next experiment leverages the popular Seq2Seq transformer as presented by (Vaswani et al., 2017), using its implementation from (Klein et al., 2017). This model has been leveraged as a base architecture for many modern DNNs including the GPT line of language models (Radford et al., 2019; Brown et al., 2020; Black et al., 2021; Ouyang et al., 2022), audio processing models (Dong et al., 2018; Gulati et al., 2020; Chen et al., 2021), and computer vision applications (Carion et al., 2020; Dosovitskiy et al., 2021). Thus, examining layer abstractions in the base model presented by (Vaswani et al., 2017) offers valuable intuition and preliminary information about layer abstractions in other, more modern, transformer-based architectures.

We begin by highlighting that the transformer architecture does not meet the constraints required by Section 3.3, as the *M* matrix must abstract *n* linear layers with ReLU activations to be an exact abstraction. Therefore, we compare the derived approximation to both the unmodified baseline and a biologically-inspired value for *M* using the identity matrix (*M=I*), as discussed in Section 3.4. Performance of both models was evaluated using the German→English translation task from the Multi30k dataset (Elliott et al., 2016). Our transformer model consisted of six encoder layers and six decoder layers. In each abstraction model the last 3 attention layers were abstracted by a single *M* within the encoder portion of the network. We evaluated performance using BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), and COMET (Rei et al., 2020) scores, all of which are established metrics in the field of machine translation.

Results are shown in Table 2. Somewhat unexpectedly, we find that the biologically-inspired gradients *M=I* performed much better than the mathematically derived gradients from Section 3.3. Despite the transformer model not meeting the constraints of our derived abstraction we did not expect performance to suffer as it did. We hypothesize that this is due to variations in the magnitude and direction of the difference between *M* and the true gradients G_i. To validate this unusual result, we applied the gradient approximation $M = I$ to a much larger dataset, IWSLT17 (Cettolo et al., 2017), again using the German→English translation as our benchmark and comparing to our baseline model. The results, shown in Table 3, confirm that the approximate gradients transmitted by *M=I* are sufficient for effective learning. This means that decoupling of the backward pass can be achieved without any significant reduction in model performance.

4.3 Ablation Study

This study shows that the strong performance of *M=I* is not caused by skipping unnecessary layers on the backward pass and that *M=I* does not cause the abstracted layers to become irrelevant. For this experiment, we set up multiple Seq2Seq transformers using the same structure presented by (Vaswani et al., 2017). We held the model dimension fixed at 512, us-

Table 1: Test set accuracy and standard deviation across five training runs for each MNIST variant. Test set accuracy is selected as the maximum test set accuracy after each epoch. Model architectures are as described in Section 4.1.

	MNIST	FMNIST	EMNIST
	(acc, stddev)	*(acc, stddev)*	*(acc, stddev)*
baseline model	(0.9763, 0.002)	(0.8793, 0.003)	(0.9022, 0.002)
abstracted gradients (derived)	(0.9762, 0.001)	(0.8724, 0.002)	(0.8948, 0.002)
abstracted gradients ($M = I_{block}$)	(0.9612, 0.002)	(0.8642, 0.003)	(0.8584, 0.003)

Table 2: Model performance of a baseline transformer as compared to models leveraging both derived and biologically inspired abstraction matrices M. Evaluations were performed using the Multi30k dataset, en$\rightarrow$de task. The first number of each tuple shows the average accuracy across ten training runs. The second number shows the standard deviation across the ten trials.

	baseline model	abstracted gradients (M=I)	abstracted gradients (derived)
	(acc, stddev)	*(acc, stddev)*	*(acc, stddev)*
BLEU	(0.386, 0.008)	(0.383, 0.008)	(0.199, 0.033)
METEOR	(0.708, 0.006)	(0.705, 0.004)	(0.477, 0.042)
COMET	(0.774, 0.004)	(0.772, 0.004)	(0.627, 0.025)

ing a batch size of 32 and Adam optimizer with adaptive learning rate. While the original structure from (Vaswani et al., 2017) used 6 encoder layers and 6 decoder layers, we also tried variants with 3 encoder layers and 3 decoder layers. We used n to represent the numbers of encoder and decoder layers. When using M and $n = 6$, the last 3 layers of the encoder block are abstracted. When $n = 3$, the last layer of the encoder block are abstracted. We trained each model setup for 14 epochs with five trials on the Multi30k German$\rightarrow$English training data. Then, we picked the model with lowest validation loss from each trial to perform the translation task on Multi30k test set. We measure each trial's BLEU and METEOR score and take average across five trials with the same model setup. We also measured scores after removing those layers which would have become abstracted layers if M had been used. The results are in Table 4.

As seen in Table 4, we can first conclude that the effectiveness of *M=I* isn't due to redundancy in the trained model. Before ablation of the baseline model, $n = 6$ baseline performs better than $n = 3$ baseline, demonstrating that the extra complexity of the model is matched by a corresponding increase in performance. It is therefore not the case that the abstraction matrix is merely approximating a smaller model; it is instead successfully retaining the complexity of the larger one. Moreover, ablation of the baseline model resulted in reduced performance, suggesting that the layers that would have been abstracted by M are impactful to the tasks. Therefore, we can conclude that M did not skip unnecessary layers during the backward pass. To determine whether layers still retain their importance after abstraction using *M=I*, we compare the performance of *M=I* and the baseline both before and after abstraction. Our data indicates that both *M=I* models and the corresponding baseline models lost similar amounts of performance after removing layers of abstraction locations. This indicates that those layers retain their importance even after the abstraction process. We present additional ablation studies varying the position and size of M in the appendix.

5 DISCUSSION

5.1 Theoretical Speedup

Efficient, large-scale parallelization of deep learning models is a highly specialized field, requiring the successful navigation of challenges including partitioning, re-materialization, and data transfer (Griewank and Walther, 2000; Chen et al., 2016; Huang et al., 2019). Such an endeavor is beyond the scope of this work, and we note in particular that a naive parallelization implementation of this novel decoupling method using, for example, `torch.multiprocessing` (Foundation, 2023) is unlikely to be effective. However, we provide here a small theoretical analysis showing the predicted impact on wall clock time of the backward pass of a parallelized implementation of our abstracted neural network.

In Section 3.2, we showed that we can approximate the gradient calculation of certain groups of layers which are abstracted by M. Our gradient derivation method for nonadjacent layers can be written as,

$$G_i M = G_j \tag{6}$$

where G_i represents the ith layer's gradient and G_j represents the jth layer's gradient. This allows the

Table 3: Model performance of a baseline transformer compared to a model using the *M=I* abstraction matrix. Evaluations were performed using the IWSLT17 dataset, en→de task. Column values show average accuracy and standard deviation across ten data runs.

	baseline model	abstracted gradients (M=I)	abstracted gradients (derived)
	(acc, stddev)	*(acc, stddev)*	*(acc, stddev)*
BLEU	(0.294, 0.002)	(0.291, 0.002)	(0.194, 0.006)
METEOR	(0.706, 0.004)	(0.705, 0.005)	(0.652, 0.006)

Table 4: Model performance of de→en translations on Multi30k test set, average of five trials. Only models from each trial, scored lowest validation loss, were picked for translation tasks. The term "ablated" following a scoring metric means that translation tasks were performed after removing certain layers from the models. *n* refers to the number of encoder layers. When *n*=6, ablation removed the last 3 attention layers from model's encoder block. When *n*=3, ablation removed the last attention layer from model's encoder. The removed layers occupied the same positions as the layers replaced by *M* in the abstracted models.

	n=6 (baseline)	n=3 (baseline)	n=6 (M=I)	n=3 (M=I)
BLEU	0.386	0.385	0.383	0.374
BLEU (ablated)	0.374	0.380	0.374	0.372
METEOR	0.708	0.709	0.705	0.702
METEOR (ablated)	0.698	0.704	0.696	0.697

gradient for the layers after layer i to be computed using M, rather than sequentially computing the gradient through layers j through i. Importantly, the layers abstracted by M are still updated using backpropagation, but this occurs after the abstracted matrix M has mediated the approximate gradients. Networks can use more than one M to have a parallelized backward pass through the layers abstracted by M, as shown in Figure 2. A speedup can be obtained even though computing M for each backward pass requires additional matrix operations.

We can model the backward process computation time of the layers abstracted by M (the light blue boxes in Figure 2) as shown below:

$$t_R \approx ml \qquad \text{and} \qquad t_M \approx mo + l. \tag{7}$$

where t_R is the backward time on a regular neural network without M, t_M is the backward time consumption on a neural network with M implemented. In these equations m is the amount of M matrices we have in a network. l is the estimated computation time needed to perform the backward pass on the layers skipped by a single M matrix. o is the amount of overhead needed to derive M and passing gradient through M.

If we require the backward pass of a network to be δ times faster, then:

$$t_M \approx t_R \frac{1}{\delta} \Rightarrow om + l \approx \frac{1}{\delta} ml \tag{8}$$

For example, when $m = 6$, $o = 2$ and $l = 6$, we have a $\delta = 2$ times speed up on a transformer model's backward processes. In other words, a 2 times speed up can be achieved when there are 6 M and overhead time for each M is only one third of the amount of time of a group of abstracted layers' backpropagation.

5.2 Optimization Algorithm Innovation and Network Modularization

With abstraction matrices used to transfer error signals across intermediate layers, abstracted layers are no longer required to perform traditional gradient descent to generate loss values for their upstream layers. Consequently, abstracted layers could potentially employ optimization algorithms other than gradient descent, while gradient descent could still be used on some layers to maintain the network's performance. This could open up research opportunities for new optimization algorithms. More concretely, a network trained with backpropagation and another optimization algorithm, denoted as algorithm A and algorithm B respectively, could utilize algorithm A in all layers except the layers abstracted by M and the layers abstracted by M could learn according to algorithm B. The incoming gradient to the layers abstracted by M could be ignored, modified or substituted according to whatever details are required by algorithm B. Thus, training using M is a robust approach to utilizing different optimization strategies in different regions of a network. We leave the exploration of these alternative optimization strategies to future work, but we acknowledge their potential in introducing network modularization.

Assuming a network trained using one such optimization method on the abstracted regions of the network can still achieve acceptable test accuracy, then an imperfect learning signal could coerce the network to learn a sort of network modularization. This behavior has been proven to be true in biological brains, where some neurons exhibit learning behavior more

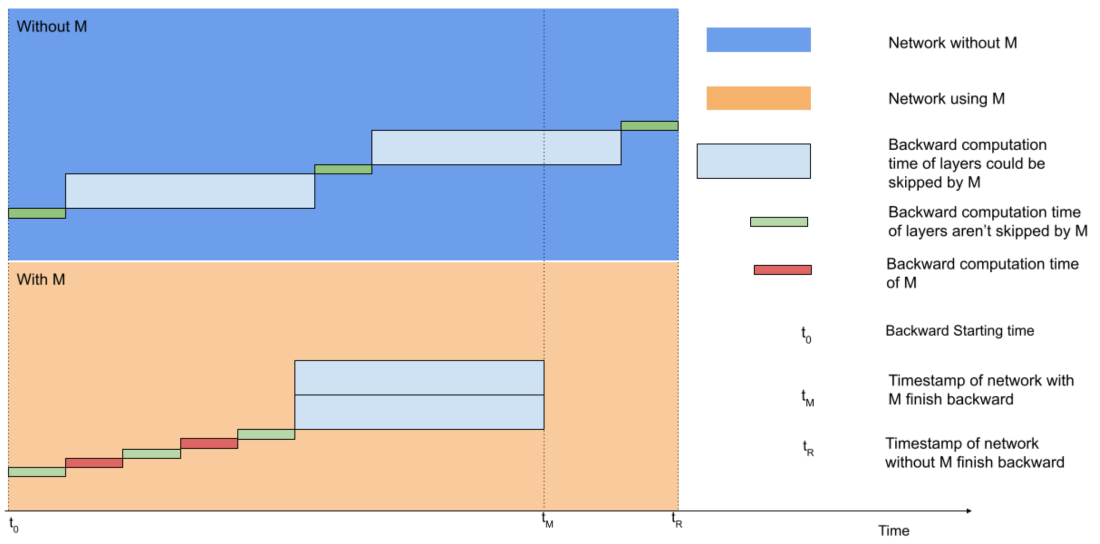
Without M
With M
Network without M
Network using M
Backward computation time of layers could be skipped by M
Backward computation time of layers aren't skipped by M
Backward computation time of M
t_0 Backward Starting time
t_M Timestamp of network with M finish backward
t_R Timestamp of network without M finish backward
t_0
t_M
t_R
Time

Figure 2: A comparison of backpropagation computation with and without M. Observe that the backward computation of the layers abstracted by multiple M matrices have their gradient computation decoupled.

attuned to learning XOR logic than to other logic (Gidon et al., 2020).

While network modularization of this form has yet to be explored in artificial neural networks, biological neural network modularization is well understood. There is a link between how optimization algorithms are associated with network modularization in the biological brain. In the biological brain, neuronal connections are maintained and updated by different neurotransmitters, which influences neurons to exhibit different weight update behaviors (Amunts et al., 2010; Huang and Reichardt, 2001). Moreover, in different brain regions, neurotransmitter types vary (Amunts et al., 2010; Amunts and Zilles, 2015; Huang and Reichardt, 2001; Paxinos and Mai, 2003). Such variation results in different weight update behaviors forming distinct brain regions which carry out different functions (Amunts et al., 2010), (Amunts and Zilles, 2015; Huang and Reichardt, 2001; Paxinos and Mai, 2003). With our M algorithm enabling usage of different optimization algorithms, we can mimic the existence of different neurotransmitters in different brain regions. Therefore, training with M enables a way toward bringing biological brain modularization into artificial neural networks.

6 CONCLUSIONS

This work has presented a biologically-inspired learning mechanism whereby approximate gradient information is propagated quickly through the network via a set of abstraction matrices M^k. This decouples gradient computation of each set of abstracted layers. Decoupled computation allows the weight updates within each block of abstracted layers to be theoretically executed in parallel, with potential applications for speeding up the backward pass of large computationally expensive networks. The next logical step in this line of research would be the utilization of abstraction matrices to create a depth-wise parallelized network architecture, and to explore potential applications toward online learning and real-time network updates.

The gradient abstraction techniques introduced in this work have research potential that extends beyond gradient decoupling. In biological brains, cortico-basal ganglia pathways – mimicked in our work by abstraction matrices – and localized logic updates are not mutually exclusive. It is often the case that imprecise learning signals are propagated quickly via the cortico-basal ganglia feedback loops, then followed by more precise updates mediated by retrograde synaptic connections between neurons (Sesack and Grace, 2010; Wilson and Nicoll, 2001). Our method for propagating abstracted gradients could be leveraged toward a similar setup where network parameters are updated both via the abstraction matrix and also via more traditional methods. This idea has particular relevance in the domain of neuromorphic computing and spiking neural networks.

REFERENCES

Alger, B. E. (2002). Retrograde signaling in the regulation of synaptic transmission: Focus on endocannabinoids. *Progress in Neurobiology*, 68(4):247 – 286. Cited by: 486.

Amari, S. (1993). Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5:185–196.

Amunts, K., Lenzen, M., Friederici, A. D., Schleicher, A., Morosan, P., Palomero-Gallagher, N., and Zilles, K. (2010). Broca's region: Novel organizational principles and multiple receptor mapping. *PLOS Biology*, 8(9):1–16.

Amunts, K. and Zilles, K. (2015). Architectonic mapping of the human brain beyond brodmann. *Neuron*, 88(6):1086–1107.

Banerjee, S. and Lavie, A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Black, S., Gao, L., Wang, P., Leahy, C., and Biderman, S. (2021). GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow.

Block, H. D., Knight Jr, B., and Rosenblatt, F. (1962). Analysis of a four-layer series-coupled perceptron. ii. *Reviews of Modern Physics*, 34(1):135.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners.

Burke, D., Kiernan, M. C., and Bostock, H. (2001). Excitability of human axons. *Clinical Neurophysiology*, 112(9):1575 – 1585. Cited by: 368.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 213–229. Springer.

Cettolo, M., Federico, M., Bentivogli, L., Niehues, J., Stüker, S., Sudoh, K., Yoshino, K., and Federmann, C. (2017). Overview of the IWSLT 2017 evaluation campaign. In *Proceedings of the 14th International Conference on Spoken Language Translation*, pages 2–14, Tokyo, Japan. International Workshop on Spoken Language Translation.

Chen, T., Xu, B., Zhang, C., and Guestrin, C. (2016). Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*.

Chen, X., Wu, Y., Wang, Z., Liu, S., and Li, J. (2021). Developing real-time streaming transformer transducer for speech recognition on large-scale dataset. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5904–5908. IEEE.

Cohen, G., Afshar, S., Tapson, J., and Van Schaik, A. (2017). Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pages 2921–2926. IEEE.

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142.

Dong, L., Xu, S., and Xu, B. (2018). Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5884–5888. IEEE.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale.

Elliott, D., Frank, S., Sima'an, K., and Specia, L. (2016). Multi30k: Multilingual english-german image descriptions. In *Proceedings of the 5th Workshop on Vision and Language*, pages 70–74. Association for Computational Linguistics.

Foundation, T. P. (2023). Multiprocessing package - torch.multiprocessing. https://pytorch.org/docs/stable/multiprocessing.html.

Gerdeman, G. L., Ronesi, J., and Lovinger, D. M. (2002). Postsynaptic endocannabinoid release is critical to long-term depression in the striatum. *Nature Neuroscience*, 5(5):446 – 451. Cited by: 607.

Gidon, A., Zolnik, T. A., Fidzinski, P., Bolduan, F., Papoutsi, A., Poirazi, P., Holtkamp, M., Vida, I., and Larkum, M. E. (2020). Dendritic action potentials and computation in human layer 2/3 cortical neurons. *Science*, 367(6473):83–87.

Günther, S., Ruthotto, L., Schroder, J. B., Cyr, E. C., and Gauger, N. R. (2019). Layer-parallel training of deep residual neural networks.

Gregor Koehler, A. A. and Markovics, P. (2020). Mnist handwritten digit recognition in pytorch. https://nextjournal.com/gkoehler/pytorch-mnist.

Griewank, A. and Walther, A. (2000). Algorithm 799: revolve: an implementation of checkpointing for the reverse or adjoint mode of computational differentiation. *ACM Transactions on Mathematical Software (TOMS)*, 26(1):19–45.

Gulati, A., Qin, J., Chiu, C.-C., Parmar, N., Zhang, Y., Yu, J., Han, W., Wang, S., Zhang, Z., Wu, Y., and Pang, R. (2020). Conformer: Convolution-augmented Transformer for Speech Recognition. In *Proc. Interspeech 2020*, pages 5036–5040.

Hardingham, N., Dachtler, J., and Fox, K. (2013). The role of nitric oxide in pre-synaptic plasticity and homeostasis. *Frontiers in Cellular Neuroscience*, (OCT). Cited by: 176; All Open Access, Gold Open Access, Green Open Access.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition.

Huang, E. and Reichardt, L. (2001). Neurotrophins: Roles in neuronal development and function. *Annual Review of Neuroscience*, 24:677 – 736. Cited by: 3394; All Open Access, Green Open Access.

Huang, Y., Cheng, Y., Bapna, A., Firat, O., Chen, D., Chen, M., Lee, H., Ngiam, J., Le, Q. V., Wu, Y., et al. (2019). Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, 32.

Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. M. (2017). Opennmt: Open-source toolkit for neural machine translation. In *Proc. ACL*.

Koehler, G. (2020). Mnist handwritten digit recognition in pytorch.

Letzkus, J. J., Kampa, B. M., and Stuart, G. J. (2006). Learning rules for spike timing-dependent plasticity depend on dendritic synapse location. *Journal of Neuroscience*, 26(41):10420 – 10429. Cited by: 217; All Open Access, Bronze Open Access, Green Open Access.

Lillicrap, T. P., Cownden, D., Tweed, D. B., and Akerman, C. J. (2016). Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications*, 7(1):13276.

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

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. (2022). Training language models to follow instructions with human feedback.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Paxinos, G. and Mai, J. K. (2003). *The Human Nervous System*. Cited by: 24.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Rei, R., Stewart, C., Farinha, A. C., and Lavie, A. (2020). COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online. Association for Computational Linguistics.

Seger, C. A. and Miller, E. K. (2010). Category learning in the brain. *Annual Review of Neuroscience*, 33:203 – 219. Cited by: 242; All Open Access, Green Open Access.

Sesack, S. R. and Grace, A. A. (2010). Cortico-basal ganglia reward network: Microcircuitry. *Neuropsychopharmacology*, 35(1):27 – 47. Cited by: 732; All Open Access, Bronze Open Access, Green Open Access.

Song, Y., Meng, C., Liao, R., and Ermon, S. (2021). Accelerating feedforward computation via parallel nonlinear equation solving.

Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015). Highway networks. *arXiv preprint arXiv:1505.00387*.

Stuart, G. J. and Häusser, M. (2001). Dendritic coincidence detection of epsps and action potentials. *Nature Neuroscience*, 4(1):63 – 71. Cited by: 279.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need.

Wilson, R. and Nicoll, R. (2001). Endogenous cannabinoids mediate retrograde signalling at hippocampal synapses. *Nature*, 410(6828):588 – 592. Cited by: 1267.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747.

APPENDIX

Extended Ablation Results and Additional Experiments

Extended Results. We present here an expanded table demonstrating results on both derived M and *M=I*, along with the additional SacreBLEU scoring metric.

This experiment seeks to answer the following question: Is it possible that the use of the *M=I* abstraction is effective, not because *M=I* is a reasonable approximation for the gradients, but because the network is learning, in effect, to ignore the intervening network layers. In other words, would it be more effective to simple train a smaller network rather than using *M=I* as an abstracted gradient representation?

We address this question by comparing abstracted models of various size with corresponding non-abstracted models in which the layers bridged by *M* have been entirely removed. If the ablated version of each model matches the performance of the abstracted model, then that would suggest that the abstraction is in fact not useful for learning, and is instead simply functioning as a mechanism to simulate a model with fewer layers overall. Results are shown in Table 5.

Abstraction Position Experiment. Here we investigate the impact of abstraction layer positioning on model performance. We provide an ablation study where we vary the position of the abstraction matrix within the transformer model. Results are shown in Table 6.

Abstraction Size Experiment. The power of potential parallelization increases as we define additional abstractions or increase abstraction size. As the results show in Table 7, abstraction on different positions perform similar to each other before ablation. This is a positive result as the abstraction's usefulness is not necessarily limited by position. However, after ablation, A3(D234) has the most amount of performance drop. We do not yet have an conrete explanation for this behavior, so we will explore it in future works.

We observe that for each pair of corresponding models, the ablated version performs less well than the abstracted version, suggesting that the model is indeed leveraging the inherent learning capacity of the additional layers. Additionally, the performance drop from the abstracted models is about the same as the drop seen in the baseline model when the same layers are removed. We therefore conclude that the question above can be answered in the negative. The abstraction *M=I* is indeed preserving useful learning capacity in the abstracted layers. We note, however, that the performance of the model does seem slightly better when only one layer is abstracted rather than three. This raises the question of how many network layers can be effectively abstracted at one time before network performance begins to degrade. Further research is needed before this question can be answered with confidence.

Additional Biological Foundations

Our work is inspired by recent findings in neuroscience. In both biological learning and machine learning via artificial neural networks, an update mechanism must exist which adjusts the synaptic weights based on observed error in the output signal. Traditional machine learning achieves this via backpropagated error signals, a method which is analogous to the neurobiological mechanisms of backpropagating action potentials (Stuart and Häusser, 2001; Letzkus et al., 2006) and release of retrograde neurotransmitters (e.g. cannabinoids and nitric oxide) (Wilson and Nicoll, 2001; Hardingham et al., 2013)

Our work expands upon this foundation by introducing an alternate approach to the transmission of error gradients. Researchers have observed that, in biological brains, neither backpropagating action potentials nor retrograde neurotransmitter signals typically propagate across multiple layers due to interference from ongoing activity and ion channel activation refractory periods (Burke et al., 2001). Instead, biological systems seem to rely on feedback loops more distant upstream layers are connected via feedback loops that bypass the initial layers (Sesack and Grace, 2010). We attempt to implement a similar system via the abstraction matrix *M*, which is inspired in particular by the cortico-basal ganglia network for reward learning (Sesack and Grace, 2010).

Research on cortico-basal ganglia dopamine network connectivity and behavioral implications is ongoing, and much of the circuit framework is still hypothetical. However, a consistent theme is that ventral tegmental area (VTA) dopamine cell bodies receive sensory input and project to the ventral striatum (i.e. nucleus accumbens) where dopamine release occurs in response to rewards and associated sensory stimuli to encode valence and form learned associations (Sesack and Grace, 2010). When a sensory stimuli is reinforcing, it drives dopamine release onto output medium spiny neurons (MSNs), concomitant to glutamate signals from cortical, thalamic, amygdala and hippocampal inputs encoding additional important aspects of the stimuli (such as its emotional value) (Seger and Miller, 2010). The dopamine signal acts as a gain modulator to facilitate or diminish propagation of that signal through MSNs. The MSNs then propagate the signals to their respective output layers called the direct and indirect pathways. Importantly, those two pathways also form two parallel feedback loops that have different numbers of layers and can thus influence future VTA dopaminergic activity through either a short or long feedback mechanism (Seger and Miller, 2010; Sesack and Grace, 2010). Furthermore, local striatal synaptic activity is still tuned by retrograde cannabinoid neurotransmission (Gerdeman et al., 2002; Alger, 2002). Thus, biological mechanisms can include backpropagating techniques (i.e. retrograde transmission) or feedback loops that skip layers to tune upstream activity.

The complexity of neurobiological feedback mechanisms in biological brains are too complex to imitate in their entirety. However, we take inspiration from the behavior of dopamine signals in the cortico-basal ganglia network in the creation of an abstraction matrix *M* which allows error signals to bypass clustered groups of layers in an artificial neural network. Traditionally, artificial neural networks have ignored these longer feedback loops and have typically focused only on backpropagating error signals between proximate neurons. We believe that this oversight fundamentally limits the opportunities for learning in deep neural networks. The abstraction matrix *M* introduces an alternative pathway for the propagation of error signals, and as such may open new computation paradigms for deep learning systems.

Table 5: Ablation study. B=BLEU, SB=SacreBLEU, M=METEOR, (a)=ablated model. Each tuple shows average final accuracy and standard deviation across five training runs.

	n=6 (baseline)	**n=3 (baseline)**	**n=6 (M=I)**	**n=3 (M=I)**	**n=6 (derived)**	**n=3 (derived)**
B	(0.386, 0.008)	(0.385, 0.011)	(0.383, 0.008)	(0.374, 0.010)	(0.199, 0.033)	(0.317, 0.017)
B(a)	(0.374, 0.010)	(0.380, 0.006)	(0.374, 0.007)	(0.372, 0.006)	(0.153, 0.038)	(0.299, 0.009)
SB	(0.386, 0.008)	(0.385, 0.011)	(0.383, 0.008)	(0.374, 0.010)	(0.199, 0.033)	(0.317, 0.017)
SB(a)	(0.374, 0.007)	(0.380, 0.006)	(0.374, 0.007)	(0.372, 0.006)	(0.153, 0.038)	(0.299, 0.009)
M	(0.708, 0.006)	(0.709, 0.007)	(0.705, 0.004)	(0.702, 0.007)	(0.477, 0.042)	(0.623, 0.016)
M(a)	(0.698, 0.006)	(0.704, 0.004)	(0.696, 0.007)	(0.697, 0.007)	(0.407, 0.058)	(0.603, 0.011)

Table 6: Performances of different abstraction sizes on Multi30k dataset over 10 trials. Models are transformer models with 6 encoder layers and 6 decoder layers. Abstraction method is $M = I$. A3(E345) means abstract 3 consecutive layers with a single M, from 3rd to 5th encoder layers. E and D means the encoder and decoder layers respectively. And (a) means performances measured after ablated abstracted layers.

	A3(E234)	A3(E345)	A3(D234)	A3(D345)
	(acc, stddev)	*(acc, stddev)*	*(acc, stddev)*	*(acc, stddev)*
BLEU	(0.368, 0.010)	(0.371, 0.008)	(0.362, 0.007)	(0.362, 0.010)
BLEU(a)	(0.332, 0.006)	(0.353, 0.011)	(0.307, 0.010)	(0.334, 0.011)
COMET	(0.759, 0.004)	(0.759, 0.005)	(0.751, 0.004)	(0.753, 0.005)
COMET(a)	(0.734, 0.005)	(0.749, 0.006)	(0.679, 0.017)	(0.715, 0.008)
METEOR	(0.692, 0.007)	(0.690, 0.006)	(0.679, 0.007)	(0.683, 0.007)
METEOR(a)	(0.642, 0.009)	(0.671, 0.009)	(0.614, 0.014)	(0.644, 0.009)

Table 7: Performances of different abstraction sizes on Multi30k dataset over 10 trials. Models above are transformer models with 6 encoder layers and 6 decoder layers. Abstraction method in this experiment is $M = I$. A6(L3-5) means abstracted 6 layers in total, with two blocks of 3 consecutive layers, from 3rd to 5th layers in both the encoders and decoders.

	A4(L4-5)	A6(L3-5)	A8(L2-5)
	(acc, stddev)	*(acc, stddev)*	*(acc, stddev)*
BLEU	(0.362, 0.010)	(0.359, 0.008)	(0.360, 0.011)
COMET	(0.754, 0.006)	(0.750, 0.005)	(0.746, 0.007)
METEOR	(0.684, 0.009)	(0.678, 0.008)	(0.678, 0.008)

Chapter 4

In preparation: Biologically-Inspired Spiking Bias Design For Use in EEG Signal Classification

This manuscript has not yet been accepted for publication.

Biologically-Inspired Spiking Bias Design For Use in EEG Signal Classification

Kyle Rogers [1], Nancy Fulda [1]

[1]Department of Computer Science
Brigham Young University
kroger25@byu.edu, nfulda@cs.byu.edu,

Abstract

We present a novel spiking bias neuron design inspired from the biological stimuli, output, and construction of dopamine neurons in biological organisms. Our spiking biases receive the input signal and supply a decaying bias current to the connected neuron(s). We show the effectiveness of these spiking biases in a fine tuning objective on EEG data to classify emotion. We compare the performance between traditional (non-spiking) biases and spiking biases and show that our method provides a 5% performance boost on average. We leverage our design of spiking bias neurons to optimize additional parameters during fine tuning to further increase model performance by 13% on average. Finally, we present a bias tuning variant, which we name volumetric tuning, where a single spiking bias neuron propagates its signal to a group or volume of neurons in the pretrained network. This method is analogous to the chemical transmission mechanisms of neurotransmitters in biological brains and results in an average test accuracy of 94% when optimizing only 1000 parameters during fine tuning.

1 Introduction

Spiking neural networks (SNNs) are a unique class of neural networks which seek to model the spiking behavior of biological neurons. Spiking neuron models have shown to be effective in processing time varying data, and have gained particular prominence with the deployment of commercially available neuromorphic accelerators (Davies et al. 2018; Orchard et al. 2021; Akopyan et al. 2015; Posey 2022). Using these accelerators, SNNs can run with incredibly low-latency and low-power. With these advantages come challenges in training high performing models on complex tasks.

Emotion classification is a complex task that has many downstream applications in the field of Affective Computing, Human Computer Interaction and Brain-Computer Interaction (BCI). One source of data to classify emotion is time-varying physiological signals such as electroencephalogram (EEG) signals. In fact, (Picard, Vyzas, and Healey 2001) observed that physiological signals can be more beneficial than visual or audio data in machine intelligence. Training an artificial SNN to classify EEG signals yields numerous opportunities in the aforementioned fields. Furthermore, if a small and low-power SNN can learn to classify EEG signals, such a network could be deployed on neuromorphic hardware to perform on-chip emotion classification.

We tackle this challenge by adapting existing network fine tuning algorithms for a small, low-power SNN. Just as SNNs are inspired by the spiking function of biological neurons, we too draw inspiration from discoveries in neuroscience to develop tuning methods. More explicitly, we design spiking biases whose stimulus, output, and structure are more biologically accurate than traditional bias input. Our contributions are summarized as below:

- We extend previous work on bias, or neuromodulatory, tuning by applying it to a LOOCV fine tuning objective on a complex EEG emotion classification dataset.
- We present a novel design of spiking bias neurons inspired from findings in neuroscience. We show this design's effectiveness in bias tuning.
- We utilize our spiking bias neuron design to present a bias tuning variant, termed volumetric tuning, to fine tune a SNN to higher accuracies than bias tuning.

This work first reviews SNNs, their training, and biases in SNNs (Section 2.1), and then discusses modern, lightweight fine tuning methods (Section 2.2). Next we introduce the DEAP emotion classification dataset (Section 2.3). In Section 3 we present biological motivations, our spiking bias neuron design, and the volumetric tuning method. Following which, we put forward a series of experiments in Section 4. Finally, we review some limitations and directions for future work in Section 5.

2 Background

2.1 Spiking Neural Networks

Spiking Neural Networks (SNNs) aim to mimic some of the functionality and capabilities of biological neural networks, but there exist numerous levels of biological abstraction and various classes of SNNs. Generally, however, SNNs process signals in the form of a spike train, and the neurons within them only fire when the input signal exceeds a certain threshold. This design can yield much more efficient networks than traditional non-spiking networks as the stored activations are binary. Recently, specialized neuromorphic accelerators have been developed which offer incredibly fast and low-power simulation of SNNs such as Intel's Loihi (Davies et al. 2018; Orchard et al. 2021), IBM's TrueNorth (Akopyan et al. 2015) and BrainChip's Akida (Posey 2022). (Carpegna, Savino, and Carlo 2024) describe the modern SNN space in detail, but we will focus on feed-forward, fully-connected SNNs utilizing Leaky Integrate-and-Fire neurons.

Surrogate Gradient Descent Gradient descent is an incredibly powerful algorithm which minimizes a neural network's loss or error over time. The algorithm accomplishes this by computing partial derivatives of different parts of the network. Therefore, using gradient descent requires a network be differentiable.

Spiking neural networks, however, utilize a non-differentiable spiking activation function. This activation function is often described by the Heaviside step function. A popular technique to leverage some of the power of gradient descent to train SNNs is to supplement the Heaviside step function with a differentiable surrogate function. The sigmoid, and variations of it, are common surrogate functions used to train SNNs.

We leverage the work presented by (Eshraghian et al. 2023) and we utilize their implementation of backpropagation through time (BPTT) to compute the error gradient using the fast sigmoid function as the Heaviside surrogate. Their approach to computing the gradient is similar to backpropagation in recurrent neural networks. BPTT essentially involves summing the loss at each time step for a particular input, and is described mathematically as,

$$\frac{\partial L}{\partial W} = \sum_t \frac{\partial L[t]}{\partial W} = \sum_t \sum_{s \leq t} \frac{\partial L[t]}{\partial W} \frac{\partial W[s]}{\partial W} \tag{1}$$

where $W[s]$ represents the application of weight W at step s, and t represents a discrete time step. Note that in the case of computing bias weights, W is in fact a vector instead of a matrix. Further explanation of the BPTT procedure is described by (Eshraghian et al. 2023).

Leaky Integrate-and-Fire Neurons Unlike the artificial neuron model utilized in modern deep networks often consisting of a ReLU activation function on the summed input, Leaky Integrate-and-Fire (LIF) neurons integrate input over time with leakage. LIF neurons describe this integrated input using an internal voltage or membrane potential. When the membrane potential exceeds some threshold, the LIF neuron emits a spike. The dynamics of the membrane potential of a LIF neuron are defined as,

$$U[t] = \beta U[t-1] + WX[t] + S_{out}[t-1]\theta \tag{2}$$

where $U[t]$ is the membrane potential of the neuron at time t, β is the decay rate, $WX[t]$ describes the input, and $S_{out}[t-1]\theta$ subtracts the threshold θ from the membrane potential if the neuron spiked at time $t-1$. $S_{out} \in \{0, 1\}$ so this term only affects the membrane potential when a spike occurred at the previous time step. These dynamics are described further in (Eshraghian et al. 2023).

The Role of Biases in SNNs Biases play an important role in training traditional feed forward neural networks as they allow a neuron to shift its activation function by a constant to better fit to data (Mitchell 1980). In SNNs, activation functions are modeled as step functions which output a 1 when the input exceeds some threshold. Biases in SNNs essentially scale this threshold by addition which can also be described as modulating the firing threshold of a spiking neuron. This mechanism is reminiscent of the role of neurotransmitters, like dopamine, in modulating the sensitivity of neurons to incoming signals. Traditional bias inputs in SNNs are active at every time step, and are sometimes modeled on hardware as a adjustable constant current.

Besides adding bias input to a spiking neuron, there exist other methods to modulate a spiking neuron's firing threshold. Some of these methods include, spike-frequency dependent dynamic thresholds (Diehl and Cook 2015), intrinsic plasticity (Weerasinghe et al. 2023), and dynamic energy-temporal thresholds (Ding et al. 2022). Other previous works have also explored incorporating bias inputs to modulate firing thresholds, such as (Lin et al. 2023) who constrained bias addition to ensure that membrane potential does not increase in the absence of non-bias input.

2.2 Low Rank Adaptations for Fine Tuning

Fine tuning pretrained models is popular for both large and small models, as it allows adapting a pretrained model to learn a specific task on a potentially small set of data. A pretrained model can be directly fine tuned by continuing to optimize the pretrained weights for a new task. This approach, however, changes and in some sense disregards the general knowledge the model learned in pretraining. To avoid this, methods such as bias tuning and low rank adaptations fine tune parameters which adapt the pretrained model to the new task.

Previous works have explored fine tuning the biases of a pretrained model as a means for a lightweight fine tuning method on larger image classification and large language models (LLMs) (Barton et al. 2022; Zaken, Ravfogel, and Goldberg 2022). (Barton et al. 2022) also present an analog chip design with bias, or neuromodulatory, tuning capabilities to solve a simple XOR task. Both works show success in fine tuning a large pretrained model, but do not explore this parameter-efficient tuning method on smaller spiking networks learning complex tasks such as emotion classification.

Another work, presented by (Hu et al. 2021), builds on this foundation and injects trainable rank decomposition matrices into LLMs such as GPT and BERT models, which can be exclusively used for fine tuning. Different from (Barton et al. 2022) and (Zaken, Ravfogel, and Goldberg 2022), (Hu et al. 2021) fine tune rank decomposition matrices rather than bias vectors which increase the number of tuned parameters, but still does not change the knowledge learned by the pretrained model.

In this work, we explore the effectiveness of bias tuning to tune a small SNN on a complex task, and we present a novel, biologically-inspired, design of spiking bias neurons which are stimulated via a trainable input projection. Furthermore, we demonstrate a novel bias tuning variant, termed volumetric tuning, which achieves higher performance in fine tuning a SNN on a complex task.

2.3 Emotion Classification Using DEAP Dataset

Spiking neural networks have promising potential applications to brain-computer interface (BCI) tasks because of their capability to run on low-power neuromorphic chips. One interesting BCI task that has been utilized in evaluating SNNs is the Database of Emotion Analysis using Physiological Signals (DEAP) (Koelstra et al. 2012). DEAP is a collection of electroencephalogram (EEG) signals recorded from 32 human subjects while the subjects were stimulated by 40, 1-minute snippets, of music videos. After each participant viewed a 1-minute video he/she then rated each video by giving it an arousal, valence, liking and dominance score on a 1-9 scale. The EEG signals were collected by 32 separate electrodes arranged on a subject's scalp.

After data collection, (Koelstra et al. 2012) preprocessed the EEG signals to use with classification or regression algorithms. Specifically, they downsample the data to 128Hz, remove EOG artefacts, apply a bandpass frequency filter from 4-45Hz, segment the data into 60 second trials, and reorder the EEG channels and trials to standardize them. Following this preprocessing stage, several researchers have utilized DEAP to evaluate the performance of various SNN architectures and training algorithms.

In many previous works, emotion classification is simplified into a binary valence classification (low vs. high valence) because of the noisy nature and limited features of EEG signals (Hasan et al. 2021; Marjit, Talukdar, and Hazarika 2021; Luo et al. 2020; Tripathi et al. 2017; Xu et al. 2024; Singh, Shaw, and Patra 2023). We show two recordings labeled low and high valence in Figure 1. Some of these works also increase the difficulty of the task by adding more classes. Splitting the dataset into binary valence classification also enables neural network-based approaches to learn with limited and imbalanced data. Even despite this split, the overall dataset is still slightly imbalanced and is more imbalanced when considering data from a single participant as shown in Table 1.

Additionally, feature extraction is an essential part of classifying these complex EEG signals. Popular feature extraction methods include Fast Fourier Transform, Discrete Wavelet Transform, and creating feature vectors from statistical measurements (Hasan et al. 2021; Marjit, Talukdar, and Hazarika 2021; Luo et al. 2020; Khateeb, Anwar, and Alnowami 2021). These feature extraction methods aim to decompose the EEG signal across time, frequency and wavelet domains. Finally, some works choose to only utilize a subset of the 32 recorded EEG channels to reduce noise and input to the model (Marjit, Talukdar, and Hazarika 2021; Khateeb, Anwar, and Alnowami 2021).

3 Methodology

3.1 Biological Motivations

Dopamine as a Neuromodulator Similar to previous works, we draw inspiration from discoveries in neuroscience to motivate our method. Specifically, we focus on the role of dopamine as a neuromodulator in the brain. Dopamine can alter the excitability and synaptic transmission of neurons without directly causing them to fire. This is accomplished via the activation of dopamine receptors located on the surface of neurons in various brain regions. When dopamine binds to these receptors it triggers intracellular signaling cascades that can modify the neuron's response to other inputs such as glutamate or GABA (Yorgason, Zeppenfeld, and Williams 2017; Beaulieu and Gainetdinov 2011; Depue and Collins 1999; Frank 2005; Stoof and Kebabian 1984; Reiner and Levitz 2018). Dopamine affects neurons by a process termed volume transmission, where a group or volume of neurons are influenced by a concentration of dopamine (Clements 1996; Agnati et al. 1995). In our work, we model this modulation of neural circuits via biases in an artificial spiking neural network, which modulate the incoming signal either positively or negatively.

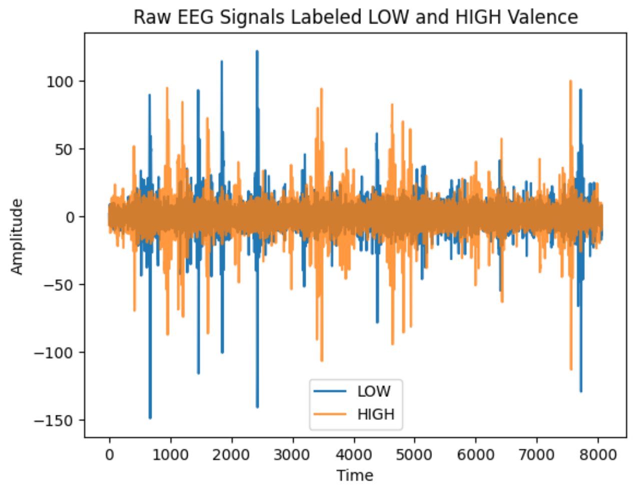

Figure 1: Two EEG recordings from participant 3 on channel 0 labeled as LOW and HIGH valence.

Stimulation of Dopamine Neurons Dopamine neurons, primarily located in the ventral tegmental area (VTA) and substantia nigra, are stimulated by a wide range of inputs from various brain regions. These inputs convey different types of information, such as sensory stimuli, reward signals, and cognitive and emotional states. The integration of these inputs to dopamine neurons plays a crucial role in modulating downstream neural circuits.

One significant input to dopamine neurons comes from brain regions involved in processing rewards. When a reward is encountered, neurons from these regions increase their firing leading to a corresponding increase in the activity of dopamine neurons in the VTA. Dopamine neurons also receive inputs from the prefrontal cortex where higher-order cognitive functions take place. The prefrontal cortex sends projections to the VTA, providing information about the current cognitive state of the organism. Additionally, dopamine neurons respond to salient sensory stimuli like sounds, visuals or tactile sensations (Yorgason, Zeppenfeld,

Table 1: DEAP dataset class distribution for each human participant's data. The number of samples of LOW and HIGH valence are shown with their percentage.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LOW	21	18	18	24	16	10	12	18	20	20	16	19	23	20	20	25
HIGH	19	22	22	16	24	30	28	22	20	20	24	21	17	20	20	15
LOW %	0.53	0.45	0.45	0.60	0.40	0.25	0.30	0.45	0.50	0.50	0.40	0.48	0.58	0.50	0.50	0.63
HIGH %	0.48	0.55	0.55	0.40	0.60	0.75	0.70	0.55	0.50	0.50	0.60	0.53	0.43	0.50	0.50	0.38

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	overall
LOW	18	16	17	17	19	22	14	22	21	14	10	15	17	13	17	20	572
HIGH	22	24	23	23	21	18	26	18	19	26	30	25	23	27	23	20	708
LOW %	0.45	0.40	0.43	0.43	0.48	0.55	0.35	0.55	0.53	0.35	0.25	0.38	0.43	0.33	0.43	0.50	0.45
HIGH %	0.55	0.60	0.58	0.58	0.53	0.45	0.65	0.45	0.48	0.65	0.75	0.63	0.58	0.68	0.58	0.50	0.55

and Williams 2017; Beaulieu and Gainetdinov 2011; Depue and Collins 1999; Frank 2005; Stoof and Kebabian 1984; Reiner and Levitz 2018).

We model this stimulus in our bias neurons by both updating the weights connecting the biases to the rest of the network via a global error signal, and by projecting the model input directly to the spiking bias neurons. In this design, bias neurons are effectively integrating information about the current state of the network and the desired output. We do acknowledge that biological dopamine neuromodulation is a highly complex mechanism and our design and modeling is a simplification of this biological process. Nevertheless, we hypothesize that our design has potential to exhibit more adaptive learning, similar to how dopamine modulates learning in the brain.

Temporal Dynamics of Dopamine Signaling When dopamine neurons fire, they release dopamine into the synaptic cleft which then binds to dopamine receptors on the postsynaptic cleft. The effect of this dopamine is not instantaneous, but rather persists for some time, influencing the postsynaptic neuron's activity. Moreover, the clearance of dopamine from the synaptic cleft is not immediate, but gradual. Dopamine transporters actively remove dopamine from the synapse, leading to a gradual decay in dopamine concentration over time (Yorgason, Zeppenfeld, and Williams 2017; Beaulieu and Gainetdinov 2011; Depue and Collins 1999; Frank 2005; Stoof and Kebabian 1984; Reiner and Levitz 2018). We model this decay via a leaking bias current which increases in magnitude only when the connected spiking bias neuron fires. Otherwise the bias current decays exponentially over time.

3.2 Spiking Bias Neurons

Overview Inspired by the biological role of dopamine neurons and to further develop biologically-inspired fine tuning algorithms, we model spiking bias neurons as LIF neurons with a leaking current over time. Unlike traditional biases in non-spiking artificial neural networks, these spiking biases only fire at a specific time step when their internal voltage exceeds a predefined threshold. When a bias spike is emitted, the bias weight is added to the internal voltage of the connected neuron(s). This bias current to connected neuron(s) then decays exponentially approximating the decay in dopamine concentration over time. This design is shown simply in Figure 2.

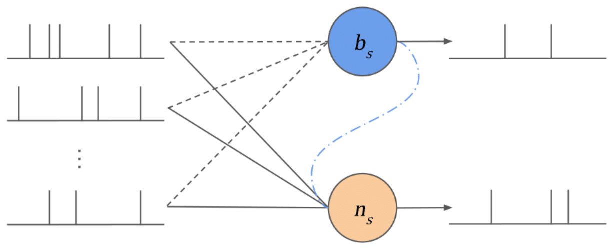

Figure 2: A high-level diagram of the spiking bias setup. The input spike train is fed to both the spiking bias neurons (in blue) and the regular LIF neurons (in orange). The spike trains are represented by the horizontal line with sparse vertical lines corresponding to spikes at discrete time steps. During network training the weights (solid lines) connecting to the regular LIF neurons can be optimized, and the weights (dashed lines) connecting to the spiking bias neurons can be optionally held fixed. The blue dashed line from the spiking bias neuron b_s connecting to n_s is the leaking bias current. The bias weights are also optimized.

Stimulating the Spiking Biases In order to increase the internal voltage of the bias neurons at a certain time step, the bias neurons are stimulated by the input spike train. As previously mentioned this is biologically motivated. Practically, a randomly-initialized matrix of weights connects each layer or group of bias neurons to the input. We explore leaving these input to bias matrices static and tuning them using surrogate gradient descent.

Leaking Bias Current Once a bias neuron, modeled by a LIF neuron, emits a spike it then supplies a bias current of the bias weight to the connected neuron(s). This current then decays across time steps as described in Equation 3. The variables are defined as: α is the decay rate, C is the leaking bias current, W_b represents the bias weights, and $S_{out_b}[t]$ is the output spike train for the bias neuron. The connected neuron(s) dynamics' are then described by Equation 4. Not only does this function mimic the decaying concentration of dopamine neurotransmitters, but it also allows a single

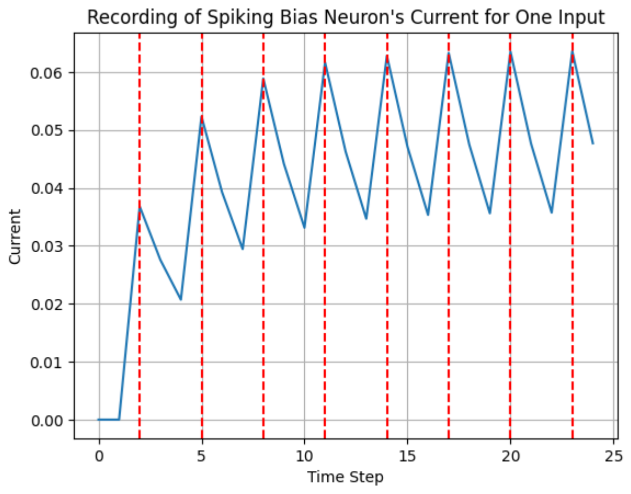

Figure 3: Recording of leaking bias current for a single neuron on one input. The recording is done at the beginning of training. The current is shown as the blue line, and spikes are shown as the vertical red lines. In this example, the bias weight is positive, but it could also be negative.

bias neuron to stimulate its connected neurons(s) across time steps, without needing to spike at every time step.

$$C[t] = \alpha C[t-1] + W_b S_{out_b}[t] \tag{3}$$

$$U[t] = \beta U[t-1] + WX[t] + \alpha C + S_{out}[t-1]\theta \tag{4}$$

A recording of the leaking current of a spiking bias neuron is shown in Figure 3. In this figure, a single bias neuron's leaking current is recorded for one input that is simulated for 25 discrete time steps. From this recording it is clear how a spiking bias neuron can affect its connected neuron(s) over multiple time steps without needing fire at each time step.

3.3 Volumetric Spiking Biases

Drawing further inspiration from biology, we also explore the volumetric property of dopamine neuromodulation. When dopamine is released its effect is often not exclusively localized to a single neuron, but rather groups or populations of neurons (Clements 1996; Agnati et al. 1995). This process is called volume transmission. Our design places more responsibility in each spiking bias neuron to distribute global error signal to groups of neurons within the network. While biologically motivated, this design can also reduce the total number of spiking bias neurons necessary and may lead to localized learning regions within an artificial network. Localized learning regions can occur in biology as a consequence of various types neurotransmitters (Amunts et al. 2010; Amunts and Zilles 2015; Huang and Reichardt 2001). We utilize our custom spiking bias neurons to define volumetric groups of neurons across layers and measure the performance of our network in Section 4.6.

4 Experiments and Analysis

4.1 DEAP Preprocessing and Feature Extraction

As discussed in Section 2.3, DEAP is a popular and challenging emotion classification task for many types of neural networks. Like many previous works, we also reduce emotion classification to binary valence classification. We acknowledge the dataset imbalance, and instead introducing a biased amount of synthesized data, we allow the training datasets to be imbalanced while the test set is forced to be balanced. We enforce this construction of the test set by repeatedly sampling the dataset until the classes are balanced within 10%. Observe that this further increases the difficultly of this task. To cope with this increased difficulty and limited data, we do augment each piece of data (regardless of class) with three different augmentations. They are: (1) a random 1D rotation or shuffling, (2) a horizontal flip, and (3) adding random Gaussian noise. These augmentations are shown in Figure 4. We also apply normalization to the raw signals. We found that these augmentations yielded much more generalized models in preliminary experiments.

Next, we utilize the Fast-Fourier Transform to extract relevant features for our network with spiking biases, as done by many previous works such as (Hasan et al. 2021). For each participant's recorded data, we decompose the raw EEG signals into predefined frequency bands. We define these frequency bands as theta (4-7) Hz, alpha (8-13) Hz, low beta (14-20) Hz, high beta (21-29) Hz and gamma (30-47) Hz as done in (Hasan et al. 2021). Next, we get the mean power of each of these frequency bands across all of a single participant's data. We define a spike threshold at this mean power to define a 5-dimensional spiking array for each channel in each instance. We utilize 32 channels for each instance so our final input to the SNN is a spiking array of length 160. Figure 5 shows this preprocessing and feature extraction process.

4.2 Network Architecture

We choose a simple three layer SNN using LIF neurons. The network is fully connected by default, and we choose the network layer sizes to be [400,800,100]. These decisions were motivated by preliminary experiments. Since our preprocessed input data is already in the form of a spike train, we can directly multiply the input signal by the first weight matrix in the network. Since we are predicting low and high valence, the final layer assigns the first 50 neurons to the label for low valence (0) and the second 50 neurons to the label for high valence (1). This is known as a population code and is common to use with SNNs (Eshraghian et al. 2023). The base network architecture does not include biases on each layer, rather we experiment with adding both traditional always-on and spiking bias neurons. The network architecture with spiking bias neurons is shown in Figure 6.

4.3 Experimental Setup

While many previous works have solved the DEAP binary valence classification, few have solved the task using leave one out cross validation (LOOCV). In this case, we refer to the **one** as all of the data of a single participant. For example,

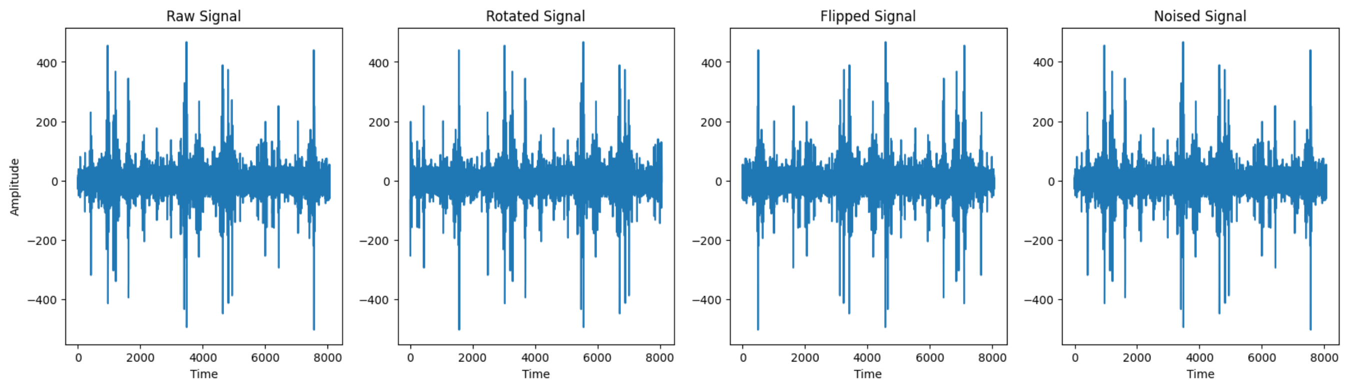

Figure 4: Visualization of the augmentation techniques applied on a single EEG channel recording from the video trial from participant 4. The label for this instance is high valence. We apply a random rotation of the 1D time series data which essentially shifts the data a random number of indices. To flip the signal, we simply reverse the indices of the time series, and finally for the noised signal, we add noise sampled from a Gaussian distribution with mean 0 and standard deviation of 0.1.

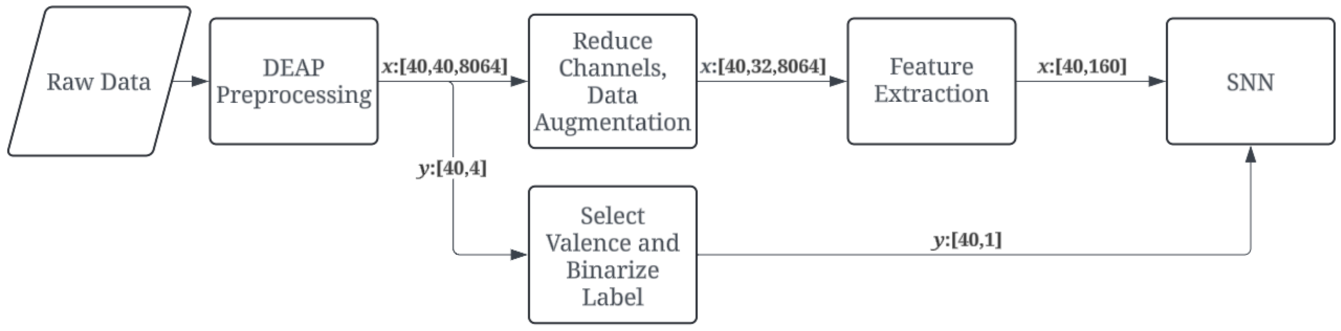

Figure 5: Data preprocessing and feature extraction method for one participant's data. The DEAP Preprocessing stage is described in detail in Secion 2.3. The shape of the data after processes is shown on the arrows. x, y refers to the input and label respectively. As explained in Section 2.3, each participant's data contains recordings from 40 trials and these recordings are preprocessed to have a length of 8064.

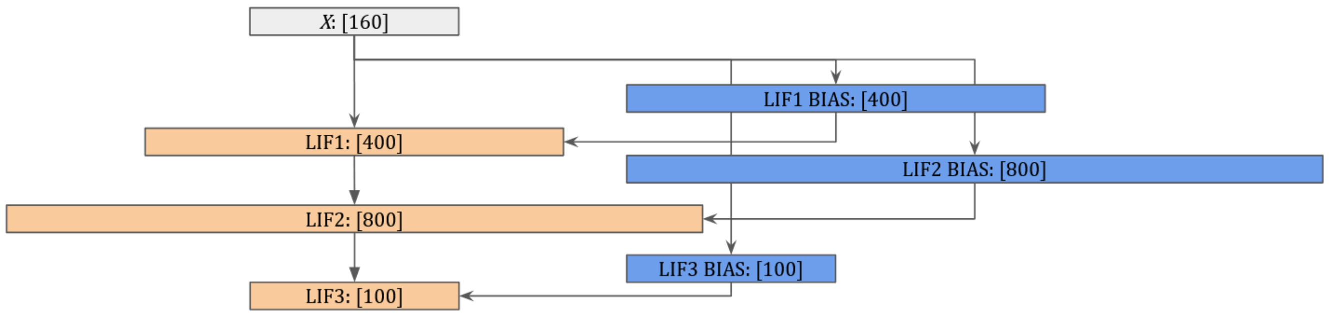

Figure 6: SNN architecture with spiking bias neurons. X describes the input. The arrows represent data passing between layers of neurons. Arrows between non-bias layers, and between the input and any layer are fully connected. The arrows between bias layers and non-bias layers are connected by a vector of bias weights. The sizes of each layer of LIF neurons are listed as well.

we can train our SNN using data from participants 1-31, but then test or fine tune on only data from participant 32. Part of the reason that few works have explored this validation technique is because EEG recordings across human participants and their corresponding recorded valence scores can vary significantly. Despite this challenge, this experimental setup has valid downstream applications, where a low-power on-chip network may be trained offline on an assortment of EEG data from different people, but it's evaluation on-chip will be specific to only the human's EEG signals in which it's deployed. Thus, exploring low-power fine tuning methods to fine tune the more general model on a specific human's EEG data is a worthy pursuit in the brain-computer interaction (BCI) field.

In addition to LOOCV, we also run five trials for each of our experiments to account for variation in network initialization and dataset ordering. For each trial, unless stated otherwise in subsequent sections, we pretrain the fully connected weight matrices on 31 participants' data for 15 epochs. Recall that each data instance is augmented by 3 augmentations to increase the size of our training dataset. Following this pretraining phase, the model is then fine tuned on the left out participant's data by training some number of the bias parameters and by leaving the standard weight matrices static. This not only limits the number of parameters needed to fine tune the network, but it also preserves the knowledge learned by the network in the pretraining phase. We fine tune the network for a maximum of 500 epochs, but we select the best validation/test accuracy and loss as our result. While 500 epochs is significantly more epochs than used in pretraining, the model actually receives nearly the same amount of total input data when pretrained for 15 epochs since the training dataset is 31 times larger. Additionally, we empirically find that this high number of epochs is necessary for fine tuning the bias parameters to converge.

We present the input spike train of features data to the SNN for $T = 25$ discrete time steps. The input remains the same on each time step. This is not uncommon approach when presenting time-invariant data to a SNN. Note that while the EEG signal was time-varying before feature extraction, it is not anymore. We utilize the Adam optimizer with a learning rate of 0.0001, and apply L1 spike sparsity regulation with a rate sparsity value of 0.001 to ensure are networks have efficient internal spiking behavior. We use a batch size of 32 and split our training/tuning dataset into train/tune and test sets using a 80-20 split.

4.4 Applying Bias Tuning on DEAP

In this first set of experiments we first examine how effective bias, or neuromodulatory, tuning is on a small network solving a complex task. Following our experimental LOOCV setup as described in Section 4.3, we find that by fine tuning only the bias parameters of each layer in our SNN we can achieve decent binary valence classification. In this specific network, 465.3K weights and biases are optimized during pretraining and only 1.3K parameters are optimized during fine tuning.

We run this experiment for both traditional biases and spiking biases stimulated by the input. For this experiment with spiking biases we do not optimize the input matrices to the biases during pretraining or fine tuning. Thus, the number of optimized parameters during pretraining and fine tuning are the same between the two types of biases. Our LOOCV results are shown in Table 2 and Figure 7. We omit the pretraining results for brevity, but we list that the mean pretraining test accuracy for with both types of biases is always between 96-98%.

Table 2: LOOCV fine tuning results when using bias/neuromodulatory tuning with both spiking and traditional biases. Test ID refers the participant's data used for fine tuning. The mean and standard deviation of accuracy across 5 trials are displayed for both our spiking biases, denoted by **(s)**, and for traditional biases. The final column lists the difference in the mean accuracy for each test participant. Positive differences showing the improvement of spiking biases are bolded. Finally, the mean across all participants is listed in the last row.

test ID	mean acc (s)	std acc (s)	mean acc	std acc	mean acc diff
0	0.86250	0.025000	0.80625	0.050000	**0.05625**
1	0.71875	0.044194	0.75000	0.044194	-0.03125
2	0.75625	0.053765	0.83125	0.061237	-0.07500
3	0.86875	0.072349	0.88125	0.036443	-0.01250
4	0.89375	0.031869	0.78125	0.073951	**0.11250**
5	0.85625	0.050775	0.76875	0.089704	**0.08750**
6	0.81250	0.088388	0.64375	0.080526	**0.16875**
7	0.91875	0.015309	0.84375	0.027951	**0.07500**
8	0.88750	0.064348	0.85000	0.053765	**0.03750**
9	0.92500	0.046771	0.85000	0.072349	**0.07500**
10	0.86250	0.080526	0.70625	0.133463	**0.15625**
11	0.78750	0.060596	0.62500	0.044194	**0.16250**
12	0.92500	0.057960	0.95000	0.050775	-0.02500
13	0.91875	0.050775	0.81875	0.036443	**0.10000**
14	0.77500	0.036443	0.85000	0.053765	-0.07500
15	0.86250	0.064348	0.75625	0.045928	**0.10625**
16	0.86250	0.046771	0.82500	0.119570	**0.03750**
17	0.90625	0.079057	0.86250	0.064348	**0.04375**
18	0.68750	0.086150	0.62500	0.094786	**0.06250**
19	0.93125	0.069597	0.93125	0.023385	0.00000
20	0.85625	0.072887	0.78750	0.072349	**0.06875**
21	0.87500	0.081490	0.84375	0.068465	**0.03125**
22	0.83750	0.023385	0.76875	0.042390	**0.06875**
23	0.83750	0.063738	0.82500	0.050775	**0.01250**
24	0.76250	0.046771	0.68125	0.077560	**0.08125**
25	0.85625	0.025000	0.86250	0.057960	-0.00625
26	0.80625	0.053765	0.73750	0.105697	**0.06875**
27	0.88125	0.072349	0.71875	0.079057	**0.16250**
28	0.79375	0.050775	0.78750	0.095607	**0.00625**
29	0.81250	0.081490	0.73750	0.080526	**0.07500**
30	0.83750	0.063738	0.76875	0.098027	**0.06875**
31	0.97500	0.030619	0.86875	0.053765	**0.10625**
Mean	0.848437	0.055969	0.791992	0.066842	0.056445

From these results, we acknowledge that fine tuning on the test participant does not reach the same test/validation accuracy obtained during pretraining. This shows the difficulty of the task and also difference in EEG recordings across people. This is further seen in the variation across test participants. We also see large standard deviations across trials for many of the participants which is explained by the importance of network initialization which is magnified for smaller and spiking networks. Without any level of fine tuning, though, the model only predicts randomly, so we do see some capacity of this small parameter set to tune the model. We do see a slight performance increase when using spiking biases, but since this improvement is within one standard de-

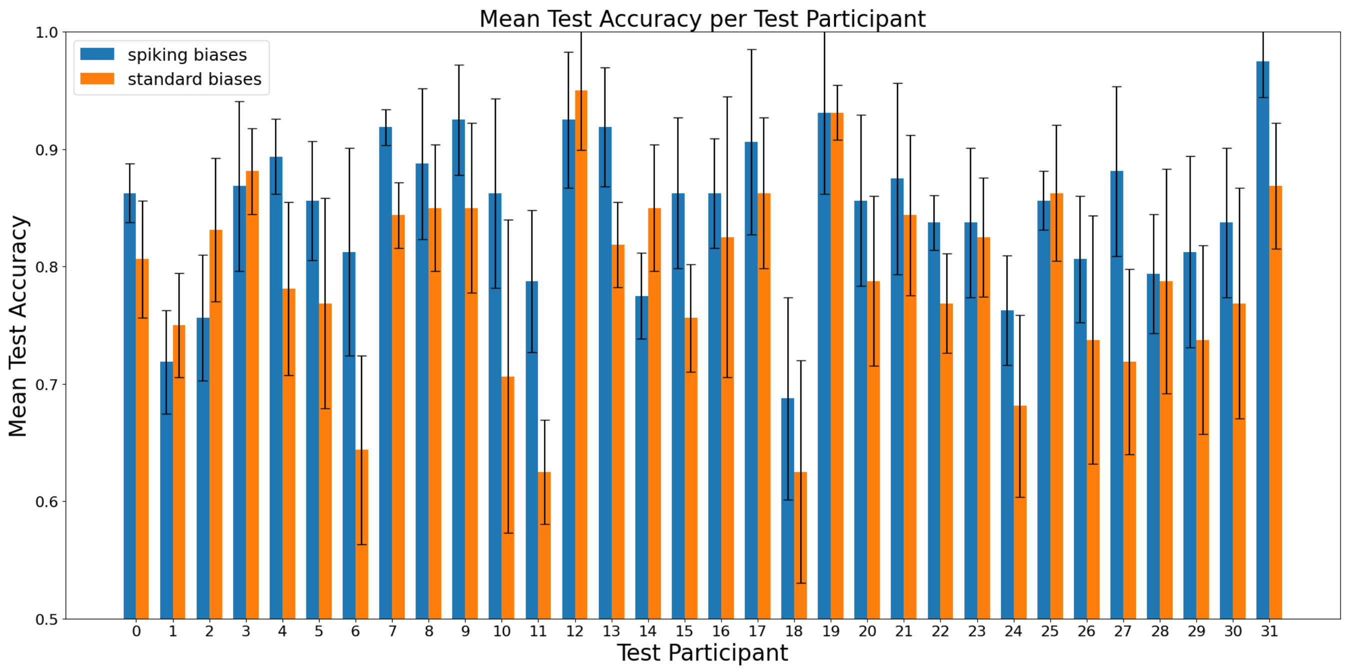

Figure 7: LOOCV fine tuning test results when using bias tuning with both spiking and traditional biases. The fine tuning test set is balanced and is 20% of the total fine tuning dataset. The mean test accuracy for spiking biases across all participants is $84.84\% \mp 5.60$, and the mean accuracy for traditional biases across all participants is $79.20\% \mp 6.68$. Error bars representing ∓ 1 standard deviation from the mean are shown as well.

viation from the traditional biases we do not consider it statistically significant across all participants. However, we do see gaps in error bars between spiking and traditional biases for several specific participants, which does not guarantee statistical significance but may suggest it.

4.5 Tuning the Input Matrices of Spiking Biases

While traditional and spiking biases performed similarly in the larger LOOCV fine tuning experiment, our spiking biases design provides the opportunity to include additional parameters when fine tuning to boost their performance. As shown in Figure 2, each spiking bias neuron is connected to the input via a randomly initialized fully connected matrix. In this set of experiments, we train some of these input matrices to the spiking biases. This significantly increases the number of optimized parameters during fine tuning, but it does not change the existing knowledge in the standard network layers learned in pretraining. We also apply different sparsity values to these input matrices to reduce the parameter count dramatically.

For the subsequent experiments we only gather results of LOOCV for the first 12 participants for the sake of computation time. First, we optimize all input matrices to the biases along with the bias weights during tuning. With the addition of thousands of weight parameters, we fine tune to a much higher accuracy as shown in Table 3. In this experiment we do not pretrain the input matrices to the biases such that the number of optimized parameters during offline training does not increase. Instead we tune these parameters only on the left out participant's data and witness 100% test accuracy on all but one of the participants. This is not surprising as our total optimized parameter count when tuning is the same amount as when training but using different parameters. Thus, we still freeze the pretrained and generalized model, but adapt the model via the spiking biases to perform well on new data from a new human subject. This is valuable in scenarios where a fine tuned model may need to be fine tuned on different tasks or data from a generalized model. In the BCI space, this may occur when calibrating, or fine tuning, a pretrained model for a specific individual, only to then fine tune the model for another individual. Since our input to biases matrices are not pretrained, we can fine tune the pretrained model any number of times for new human data by simply tuning the spiking bias parameters.

Next, we introduce sparsity into the input matrices to the biases to reduce the number of optimized parameters during fine tuning. We make these input matrices to the biases 90% sparse, and find that we can still achieve decent performance with much fewer parameters. Concretely we reduce the number of optimized parameters during tuning from 465K to 48K. These results are shown in Table 4. Similar to the result for test participant 2 in Table 3, performance suffered the most for this same participant with sparse input matrices. One challenge with solving the DEAP dataset is its limited amount of data. Despite applying augmentations, the total number of data instances used during pretraining is less than 5K. We suspect that with additional data used in pretraining, we could obtain a more generalized model

test ID	0	1	2	3	4	5	6	7	8	9	10	11
mean	1.000	1.000	0.988	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
std	0.000	0.000	0.015	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table 3: Mean and standard deviation of 5 trials LOOCV results on the first 12 participant when tuning the input matrices to the spiking biases.

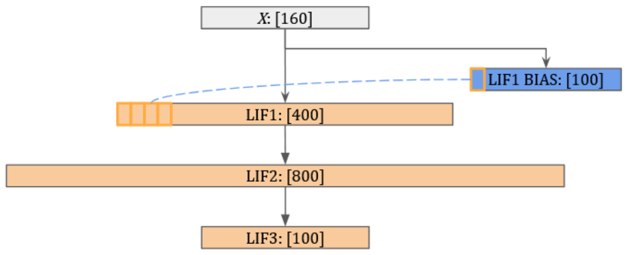

Figure 8: Our experimental setup for evaluating volumetric tuning on the original network architecture. Each spiking bias neuron is connected to four spiking neurons in LIF1. The connection between the first spiking bias neuron is shown by the blue dashed line, and is a leaking current as in previous experiments.

which would increase the performance for test participant 2.

4.6 Volumetric Tuning for Spiking Biases

Now that we have shown the value of both bias tuning and tuning the input to our spiking biases, we explore another biological trait of dopamine pertaining to the diffusion of dopamine across a volume of neurons. We modify our network such that we define spiking bias neurons numbering only $\frac{1}{10}$ of the neurons in the first layer. Thus, each spiking bias neuron is connected, by the same bias weight and current, to 10 neurons in the first layer of the network. Additionally, drawing on our results in the previous section, we define the input matrix to these 40 spiking bias neurons to be 85% sparse. This setup is visualized in Figure 8. In this construction, we have exactly 1000 tunable parameters, which is 300 less than we use in bias tuning using either traditional or spiking biases. In a set of intermediate experiments we discover that we can achieve similar performance using volumetric tuning when only tuning the first layer biases. We find this volumetric tuning, incredibly effective, and we present our results in Table 5. The volumetrically tuned model significantly outperforms our previous results shown in Table 2.

To ensure that volumetric spiking bias neurons are the cause for such high performance, we conduct an ablation study where we utilize 1000 traditional bias parameters in the first layer of our network. To make a fair comparison, we also ensure that this ablated model has nearly the same number of total parameters in the model. Specifically, our abalated model features an architecture of 3 layers with sizes [1000,277,98]. Using this architecture we obtain a difference of only 221 optimized parameters during pretraining between our ablated model and our model tuned by volumetric tuning; where the ablated model has 221 more parameters. Recall, that spiking biases and their input matrix are not used during pretraining. During testing the number of optimized parameters is exactly 1000 in both cases. We also ensure that the ablated model still achieves the same test accuracy during pretraining as our original network. The results of our ablated model are shown in Table 6.

From these results, we see that volumetric tuning is in fact more effective on all but one of the twelve participants when fine tuning on EEG data from a new participant. For participant 1 we see a slight drop in performance, but it is not significant when consider the standard deviation of the ablated result for this participant. Furthermore, we see that the mean performance across all participants is significantly higher. Interestingly, we also see a smaller deviation across all participants when using volumetric tuning which shows that it is less volatile and less dependent on the exact pretrained network weights.

5 Limitations and Future Work

We have presented biologically-inspired spiking bias neurons which model some of the stimulus and output mechanisms of dopamine neurons in the VTA. While our results showed promising results, the path to utilize spiking bias tuning (volumetric or otherwise) on on-chip networks is not short. The next step in applying our method to low-power neuromorphic devices depends largely on the device specifications, but we caution researchers that creating many spiking bias neurons may not be power efficient compared to the implementation of traditional biases on hardware. Non-spiking biases can be implemented via a variable current directly to each spiking neuron in the network. On the contrary, our spiking biases require that the biases spike which requires additional hardware neurons. This problem is somewhat reduced by our development of volumetric tuning where the number of spiking bias neuron drops significantly. However, there does exist certain design concerns with our method on physical hardware that require further exploration.

Another area for future work revolves around the use of the input matrix to a group of bias neurons. From our course of experiments we determined to use gradient descent on some elements within those matrices. Conversely, a different weight adjustment algorithm that is more lightweight could be utilized. One such example is Spike Time Dependent Plasticity (STDP) which is an unsupervised spike-frequency based algorithm that is popular in the SNN field. Several recent works exist that consider combining backpropagation in SNNs with STDP (Gong et al. 2024; Furuya and Ohkubo 2021; Bengio et al. 2017) that may provide insight in adjust-

test ID	0	1	2	3	4	5	6	7	8	9	10	11
mean	0.993	1.000	0.825	0.938	1.000	0.988	1.000	1.000	1.000	1.000	1.000	1.000
std	0.013	0.000	0.025	0.000	0.000	0.025	0.000	0.000	0.000	0.000	0.000	0.000

Table 4: Mean and standard deviation of 5 trials LOOCV results on the first 12 participant when tuning the 90% sparse input matrices to the spiking biases.

Table 5: Original network architecture using volumetric tuning to fine tune on each left out participant. The mean test accuracy and standard deviation over 5 trials are shown, with the mean of all participants are bolded and shown in the final column.

test ID	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**Mean**
mean	0.994	0.819	0.894	0.956	1.000	0.975	0.931	0.981	0.938	0.994	0.988	0.831	**0.942**
std	0.012	0.023	0.025	0.015	0.000	0.031	0.061	0.025	0.034	0.012	0.025	0.054	**0.027**

Table 6: Ablated network using bias tuning to fine tune on each left out participant. The mean test accuracy and standard deviation over 5 trials are shown, with the mean of all participants are bolded and shown in the final column.

test ID	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**Mean**
mean	0.806	0.850	0.888	0.900	0.788	0.763	0.788	0.950	0.938	0.831	0.875	0.800	**0.848**
std	0.061	0.041	0.015	0.046	0.127	0.051	0.130	0.032	0.020	0.058	0.044	0.106	**0.059**

ing the input matrices to the spiking biases via STDP while still using BPTT for the bias weights.

6 Conclusions

In this work we have designed biologically-inspired spiking bias neurons which performs similarly in fine tuning a small pretrained SNN to classify valence from EEG signals from a new subject. Our spiking bias neuron design mimics some of stimuli, output mechanisms and structures of biological dopamine neurons. We first extend previous work using bias, or neuromodulatory, tuning on larger models, to our smaller model solving binary valence classification by evaluating the method using traditional bias inputs and our spiking biases. We found that on some subjects, that our spiking biases outperformed the traditional biases without adding additional optimized parameters during tuning.

Given the construction of our spiking biases we take advantage of the stimulus mechanism, the input matrix to the bias, and show by tuning only 10% of the matrix, we can achieve much higher accuracies without changing the learned features in the pretrained model. This is not unsurprising, as this approach can be viewed as a modification of the work presented by (Hu et al. 2021). Finally, we apply a simplified form of biological volumetric transmission where we reduce the number of bias neurons for a certain layer. Each bias neuron then distributes its signal to a group of neurons rather than a single neuron. Additionally, we optimize 15% of the input weights to the spiking biases. This construction places more responsibility in each spiking bias neuron to learn to modify input signals to adapt groups of neurons similar to dopamine neurons responsibility to integrate information from multiple sources and affect a volume of neurons. Our variant of neuromodulatory tuning, termed volumetric tuning, achieves higher accuracy than neuromodulatory tuning while utilizing a fewer number of optimized parameters during fine tuning.

7 Acknowledgements

The authors would like to thank Hao Yu for his conceptual contributions to the spiking bias neuron design, and Daniel Isemonger for his contributions in reviewing biological processes.

References

Agnati, L. F.; Zoli, M.; Strömberg, I.; and Fuxe, K. 1995. Intercellular communication in the brain: wiring versus volume transmission. *Neuroscience*, 69(3): 711–726.

Akopyan, F.; Sawada, J.; Cassidy, A.; Alvarez-Icaza, R.; Arthur, J.; Merolla, P.; Imam, N.; Nakamura, Y.; Datta, P.; Nam, G.-J.; et al. 2015. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE transactions on computer-aided design of integrated circuits and systems*, 34(10): 1537–1557.

Amunts, K.; Lenzen, M.; Friederici, A. D.; Schleicher, A.; Morosan, P.; Palomero-Gallagher, N.; and Zilles, K. 2010. Broca's Region: Novel Organizational Principles and Multiple Receptor Mapping. *PLOS Biology*, 8(9): 1–16.

Amunts, K.; and Zilles, K. 2015. Architectonic Mapping of the Human Brain beyond Brodmann. *Neuron*, 88(6): 1086–1107.

Barton, T.; Yu, H.; Rogers, K.; Fulda, N.; Chiang, S.-h. W.; Yorgason, J.; and Warnick, K. F. 2022. Towards Low-Power Machine Learning Architectures Inspired by Brain Neuromodulatory Signalling. *Journal of Low Power Electronics and Applications*, 12(4).

Beaulieu, J.-M.; and Gainetdinov, R. R. 2011. The physiology, signaling, and pharmacology of dopamine receptors. *Pharmacological reviews*, 63(1): 182–217.

Bengio, Y.; Mesnard, T.; Fischer, A.; Zhang, S.; and Wu, Y. 2017. STDP-compatible approximation of backpropagation in an energy-based model. *Neural computation*, 29(3): 555–577.

Carpegna, A.; Savino, A.; and Carlo, S. D. 2024. Spiker+: a framework for the generation of efficient Spiking Neural Networks FPGA accelerators for inference at the edge. arXiv:2401.01141.

Clements, J. 1996. Transmitter timecourse in the synaptic cleft: its role in central synaptic function. *Trends in neurosciences*, 19(5): 163–171.

Davies, M.; Srinivasa, N.; Lin, T.-H.; Chinya, G.; Cao, Y.; Choday, S. H.; Dimou, G.; Joshi, P.; Imam, N.; Jain, S.; et al. 2018. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1): 82–99.

Depue, R. A.; and Collins, P. F. 1999. Neurobiology of the structure of personality: Dopamine, facilitation of incentive motivation, and extraversion. *Behavioral and brain sciences*, 22(3): 491–517.

Diehl, P.; and Cook, M. 2015. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in Computational Neuroscience*, 9.

Ding, J.; Dong, B.; Heide, F.; Ding, Y.; Zhou, Y.; Yin, B.; and Yang, X. 2022. Biologically inspired dynamic thresholds for spiking neural networks. *Advances in Neural Information Processing Systems*, 35: 6090–6103.

Eshraghian, J. K.; Ward, M.; Neftci, E.; Wang, X.; Lenz, G.; Dwivedi, G.; Bennamoun, M.; Jeong, D. S.; and Lu, W. D. 2023. Training Spiking Neural Networks Using Lessons From Deep Learning. arXiv:2109.12894.

Frank, M. J. 2005. Dynamic dopamine modulation in the basal ganglia: a neurocomputational account of cognitive deficits in medicated and nonmedicated Parkinsonism. *Journal of cognitive neuroscience*, 17(1): 51–72.

Furuya, K.; and Ohkubo, J. 2021. Semi-Supervised Learning Combining Backpropagation and STDP: STDP Enhances Learning by Backpropagation with a Small Amount of Labeled Data in a Spiking Neural Network. *Journal of the Physical Society of Japan*, 90(7): 074802.

Gong, Y.; Chen, T.; Wang, S.; Duan, S.; and Wang, L. 2024. Lightweight spiking neural network training based on spike timing dependent backpropagation. *Neurocomputing*, 570: 127059.

Hasan, M.; Rokhshana-Nishat-Anzum; Yasmin, S.; and Pias, T. S. 2021. Fine-Grained Emotion Recognition from EEG Signal Using Fast Fourier Transformation and CNN. In *2021 Joint 10th International Conference on Informatics, Electronics Vision (ICIEV) and 2021 5th International Conference on Imaging, Vision Pattern Recognition (icIVPR)*, 1–9.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2021. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685.

Huang, E. J.; and Reichardt, L. F. 2001. Neurotrophins: roles in neuronal development and function. *Annual review of neuroscience*, 24: 677–736.

Khateeb, M.; Anwar, S. M.; and Alnowami, M. 2021. Multi-Domain Feature Fusion for Emotion Classification Using DEAP Dataset. *IEEE Access*, 9: 12134–12142.

Koelstra, S.; Muehl, C.; Soleymani, M.; Lee, J.-S.; Yazdani, A.; Ebrahimi, T.; Pun, T.; Nijholt, A.; and Patras, I. 2012. DEAP: A Database for Emotion Analysis using Physiological Signals (PDF). *EEE Transactions on Affective Computing*, 3(1): 18–31.

Lin, R.; Dai, B.; Zhao, Y.; Chen, G.; and Lu, H. 2023. Constrain Bias Addition to Train Low-Latency Spiking Neural Networks. *Brain Sciences*, 13(2): 319.

Luo, Y.; Fu, Q.; Xie, J.; Qin, Y.; Wu, G.; Liu, J.; Jiang, F.; Cao, Y.; and Ding, X. 2020. EEG-based emotion classification using spiking neural networks. *IEEE Access*, 8: 46007–46016.

Marjit, S.; Talukdar, U.; and Hazarika, S. M. 2021. EEG-Based Emotion Recognition Using Genetic Algorithm Optimized Multi-Layer Perceptron. In *2021 International Symposium of Asian Control Association on Intelligent Robotics and Industrial Automation (IRIA)*, 304–309.

Mitchell, T. M. 1980. The need for biases in learning generalizations.

Orchard, G.; Frady, E. P.; Rubin, D. B. D.; Sanborn, S.; Shrestha, S. B.; Sommer, F. T.; and Davies, M. 2021. Efficient Neuromorphic Signal Processing with Loihi 2. arXiv:2111.03746.

Picard, R. W.; Vyzas, E.; and Healey, J. 2001. Toward machine emotional intelligence: Analysis of affective physiological state. *IEEE transactions on pattern analysis and machine intelligence*, 23(10): 1175–1191.

Posey, B. M. 2022. What Is the Akida Event Domain Neural Processor? *2020*.

Reiner, A.; and Levitz, J. 2018. Glutamatergic signaling in the central nervous system: ionotropic and metabotropic receptors in concert. *Neuron*, 98(6): 1080–1098.

Singh, U.; Shaw, R.; and Patra, B. K. 2023. A data augmentation and channel selection technique for grading human emotions on DEAP dataset. *Biomedical Signal Processing and Control*, 79: 104060.

Stoof, J.; and Kebabian, J. 1984. Two dopamine receptors: biochemistry, physiology and pharmacology. *Life sciences*, 35(23): 2281–2296.

Tripathi, S.; Acharya, S.; Sharma, R.; Mittal, S.; and Bhattacharya, S. 2017. Using deep and convolutional neural networks for accurate emotion classification on DEAP data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 4746–4752.

Weerasinghe, M. M. A.; Wang, G.; Whalley, J.; and Crook-Rumsey, M. 2023. Mental stress recognition on the fly using neuroplasticity spiking neural networks. *Scientific Reports*, 13(1): 14962.

Xu, F.; Pan, D.; Zheng, H.; Ouyang, Y.; Jia, Z.; and Zeng, H. 2024. EESCN: A novel spiking neural network method for EEG-based emotion recognition. *Computer Methods and Programs in Biomedicine*, 243: 107927.

Yorgason, J. T.; Zeppenfeld, D. M.; and Williams, J. T. 2017. Cholinergic interneurons underlie spontaneous dopamine release in nucleus accumbens. *Journal of Neuroscience*, 37(8): 2086–2096.

Zaken, E. B.; Ravfogel, S.; and Goldberg, Y. 2022. BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models. arXiv:2106.10199.

Chapter 5

Conclusion

This thesis has demonstrated the substantial benefits of leveraging insights from neuroscience to address persistent challenges in machine learning, particularly in the areas of parameter-efficient fine-tuning and depth-wise parallelization. By mimicking biological structures and processes, we have shown that it is possible to overcome some of the limitations inherent in conventional methods such as backpropagation. The introduction of neuromodulatory tuning has proven to be a significant advancement, allowing fine-tuning of pretrained networks with a dramatically reduced parameter count, while our exploration of abstracted gradients has introduced a novel paradigm that challenges the necessity of sequential backpropagation.

Moreover, the development of biologically-inspired spiking bias neurons and volumetric tuning has shown promise in improving the performance of neuromodulatory tuning on complex tasks like EEG emotion classification. These innovations not only address specific technical challenges but also open up exciting new possibilities for the implementation of machine learning algorithms in hardware, particularly for low-power and edge computing applications. The practical applications of this work, from tunable CMOS hardware to personalized BCI devices, underscore the potential of continued interdisciplinary research. By drawing inspiration from biological intelligence, we can push the boundaries of what is possible in artificial intelligence, paving the way for future breakthroughs in the field.

References

[1] Timothy P Lillicrap, Adam Santoro, Luke Marris, Colin J Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346, 2020.

[2] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.