International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2026.0243
e ISSN: 2584-2854
Volume: 04 Issue: 05
May 2026
Page No: 1650 - 1655

A Comprehensive Study On ANN-To-SNN Conversion for Energy-Efficient Neural Network Deployment

Asmeena[1], Devanand P Ramesan[2], Febin Soyichan[3], Devadutt V[4], Adwaith T K[5], Jaseem K[6]
[1]Assistant Professor, Yenepoya Deemed to Be University, Bangalore, Karnataka, India.
[2,3,4,5,6]UG – BCA, Yenepoya Deemed to Be University, Bangalore, Karnataka, India.
Email ID: *asmeena.blr@yenepoya.edu.in[1], devanand0903@gmail.com[2], febinsoyichan36@gmail.com[3], devaduttvipin@gmail.com[4], adwaithtk27@gmail.com[5], Jaseemkoppilakath@gmail.com[6]*

Abstract
These days deep neural networks are being integrated into each and every device that exists, from smartphones to portable home devices that run on batteries. Even though these models are smarter, they are also very energy-consuming. This is very true in the case of standard Artificial Neural Networks (ANNs). In order to perform accurately, they never really turn off. They constantly process data in large amounts, which drains the batteries. If we want AI to work on portable edge devices without draining their batteries, we need a much more efficient way to handle the computing workload. This makes Spiking Neural Networks (SNNs) a strong alternative. They are different from regular neural networks because they are event-driven . It only generates signals when necessary to avoiding unnecessary activity, Which is similar to how neurons function in the human brain. This makes them far better for saving energy. One of the main challenges is that SNNs are difficult to train from scratch. Because of this, a practical solution is to train an Artificial Neural Network (ANN) first and then convert it into a Spiking Neural Network (SNN). In this paper, we will look into the total conversion of the ANN to SNN and the model performance. Our main goal is to implement high performance models in small devices without frequent failures and low power consumption. Also this research paper contains the current challenges and future research ideas for creating energy-efficient models

Keywords: *Deep Neural Networks; Artificial Neural Networks (ANNs); Spiking Neural Networks (SNNs); Energy-Efficient; Conversion Techniques*

1. Introduction
When we started to study about this topic, we came into the problem that the ANN models are using high power. These models are mainly focused on getting high accuracy without looking into power consumption. For small devices it can be a serious problem as it runs on small battery. While exploring ways to solve this problem, we found that Spiking Neural Networks (SNNs) . Further analysis revealed that, The process information in different way compared to conventional neural networks. Traditional neural networks work continuously, but SNNs follow an event-driven approach. This means that neurons only become active when a meaningful signal occurs. This behavior is inspired by the way neurons work in the human brain. In biological systems, neurons are communicate through short electrical spikes. SNNs try the same mechanism. Because computation only happens when spikes occur, the overall energy consumption can be much lower, especially when they run on neuromorphic hardware. The training SNNs directly is not easy. One of the main challenges is the spike generation function used in these networks is not differentiable. Since the most deep learning training methods rely on the gradient descent, because this makes the training process very complicated. To solve the issue, many researchers use a method known as ANN-to-SNN-conversion. Initially, a conventional Artificial Neural Network (ANN) undergoes training employing established deep learning methodologies. After

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2026.0243
e ISSN: 2584-2854
Volume: 04 Issue: 05
May 2026
Page No: 1650 - 1655

completing the training phase, the learned weights and architecture of the Artificial Neural Network (ANN) are converted into an equivalent Spiking Neural Network (SNN) representation. With this approach, researchers can use the proven training methods of ANNs while still achieving the energy efficiency offered by SNNs. In this paper we have researched on how the conversion actually works, which are the approaches that have been tried, which are the successful one, and what still needs to be solved. Our research paper covers a few main things. First, we will talk about the improvement in the conversion methods. Then about the speed and accuracy of the models. We also checked about the hardware for the SNN to be used in the real world. Finally we talk about the problems still out there and what we think will happen next.

1.1. Why We Chose This Topic

We noticed that the ANN models have high accuracy but it was using high power as well. So we needed to find ways in which we can build models that is accurate and also uses less power. If we look into real-world, IoT and embedded devices, uses high power and also the models we currently use require high power which cannot be used everywhere. So doing a study on Spiking Neural Network will help us with this problem, So it can be used in small devices which have limited hardware as well.

2. Related Work

We knew that Artificial Neural Network has grown rapidly. At the beginning, converting a traditional Artificial Neural Network (ANN) into a Spiking Neural Network (SNN) was mostly a research theory problem, as time passed researchers started solving the problem and many groups began trying to improve the methods. There are several key papers that demonstrate the current progress on this topic.

2.1. The Foundation: Rate-Coded Conversion

The conversion of an ANN to a SNN is based on a simple yet insightful observation. If you provide enough time steps, a LIF neuron will have an average firing rate that is very similar to a ReLU activation. Cao et al. [4] were the first to formally show that you can effectively cut and paste the weights from a trained ANN into an SNN. Since this paper came out, the basic idea of converting ANNs to SNNs is largely based on this work.

2.2. Threshold Balancing — The Breakthrough

Initially the attempts at ANN – SNN conversion, directly coping the weights from a trained ANN into a SNN often led to very poor results. Diehl et al. [3] later identifies that this problem is mainly caused by incorrectly set firing thresholds. If the threshold is too low, neurons tend to fire continuously, if it is too high, the neutron may not fire at all. Both cases result in extremely poor model performance. To solve this problem, they developed a technique known as threshold balancing. In this technique, the firing threshold of each layer is adjusted based on the activation statistics observed during the ANN training phase. This approach represented a major breakthrough in ANN – SNN conversion, achieving nearly perfect accuracy on the MNIST dataset at T = 400 time steps. As a result, the paper become one of the most widely cited works in this research area.

2.3. Scaling to Real-World Datasets

The next big issue was scaling up. Standard networks use things like batch normalization and max pooling, which don't have an obvious "spiking" version. Rueckauer et al [5] bridged this gap, showing that we could actually convert massive architectures like VGG-16 and Inception. But the problem was, It took over 2,500 time steps to stay accurate. So because of the large delay, the system became too slow for the real life applications, but it proved that the concept can scale to large neural networks.

2.4. Cutting the Time-Steps Down

In recent years, researchers have increasingly focused on improving the speed and efficiency of these networks. Han et al. [6] introduced a model called Residual Membrane Potential (RMP) Spiking Neural Networks, where neurons keep a small amount of their membrane potential even after they fire, rather than resetting entirely to zero. Although this change appears minor, it significantly improved performance, achieving 93.63% accuracy on the CIFAR – 10 dataset at T = 2048 time steps. In the next year, Deng and Gu [7] further advanced the field by introducing a method for optimizing weights and thresholds jointly; this was demonstrated to achieve good results on CIFAR-100 in only 256 time steps.

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2026.0243
e ISSN: 2584-2854
Volume: 04 Issue: 05
May 2026
Page No: 1650 - 1655

2.5. Hybrid Fine-Tuning Methods

At present, the best results are usually achieved through a "the best of two worlds" approach. Instead of performing a onetime conversion and using the model as it is, many researches now treat ANN – SNN conversion as an initial step and further refine the network using surrogate gradient training methods. An excellent example of this is DIET-SNN by Rathi and Roy [9]; DIET-SNN takes a converted SNN and refines it for an additional number of epochs. The resulting performance is quite high with very low latency at T=64. While hybrid methods clearly represent the gold standard for performance, there is a trade-off involved. In contrast to one-shot conversion methods, which allow a converted model to be deployed almost immediately, hybrid approaches typically include an additional fine tuning phase. This stage requires significantly more computational power along with a well-structured training environment. As a result, if the required infrastructure is not available, implementing such hybrid techniques in practical applications may be challenging.

3. Methodology

In this study, we built a four-stage pipeline for converting ANNs to SNNs. We pulled ideas from the papers we went through and tried to fix the common weaknesses we noticed in each individual approach.

3.1. The Neuron Model

The discrete-time Leaky Integrate-and-Fire (LIF) model will be used throughout. At each time-step t, the membrane potential of neuron i updates as:

u_(i)^t= λ · u_i^((t-1)) (1-s_i^(t-1))+Σ_j w_(ij)· s_j^t (1)

Here, λ is the leak factor, w is the synaptic weight, and s_j^t

is the binary spike input. A spike fires when the membrane potential exceeds threshold ϑ , and the potential is then I reduced by that threshold (reset-by-subtraction). Under these conditions, the mean firing rate of a neuron closely tracks the corresponding ANN ReLU output — which is the whole basis for conversion working at all.

3.2. Stage 1 — Training the ANN with Sparsity in Mind

One thing we learned quickly: the quality of the converted SNN depends heavily on how the source ANN was trained. If the ANN produces large, dense activations, the SNN will have neurons firing constantly — destroying the energy efficiency you were trying to achieve. We added an L1 penalty on intermediate ReLU activations to the training loss:

L = L (θ; D) + γ · Σ Σ E [a (l)(x)] (2)
CE l i x i

With γ = 10−5, the ANN accuracy during the training will not be hurt but it may significantly reduce the firing rates after conversion. That's the reason we felt like this step was basically getting the model ready for how it would actually behave once it became an SNN..

3.3. Stage 2 — Threshold Balancing

By following Diehl et al. [3], we have set the threshold of each layer using the 99.9th percentile of activations over 1,024 calibration samples:

ϑ(l) = p (a(l)) (3)
99.9

At the start of our tests, the threshold was so big because of few really big signals skipped connections and the after effect was, most of the neutrons just barely fired. Then we used a percentile, that made the problem to go away and made things more accurate by over 1%.

3.4. Stage 3 — Layer-wise Weight Rescaling

After threshold calibration, weights are rescaled to preserve the correct signal relationship across layers:

■(_ij^)(l)=w_ij^((l)) ·(ϑ_^((1-1)))/(ϑ_^((1))) (4)

Without this, deeper layers receive input currents mismatched with their calibrated thresholds, and accuracy degrades progressively with depth — especially visible in VGG-16.

Stage 4 — Adaptive Time-Step Selection

Rather than fixing T at a large default, we run a bisection search over T ∈ {8, 16, 32, 64, 128, 256} to find the smallest T satisfying an accuracy tolerance δ:

T* = min T s.t. Acc (T) ≥ Acc − δ (5)
SNN ANN

With δ = 0.5%, this converges in at most six validation passes. In practice, it let us say definitively: for CIFAR-10 at this accuracy budget, T = 32 is enough — no need for 2,500 steps.

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2026.0243
e ISSN: 2584-2854
Volume: 04 Issue: 05
May 2026
Page No: 1650 - 1655

4. Results And Discussion

4.1.Experimental Setup

We tested on CIFAR-10 and CIFAR-100 using VGG-16 [10] and ResNet-20 [11] as source architectures. ANNs were trained for 200 epochs with SGD (momentum 0.9, weight decay 10−4), cosine annealing LR starting at 0.1, and $\gamma = 10-5$. Experiments were implemented in PyTorch on an NVIDIA A100 GPU. Baselines: TB-SNN [3], RMP-SNN [6], OC-SNN [7], DIET-SNN [9]

4.2.Classification Accuracy

Table 1 shows top-1 accuracy results on CIFAR-10 and CIFAR-100 using VGG-16. The ANN baseline is 93.96% on CIFAR-10 and 74.31% on CIFAR-100.

Table 1 Top-1 Accuracy (%) — VGG-16

Method	T	CIFAR-10	CIFAR-100
ANN baseline	—	93.96%	74.31%
TB-SNN [3]	2500	91.55%	70.97%
RMP-SNN [6]	2048	93.63%	73.09%
OC-SNN [7]	256	93.71%	73.55%
DIET-SNN [9]	64	93.44%	74.24%
Ours	64	93.52%	74.08%
Ours	32	93.46%	73.87%

The number that stood out most to us was T = 32 on CIFAR-10: 93.46%, just 0.5% below the ANN. OC-SNN reaches similar accuracy but needs T = 256 — eight times more time-steps. On CIFAR-100 at T = 64, we got 74.08% compared to DIET-SNN's 74.24% — a gap of only 0.16% — without any post-conversion fine-tuning. That felt like a fair trade to us.

4.3. Energy Consumption

Table 2 shows normalised energy vs. the GPU ANN baseline. A value below 1.0× means the SNN is more efficient. TB-SNN at 3.41× was a genuine surprise — running 2,500 time-steps with a high firing rate costs more than the original ANN. SNNs are not automatically efficient; they only save energy when both firing rates and time-steps are kept low.

Table 2 Normalised Energy Consumption (ANN GPU = 1.00×)

Method	T	Fire Rate	Norm. Energy
ANN (MACs)	—	—	1.00×
TB-SNN	2500	0.62	3.41×
RMP-SNN	2048	0.51	2.18×
OC-SNN	256	0.44	0.47×
DIET-SNN	64	0.38	0.23×
Ours	64	0.31	0.19×
Ours	32	0.29	0.17×

Our result at T = 32 achieves a 5.8× energy reduction. The low firing rate of 0.29 is a direct result of the L1 sparsity regularisation applied during ANN training in Stage 1.

4.4.Ablation Study

Table 3 Ablation — CIFAR-10, ResNet-20, T = 32

Configuration	Acc.	Fire Rate	Norm. E
w/o Sparsity Reg.	91.32%	0.48	0.29×
w/o Wt. Rescaling	92.11%	0.33	0.19×
w/o Adaptive T*	91.87%	0.30	0.21×
Full Pipeline (Ours)	93.41%	0.29	0.17×

When we removed the sparsity regularization, there was the biggest performance drop in the model's accuracy. By this we understood that training an ANN model and Converting to an SNN Model is equally important. Then the four stages of conversion must be properly done. If any one of the stage went wrong then it will reduce the accuracy or the complete model's working.

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2026.0243
e ISSN: 2584-2854
Volume: 04 Issue: 05
May 2026
Page No: 1650 - 1655

4.5.Key Takeaways

We understood three things from our analysis. First, source network preparation is underrated — training the ANN with sparsity in mind is the single highest-impact step. Second, percentile-based threshold calibration is noticeably more stable than maximum-based calibration for deep networks with residual connections. Third, our pure-conversion approach trails DIET-SNN by only 0.16% on CIFAR-100, without requiring any fine-tuning — a worthwhile trade-off in compute-constrained settings.

Conclusion

In conclusion, this study explored different ANN-to-SNN conversion techniques and how they help make neural networks more energy efficient.

From the beginning our aim was to understand how ANN-to-SNN conversion works and how much energy could be saved. During the study, we found that the results were more detailed and interesting than expected. The conversion pipeline used in our work achieved an accuracy that was very close to the original ANN model, with only about a 0.5% difference at T=32. At the same time, the energy consumption was reduced by nearly 5.8 times compared to GPU-based inference. One important observation from our ablation study was that the way the source ANN is trained plays a major role in the final performance. Training the ANN while considering activation sparsity did not reduce its accuracy, but it significantly lowered the firing rates after conversion. When the value of firing activity decreases then we can understand the SNN Model is saving energy. That's how they are more efficient. We know that there is a big difference in the pure conversion techniques and hybrid methods. To make it pure conversion technique it require extra training. More studies could be done to test the framework on neuromorphic hardware like Intel Loihi.

Acknowledgements

We're really thankful to the Computer Department at Yenepoya Deemed to be University in Bangalore for this opportunity and all their help.Thanks a lot to the Computer department at Yenepoya Deemed to be University, Bangalore, for all their help and this opportunity.This was a big help.We also want to thank the researchers and developers who shared their datasets and code with everyone. That was a really big help for our project.That really gave our project a boost.

References

[1]. P. U. Diehl, D. Neil, J. Binas, M. Cook, S. C. Liu, and M.Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in Proc. IJCNN, 2015, pp. 1–8. https://ieeexplore.ieee.org/document/7280696

[2]. M. Davies et al., "Loihi: A neuromorphic manycore processorwith on-chip learning," IEEE Micro, vol. 38, no. 1, pp. 82–99, 2018. https://ieeexplore.ieee.org/document/8259423

[3]. P. A. Merolla et al., "A million spiking-neuron integrated circuitwith a scalable communication network and interface," Science, vol. 345, no. 6197, pp. 668–673, 2014. https://www.science.org/doi/10.1126/science.1254642

[4]. Y. Cao, Y. Chen, and D. Khosla, "Spiking deep convolutionalneural networks for energy-efficient object recognition," International Journal of Computer Vision, vol. 113, no. 1, pp. 54–66, 2015.https://link.springer.com/article/10.1007/s11263-014-0788-3

[5]. B. Rueckauer, I. Lungu, Y. Hu, M. Pfeiffer, and S. C. Liu,"Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," Frontiers in Neuroscience, vol. 11, p. 682, 2017. https://www.frontiersin.org/articles/10.3389/fnins.2017.00682/full

[6]. B. Han, G. Srinivasan, and K. Roy, "RMP-SNN: Residualmembrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network," in Proc. CVPR, 2020, pp. 1355813567.https://openaccess.thecvf.com/content_CVPR_2020/html/Han_RMP-SNN_Residual_Membrane_Potential_Neuron_for_Enabling_Deeper_HighAccuracy_and_CVPR_2020_ paper.html

[7]. S. Deng and S. Gu, "Optimal conversion of conventional artificialneural networks to spiking neural networks," in Proc. ICLR,2021.

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2026.0243
e ISSN: 2584-2854
Volume: 04 Issue: 05
May 2026
Page No: 1650 - 1655

https://openreview.net/forum?id=FZ1oTwcXchK

[8]. E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradientlearning in spiking neural networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 51–63, 2019. https://ieeexplore.ieee.org/document/8891809

[9]. N. Rathi and K. Roy, "DIET-SNN: A low-latency spiking neuralnetwork with direct input encoding and leakage and threshold optimization," IEEE Trans. Neural Networks and Learning Systems, vol. 34, no. 6, pp. 3174–3182, 2023.https://ieeexplore.ieee.org/document/9556508

[10]. K. Simonyan and A. Zisserman, "Very deep convolutionalnetworks for large-scale image recognition," in Proc. ICLR, 2015. https://arxiv.org/abs/1409.1556

[11]. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning forimage recognition," in Proc. CVPR, 2016, pp. 770–778. https://arxiv.org/abs/1512.03385