Exploring Spiking Neural Networks (SNN) for Low Size, Weight, and Power (SWaP) Benefits

Trevor J. Bihl
Air Force Research Laboratory
Ohio, USA
Trevor.Bihl.2@us.af.mil

Patrick Farr
Applied Research Solutions
Ohio, USA
pfarr@appliedres.com

Gaetano Di Caterina,
Paul Kirkland,
Alex Vicente Sola,
Davide Manna
University of Strathclyde (UK)
{gaetano.di-caterina,
paul.kirkland,alex.vicente-
sola,davide.manna}
@strath.ac.uk

Jundong Liu
Ohio University
Ohio, USA
liuj1@ohio.edu

Kara Combs
Air Force Research Laboratory
Ohio, USA
Kara.Combs.1@us.af.mil

Abstract

Size, Weight, and Power (SWaP) concerns are growing as artificial intelligence (AI) use spreads in edge applications. AI algorithms, such as artificial neural networks (ANNs), have revolutionized many fields, e.g. computer vision (CV), but at a large computational/power burden. Biological intelligence is notably more computationally efficient. Neuromorphic edge processors and spiking neural networks (SNNs) aim to follow biology closer with spike-based operations resulting in sparsity and lower-SWaP operations than traditional ANNs with SNNs only "firing/spiking" when needed. Understanding the trade space of SWaP when embracing neuromorphic computing has not been studied heavily. To addresses this, we present a repeatable and scalable apples-to-apples comparison of traditional ANNs and SNNs for edge processing with demonstration on both classical and neuromorphic edge hardware. Results show that SNNs combined with neuromorphic hardware can provide comparable accuracy for CV to ANNs at 1/10th the power.

1. Introduction

Artificial Intelligence (AI) and Machine Learning (ML), colloquially "AI/ML", are finding increasing uses in daily life, including generative AI methods for text/image/video creation from prompts, electronic government (e-Gov), computer vision (CV), business analysis, personal applications on smartphones, and security. Many such applications are edge in nature with processing at the devices, e.g. mobile computing Internet of Things (IoT) applications. Notably and fundamentally, AI/ML are complex algorithms with abilities that grow with computational complexity and problem complexity (Kühl, et al., 2019). However, increasing abilities in AI typically come at a computational price (Li, et al., 2016). Notably such computational prices are not free, as these are associated with electrical power costs (Strubell, et al., 2019). Logically, such power demands will increase as AI/ML expands in application/use.

While such power demands are performed on relatively cheap electricity for large scale applications, even then it is not free (Denisova, et al., 2019). In edge-based AI/ML applications, power, hardware and software constraints are even more critical; for example, a complex and accurate computer vision system might require gigabytes on disk and consume significant power, possibly precluding smartphone or unmanned aerial vehicle (UAV) use. Similarly, further expansion of the IoT logically expands the use of edge computing, as cloud resources become more prohibitive to use for some applications (Shi & Dustdar, 2016). Thus, considering the Size, Weight, and Power (SWaP) demands of AI/ML solutions are necessary to their wider expansion and IoT use. Additionally, lower algorithm energy consumption can facilitate speed advantages since Power equals Energy over Time.

As AI/ML increasingly becomes democratized, more and more applications can be found with real-time demands and on-board processing requirements, as is the case with autonomous vehicles, single CPU computers, and smartphones, where power budgets may be more limited (Bihl & Talbert, 2020). Some solutions to SWaP issues involve selecting the appropriate method, e.g. (Thórisson & Helgasson, 2012), incorporating "AI Accelerators", advanced electronics to get around a Moore's Law bottleneck on computation abilities available on CPUs, and developing simple algorithms at an acceptable level of performance (Boubin, et al., 2019). Of particular interest herein is further understanding this trade-space with particular concern to the combination of AI/ML methods and edge-based AI accelerators.

URI: https://hdl.handle.net/10125/107294
978-0-9981331-7-1

Biologically inspired Artificial Neural Networks (ANNs) have received wide interests and applications, particularly due to the deep learning advances in the 2010s (Hao, 2019). However, while ANNs and deep learning ANNs have provided revolutionary new abilities, their computational demands grow significantly as ANNs grow in size (Li, et al., 2016) (Strubell, et al., 2019). Increasing computational demands from ANNs impact the SWaP performance of algorithms with the possibility that high performing, but high-SWaP, algorithms might not be deployable on edge processors, among other concerns (Strubell, et al., 2019).

Spiking Neural Networks (SNNs) are third generation ANNs whose neurons imitate the temporal, and sparse, spiking nature of biological neurons which are typically in an off state unless spiked high for a short period of time (Ghosh-Dastidar & Adeli, 2009), which results in natural SWaP reduction. Initial work in comparing SNNs and CNNs for power consumption showed the advantages in reduced power per inference but at lower accuracy than traditional ANNs (Cao, et al., 2014)(Blouw, et al., 2019). However, neither Cao et al. (2014) and Blouw et al. (2019) explored the tradeoff between complexity and power and accuracy, which is critical to understand to facilitate further low-SWaP use of SNNs as alternatives to traditional ANNs.

While there are theoretical differences between SNNs and more traditional ANNs, due to their novelty, the practical trade-space of when to use SNNs is largely unknown and unexplored. Generally, we are interested in providing answer or further understanding to the following research question:

RQ1: What are the practical benefits to using SNNs?

Based on prior work, the underlying hypothesis is that SNNs will be lower in SWaP than traditional ANNs but at the expense of accuracy (Blouw, et al., 2019)(Li, et al., 2021). But RQ1 is hard to answer, given differences between the maturity of ANN/CNN focused edge processors and the relative immaturity of SNN edge processors. Based on this and coding nuances, one could easily construct problems that are best solvable by any given algorithm. Therefore, we must further explore the following research questions as well to look at the SNN opportunity space:

RQ2: What are appropriate methods to compare SNNs with ANNs?

In order to test RQ1 and RQ2, we develop an apples-to-apples test and evaluation (T&E) framework using software and edge hardware optimized, respectively, for both ANNs and SNNs. With this T&E framework we explore the tradeoffs between architecture size and power consumption by considering the following:

RQ3: How do SNNs and ANNs compare on accuracy at the same task?
RQ4: How do SNNs and ANNs compare on SWaP when employed for the same task?

To answering RQ1-RQ4, the authors provide justification for the claims of the neuro-benefits that are realized in hardware when combining SNNs with edge-neuromorphics versus ANNs with traditional edge-processors. This is considered for CV from two different perspectives: 1) a simple ANN architecture which is expanded/reduced from a reference point, to characterize the relationships between architecture and performance; and 2) state-of-the-art SNN algorithms and their implementation nuances. In summary, the contributions of this work are: 1) We tackle a crucial void in existing research by concentrating on the trade space of SWaP in the context of neuromorphic computing; 2) We propose a scalable apples-to-apples comparison approach between traditional ANNs and SNNs for edge processing; and 3) Our results demonstrate that SNNs, when integrated on neuromorphic hardware, compare well with ANNs, but at one-tenth the power.

2. Background

Throughout the space of AI/ML, users must decide which algorithms to use and this involves a complex trade-space due to the wide proliferation of ML methods (Domingos, 2015). Within ML, one can coarsely group methods into five "tribes" (symbolists, connectionists, Bayesians, evolutionaries, and analogizers), depending on what family of algorithms one uses (Domingos, 2015). Of these tribes, connectionists, which focuses on biologically inspired ANNs, have received wide interests and applications, particularly due to the deep learning advances in the 2010s (Hao, 2019).

2.1. Artificial Neural Networks

ANNs are interconnected networks, whereby nodes and the weights connecting each node are trained to learn patterns in the data (Jain, Duin, & Mao, 2000). A wide variety of ANN variants exist and these range in complexity, architecture, philosophy, and training approaches (Bihl, et al., 2022). Inherently, ANNs are statistical in nature and epistemologically similar to Bayesian and likelihood methods (Bihl, et

al., 2022). ANNs are neurologically inspired constructs, which represent complex non-linear input and output relationships through interconnected nodes, "neurons," along with weights between inputs, nodes, and outputs (Jain, et al., 2000).

Computational ANNs are biologically inspired and base their structure around analogous relationships found in biological neurons, as seen in Figure 1, where inputs to ANNs are in data form and are analogous to biological axons from other neurons feeding into a neuron's dendrites (Bihl, et al., 2022). The cell then processes the inputs, which is represented as a transfer and activation function in an ANN, i.e. perceptrons when expressed in node form (Bihl, et al., 2022). The outputs are then analogous as axons in biology and probabilities in ANNs (Bihl, et al., 2022). The end result is a mapping between independent inputs (X_1, X_2, X_3, and X_p), connection weights (W_1, W_2, W_3, and W_p) and bias (B), and a dependent variable (Y) (Bihl, et al., 2022). Weights representing the strength of the association between independent and dependent variables (positive, negative, or zero) are then determined through statistical methods (Bihl, et al., 2022).

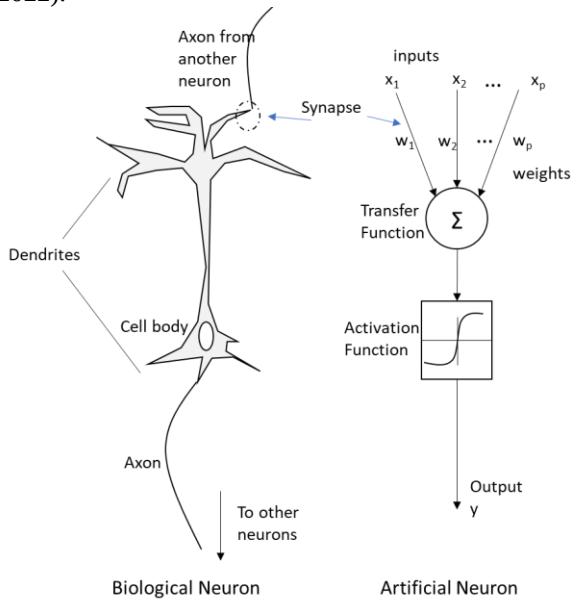

Figure 1. General Conceptualization of Biological and Biologically Inspired Artificial Neurons

The inspiration of biology in ANNs has yielded multiple generations of meta-architectural approaches, as conceptualized in Figure 2. The first generation of ANNs began with McCulloch and Pitts (McCulloch & Pitts, 1943) developing the earliest known ANN model, which included a non-differentiable step function for the neuron. However, the use of this function precluded efficient training methods. Second generation ANN neuron models included differentiable activation functions, such as a logistic sigmoid, which enable gradient descent methods, e.g. backpropagation, to be used for training. Notably, the current state of the practice in ANN implementation resides within second generation ANNs and it includes recent advances in convolution neural networks and deep learning, see (Bihl, et al., 2022), and many other methods.

While, individually, a single neuron ANN with a logistic activation function is essentially a logistic regression model (Timmerman, et al., 1999), the real power of ANNs comes into play with multiple hidden layer nodes, as these permit an ANN to learn a mapping in complex data. The interconnectivity is one inspiration taken from biological neuron models, whereby multiple interconnected nodes learn patterns between inputs and outputs through organizational, statistical, and iterative principles (Jain, et al., 2000). The result is a nonlinear model; even an ANN developed using linear activation functions will result in a nonlinear mapping, due to the interconnections and weights learned in training (Bihl, et al., 2022).

2.2. Spiking Neural Networks

As conceptualized in Figure 2, both first and second generation neuron models considered data in a continuous sense as either binary (first gen.) or real numbers (second gen.). Notably, biological neurons consider information in a more complex manner, whereby signals are sent from an axon after a neuron collects and processes a complex grouping of biochemical, molecular, electrical, cellular, behavioral, and systems information (Bhalla, 2014). Thus, biological neurons do not operate in a continuous manner, but rather a highly complex and interactive manner (Bihl, et al., 2022). A significant benefit of neurobiology is highly efficient operations and comparatively low-SWaP, when compared to ANNs for the same task (Strubell, et al., 2019). Third generation ANN models aim to model more of the operating characteristics of biological neurons, in an effort to both be more biologically plausible while providing engineering advantages (Maass, 1997) (Vicente-Sola, et al., 2023).

It is known that the human brain has considerably lower power requirements than a typical computer processor (10W in the neocortex vs 100W/cm^2 in a processor) (Cantley, et al., 2011). SNNs leverage concepts that result in the similar efficiencies as those built in efficiencies of biology and through being tethered to biologically models of neurons, have

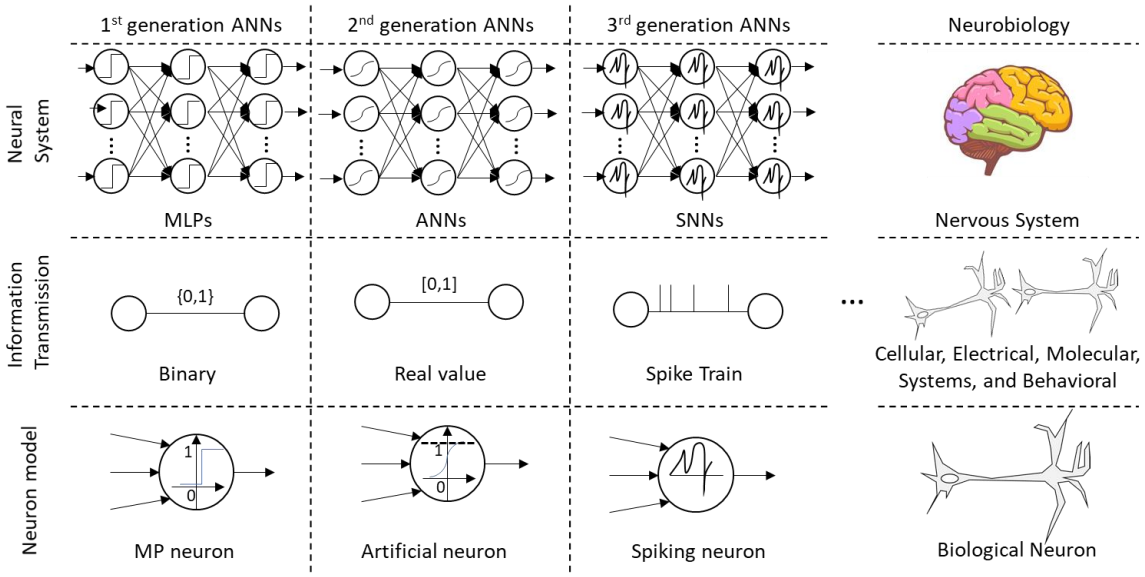

Figure 2. Conceptualization ANN generations with analogs to biology, adapted and extended from (Bihl, et al., 2022)

potential efficiency advantages over traditional ANNs and are thus of interest due to SWaP concerns alone.

SNNs aim to be closer to biology by exhibiting a spiking behavior (Maass, 1997). Thus, while traditional ANNs will output for all data, an SNN will only generate a spike when trained to do so (Maass, 1997). The result is a sparser operation than a traditional ANN. It is known that this sparsity can lead to improvements in power consumption, on specialized hardware (Blouw, et al., 2019). From an engineering point of view, these spikes are typically interpreted either in a rate-based manner, where the frequency of spiking is decoded as the value of interest, or in a latency-based manner, using precise spike timing as the information-carrying value.

However, limitations and a complex trade-space exist in developing SNNs. For instance, multiple models exist for capturing the spiking behavior, as reviewed in (Manna, et al., 2022). Notably, the most popular (and used herein) neuron model is the leaky-integrate and fire (LIF) which considers the potential to spike as: $\tau_m \frac{dV(t)}{dt} = -V(t) + RI(t)$, where the membrane voltage is $v(t)$, the input (data, but considered as current) is $I(t)$, R is membrane resistance, and τ_{RC} represents the membrane time constant (Manna, et al., 2022). The neuron fires a spike when the voltage $Vth = 1$. The voltage is then set and remains at zero until a refractory period has passed. For data processing and equivalency to 2nd generation ANNs, $I(t) = \sum_{i=1}^{n} w_i x_{i,t}$, where $x_{i,t}$ is the input from the *i-th* synapse at time *t* and w_i is the weight of the *i-th* synapse (Manna, et al., 2022). For comparison, a 2nd generation ANN neuron model would be $y(t) = ReLU(I(t) + b)$, for a general rectified linear unit activation function with a bias *b* (Manna, et al., 2022).

Beyond these considerations, as SNNs are an emerging area of ANN research, the current hardware and software solutions are less mature than those developed and optimized for second generation ANNs. For instance, when developing conventional ANNs, researchers have access to commercial off-the-shelf advanced hardware and established software frameworks. However, in neuromorphics and SNNs, while frameworks for developing SNNs exist, there is not the ability yet to seamlessly implement any type of algorithmic solution and not all neuromorphic software libraries support the same types of neurons or networks (Manna, et al., 2023). Similarly, different neuromorphic hardware have different levels of quantization, precision, scale (number of neurons/synapses), and types of neurons that they can provide or support. Thus, there are limited abilities to implement end-to-end neuromorphic solutions.

2.2.1 SNN Training. A primary bottleneck in implementing SNNs is that traditional ANN training methods, such as gradient descent, cannot be applied to SNNs, since the underlying approach of SNNs is non-differentiable. Thus, training and developing SNNs models traditionally involved an entirely different process than CNNs or ANNs in general. A general model for spiking neural learning includes the concept of Spike Timing Dependent Plasticity (STDP) (Kempter, et al., 1999). STDP works under the following principle: If a spike inputs to a given neuron, and that neuron outputs a spike shortly after, then the connection (i.e. weight) between spiking source and

the neuron tends to become stronger and vice versa. Alternatively, systems requiring to train deeper networks, resorted to training methods which allow to bypass the non-differentiability of SNNs and apply backpropagation, such as surrogate gradients (Neftci, 2019) or SLAYER (Shrestha & Orchard, 2018).

2.2.2 ANN to SNN Conversion. Due to the issues in training SNNs, an alternative family of training approaches was also developed whereby a traditional ANN is first trained and then it is mapped over to be an SNN, which is then implemented asynchronously. A variety of methods exist for ANN to SNN conversion like sigma-delta quantization (Yousefzadeh, et al., 2019) or directly mapping learned weights from a CNN to an SNN (Cao, et al., 2014), (Hunsberger & Eliasmith, 2016).

2.3. AI Accelerators for Edge Computing

Edge computing involves placing the computations, decision making, and inferences as close to the devices and sensor as possible. This improves efficiencies, reduces communications, and reduces latency in data-to-decisions. Of interest herein is edge computing where the computing is done at the device itself, e.g. classifier training is performed on a CPU/GPU and then deployed to the edge-based processor for operations.

A wide variety of AI accelerators exist for this purpose, see (Spiller, et al., 2022), but of interest herein are representative AI accelerators from the traditional, i.e. von Neumann, and neuromorphic approaches. Traditional hardware and neuromorphic hardware can be regarded as distinct generations. They differ in terms of operation (sequential processing as opposed to massively parallel processing), structure (discrete computation versus integrated processing and memory), coding (binary instruction as opposed to spiking neural networks), and timing (synchronous versus asynchronous) (Schuman, et al., 2022).

For this study, two edge-based devices were selected that represent both approaches: the Intel Loihi, a neuromorphic device, and the Intel Neural Compute Stick, colloquially "Movidius," which implements a computer vision optimized traditional von Neumann architecture (Davies, et al., 2018). The Loihi neuromorphic processor natively implements spiking neurons with hardware sections dedicated to synaptic, spiking, and dendrite activity (Davies, et al., 2018). In operating either USB edge-based processors, a trained ANN is saved to the device for processing data (Xu, et al., 2017). A brief comparison of the two devices used herein is presented in Table 1.

Table 1. Comparison of Loihi and Movidius, details from: (FRENKEL, ET AL., 2018) (DAVIES, ET AL., 2018) (INTEL, 2017)

DEVICE	LOIHI	MOVIDIUS
MODEL	Kapoho Bay	Myriad
VERSION	1	2
TECHNOLOGY	14nm FinFET	28nm
PROCESSOR TYPE	Loihi	SHAVE
# PROCESSORS	2	12
THROUGHPUT	3.44 Gspikes/s	1.5 Gb/s

3. Developing an Apples-to-Apples Evaluation Framework

Of particular interest in answering RQ1-RQ4 and understanding the differences in performance between ANNs and SNNs is to deploy the algorithms on appropriate hardware and software. Thus, selecting hardware and software optimized for the respective ANN approach are needed.

3.1. Coding and Hardware Consideration

Multiple issues exist in comparing the performance of AI/ML algorithms, no matter the measure of interest. Accuracy/performance metrics are subject to differences in datasets and experimental conditions; time and power metrics are subject to differences and quality in coding; which is compounded if special libraries are used in one trial and not another. To provide an apples-to-apples evaluation, the authors developed a framework to isolate such issues.

3.2. Metrics and Data Collection

The first question for evaluation is that of appropriate metrics to answer RQ2-RQ4. To this aim, the authors considered metrics for accuracy, time, and power. For edge-based applications with *a priori* classifier training, use cases for Loihi and Movidius, two metrics are of interest:

- Testing set classification accuracy
- Power consumption for test set validation.

Training accuracy was not considered since this was performed *a priori* to edge device deployment. For power, the authors focused on running the test set through fully trained algorithms on each edge device. In trial runs, it was seen that the communication time for Loihi was higher than Movidius, this could be due to Loihi being a prototype in nature, whereas the Movidius is a commercial product; thus, computation time was not considered to avoid inadvertent biases.

Accuracy was considered using the test data allocations in tensorflow.datasets. Power consumption

was computed using a UM25C power meter. To provide a stable baseline, a MacBook Pro was used for all assessments with the edge device and UM25C connected in series to a USB-port on the computer with the same port used for all tests. Due to the stochastic nature of ANNs, 3 replications were performed for each model and 95% mean (Student's t) confidence intervals were computed for all results. For all algorithms, Python 3.7.6, tensorflow 2.1, and nengo/nengoDL 3.0 were used. A MacBook Pro with an Intel i9 processor was used for all USB communication. The Loihi tests used Python 3.5.2, tensorflow 2.3, nengo 3.1, nengo-dl 3.1, and nengo-loihi 1.0. The Movidius tests used Python 3.7.11 and tensorflow 2.4. Herein, Nengo was used for SNNs since it provides neuromorphic hardware applications compatible with the Intel Loihi edge device.

3.3. Example Datasets

The authors considered common benchmarking datasets: MNIST (LeCun, et al., 1995), Fashion-MNIST (Xiao, et al. 2017), and CIFAR-10 (Krizhevsky & Hinton, 2009). Both MNIST and Fashion-MNIST are conceptually similar, n_C = 10 classes of 28x28 grayscale images of digits (MNIST) and fashion products (Fashion-MNIST). CIFAR-10 (Krizhevsky & Hinton, 2009) is a set color images (32x32x3), equally distributed into 10 categories (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck). All three datasets have 60,000 images which are separated into a predefined 50,000 image training set and a 10,000 image testing set. As of writing, test set benchmark accuracy for these are, approximately: 99.87% (MNIST), 96.91% (Fashion-MNIST), and 99.5% (CIFAR-10) (Image Classification, 2023). Notably, the algorithms that yielded these current benchmark performance levels are complex, have many layers, and include components which are not currently replicable in SNNs.

4. Experiment 1: Trade-space Study to Understand SNNs vs ANNs

To understand and characterize performance differences in ANNs and SNNs, first, a simplistic architecture approach was used to focus on understanding the research questions to benchmark ANN and SNN algorithms. To focus on RQ2, of particular interest is the effect of scaling.

4.1. Experiment Conceptualization

In order to compare intrinsic difference between ANNs and SNNs, the authors first focused on a consistent comparison, with simple architectural design having one input layer, one hidden layer, and one output layer. While such architectures are not as highly tuned as many deep learning methods, and high accuracy is not expected, they provide a simplicity which engenders comparisons.

To address RQ2, the authors' approach with this basis is presented in Figure 3, which presents the reference architecture (4 hidden nodes) in the center, along with a scaling in both directions by 2, for larger size (8 hidden nodes) and smaller size (2 hidden nodes). Thus, this experiment only changes the hidden layer, with the input and output layer kept consistent.

4.2. Experimental Settings and Setup

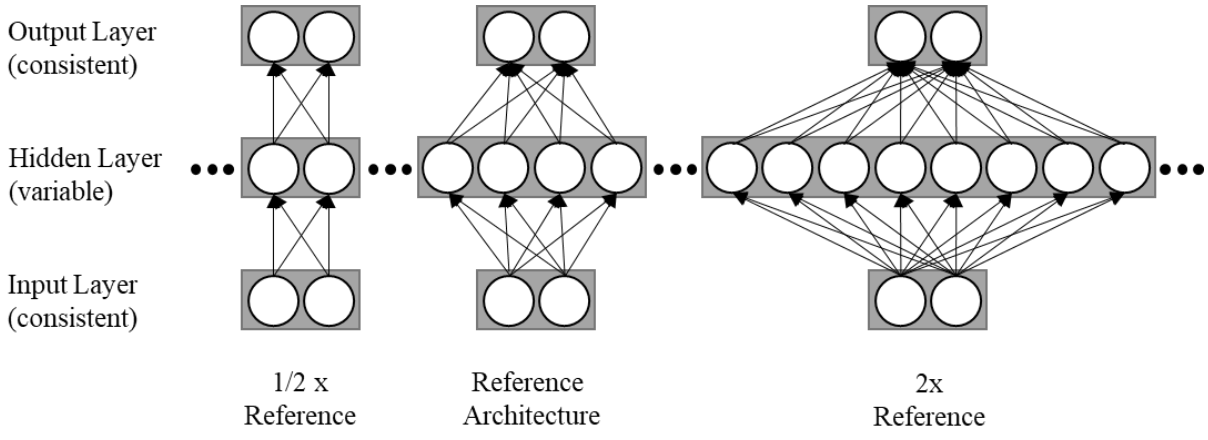

Figure 3. Conceptualization of the general experimental approach on a 3 layer neural network for the architecture changing from the baseline to 1/2 the baseline quantity of hidden nodes to 2x the quantity.

For the comparison of ANNs and SNNs in as apples-to-apples sense, three types of ANNs were created:

- ANNs – baseline 2nd generation ANNs
- SNNs – 3rd generation SNNs
- CSNNs – ANNs converted to SNNs

All SNNs were created in nengo/nengoDL 3.0, and all ANNs were created in tensorflow 2.1. The CSNNs were created by taking the respective ANN, created in tensorflow 2.1, and then converted each to SNNs using nengo/nengoDL 3.0 via the rate based method of (Hunsberger & Eliasmith, 2016). This method softens the neural response function and bypasses the non-differentiability problems by using an approximation of the spiking function during training and then replacing these neurons with actual spiking neurons during inference. For each SNN implementation, *dt*, the duration between consecutive computations of spikes, was implemented for *dt* = [0.05, 0.10].

Following the framework of Section 4.1, the authors developed ANNs which had N_{Hidden} = [6, 12, 24, 49, 98, 196] hidden nodes. The sweeping of the architecture of the study framework was increased by a factor of 2, to create ANNs and SNNs in the form, following the notation of (Bihl, et al., 2020), of:

$$28x28\text{-}N_{Hidden}ReLu\text{-}CL \quad (1)$$

where *ReLu* are rectified linear units (output linear for inputs greater than 0; 0 otherwise) for ANNs, or spiking rectified linear units (spiking rate is linear for inputs greater than 0; 0 otherwise) for all SNNs.

4.3. Experimental Results

Results for MNIST and Fashion-MNIST are presented in Figures 5 through 8, with the following designations in the legends: a) ANN (blue), b) SNN with dt = 0.1 (orange), c) SNN with dt = 0.05, d) CSNN with dt = 0.1 (red), and e) CSNN with dt = 0.05 (purple). Figures 5 and 6 present test set accuracy for MNIST and Fashion-MNIST, respectively. These figures present broad comparisons of test set classification accuracy with 95% t-test confidence intervals, for 3 replications, versus the architecture size, for N_{Hidden}. Notably, we see in both Figures 4 and 5 that ANN performance rises slowly and generally outperforms SNNs and CSNNs until sufficiently large architecture exists, at around N_{Hidden} = 24. After N_{Hidden} = 24, accuracy is statistical equivalent for the ANNs, SNNs, and CSNNs at each N_{Hidden} point, whereby the confidence intervals of accuracy overlap considerable, or entirely, at each N_{Hidden} point. Thus, RQ3 is answered with ANNs and SNNs performing similarly at this task when sufficient architecture is instantiated.

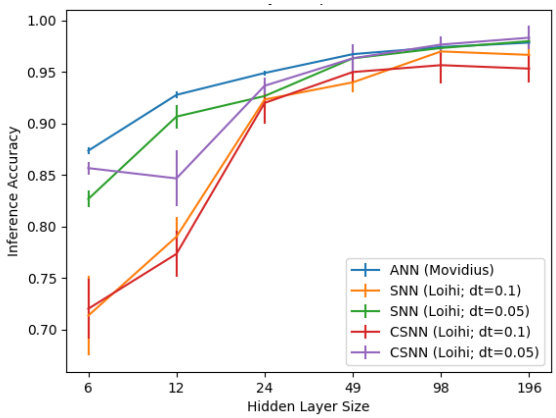

Figure 4. Test set accuracy for MNIST

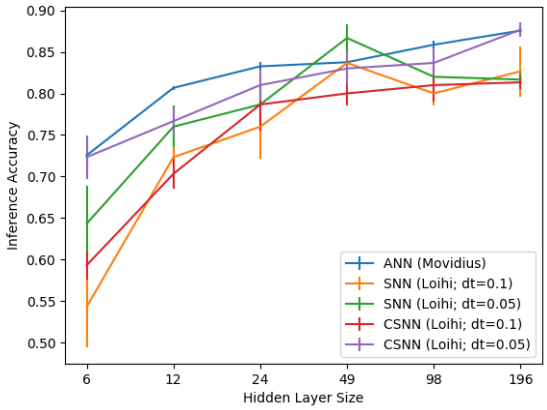

Figure 5. Test set accuracy for Fashion MNIST

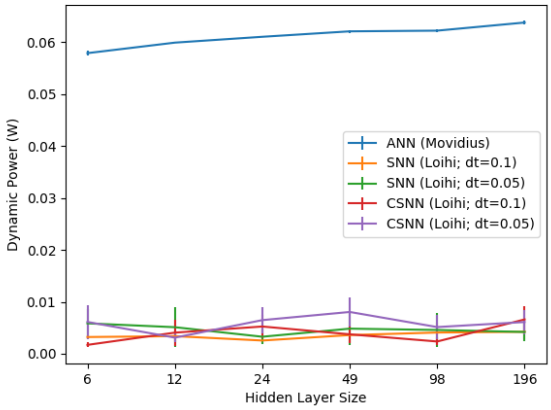

Figure 6. Test set dynamic power (Watts) for MNIST

Given that the accuracy was comparable between ANNs, SNNs, and CSNNs, the SWaP concerns are then of interest. Figures 6 and 6 present dynamic power for each network on MNIST, Figure 6, and Fashion MNIST, Figure 7. Notably, we see in both that SNNs and CSNNs offer significant performance advantages in dynamic power, which is flat (within adjacent confidence intervals) across explored N_{Hidden}

points. In contrast, the ANNs required 5-10 times more power which was increasing across explored N_{Hidden} points. Thus, RQ4 is answered with whereby SNNs provide a performance advantage over ANNs.

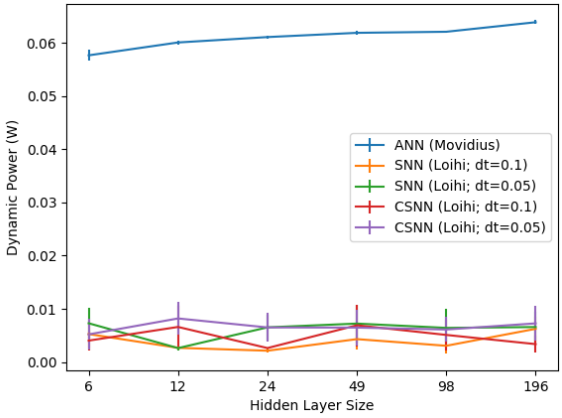

Figure 7. Test set dynamic power (Watts) for Fashion MNIST

5. Experiment 2: Focused Comparison of SNNs vs ANNs

.

The results in Section 4 illustrated the differences of ANNs and SNNs on a relatively simplistic problem with performance comparable in accuracy (once sufficient SNN architecture is reached) but with SNNs providing that performance with less SWaP demands. However, the apparent advantages of SNNs and their deployment on neuromorphic hardware are only potentials. Of interest next is exploring the research questions on a more meaningful problem, which will be considered using competitive a SNN benchmark algorithm and ANN equivalent.

5.1. Algorithm Development

As the complexity of the problem at hand increases, so generally does the required complexity of the network employed. Therefore, in this section we evaluate a complex SNN architecture taken from the current state-of-the-art in neuromorphic image classification, the S-ResNet (Vicente-Sola, et al., 2022). S-ResNet provides the current benchmark setting accuracy for SNNs on imagery; however, it has only been demonstrated in software. On CIFAR-10, S-ResNet, when using the widest version and all developed embellishment (not implementable on Loihi), yielded test-set classification performance of 94.14%. The success of S-ResNet is based on the usage of spiking residual connections. Adding residual connections to the ANNs in general allows one train deeper architectures without making the optimization problem more complex, giving rise to deeper and more powerful topologies (Vicente-Sola, et al., 2022). The topology of the system in (Vicente-Sola, et al., 2022) is based on the CNN architecture proposed by the original ResNet (He, et al., 2016), the neuron model used for the spiking layers is the LIF and as normalization strategy Batch Normalization Through Time (BNTT) (Kim & Panda, 2020) is used. BNTT being a version of Batch Normalization that de-correlates activation statistics through time and has been proven to improve performance for SNN.

S-ResNet is formulated as

$$3x32x32 - 32C3_{s1} - BNTT - SN - (RB_{32,s1})*5 - RB_{32,s2} - (RB_{64,s1})*5 - RB_{64,s2} - (RB_{128,s1})*5 - RB128_{,s2} - GAP - CL \quad (2)$$

where BNTT indicates the process of (Vicente-Sola, et al., 2022), s1 means stride=1, s2 means stride=2, SN are spiking neurons, GAP indicates global average pooling (GAP) to 1x1 resolution, and RB is a residual block defined as

$$RB_{n,sm} = nC3s1 - BNTT - SN - nC3sm - BNTT - SN - residual\ input \quad (3)$$

with "s" in the convolutions to indicate the stride, *n* indicating the output channels of the convolutional layers in the block, and *m* indicating the stride of its convolutions.

However, to implement S-ResNet on Loihi and in Nengo, some changes had to be made due to hardware and software limitations for both size allowable on Loihi and types of connections permissible in Nengo. In this work we thus implemented the 38-layer S-ResNet38 with only the S2M (Spiking Output to Membrane Potential) approach which was shown in (Vicente-Sola, et al., 2022) to have a test set accuracy of 89.27% on CIFAR-10. Herein, the SNNs implemented used spiking ReLu and, since BNTT was not previously instantiated in Nengo, this work developed a custom object for BNTT in Nengo. S-ResNet is natively in SNN form, thus to create an ANN equivalent, ANN-S-ResNet, the following changes were made: BNTT was changed to Batch Normalization, ReLu was used in place of spiking ReLu, and the residual connections were output to input rather than S2M.

5.2. Experimental Results

With implementations for ANN-S-ResNet, S-ResNet, and CS-ResNet (ANN-S-ResNet converted to an SNN), the comparison process of Section 3 was repeated, again with 3 replications for each algorithm,

but with no sweeping architectural changes of Figure 3 since the many layers would make such an approach seemingly arbitrary.

Results are presented in Table 2 show that ANN accuracy outperforms both the SNN and CSNN; however, the drop in accuracy is no more than 4.3%. However, it is likely that architecture changes made for edge implementation of S-ResNet38 with S2M caused the drop in accuracy, since the discussion in Section 4 showed statistically similar performance of SNNs and ANNs for equivalent networks. For power, Table 2 shows that the SNN and CSNN are seen to require orders of magnitude less dynamic power for their inferences.

Table 2. Results of S-ResNet on Loihi and Movidius with mean and confidence interval

ALGORITHM	ANN-S-RESNET	S-RESNET	CS-RESNET
DEVICE	MOVIDIUS	LOIHI	LOIHI
ACCURACY (%)	90.6% ± 0.001%	86.3% ± 0.002%	87.2% ± 0.002%
DYNAMIC POWER (W)	0.105 ± 9.5E-6	0.0092 ± 4E-6	0.0072 ± 1.2E-5

Thus, further work is needed to mature the software available for edge neuromorphics to close the gap in capabilities. Additionally, Table 2 shows that the CSNN implementation of S-ResNet slightly outperforms S-ResNet in both accuracy and dynamic power consumption. As this could indicate an advantage from converting an ANN to SNN over direct SNN development, more investigation into why this occurred is of interest.

6. Conclusions

This contribution addresses recent concerns in AI/ML literature involving expanding computational and power demands for ever increasing algorithm sizes. To this aim, the authors presented a systematic study to understand how Spiking Neural Networks (SNNs) might be advantageous over traditional Artificial Neural Networks (ANNs) with implementation on edge devices. Prior studies that compared SNNs and ANNs had limitations which yield not always apples-to-apples comparisons. This study, answering RQ2, thus first developed a simple framework to change ANN architecture size, while evaluating performance by accuracy, time, power, and size, key SWaP (Size, Weight, and Power) concerns. ANNs, SNNs, and ANN-to-SNN converted networks were all explored with an aim for equivalency maintained throughout. The results further provided verification for the performance claims of state-of-the-art neuromorphic classification algorithms.

Answering RQ1, the results show that SNNs offer distinct power advantages over traditional ANNs (answering RQ4), while providing comparable accuracy (answering RQ3). This is particularly seen when "fully embracing" neuromorphics whereby a neuromorphic edge device is combined with neuromorphic SNN algorithms. Interesting, the ANN-to-SNN converted models provided better accuracy and less power than both ANNs and SNNs, a result that needs further consideration since it could indicate either issues in training SNNs natively or a unique advantage from ANN-to-SNN conversion.

Further studies and analysis can aim to understand this trade-space which logically involves extending this study to include other dedicated AI/ML hardware. Additionally of interest are exploring additional network structures, and expanding the current study to different applications, e.g. CV or speech processing.

7. Acknowledgements

The views expressed in this paper are those of the authors and do not necessarily represent any views of the U.S. Government, U.S. Department of Defense, or U.S. Air Force. This work was cleared for unlimited release under AFRL-2023-2753.

8. References

Bhalla, U. (2014). Molecular computation in neurons: a modeling perspective. . *Current opinion in neurobiology, 25*, 31-37.

Bihl, T. J., et al. (2022). Artificial Neural Networks and Data Science. *Encycl. of Data Science and Machine Learning*.

Bihl, T., & Talbert, M. (2020). Analytics for Autonomous C4ISR within e-Government: a Research Agenda. *HICSS*, 2218-2227.

Bihl, T., et al. (2020). Easy and Efficient Hyperparameter Optimization to Address Some Artificial Intelligence "ilities". *HICSS*, 943-952.

Blouw, P., et al. (2019). Benchmarking Keyword Spotting Efficiency on Neuromorphic Hardware. *Proc. 7th Ann. Neuro-inspired Comput. Elements*.

Boubin, J., et al. (2019). Neurowav: Toward real-time waveform design for vanets using neural networks. *Vehicular Network. Conf. (VNC)*, 1-4.

Cantley, K., et al. (2011). Hebbian learning in spiking neural networks with nanocrystalline silicon TFTs and memristive synapses. *IEEE Trans on Nanotechnology, 10*(5), 1066-1073.

Cao, Y., et al. (2014). Spiking Deep Convolutional Neural Networks for Energy-Efficient Object Recognition. *Int'l J. Computer Vision, 113*(1), 54–66.

Davies, M., et al. (2018). Loihi: A neuromorphic manycore processor with on-chip learning. . *Ieee Micro, 38*(1), 82-99.

Denisova, V., et al. (2019). Blockchain infrastructure and growth of global power consumption. *Int'l J. Energy Econ Policy, 9*(4), 22-29.

Domingos, P. (2015). *The master algorithm: How the quest for the ultimate learning machine will remake our world.* . Basic Books.

Frenkel, C., et al. (2018). A 0.086-mm $^ 2 $12.7-pJ/SOP 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28-nm CMOS. *IEEE Trans. biomedical circuits and systems, 13*(1), 145-158.

Ghosh-Dastidar, S., & Adeli, H. (2009). Third generation neural networks: Spiking neural networks. *Adv. Computational Intelligence*, 167-178.

Hao, K. (2019, Jan. 25). *We analyzed 16,625 papers to figure out where AI is headed next*. (Technology Review) Retrieved Apr. 1, 2019, from https://www.technologyreview.com/s/612768/we-analyzed-16625-papers-to-figure-out-where-ai-is-headed-next/

He, K., et al. (2016). Deep residual learning for image recognition. *IEEE conf. on computer vision and pattern recognition*, 770-778.

Hunsberger, E., & Eliasmith, C. (2016). Training spiking deep networks for neuromorphic hardware. . *arXiv preprint arXiv:1611.05141*.

Image Classification. (2023, May 10). (Papers with Code) Retrieved May 10, 2022, from Papers with Code: https://paperswithcode.com/task/image-classification

Intel. (2017). *Intel Movidius Myriad 2 Vision Processing Unit (VPU)*. Retrieved May. 10, 2022, from https://newsroom.intel.com/wp-content/uploads/sites/11/2017/06/Myriad-2-VPU-Fact-Sheet.pdf

Jain, A. K., et al. (2000, Jan.). Statistical Pattern Recognition: A Review. *IEEE Trans. Pattern Analysis and Machine Intelligence, 22*(1), 4-37.

Kempter, R., et al. (1999). Hebbian learning and spiking neurons. . *Physical Review E, 59*(4), 4498.

Kim, Y., & Panda, P. (2020). Revisiting batch normalization for training low-latency deep spiking neural networks from scratch. *Frontiers in neuroscience*.

Krizhevsky, A., & Hinton, G. (2009). *Learning multiple layers of features from tiny images.* Toronto, CA: University of Toronto. Retrieved from https://www.cs.toronto.edu/~kriz/cifar.html

Kühl, N., et al. (2019). Machine Learning in Artificial Intelligence: Towards a Common Understanding. *HICSS*, 5236-5245.

LeCun, Y., et al. (1995). Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective*.

Li, D., et al. (2016). Evaluating the energy efficiency of deep convolutional neural networks on CPUs and GPUs. *IEEE Int'l Conf big data and cloud computing (BDCloud)*, 477-484.

Li, Y., et al. (2021). A free lunch from ANN: Towards efficient, accurate spiking neural networks calibration. *Int'l Conf. Mach. Learn.*, 6316-6325.

Maass, W. (1997). Networks of spiking neurons: the third generation of neural network models. *Neural networks, 10*(9), 1659-1671.

Manna, D.L., et al. (2022). Simple and complex spiking neurons: Perspectives and analysis in a simple STDP scenario. *Neuromorphic Computing and Engineering*, 2(4), p.044009.

Manna, D. L., et al. (2023) Frameworks for SNNs: a review of data science-oriented software and an expansion of SpykeTorch *Int'l Conf Engineering Applications of Neural Networks*, pp. 227-238.

McCulloch, W., & Pitts, W. (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics, 5*, 115-133.

Schuman, C., et al. (2022). Opportunities for neuromorphic computing algorithms and applications. *Nature Computational Science, 2*(1), 10-10.

Shi, W., & Dustdar, S. (2016). The promise of edge computing. *Computer, 49*(5), 78-81.

Shrestha, S., & Orchard, G. (2018). Slayer: Spike layer error reassignment in time. *Advances in neural information processing systems, 31*.

Spiller, D., et al. (2022). Wildfire segmentation analysis from edge computing for on-board real-time alerts using hyperspectral imagery. *IEEE MetroXRAINE*, 725-730.

Strubell, E., et al. (2019). Energy and policy considerations for deep learning in NLP. *arXiv preprint arXiv:1906.02243*.

Thórisson, K., & Helgasson, H. (2012). Cognitive architectures and autonomy: A comparative review. *J. Artificial General Intell., 3*(2), 1-30.

Timmerman, D., et al. (1999). Artificial neural network models for the preoperative discrimination between malignant and benign adnexal masses. *Ultrasound Obstetrics Gynecology, 13*(1), 17-25.

Vicente-Sola, A., et al. (2022). Keys to accurate feature extraction using residual spiking neural networks. *Neuromorphic Comput. & Engineering*, 044001.

Vicente-Sola, A., et al. (2023). Spiking Neural Networks for event-based action recognition: A new task to understand their advantage. *arXiv:2209.14915*.

Xiao, H., et al. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Xu, X., et al. (2017). Classify 3D voxel based point-cloud using convolutional neural network on a neural compute stick. *3th Int'l Conf. Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, 37-43.

Yousefzadeh, A., et al. (2019). Conversion of synchronous artificial neural network to asynchronous spiking neural network using sigma-delta quantization. *IEEE Int'l Conf on Artificial Intelligence Circuits and Systems (AICAS)*, 81-85.