ORIGINAL RESEARCH
published: 24 June 2020
doi: 10.3389/fnins.2020.00535

Exploring the Connection Between Binary and Spiking Neural Networks

*Sen Lu and Abhronil Sengupta**

School of Electrical Engineering and Computer Science, The Pennsylvania State University, University Park, PA, United States

On-chip edge intelligence has necessitated the exploration of algorithmic techniques to reduce the compute requirements of current machine learning frameworks. This work aims to bridge the recent algorithmic progress in training Binary Neural Networks and Spiking Neural Networks—both of which are driven by the same motivation and yet synergies between the two have not been fully explored. We show that training Spiking Neural Networks in the extreme quantization regime results in near full precision accuracies on large-scale datasets like CIFAR-100 and ImageNet. An important implication of this work is that Binary Spiking Neural Networks can be enabled by "In-Memory" hardware accelerators catered for Binary Neural Networks without suffering any accuracy degradation due to binarization. We utilize standard training techniques for non-spiking networks to generate our spiking networks by conversion process and also perform an extensive empirical analysis and explore simple design-time and run-time optimization techniques for reducing inference latency of spiking networks (both for binary and full-precision models) by an order of magnitude over prior work. Our implementation source code and trained models are available at https://github.com/NeuroCompLab-psu/SNN-Conversion.

Keywords: Spiking Neural Networks, Binary Neural Networks, In-Memory computing, neuromorphic computing, ANN-SNN conversion

Edited by:
Arindam Basu,
Nanyang Technological University, Singapore

Reviewed by:
Guoqi Li,
Tsinghua University, China
Mostafa Rahimi Azghadi,
James Cook University, Australia

Correspondence:
Abhronil Sengupta
sengupta@psu.edu

Specialty section:
This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience

Received: *04 March 2020*
Accepted: *30 April 2020*
Published: *24 June 2020*

Citation:
Lu S and Sengupta A (2020) Exploring the Connection Between Binary and Spiking Neural Networks. Front. Neurosci. 14:535. doi: 10.3389/fnins.2020.00535

1. INTRODUCTION

The explosive growth of edge devices such as mobile phones, wearables, smart sensors and robotic devices in the current Internet of Things (IoT) era has driven the research for the quest of machine learning platforms that are not only accurate but are also optimal from storage and compute requirements. On-device edge intelligence has become increasingly crucial with the advent of a plethora of applications that require real-time information processing with limited connectivity to cloud servers. Further, privacy concerns for data sharing with remote servers have also fueled the need for on-chip intelligence in resourced constrained, battery-life limited edge devices.

To address these challenges, a wide variety of works in the deep learning community have explored mechanisms for model compression like pruning (Han et al., 2015; Alvarez and Salzmann, 2017), efficient network architectures (Iandola et al., 2016), reduced precision/quantized networks (Hubara et al., 2017), among others. In this work, we primarily focus on "Binary Neural Networks" (BNNs)—an extreme form of quantized networks where the neuron activations and synaptic weights are represented by binary values (Courbariaux et al., 2016; Rastegari et al., 2016). Recent experiments on large-scale datasets like ImageNet (Deng et al., 2009) have demonstrated acceptable accuracies of BNNs, thereby leading to their current popularity. For instance, Rastegari et al. (2016) has shown that 58× reduction in computation time and 32× reduction in model size can be achieved for a BNN over a corresponding full-precision model. The drastic reductions in computation time simply result from the fact that costly Multiply-Accumulate operations required in a standard deep network can be simplified to simple XNOR and Pop-Count Operations. While current commercial hardware (Cass, 2019) already supports fixed point precision (as low as 4 bits), algorithmic progress on BNNs have contributed to the recent wave of specialized "In-Memory" BNN hardware accelerators using CMOS (Zhang et al., 2017; Biswas and Chandrakasan, 2018) and post-CMOS technologies (Sun et al., 2018) that are highly optimized for single-bit state representations.

As a completely parallel research thrust, neuromorphic computing researchers have long advocated for the exploration of "brain-like" computational models that abstract neuron functionality as a binary output "spike" train over time. The binary nature of neuron output can be exploited to design event-driven hardware that is able to demonstrate significantly low power consumption by exploiting event-driven computation and data communication (Deng et al., 2020). IBM TrueNorth (Akopyan et al., 2015) and Intel Loihi (Davies et al., 2018) are examples of recently developed neuromorphic chips. While the power advantages of neuromorphic computing have been apparent, it has been difficult to scale up the operation of such "Spiking Neural Networks" (SNNs) to large-scale machine learning tasks. However, recent work has demonstrated competitive accuracies of SNNs in large-scale image recognition datasets like ImageNet by training a non-spiking deep network and subsequently converting it to a spiking version for event-driven inference (Rueckauer et al., 2017; Sengupta et al., 2019).

There has not been any exploration or empirical study at exploring whether SNNs can be trained with binary weights for large-scale machine learning tasks. Note that this is not a trivial task since training standard SNNs itself from non-spiking networks has been a challenge due to the several constraints imposed on the base non-spiking network (Sengupta et al., 2019). If we assume that, in principle, such a network can be trained then the underlying enabling hardware for both BNNs and SNNs become equivalent[1] (due to the binary nature of neuron/synapse state representation) except for the fact that the SNN needs to be operated over a number of time-steps. This work is aimed at exploring this connection between BNN and SNN.

While a plethora of custom BNN hardware accelerators have been developed recently, it is well-known that BNNs suffer from significant accuracy degradation in complex datasets in contrast to full-precision networks. Recent work has demonstrated that while weight binarization can be compensated by training the network with the weight discretization in-loop, neuron activation binarization is a serious concern (Hubara et al., 2017). Interestingly, it has been shown that although SNNs represent neuron outputs by binary values (Maass, 1997), the information integration over time can be approximated as a Rectified Linear transfer function (which is the most popular neuron transfer function used currently in full-precision deep networks). Drawing inspiration from this fact, we explore whether SNNs can be trained with binary weights as a means to bridge the accuracy gap of BNNs. This opens up the possibility of using BNN hardware accelerators for resource constrained edge devices without compromising on the recognition accuracy. This work also serves as an important application domain for SNN neuromorphic algorithms that can be viewed as augmenting the computational power of current non-spiking binary deep networks.

[1]"near-equivalent" since neuron states are discretized as $-1, +1$ in BNN while SNN neuron outputs are discretized as $0, 1$.

2. RELATED WORK AND MAIN CONTRIBUTIONS

The obvious comparison point of this paper would be recent efforts at training quantized networks with bit-precision greater than single bit. There have been a multitude of approaches (Li et al., 2016; Zhou et al., 2016, 2017; Choi et al., 2018; Deng et al., 2018; Zhang et al., 2018) with recent efforts aimed at designing networks with hybrid precision where the bit-precision of each layer of the network can vary (Prabhu et al., 2018; Wu et al., 2018; Chakraborty et al., 2019; Wang et al., 2019). However, in order to support variable bit-precision for each layer, the underlying hardware would need to be designed accordingly to handle mixed-precision (which usually is characterized by much higher area, latency and power consumption than BNN hardware accelerators. Further, peripheral circuit complexities like sense amplifier input offset, parasitics limit their scalability; Xue et al., 2019). This work explores a complementary research domain where the core underlying hardware can be simply customized for a BNN. This enables us to leverage the recent hardware developments of "In-Memory" BNN accelerators and provides motivation for the exploration of *time* (SNN computing framework) rather than *space* (Mixed Precision Neural Networks) as the information encoding medium to compensate for accuracy loss exhibited by BNNs. Distributing the computations over time also implies that the instantaneous power consumption of the network would be much lower than mixed-precision networks and approach that of a BNN in the worst-case (savings observed due to SNN event-driven behavior discussed in the next section) which is the critical parameter governing power-grid design and packaging cost for low-cost edge devices.

There has been also recent efforts by the neuromorphic hardware community at training SNNs for unsupervised learning with binary weights enabled by stochasticity of several emerging post-CMOS technologies (Suri et al., 2013; Sengupta et al., 2018; Srinivasan and Roy, 2019). Earlier works on analog CMOS VLSI implementations of bistable synapses have been also explored (Indiveri et al., 2006). However, such works have been typically limited to shallow networks for simple digit recognition frameworks and do not bear relevance to our current effort at training supervised deep BNNs/SNNs.

We utilize standard training techniques for non-spiking networks and utilize the trained models for conversion to a spiking network. We perform an extensive empirical analysis and substantiate several optimization techniques that can reduce the inference latency of spiking networks by an order of magnitude without compromising on the network accuracy. A key facet of our proposal is the run-time flexibility. Depending on the application level accuracy requirement, the network can be simply run for multiple time-steps while leveraging the core BNN-catered "In-Memory" hardware accelerator.

3. B-SNN PROPOSAL

We first review preliminaries of BNNs and SNNs from literature and subsequently describe our proposed B-SNN (SNN with binary weights).

3.1. Binary Networks

Our BNN implementation follows the XNOR-Net proposal in Rastegari et al. (2016). While the feedforward dot-product is performed using binary values, BNNs maintain proxy full-precision weights for gradient calculation. To formalize, the dot-product computation between the full-precision weights and inputs is simplified in a BNN as follows:

$$I * W \approx (\text{sign}(I) * \text{sign}(W))\alpha \tag{1}$$

where, α is a non-binary scaling factor determined by the L1-norm of the full-precision proxies (Rastegari et al., 2016). Straight-Through Estimator (STE) with gradient-clipping to $(-1, +1)$ range is used during the training process (Rastegari et al., 2016). Note that the above formulation reduces both weights and neuron activations to $-1, +1$ values. Although a non-binary scaling factor is introduced per layer, yet the number of non-binary operations due to the scaling factor is significantly low.

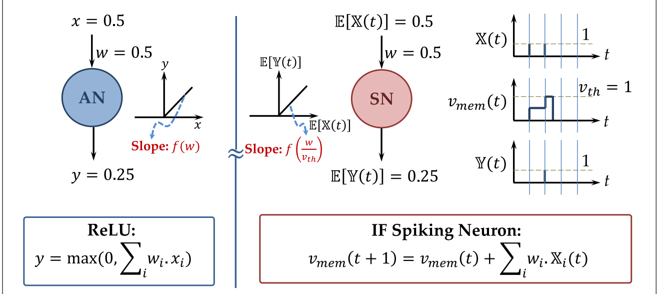
3.2. Spiking Networks

SNN training can be mainly divided into three categories: ANN[2]-SNN conversion, backpropagation through time from scratch and unsupervised training through Spike-Timing Dependent Plasticity (Pfeiffer and Pfeil, 2018). Since ANN-SNN conversion relies on standard backpropagation training techniques, it has been possible to scale SNN training using such conversion methods to large-scale problems (Sengupta et al., 2019). ANN-SNN conversion is driven by the observation that an Integrate-Fire spiking neuron is functionally equivalent to a Rectified Linear ANN neural transfer function. The functionality of an Integrate-Fire (IF) spiking neuron can be described by the temporal dynamics of a state variable, v_{mem}, that accumulates incoming spikes and fires an output spike whenever the membrane potential crosses a threshold, v_{th}.

$$v_{mem}(t+1) = v_{mem}(t) + \sum_i w_i.\mathbb{X}_i(t) \tag{2}$$

Considering $\mathbb{E}[\mathbb{X}(t)]$ to be the input firing rate (total spike count over a given number of time-steps), the output spiking rate

[2]ANN refers to standard non-spiking networks, Analog Neural Networks (Diehl et al., 2015), where the neuron state representations are analog or full-precision in nature, instead of binary spikes.



$x = 0.5$
$w = 0.5$
AN
y
x
Slope: $f(w)$
$y = 0.25$
ReLU:
$y = \max(0, \sum_i w_i.x_i)$
$\approx$
$\mathbb{E}[\mathbb{X}(t)] = 0.5$
$w = 0.5$
$\mathbb{E}[\mathbb{Y}(t)]$
$\mathbb{E}[\mathbb{X}(t)]$
SN
Slope: $f\left(\frac{w}{v_{th}}\right)$
$\mathbb{E}[\mathbb{Y}(t)] = 0.25$
$\mathbb{X}(t)$
$v_{mem}(t)$
$v_{th} = 1$
$\mathbb{Y}(t)$
1
t
IF Spiking Neuron:
$v_{mem}(t+1) = v_{mem}(t) + \sum_i w_i.\mathbb{X}_i(t)$

FIGURE 1 | An example to illustrate the mapping of ReLU to IF-Spiking neuron.

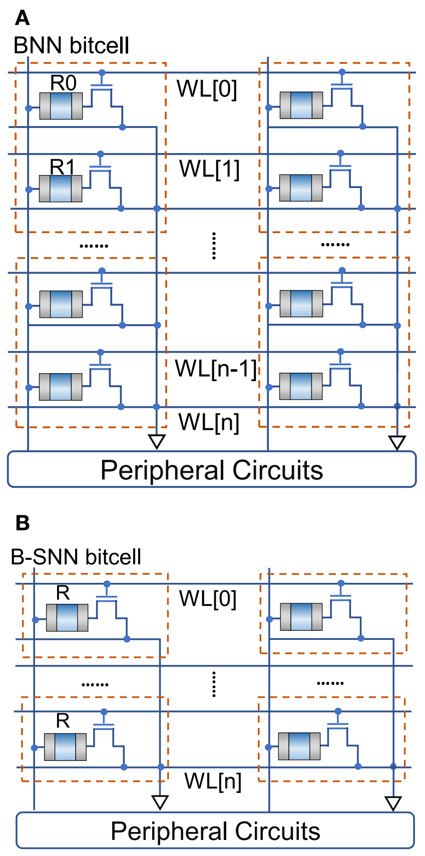
of the neuron is given by $\mathbb{E}[\mathbb{Y}(t)] = \frac{w.\mathbb{E}[\mathbb{X}(t)]}{v_{th}}$ (considering the neuron being driven by a single input $\mathbb{X}(t)$ and a positive synaptic weight w). In case the synaptic weight is negative, the neuron firing rate would be zero since the neuron membrane potential would be unable to cross the threshold. This is in direct correspondence to the Rectified Linear functionality and is described by an example in **Figure 1**. An ANN trained with ReLU neurons can therefore be transformed to an SNN with IF spiking neurons with minimal accuracy loss. The sparsity of binary neuron spiking behavior can be exploited for event-driven inference resulting in significant power savings (Sengupta et al., 2019).

3.3. Connecting Binary and Spiking Networks

Our B-SNN is trained by using BNN training techniques described earlier. However, we utilize analog ReLU neurons instead of binary neurons. Conceptually, the network structure is analogous to Binary-Weight Networks (BWNs) introduced in Rastegari et al. (2016). However, we also include additional constraints like bias-less neural units and no batch-normalization layers in the network structure (Sengupta et al., 2019). This is due to the fact that including bias and batch-normalization can potentially result in huge accuracy loss during the conversion process (Rueckauer et al., 2017). Much of the success of training BNNs can be attributed to Batch-Normalization. Hence, it is not trivial to train such highly-constrained ANNs with binary weights and without Batch-Normalization aiding the training process. Additional constraints like the choice of pooling mechanism, spiking neuron reset mechanism are discussed in details in the next section. This work is aimed at performing an extensive empirical analysis to substantiate the feasibility of achieving high-accuracy and low-latency B-SNNs.

Note that the threshold of each network layer is an additional hyper-parameter introduced in the SNN model and serves as an important trade-off factor between SNN latency and accuracy. Due to the neuron reset mechanism, the SNN neurons are characterized by a discontinuity at the reset time-instants. If the threshold is too low, the membrane potential accumulations would be always higher than the threshold causing the neuron to continuously fire. On the other hand, too high thresholds result in increased latency for neurons to fire. In this work, we normalize the neuron thresholds to the maximum ANN activation (recorded by passing the training set once after the ANN has been trained) (Rueckauer et al., 2017). Other thresholding schemes can be also applied (Sengupta et al., 2019) to minimize the conversion accuracy loss further.

Considering that the SNN is operated for N time-steps, the network converges to a Binary-Weight Network as $N \rightarrow \infty$. However, for a finite number of timesteps, we can consider the network to be a discretized ANN, where the weights are binary but the neuron activations are represented by $B = log_2 N$ number of bits. However, since the neuron states are represented by 0 and 1 values, B-SNNs are event-driven, thereby resulting in power consumption only when triggered, i.e., on receiving a spike from the previous layer. Hence, while the representative bit-precision can be ∼7 bits for networks simulated over 100 timesteps, the network's computational power does not scale-up corresponding to a multi-bit neuron model. This is explained in **Figures 2A–E**. The left-panel depicts a bit-cell for an "In-Memory" Resistive Random Access Memory (RRAM) based BNN hardware accelerator (Yin et al., 2019). The RRAM can be programmed to either a high resistive state (HRS) or a low resistive state (LRS). The RRAM states and input conditions for $+1, -1$ are tabulated in **Figure 2** and shows the correspondence to the binary dot-product computation. Note that two rows per input are used due to the differential nature $+1, -1$ of the neuron inputs. Hence, irrespective of the value of the input, one of the rows of the array will be active resulting in power consumption. **Figure 2B** depicts the same array for the B-SNN scenario. Since, in a B-SNN, the neuron outputs are 0 and 1, we can use just one row per bit-cell, thereby reducing the array area by 50%. Note that a dummy column will be required for referencing purposes of sense amplifiers interfaced with the array (Yin et al., 2019). Additionally, the neuron circuits interfaced with the array need to accumulate the dot-product evaluation over time. Such an accumulation process can be accomplished using digital accumulators (Han et al., 2017) or non-volatile memory technologies (Sengupta et al., 2016; Wijesinghe et al., 2018). Note that energy expended due to this accumulation process is minimal in contrast to the overall crossbar power consumption (Ankit et al., 2017). However, the input to the next layer will be a binary spike, thereby enabling us to utilize the "In-Memory" computing block as the core hardware primitive. It is worth noting here that the power-consumption involved in accessing the rows of the array occurs only on a spike event, thereby resulting in event-driven operation.



C

Input	WL[0]	WL[0]
−1	1	0
+1	0	1

D

Input	WL[0]
0	0
1	1

E

Weight	R0	R1
−1	LRS	HRS
+1	HRS	LRS

F

Weight	R
−1	HRS
+1	LRS

FIGURE 2 | BNN vs. B-SNN RRAM based "In-Memory" computing kernel. **(A)** BNN RRAM array. **(B)** B-SNN RRAM array. **(C)** BNN input encoding. **(D)** B-SNN input encoding. **(E)** BNN resistor state. **(F)** B-SNN resistor state.

4. EXPERIMENTS AND RESULTS

4.1. Datasets and Implementation

We evaluate our proposal on two popular, publicly available datasets, namely the CIFAR-100 (Krizhevsky et al., 2009) and large-scale ImageNet (Deng et al., 2009) dataset. CIFAR-100 dataset contains 100 classes with 60, 000 32 × 32 colored images where 50, 000 images were used for training and 10, 000 images were used for testing. The more challenging ImageNet 2012 dataset contains 1, 000 classes of images of various objects. The dataset contains 1.28 million training images and 50, 000 validation images. Randomly cropped 224 × 224 pixel regions were used for the ImageNet dataset. All empirical analysis and optimizations were performed on the CIFAR-100 dataset and the resultant conclusions and settings were used for the final ImageNet simulation. All experiments are run in PyTorch framework using two GPUs. For both datasets, the image pixels were normalized to have zero mean and unit variance. Other standard pre-processing techniques used in this work can be found at https://github.com/NeuroCompLab-psu/SNN-Conversion. It is worth mentioning here that while evaluations in this paper are based on static datasets, SNNs are inherently suited for spatio-temporal datasets generated from event-driven sensors (Amir et al., 2017; Li et al., 2017) and such sensor-algorithm co-design is currently an active research area (Rückauer et al., 2019).

Our network architecture follows a standard VGG-16 model. We purposefully chose the VGG architecture since many of the inefficiencies and accuracy degradation effects of BNNs are not reflected in shallower models like AlexNet or already-compact models like ResNet. However, we observed that VGG XNOR-Nets could not be trained successfully with 3 fully connected layers at the end. Hence, to reduce the training complexity, we considered a modified VGG-15 structure with one less linear layer. Note that only top-1 accuracies are reported in the paper.

As mentioned earlier, we used ANN-SNN conversion technique to generate our B-SNN. While ANN-SNN conversion is currently the most scalable technique to train SNNs, it suffers from high inference latency. However, recent work has shown SNNs trained directly through backpropagation are characterized by much lower latency than networks obtained through ANN-SNN conversion, albeit for simpler datasets and shallower networks (Lee et al., 2016). Due to the fact that such training schemes are computationally much more exhaustive, a follow-up work has explored a hybrid training approach comprising ANN-SNN conversion followed by backpropagation-through-time fine-tuning to scale the latency reduction effect to deeper networks (Rathi et al., 2020). However, as we show in this work, the full design space of ANN-SNN conversion has not been fully explored. Prior work on ANN-SNN conversion (Sengupta et al., 2019) has mainly considered conversion techniques optimizing accuracy, thereby incurring high latency. In this work, we show that there exists extremely simple control knobs (both at design time and at run time) that can be also used to reduce inference latency drastically in ANN-SNN conversion methods without compromising on the accuracy or involving computationally expensive training/fine-tuning approaches. Since our SNN training optimizations are equally valid for full-precision networks, we report accuracies for full-precision models along with their binary counterparts in order to compare against prior art.

Our ANNs were trained with constraints of no bias and batch-normalization layers in accordance with previous work (Sengupta et al., 2019). A dropout layer was inserted after every ReLU layer (except those followed by a pooling layer) to aid the regularization process in absence of batch-normalization. Our XNOR and B-SNN networks do not binarize the first and last layers as in previous BNN implementations. We apply the pixel intensities directly as input to the spiking networks instead of an artificial Poisson spike train (Rueckauer et al., 2017). Once the ANN is trained, it is converted to an iso-architecture SNN by replacing the ReLUs with IF spiking neuron nodes. The SNN weights are normalized by using a randomly sampled subset of images from the training set and recording the maximum ANN activities. Note that normalization based on SNN activities can be used to further reduce the ANN-SNN accuracy gap (Sengupta et al., 2019). The SNN implementation is done using a modified version of the mini-batch processing enabled SNN simulation framework (Saunders et al., 2019) in BindsNET (Hazan et al., 2018), a PyTorch based package.

4.2. Training B-SNNs

In order to train the B-SNN, we first trained a constrained-BWN, as mentioned previously. ADAM optimizer is used with an initial learning rate of $5e-4$ and a batch size of 128. Lower learning rates for training binary nets have proven to be also effective in a recent study (Tang et al., 2017). The learning rate is subsequently halved every 30 epochs for a duration of 200 epochs. The weight decay starts from $5e-4$ and is then set to 0 after 30 epochs similar to XNOR-Net training implementations (Rastegari et al., 2016). As shown in **Figure 3**, we find that the final validation accuracy improvement for the constrained-BWN is minimal over an iso-architecture XNOR-Net. This is primarily due to the constrained nature of models suitable for ANN-SNN conversion coupled with weight binarization.

However, previous work has indicated that careful weight initialization is crucial for training networks without batch-normalization (Sengupta et al., 2019). Drawing inspiration from that observation, we performed a hybrid training approach, where a constrained full-precision model was first trained and then subsequently binarized with respect to the weights. The resultant constrained-BWNs exhibited accuracies close to original full-precision accuracies, as shown in **Figure 3**. A similar hybrid training approach was also recently observed to speed up the training process for normal BNNs (Alizadeh et al., 2019). Note that the full precision networks are trained for 200 epochs with a batch size of 256, an initial learning rate of $5e-2$, weight decay of $1e-4$ and SGD optimizer with a momentum of 0.9. The learning rate was divided by 10 at 81 and 122 epochs. The trained full-precision models are also used for substantiating the benefits of the SNN optimization control knobs discussed next.

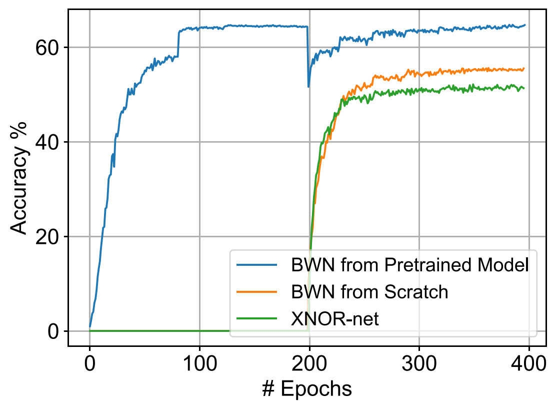


FIGURE 3 | Validation results on CIFAR-100 dataset. Note that full-precision model training is plotted from 0 to 200 epochs. The constrained-BWN model is trained subsequently from the 200-th epoch. The BWN model trained from scratch and XNOR-Net convergence plots are also shown for comparison.

4.3. Design-Time SNN Optimizations

4.3.1. Architectural Options

An important design option in the SNN/BNN architecture is the type and location of pooling mechanism. Normal deep networks usually have pooling layers after the neural node layer to compress the feature map. Among the two options typically used—Max Pooling and Average Pooling—architectures with Max Pooling are usually characterized by higher accuracy. However, because of the binary nature of neuron outputs in BNN/SNN, Max Pooling after the neuron layer should result in accuracy degradation. To circumvent this issue, BNN literature has explored using Max Pooling before the neuron layer (Rastegari et al., 2016) while SNN literature has considered Average Pooling after the neuron layer (Sengupta et al., 2019). A comprehensive analysis in this regard is missing.

In this work, we trained network architectures with four possible options—Average/Max-Pooling before/after the ReLU/IF neuron layer (**Figure 4**). All four constrained-BWN architectures perform similarly on CIFAR-100, as full-precision ANNs, and converge to accuracy of $64.9\%, 65.8\%, 67.7\%$ and 67.6% for Average-Pooling before and after ReLU, Max-Pooling before and after ReLU respectively. As expected, the Max-Pooling architectures perform slightly better. However, converted SNNs with Max-Pooling would result in accuracy degradation during the conversion process since the max-pooling operation is not distributed linearly over time. In contrast, the linear Avg-Pooling operation would not involve such issues during the conversion process. This tradeoff was evaluated in this design space analysis. We would like to mention here that two architectural modifications were performed while converting the constrained-BWN to B-SNN. First, as shown in **Figure 4B**, an additional IF layer was added after the Average-Pooling layer to ensure that the input to the next Convolutional layer is binary (to utilize the underlying binary hardware primitive). Also, for the Max-Pooling before ReLU option (**Figure 4C**), we inserted an additional IF neuron layer after the Convolutional layer. We observed that absence of this additional layer resulted in extremely low SNN accuracy (33%). We hypothesize this to occur due to Max-Pooling the Convolutional outputs directly over time at every time-step.

The variation of SNN accuracy with time-steps is plotted in **Figure 5** for full-precision and B-SNN models respectively. While the baseline ANN Max-Pooling architectures provide better accuracies, they undergo higher accuracy degradation during the conversion process. For the Average-Pooling models, the option with pooling after the neuron layer have higher latency due to additional spiking neuron layers. We find that the Average-Pooling before ReLU/IF neuron layer offers the best tradeoff between inference latency and final accuracy. We therefore chose this design option for the next set of experiments. Note that **Figure 5** shows the convergence graph for this architecture. Similar variation was also observed for the other options. For this architecture option, the full-precision (binary) SNN accuracy is **63.2%** (**63.7%**) in contrast to full-precision (binary) ANN accuracies of **64.9%** (**64.8%**).

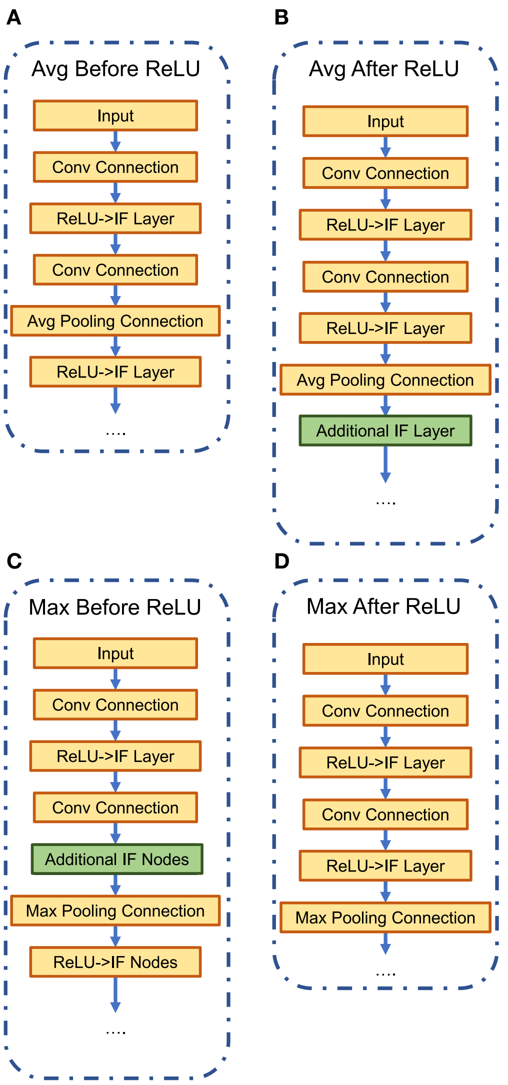


FIGURE 4 | Network architectural options: average/max-pooling before/after ReLU/IF neuron layers. **(A)** Avg-Pooling Before ReLU. **(B)** Avg-Pooling After ReLU. **(C)** Max-Pooling Before ReLU. **(D)** Max-Pooling After ReLU.

4.3.2. Neural Node Options

Another underexplored SNN architecture option is the choice of the spiking neuron node. While prior literature has mainly considered IF neurons where the membrane potential is reset upon spiking, Rueckauer et al. (2017) considers the membrane potential subtracted by the threshold voltage at a firing event. We will refer to the two neuron types as Reset-IF (RIF) and Subtractive-IF (SIF) respectively. SIF neurons assist in reducing the accuracy degradation of converted SNNs by removing the discontinuity occurring in the neuron function at a firing event (Rueckauer et al., 2017). However, this is achieved at the cost of higher spiking activity. We would like to stress here that while SNNs reduce the power consumption due to time-domain redistribution of computation, optimizing the SNN energy consumption is a tradeoff between the power benefits

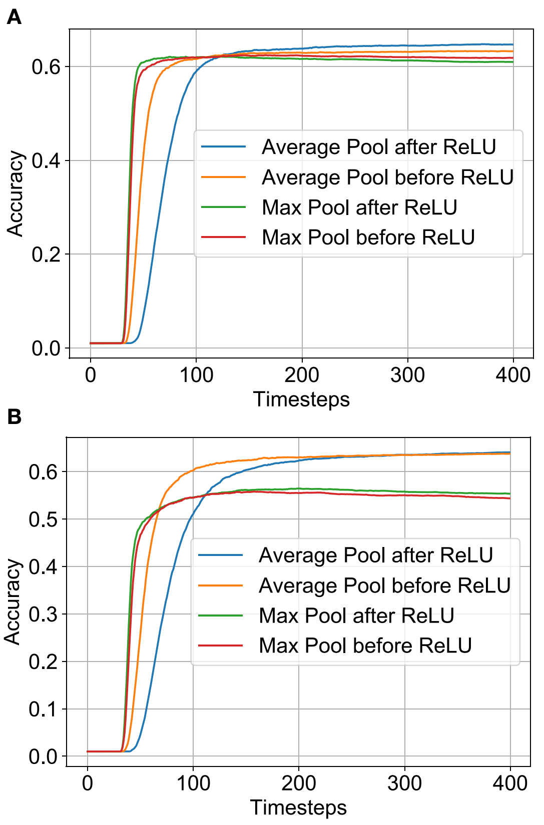


FIGURE 5 | SNN accuracy variation with time-steps on CIFAR-100 dataset for various architectural options. Note that all SNNs used here have Subtractive-IF layers. **(A)** Full-precision SNN. **(B)** Binary SNN.

and latency overhead—which is a function of such architectural options considered herein.

For our analysis, we consider the following proxy metrics for the energy consumption of the ANN and SNN. Assuming that the major energy consumption would occur in the "In-Memory" crossbar arrays discussed previously, the energy consumption of the ANN will be proportional to the sum of the number of operations in the convolutional and linear layers (due to corresponding activations of the rows of the crossbar array). However, in case of SNNs, the operation is conditional in the case of a spiking event. The calculation for ANN operations in convolutional and linear layers are performed using Equations (3) and (4).

$$\text{Convolution Layer \#OPS} = nIP * kH * kW * nOP * oH * oW \quad (3)$$

$$\text{Linear Layer \#OPS} = iS * oS \quad (4)$$

where, nIP is the number of input planes, kH and kW are the kernel height and width, nOP is the number of output planes, oH and oW are the output height and width, and iS and oS are the input and output sizes for linear layers.

In order to measure the efficiency of the SNN with respect to ANN in terms of energy consumption, we use the following Normalized Operations count henceforth.

$$\text{Normalized \#OPS} = \frac{\sum_{i=2}^{L-1} \text{IFR}_i * \text{Layer \#OPS}_{i+1}}{\sum \text{Layer \#OPS}} \quad (5)$$

where, IFR stands for IF Spiking Rate (total number of spikes over the inference time window averaged over number of neurons), and Layer #OPS include the operation counts in convolution and linear layers. L represents the total number of layers in the network. Note that, lower the value of normalized operations, higher is the energy efficiency of converted SNN, with 1 reflecting iso-energy case. Note that we do not consider the operation count for the first and last layers since they are not binarized.

Considering a baseline accuracy of 62%, **Figures 6A,C** shows that the SNN structure with SIF has a much smaller delay than the RIF structure. This is intuitive since the spiking rate is much higher in SIF due to subtractive reset. We also observed that the RIF topology was more error-prone during conversion due to the discontinuity occurring on reset to zero. For instance, the full-precision RIF SNN model was unable to reach 62% during 400 timesteps. The total number of normalized operations for SIF and RIF are 4.40 and 4.38 respectively for the B-SNN implementation, and 2.35 and 6.40 (did not reach 62% accuracy) respectively for the full-precision SNN. The layerwise spiking activity is plotted in **Figures 6B,D** (the numbers in figure inset represent the timesteps required to reach 62% accuracy). Since the number of computations does not greatly increase for the SIF model with significantly less delay and better accuracy, we choose the SIF model for the remaining analysis.

4.4. Run-Time SNN Optimizations

4.4.1. Threshold Balancing Factor

Prior work has usually considered the maximum activation of the ANN/SNN neuron as the neuron firing threshold for a particular layer, as explained in section 4.3. **Figure 7A** plots the histogram of the maximum ANN activations of a particular layer. The distribution is characterized by a long tail (**Figure 7B**) which results in an unnecessarily high SNN threshold, since most of the actual SNN activations would be much lower at inference time. This observation was consistently observed for other layers. Hence, while prior work has shown ANN-SNN conversion to be characterized by extremely high latency, it is due to the fact that the model is optimized for high accuracy, which translates to high latency. In this work, we analyze the effect of varying the threshold balancing factor by choosing a particular percentile from the activation histogram.

Figures 8A,C depicts the variation of accuracy vs. timesteps for different percentiles chosen from the activation histogram during threshold balancing. It is obvious that the network's latency reduces as the normalization percentile decreases due to a less conservative threshold choice. However, the accuracy degradation due to threshold relaxation seems to be minimal. Furthermore, no significant change in the number of computations are observed despite changing percentiles for both the full-precision and binary SNN models, as shown

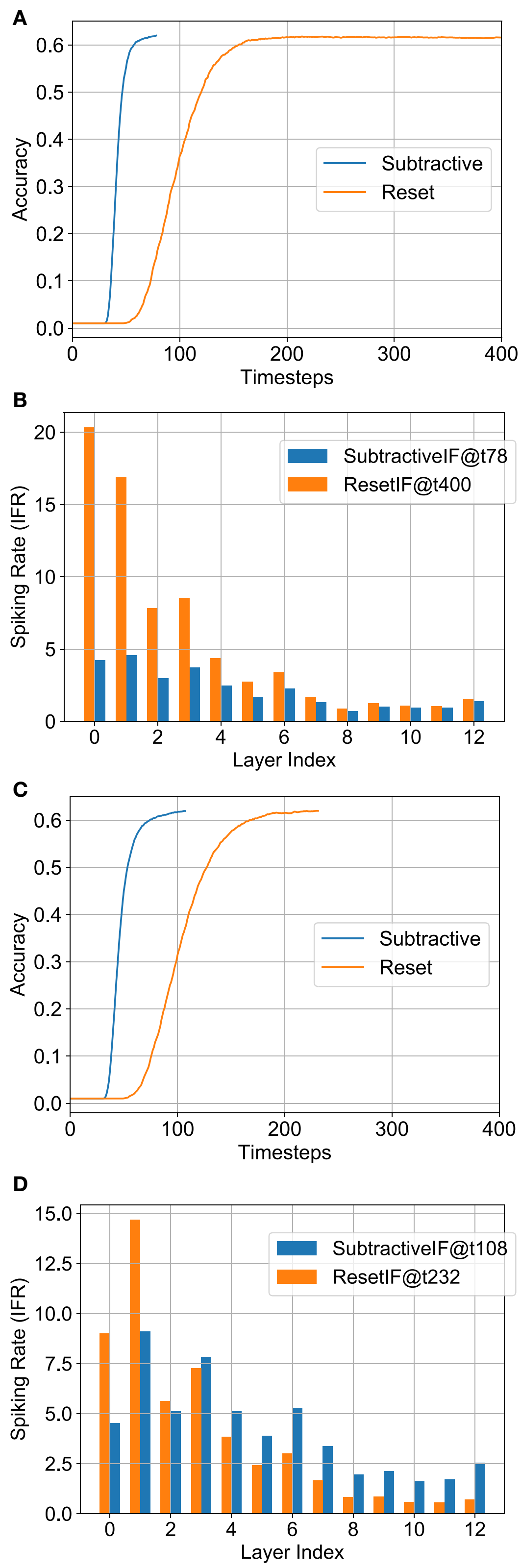


FIGURE 6 | Analysis for neural node options—SIF vs. RIF. **(A)** Accuracy vs. timesteps for full-precision model. **(B)** Layerwise IFR for full-precision model. **(C)** Accuracy vs. timesteps for binary model. **(D)** Layerwise IFR for binary model.

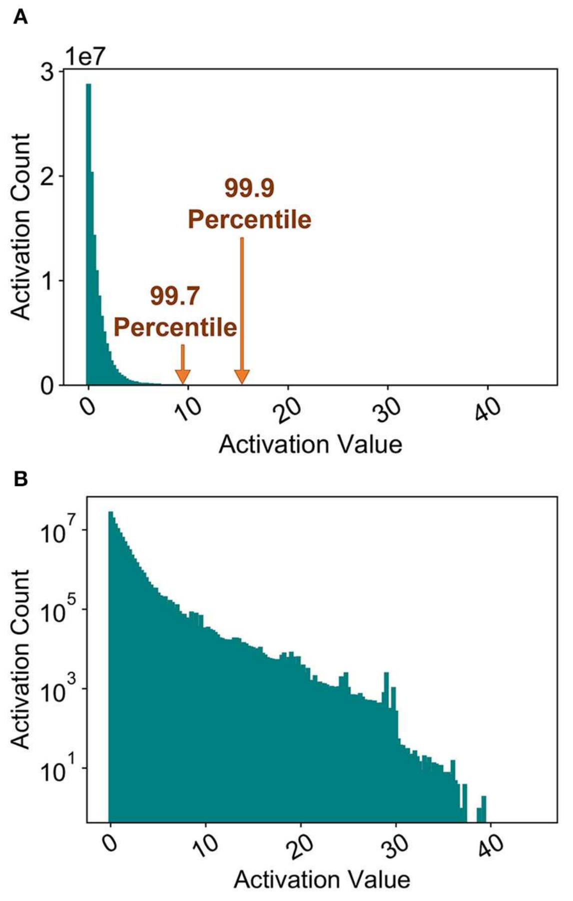


FIGURE 7 | Maximum ANN activations for a particular layer. **(A)** Histogram. **(B)** Histogram (a) in log-scale.

in **Figures 8B,D**. The number of timesteps required to reach 62% accuracy are also noted in the figure. We chose 99.7 percentile (a subset of 3, 500 training set images were used for measuring the statistics) for our remaining analysis since degradation in accuracy was observed for lower values in case of the binary model.

In order to explore additional opportunities for reduction in number of computations for the SNN models, we observed that the number of computations increases exponentially after a certain limit ∼60% accuracy. This has been plotted in **Figure 9** (combination of data shown in **Figures 8C,D**). Hence, computation costs for the B-SNN can be significantly reduced with small relaxation of the accuracy requirement. This is a major flexibility in our proposal unlike prior mixed-precision network proposals to circumvent the accuracy degradation issue of XNOR-Nets. The core hardware framework and operation remains almost similar to the XNOR-Net implementation with the flexibility to increase accuracy to full-precision limits as desired.

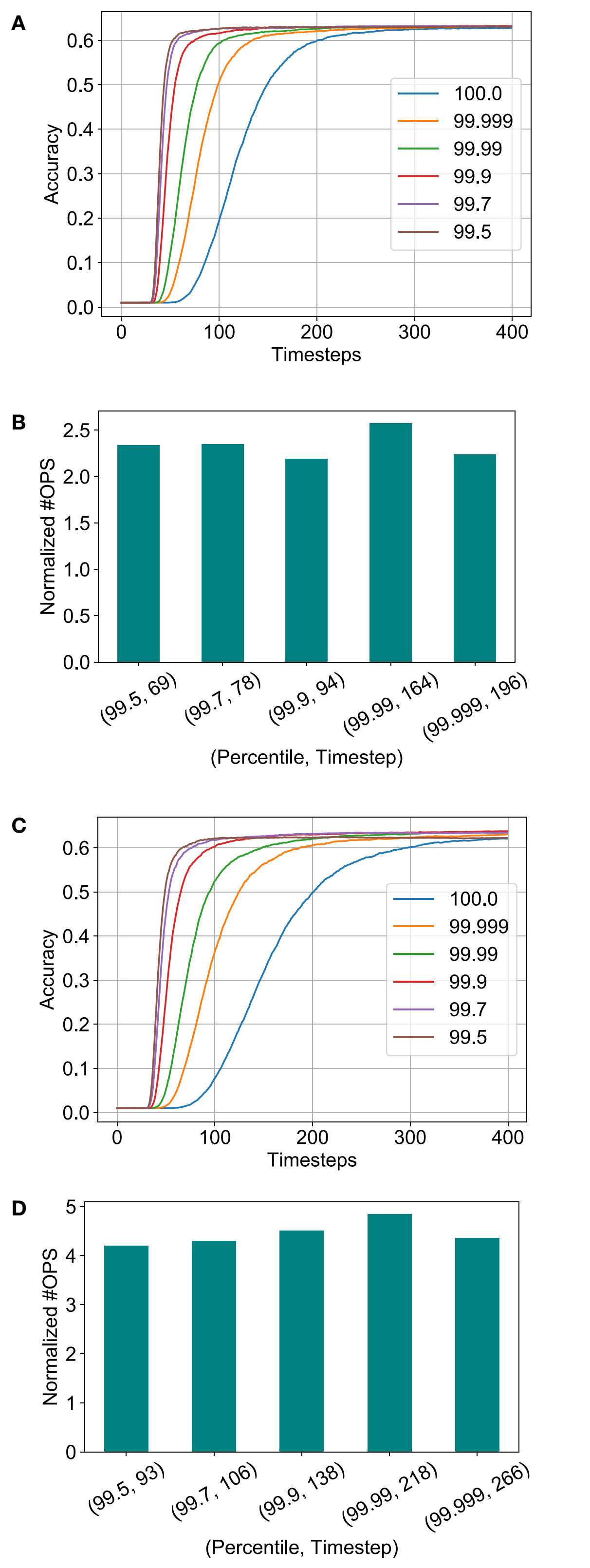


FIGURE 8 | Analysis for threshold balancing factor. **(A)** Accuracy vs. timesteps for full-precision model. **(B)** Normalized #OPS with varying percentile for the full-precision model. **(C)** Accuracy vs. timesteps for binary model. **(D)** Normalized #OPS with varying percentile for the binary model.

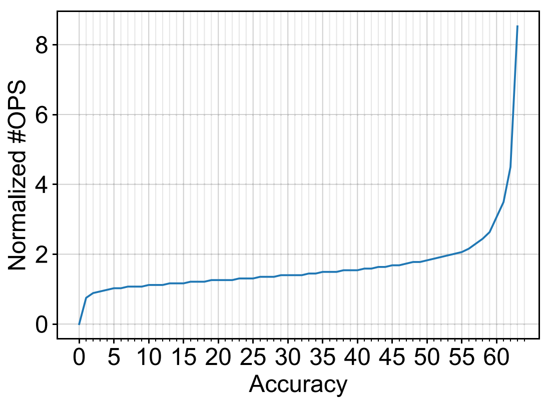


FIGURE 9 | Normalized #OPS of B-SNN as a function of accuracy.

4.4.2. Early Inference

Another conclusion obtained from the exponential increase in number of computations with accuracy beyond ~60% (**Figure 9**) is that a few difficult images require longer evidence integration for the SNN. However, it is unnecessary to run the SNN for an extended period of time for easy image instances that could have been classified earlier. Driven by this observation, we explored an "Early Exit" inference method for SNNs wherein we consider the SNN inference to be completed when the maximum membrane potential of the neurons in the final layer is above a given threshold[3]. This results in a dynamic SNN inference process where easier instances resulting in faster evidence integration can be classified earlier, thereby reducing the average inference latency and, in turn, the number of unnecessary computations.

Figures 10A,B depicts the variation of final SNN accuracy with the confidence threshold value for the maximum membrane potential of the final B-SNN layer. This optimization is equally applicable for the full-precision model. We considered that in the worst case the SNN runs for 105 timesteps (time required to reach baseline accuracy of 62%—obtained from **Figure 8C**). Indeed, we observed a reduction in computation from **4.30** to **3.55** with early inference without any compromise in accuracy (62%) for the binary model. The histogram of the required inference timesteps is shown in **Figure 11**. The average value of inference timesteps is **62.4**, which is significantly lower than **105** for the case without early exit. As a comparison point, we can achieve the XNOR-Net accuracy (47.16%) with threshold value 0.90 as shown in **Figure 10B**, and the corresponding number of normalized computations is 1.49 as compared to that of 1.0 of XNOR. Note that the 50% increment in computations for the XNOR-Net accuracy is a result of the fact that our model was optimized for a baseline accuracy of 62%. Hence, relaxing constraints explained in the previous subsections can potentially be used for the B-SNN to achieve XNOR-Net level accuracy at iso-computations. The

[3]It is worth noting here the final neuron layer does not have any intrinsic membrane potential threshold, i.e., the membrane potential accumulates over time. Normal inference involves determination of the neuron with maximum membrane potential.

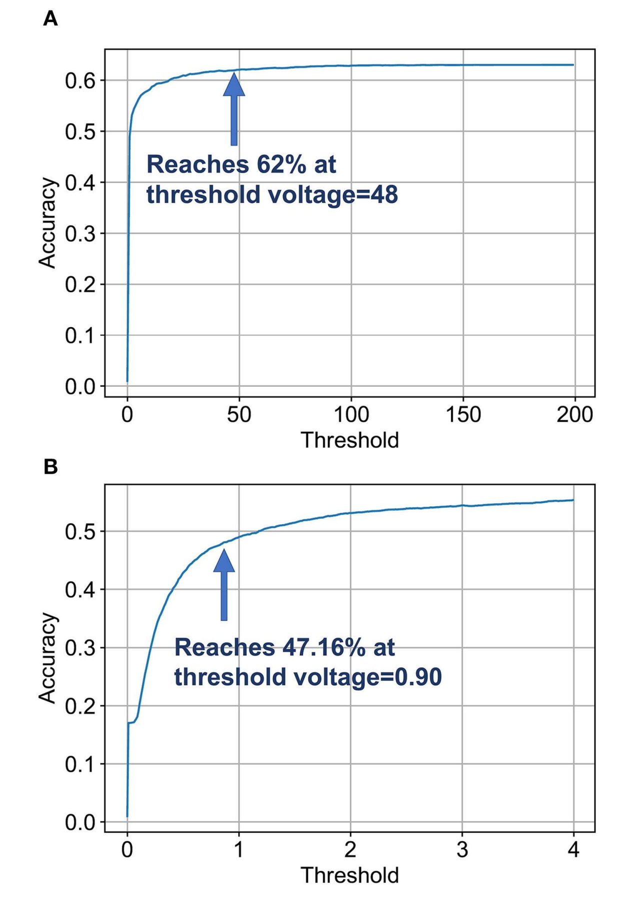


FIGURE 10 | Analysis for early inference. **(A)** The accuracy reaches 62% at around voltage of 48 and reaches 63% at around voltage of 106. **(B)** Panel **(A)** at finer granularity. It reaches XNOR accuracy at threshold value 0.90.

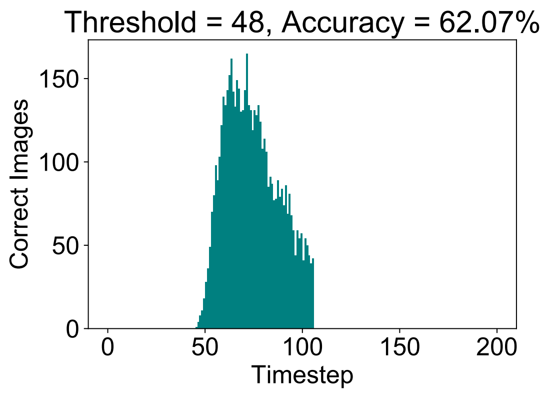


FIGURE 11 | Evaluation with Early Inference. Note that the there are 803 images predicted at the 105-th timestep which is not included in the graph.

results for CIFAR-100 dataset are summarized in **Table 1**. The B-SNN VGG model achieves near full-precision accuracy while only requiring 3.55× more operations integrated over the entire inference time window.

TABLE 1 | Results for CIFAR-100 Dataset.

Network model	Accuracy (%)	Normalized #OPS
Full precision ANN	64.9	32
XNOR Net	47.16	1
B-SNN	62.07	3.55

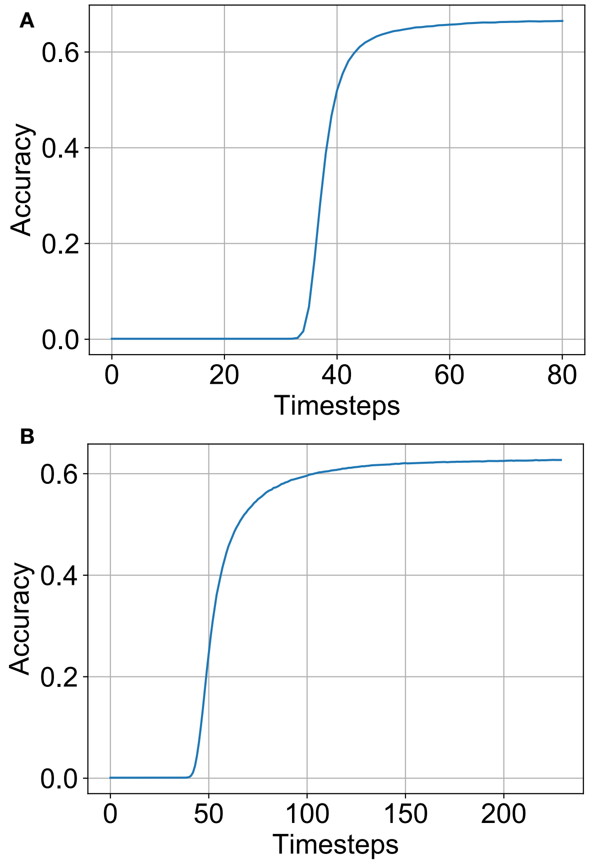


FIGURE 12 | Performance on the ImageNet dataset. **(A)** Accuracy vs. timesteps for full-precision model. **(B)** Accuracy vs. timesteps for binary model.

4.5. ImageNet Results

The full-precision VGG-15 model is trained on ImageNet dataset for 100 epochs with a batch size of 128, a learning rate of $1e-2$, a weight decay of $1e-4$ and the SGD optimizer with a momentum of 0.9. Note that the learning rate was divided by 10 at 30, 60, and 90 epochs similar to that of CIFAR-100 training. The final top-1 accuracy of the full-precision ANN was 69.05%.

Similarly, we binarized the network from the pre-trained ANN using the hybrid methodology described previously and we also observed a drastic increase in B-SNN accuracy (in contrast to training the model from scratch) similar to **Figure 3**. The initial parameters used for the Adam optimizer were learning rate of $5e-4$, weight decay of $5e-4$ (and 0 after 30 epochs), and beta values (the decay rates of the exponential moving averages) of (0.0, 0.999). Note that we observed proper setting of the

TABLE 2 | Results for ImageNet dataset.

Network model	Accuracy (%)	Timesteps
Full precision ANN	69.05	–
XNOR Net	49.77	–
Full precision SNN [ANN-SNN conversion (Rathi et al., 2020)]	62.73	250
Full precision SNN [Hybrid training (Rathi et al., 2020)]	65.19	250
Full precision SNN (This work)	66.56	64
B-SNN (This work)	62.71	148

beta values to be crucial for higher accuracy of the B-SNN training, as suggested in a recent work (Alizadeh et al., 2019). We achieved 65.4% top-1 accuracy for the constrained BWN model after 40 epochs of training (the binarization phase after full-precision training).

Optimization settings derived from the previous CIFAR-100 experiments were applied to our ImageNet analysis, namely, the pooling architecture, neural node type and relaxing the threshold balancing (99.9% percentile was used by recording maximum ANN activations for a subset of 80 images from the training set). The top-1 SNN (ANN) accuracy was 66.56% (69.05%) for the full-precision model and 62.71% (65.4%) for the binarized model respectively. The accuracy vs. timesteps variation for the two models are depicted in **Figures 12A,B**. The binary SNN model achieves near full-precision accuracy on the ImageNet dataset as well with 5.09 Normalized #OPS count. Note that the latency and #OPS count can be further reduced by early exit. We did not include the early exit optimization in order to achieve a fair comparison with previous works. A summary of our results on the ImageNet dataset and results from other competing approaches are shown in **Table 2**. Apart from the B-SNN proposal, our simple optimization procedures involving standard non-spiking network based training is able to achieve extremely low-latency deep SNNs.

5. CONCLUSIONS AND FUTURE WORK

While most of the current efforts at solving the accuracy degradation issue of BNNs have been focused on mixed-precision networks, we explore an alternative time-domain encoding procedure by exploring synergies with SNNs. ANN-SNN conversion provides a mathematical formulation for expressing multi-bit precision of ANN activations as binary values over time. Our binary SNN models achieve near full-precision accuracies on large-scale image recognition datasets, while utilizing similar hardware backbone of BNN-catered "In-Memory" computing platforms. Further, we explore several design-time and run-time optimizations and perform extensive empirical analysis to demonstrate high-accuracy and low-latency SNNs through ANN-SNN conversion techniques. Future work will explore algorithms to reduce the accuracy gap between full-precision and binary SNNs even further along with substantiating the generalizability of the proposal to advanced network architectures like residual connections (that may require additional design considerations Sengupta et al., 2019). Further, hardware benefits of the B-SNN proposal against mixed/reduced-precision implementations will be evaluated.

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author/s.

AUTHOR CONTRIBUTIONS

AS developed the main concepts. SL performed all the simulations. All authors assisted in the writing of the paper and developing the concepts.

FUNDING

This work was supported in part by the National Science Foundation.

REFERENCES

Akopyan, F., Sawada, J., Cassidy, A., Alvarez-Icaza, R., Arthur, J., Merolla, P., et al. (2015). Truenorth: design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE Trans. Comput. Aided Design Integr. Circ. Syst*. 34, 1537–1557. doi: 10.1109/TCAD.2015.2474396

Alizadeh, M., Fernández-Marqués, J., Lane, N. D., and Gal, Y. (2019). "An empirical study of binary neural networks' optimisation," in *International Conference on Learning Representations* (New Orleans). Available online at: https://openreview.net/forum?id=rJfUCoR5KX

Alvarez, J. M., and Salzmann, M. (2017). "Compression-aware training of deep networks," in *Advances in Neural Information Processing Systems*, eds I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Long Beach, CA: Curran Associates, Inc.), 856–867.

Amir, A., Taba, B., Berg, D., Melano, T., McKinstry, J., Di Nolfo, C., et al. (2017). "A low power, fully event-based gesture recognition system," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Honolulu, HI), 7243–7252. doi: 10.1109/CVPR.2017.781

Ankit, A., Sengupta, A., Panda, P., and Roy, K. (2017). "Resparc: a reconfigurable and energy-efficient architecture with memristive crossbars for deep spiking neural networks," in *Proceedings of the 54th Annual Design Automation Conference* (Austin, TX), 1–6. doi: 10.1145/3061639.3062311

Biswas, A., and Chandrakasan, A. P. (2018). "Conv-RAM: an energy-efficient SRAM with embedded convolution computation for low-power CNN-based machine learning applications," in *2018 IEEE International Solid-State Circuits Conference-(ISSCC)* (San Francisco, CA: IEEE), 488–490. doi: 10.1109/ISSCC.2018.8310397

Cass, S. (2019). Taking AI to the edge: Google's TPU now comes in a maker-friendly package. *IEEE Spectrum* 56, 16–17. doi: 10.1109/MSPEC.2019.8701189

Chakraborty, I., Roy, D., Garg, I., Ankit, A., and Roy, K. (2019). PCA-driven hybrid network design for enabling intelligence at the edge. *arXiv [Preprint], arXiv:1906.01493*. doi: 10.1038/s42256-019-0134-0

Choi, J., Wang, Z., Venkataramani, S., P. Chuang, I.-J., Srinivasan, V., and Gopalakrishnan, K. (2018). Pact: parameterized clipping activation for quantized neural networks. *arXiv [Preprint], arXiv:1805.06085*.

Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., and Bengio, Y. (2016). Binarized neural networks: training deep neural networks with weights and activations constrained to $+1$ or -1. *arXiv [Preprint], arXiv:1602.02830*.

Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S. H., et al. (2018). Loihi: a neuromorphic manycore processor with on-chip learning. *IEEE Micro* 38, 82–99. doi: 10.1109/MM.2018.112130359

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). "Imagenet: a large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition* (Miami Beach, FL: IEEE), 248–255. doi: 10.1109/CVPR.2009.5206848

Deng, L., Jiao, P., Pei, J., Wu, Z., and Li, G. (2018). Gxnor-net: training deep neural networks with ternary weights and activations without full-precision memory under a unified discretization framework. *Neural Netw*. 100, 49–58. doi: 10.1016/j.neunet.2018.01.010

Deng, L., Wu, Y., Hu, X., Liang, L., Ding, Y., Li, G., et al. (2020). Rethinking the performance comparison between SNNs and ANNs. *Neural Netw*. 121, 294–307. doi: 10.1016/j.neunet.2019.09.005

Diehl, P. U., Neil, D., Binas, J., Cook, M., Liu, S.-C., and Pfeiffer, M. (2015). "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *2015 International Joint Conference on Neural Networks (IJCNN)* (Killarney, IL: IEEE), 1–8. doi: 10.1109/IJCNN.2015.7280696

Han, B., Ankit, A., Sengupta, A., and Roy, K. (2017). Cross-layer design exploration for energy-quality tradeoffs in spiking and non-spiking deep artificial neural networks. *IEEE Trans. Multi Scale Comput. Syst*. 4, 613–623. doi: 10.1109/TMSCS.2017.2737625

Han, S., Pool, J., Tran, J., and Dally, W. (2015). "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processing Systems*, eds C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Montreal, CA: Curran Associates, Inc.), 1135–1143.

Hazan, H., Saunders, D. J., Khan, H., Patel, D., Sanghavi, D. T., Siegelmann, H. T., et al. (2018). Bindsnet: a machine learning-oriented spiking neural networks library in Python. *Front. Neuroinform*. 12:89. doi: 10.3389/fninf.2018.00089

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. (2017). Quantized neural networks: training neural networks with low precision weights and activations. *J. Mach. Learn. Res*. 18, 6869–6898. doi: 10.5555/3122009.3242044

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., and Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and $<$ 0.5 MB model size. *arXiv [Preprint], arXiv:1602.07360*.

Indiveri, G., Chicca, E., and Douglas, R. (2006). A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *IEEE Trans. Neural Netw*. 17, 211–221. doi: 10.1109/TNN.2005.860850

Krizhevsky, A., Nair, V., and Hinton, G. (2009). *Cifar-100*. Canadian Institute for Advanced Research. Available online at: https://www.cs.toronto.edu/~kriz/cifar.html

Lee, J., Delbruck, T., and Pfeiffer, M. (2016). Training deep spiking neural networks using backpropagation. *Front. Neurosci*. 10:508. doi: 10.3389/fnins.2016.00508

Li, F., Zhang, B., and Liu, B. (2016). Ternary weight networks. *arXiv [Preprint], arXiv:1605.04711*.

Li, H., Liu, H., Ji, X., Li, G., and Shi, L. (2017). Cifar10-dvs: an event-stream dataset for object classification. *Front. Neurosci*. 11:309. doi: 10.3389/fnins.2017.00309

Maass, W. (1997). Networks of spiking neurons: the third generation of neural network models. *Neural Netw*. 10, 1659–1671. doi: 10.1016/S0893-6080(97)00011-7

Pfeiffer, M., and Pfeil, T. (2018). Deep learning with spiking neurons: opportunities and challenges. *Front. Neurosci*. 12:774. doi: 10.3389/fnins.2018.00774

Prabhu, A., Batchu, V., Gajawada, R., Munagala, S. A., and Namboodiri, A. (2018). "Hybrid binary networks: optimizing for accuracy, efficiency and memory," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)* (Lake Tahoe, NV/CA: IEEE), 821–829. doi: 10.1109/WACV.2018.00095

Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. (2016). "Xnor-net: imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision* (Amsterdam: Springer), 525–542. doi: 10.1007/978-3-319-46493-0_32

Rathi, N., Srinivasan, G., Panda, P., and Roy, K. (2020). "Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation," in *International Conference on Learning Representations* (Virtual Conference).

Rückauer, B., Känzig, N., Liu, S.-C., Delbruck, T., and Sandamirskaya, Y. (2019). Closing the accuracy gap in an event-based visual recognition task. *arXiv [Preprint], arXiv:1906.08859*.

Rueckauer, B., Lungu, I.-A., Hu, Y., Pfeiffer, M., and Liu, S.-C. (2017). Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Front. Neurosci*. 11:682. doi: 10.3389/fnins.2017.00682

Saunders, D. J., Sigrist, C., Chaney, K., Kozma, R., and Siegelmann, H. T. (2019). Minibatch processing in spiking neural networks. *arXiv [Preprint], arXiv:1909.02549*.

Sengupta, A., Shim, Y., and Roy, K. (2016). Proposal for an all-spin artificial neural network: Emulating neural and synaptic functionalities through domain wall motion in ferromagnets. *IEEE Trans. Biomed. Circ. Syst*. 10, 1152–1160. doi: 10.1109/TBCAS.2016.2525823

Sengupta, A., Srinivasan, G., Roy, D., and Roy, K. (2018). "Stochastic inference and learning enabled by magnetic tunnel junctions," in *2018 IEEE International Electron Devices Meeting (IEDM)* (San Francisco, CA: IEEE), 15–6. doi: 10.1109/IEDM.2018.8614616

Sengupta, A., Ye, Y., Wang, R., Liu, C., and Roy, K. (2019). Going deeper in spiking neural networks: VGG and residual architectures. *Front. Neurosci*. 13:95. doi: 10.3389/fnins.2019.00095

Srinivasan, G., and Roy, K. (2019). Restocnet: Residual stochastic binary convolutional spiking neural network for memory-efficient neuromorphic computing. *Front. Neurosci*. 13:189. doi: 10.3389/fnins.2019.00189

Sun, X., Yin, S., Peng, X., Liu, R., Seo, J.-S., and Yu, S. (2018). "XNOR-RRAM: a scalable and parallel resistive synaptic architecture for binary neural networks," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (Dresden: International Congress Center Dresden), 1423–1428. doi: 10.23919/DATE.2018.8342235

Suri, M., Querlioz, D., Bichler, O., Palma, G., Vianello, E., Vuillaume, D., et al. (2013). Bio-inspired stochastic computing using binary CBRAM synapses. *IEEE Trans. Electron Devices* 60, 2402–2409. doi: 10.1109/TED.2013.2263000

Tang, W., Hua, G., and Wang, L. (2017). "How to train a compact binary neural network with high accuracy?" in *Thirty-First AAAI Conference on Artificial Intelligence* (San Francisco, CA).

Wang, K., Liu, Z., Lin, Y., Lin, J., and Han, S. (2019). "Haq: hardware-aware automated quantization with mixed precision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Long Beach, CA), 8612–8620. doi: 10.1109/CVPR.2019.00881

Wijesinghe, P., Ankit, A., Sengupta, A., and Roy, K. (2018). An all-memristor deep spiking neural computing system: a step toward realizing the low-power stochastic brain. *IEEE Trans. Emerg. Top. Comput. Intell*. 2, 345–358. doi: 10.1109/TETCI.2018.2829924

Wu, B., Wang, Y., Zhang, P., Tian, Y., Vajda, P., and Keutzer, K. (2018). Mixed precision quantization of convnets via differentiable neural architecture search. *arXiv [Preprint], arXiv:1812.00090*.

Xue, C.-X., Chen, W.-H., Liu, J.-S., Li, J.-F., Lin, W.-Y., Lin, W.-E., et al. (2019). "24.1 A 1Mb Multibit ReRAM computing-in-memory macro with 14.6 ns parallel MAC computing time for CNN based AI edge processors," in *2019 IEEE International Solid-State Circuits Conference-(ISSCC)* (San Francisco, CA: IEEE), 388–390. doi: 10.1109/ISSCC.2019.8662395

Yin, S., Sun, X., Yu, S., and Seo, J.-S. (2019). High-throughput in-memory computing for binary deep neural networks with monolithically integrated rRAM and 90nm CMOS. *arXiv [Preprint], arXiv:1909.07514*.

Zhang, D., Yang, J., Ye, D., and Hua, G. (2018). "Lq-nets: learned quantization for highly accurate and compact deep neural networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 365–382. doi: 10.1007/978-3-030-01237-3_23

Zhang, J., Wang, Z., and Verma, N. (2017). In-memory computation of a machine-learning classifier in a standard 6T SRAM array. *IEEE J. Solid State Circ*. 52, 915–924. doi: 10.1109/JSSC.2016.2642198

Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H., and Zou, Y. (2016). Dorefa-net: training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv [Preprint], arXiv:1606.06160*.

Zhou, S.-C., Wang, Y.-Z., Wen, H., He, Q.-Y., and Zou, Y.-H. (2017). Balanced quantization: an effective and efficient approach to quantized neural networks. *J. Comput. Sci. Technol*. 32, 667–682. doi: 10.1007/s11390-017-1750-y

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.