LightSNN: Lightweight Architecture Search for Sparse and Accurate Spiking Neural Networks

Yesmine Abdennadher*, Giovanni Perin^{†,*}, Riccardo Mazzieri*, Jacopo Pegoraro*, and Michele Rossi*

*Department of Information Engineering (DEI), University of Padova, Padova, Italy

†Department of Information Engineering (DII), University of Brescia, Brescia, Italy

Abstract—Spiking Neural Networks (SNNs) are highly regarded for their energy efficiency, inherent activation sparsity, and suitability for real-time processing in edge devices. However, most current SNN methods adopt architectures resembling traditional artificial neural networks (ANNs), leading to suboptimal performance when applied to SNNs. While SNNs excel in energy efficiency, they have been associated with lower accuracy levels than traditional ANNs when utilizing conventional architectures. In response, in this work we present LightSNN, a rapid and efficient Neural Network Architecture Search (NAS) technique specifically tailored for SNNs that autonomously leverages the most suitable architecture, striking a good balance between accuracy and efficiency by enforcing sparsity. Based on the spiking NAS network (SNASNet) framework, a cell-based search space including backward connections is utilized to build our trainingfree pruning-based NAS mechanism. Our technique assesses diverse spike activation patterns across different data samples using a sparsity-aware Hamming distance fitness evaluation. Thorough experiments are conducted on both static (CIFAR10 and CIFAR100) and neuromorphic datasets (DVS128-Gesture). Our LightSNN model achieves state-of-the-art results on CIFAR10 and CIFAR100, improves performance on DVS128Gesture by 4.49%, and significantly reduces search time most notably offering a 98× speedup over SNASNet and running 30% faster than the best existing method on DVS128Gesture. Code is available on Github at: https://github.com/YesmineAbdennadher/LightSNN.

I. Introduction

After the development of perceptrons and artificial neural networks (ANNs) [1], spiking neural networks (SNNs) [2] [3] emerge as the third generation of neural networks. SNNs mimic the behavior of biological neurons through discrete and time-dependent signals known as spikes. This makes them suitable for temporal (1D) and spatiotemporal (3D) data processing, offering better efficiency and reduced energy consumption compared to conventional neural networks. Because of the event-driven and low-power nature of SNNs, they have attracted major attention in the fields of edge computing, wearable devices, and signal processing at the physical layer of wireless systems. In fact, SNNs are eminently suitable for use in battery-powered or resource-limited devices due to their low energy consumption and real-time adaptation capabilities. Thanks to their amenability to on-chip implementation [4] [5] and low-latency handling of signals, they bear the promise

This work has been supported by the EU H2020 MSCA ITN project Greenedge (grant no. 953775), by the EU through the Horizon Europe/JU SNS project ROBUST-6G (grant no. 101139068), and by the EU under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE0000001 - program "RESTART").

to suit those applications where efficiency and speed are of the essence, such as wireless communications [6] [7] and biomedical signal processing [8] [9], among others.

Previous work has focused on developing learning algorithms and training protocols; for example, based on spike-timing-dependent plasticity (STDP) or event-driven methods, etc. [10]–[16]. However, minor attention has so far been paid to architectural designs, preventing SNNs from fully leveraging their characteristics. Consequently, SNNs are still behind ANNs in terms of scalability to deep network models and performance (especially accuracy). To address these performance gaps and strike a balance in the trade-off between accuracy and energy efficiency, novel architectural designs are to be explored.

Neural architecture search (NAS) [17] has propelled AI forward by automatically exploring the design space to identify high-performance architectures, minimizing manual tuning and producing highly efficient models. In the past few years, NAS has found very effective neural network architectures that have succeeded in several tasks, including image segmentation [18]–[21], object detection [22]–[25], and other challenging domains such as speech [26] and image recognition [27], [28]. With NAS, a systematic search for optimal architectural designs is performed. In doing so, the event-driven nature of SNNs can be exploited, and the need for manual experimentation can be substantially reduced. Through a methodical assessment of various network topologies, NAS can identify configurations that effectively balance energy efficiency and accuracy, by tailoring the search strategy to the specific dynamics of SNNs.

NAS techniques have recently been investigated for SNNs. However, previous works ignored the effectiveness (e.g., computation cost) of the search methods used. For example, the techniques presented in [29] and [30] involve training a supernet prior to the actual search stage which can be very expensive in terms of both GPU hours and memory usage, while [31] uses a computationally demanding performance predictor, which requires prior training on a small subset of potential architectures. During this initial training phase, performance indicators such as early accuracy are collected are used to train a regression model. Afterwards, the trained predictor infers how unseen architectures might perform without needing to train these all the way. The approach in [32] shows good accuracy performance, but its main drawback is that this comes at the cost of a high number of floating-

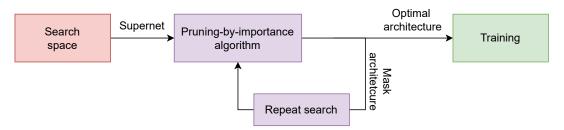


Figure 1: High-level diagram of the proposed NAS framework.

point operations (FLOPS), which goes against our efficiency requirement. SNASNet [33] is the first algorithm to use a *training-free* NAS technique for SNNs, thus enabling a more efficient search phase. SpikeNas, the recently proposed optimizations technique of [34] modify the original SNASNet framework to further reduce the search space, while at the same time coping with memory constraints. To the best of our knowledge, this is to date the most effective algorithm in terms of search time, accuracy and sparsity of the found network models. Our solution will achieve notable improvements, especially in dynamic datasets.

To devise our proposed technique, LightSNN, we first analyze SNASNet and highlight its limitations. Next, we delve into our enhancements, outlining how we modified the framework to get around its shortcomings and obtaining improved results. In Fig. 1, we show an overview of the general phases that drive the proposed NAS algorithm. The main objectives of our design are as follows:

- Improving model accuracy. By effectively searching across the entire search space, to evaluate the importance of each operation, we achieve new state-of-the-art accuracies on static datasets, and a substantial 4.94% accuracy improvement on an event-based dataset.
- Reducing network complexity. The search space has been reduced by eliminating those benchmark operations that lead to a minor improvement in the network task performance.
- Enforcing sparsity in the final architecture. Using various operations, such as zeroize and max-pooling, allowed us to reduce the sparsity of the final model that is outputted by our NAS algorithm.

The paper is organized as follows. Section II briefly reviews the baseline frameworks and methods that were considered as a starting point for the design of LightSNN, our newly proposed NAS algorithm. LightSNN is presented in Section III alongside its design principles. The results for the new NAS framework are reported in Section IV for both static and dynamic datasets. Our final considerations are drawn in Section V.

II. BASELINE APPROACHES

A. Spiking neuron dynamics

In contrast to traditional artificial neurons, which accumulate real-valued inputs and apply a non-linear activation func-

tion (such as ReLU) to produce real-valued outputs, spiking neurons operate differently, by mimicking the fundamental behavior of biological neurons. These computational units aggregate inputs over a number of timesteps within their membrane potential, and an output spike is generated only when the membrane potential reaches a predefined threshold. This characteristic spiking behavior is captured by the leaky integrate and fire (LIF) neuron model [35], where the neuron's membrane potential v(t) increases with each incoming spike, but experiences a "leakage" with time t, causing the membrane potential to decay. Upon reaching a threshold $V_{\rm th}$, the neuron fires an output spike and the membrane potential is reset to a resting value $V_{\rm reset}$.

B. SNASNet

SNASNet [33] employs iterative search within a subset of architectures chosen from a larger pool of more than 200 million candidates randomly generated from a cell-based search space. It assesses *a priori* each architecture using a *training-free* metric to determine which architecture could obtain the highest accuracy after training.

1) Cell-based search space: The search space comprises a macro skeleton and a micro skeleton, the first includes a stem layer consisting of a convolution layer that extracts the first feature maps, two identical cells to be searched, a reduction cell, and a classifier.

2) Training-free NAS approach: The sparsity-aware Hamming distance (SAHD) evaluates architectures based on their performance at initialization without requiring training. Architectures that generate distinct representations across different samples are likely to achieve high accuracy after training [33].

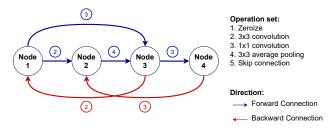


Figure 2: Example of a candidate cell.

Specifically, the SAHD measures the difference between binary codes (activation patterns) produced by the untrained network for input data pairs within a mini-batch. A greater distance between these activation patterns suggests a higher post-training accuracy. To analyze the relation between binary codes for an entire mini-batch of size N, we calculate the kernel matrix $K_H^{(t)}$ as

$$K_H^{(t)} = \begin{pmatrix} N_A - d^{(t)}(c_1, c_1) & \cdots & N_A - d^{(t)}(c_1, c_N) \\ \vdots & \ddots & \vdots \\ N_A - d^{(t)}(c_N, c_1) & \cdots & N_A - d^{(t)}(c_N, c_N) \end{pmatrix},$$
(1)

where N_A is the number of LIF neurons and $d(c_i,c_j)$ represents the SAHD between binary activations c_i and c_j for data samples i and j. The global SAHD score is computed by accumulating the SAHD across all layers, and it is used to generate the kernel matrix (1) at each timestep t. Next, we use the following equation to aggregate the kernel matrices and determine the final score s [33]

$$s = \log \left[\det \left(\left| \sum_{t} K_{H}^{(t)} \right| \right) \right]. \tag{2}$$

III. LIGHTSNN: RATIONALE AND METHODS

A. Pruning-by-importance algorithm

The iterative random search method basically relies on chance-driven selection over only a small subset of the entire search space (5,000 candidates) and, thus, may result in many suboptimal architectures, which increases the chance of missing other promising designs in the more extensive search space. Furthermore, the individual evaluation of each architecture can be time-consuming and computationally intensive. A faster and more reliable search technique is required to circumvent these restrictions and increase the likelihood of finding an architecture with good performance. The application of the *pruning-by-importance* algorithm, as shown in [36], is a good strategy to achieve this goal. To the best of our knowledge, our work is the first to apply this approach to NAS for SNNs.

While exploring the search space, possible architecture candidates in each cell have E edges connecting the nodes and various operators in the previously defined set \mathcal{O} , of cardinality O. Sampling methods require examining O^E unique cells, leading to a search complexity of $\Theta\left(O^E\right)$. For SNASNet, this means that we would need to evaluate $5^{12}\approx 2.4\times 10^8$ possible

Algorithm 1 Operator Pruning Algorithm

Input: Supernet N stacked by cells, each cell with E edges, each edge with O operators.

```
1: while N is not a single-path network do

2: for each operator o_j in N do

3: s_{N\setminus o_j} \leftarrow \mathrm{SAHD}_{N\setminus o_j} \triangleright The higher s_{N_t\setminus o_j} the more likely we prune o_j

4: end for

5: for each edge e_i, i=1,\ldots,E do

6: j^* \leftarrow \arg\max_j \{s_{N\setminus o_j}: o_j \in e_i\}

7: N \leftarrow N\setminus o_{j^*}

8: end for

9: end while

10: return Pruned single-path network N
```

architectures (including backward connections). The pruning-by-importance algorithm [36], in contrast, takes a different approach by assessing a supernet that includes every possible operator and edge. This approach significantly reduces the search complexity, by boosting effectiveness. In fact, the exploration cost is lowered from $\Theta\left(O^{E}\right)$ to $\Theta\left(O\cdot E\right)$, providing a more effective and economical resource consumption. This means that, in our case, this strategy reduces the complexity of the evaluation to $5\cdot 12=60$ iterations.

The pruning-by-importance algorithm is composed of two loops, detailed in what follows and referring to Algorithm 1.

Outer loop: At each round, a single operator is pruned (removed) from each edge. This outer loop continues until the current supernet transforms into a single-path network, which represents the stopping condition. This returns the architecture identified through the search (line 10).

Inner loop: The significance of individual operators is evaluated by computing $SAHD_{N\setminus o_j}$. With the notation $N\setminus o_j$ we refer to a network N where the operation o_j has been pruned. The operator with the least impact on the SAHD is considered the least influential (line 6). Thus, the operation o_j^* whose removal leads to the highest score $SAHD_{N\setminus o_j^*}$ is removed from each edge (line 7).

In Fig. 3, we present a simplified representation of the pruning process within a cell consisting of forward connections, each with three possible operations. Each operation is denoted by a distinct color. The process unfolds as follows:

- In the initial state, all operations are considered (O operations are available at each node, see Fig. 3a).
- The importance of each operation is evaluated and the least significant ones are pruned (Fig. 3b).
- The process is iterated, eliminating the second least important operations (Fig. 3c).
- Finally, a single-path network is obtained, retaining only the most crucial operations (Fig. 3d).

We conducted a comparison on the SNASNet search space using the original search algorithm and then the proposed pruning algorithm, on the CIFAR10 dataset. The statistics in Table Tab. 1 show the promising results of the pruning-by-importance in both accuracy and search time. While the

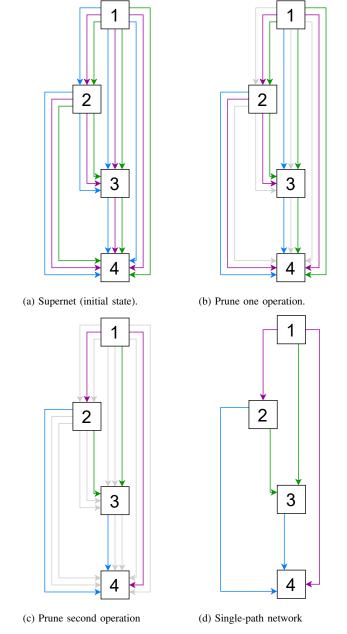


Figure 3: Pruning by importance steps. Pruned operations are shown with a light-gray color.

random selection of a small set of candidates leaves some regions unturned, our greedy approach can navigate through a larger portion of the search space. As a consequence, using the pruning-by-importance algorithm not only reaches better performance in accuracy (92.59% compared to 91.83% of the random search) but achieves a search time of only 2 hours and 16 minutes unlike the 2 hours and 49 minutes of the random search method (i.e., it is 20% faster). Thus, the efficiency of the algorithm has drastically improved. Similar results were consistently obtained throughout our experiments, providing empirical evidence that pruning by importance can better cover the search space and find good architectures within a shorter time span.

Search algorithm	Accuracy	Search time
Random search	91.83%	2h 49min
Pruning-by-importance	92.59%	2h 16min

Table 1: Search algorithms comparison.

B. Lightweight-and-sparsity-aware NAS

1) Search space refinements: It has been noted that a sparser SNN provides better efficiency and performance [37]. Sparsity consists in the reduction of the number of active connections and neurons over time, which leads to enhanced energy efficiency. This also allows for a lower memory footprint and, hence, more efficient hardware utilization. Additionally, sparsity improves the network generalization capabilities thanks to its inherent regularization effect [38]. To leverage a sparser spiking neural network, we perform some design modifications to the search space, creating a more efficient and lightweight framework via the following expedients.

Max pooling: Replacing average pooling with max pooling in SNNs preserves the binary spiking behavior by selecting the most significant spike within a pooling window, ensuring that only the most critical spikes are propagated. Hence, this approach maintains the binary nature of SNNs while promoting sparsity, leading to energy efficiency by reducing the number of spikes [30].

3-operation-cell: Building on the analytical comparison of operations conducted in [34] to quantify the importance of each operation, we opted to reduce the number of operations in our search space to three. The ranking conducted in [34] prioritized operations as follows: (1) 3×3 convolution, (2) skip connection, and (3) zeroize, 1×1 convolution, and 3×3 average pooling have the same importance. They selected 3×3 convolution, skip connection, and average pooling for their architecture search space. Thus, for our own solution, we choose to work with 3×3 convolution, skip connection, and in contrast, we replace average pooling with zeroize in our design to enhance network sparsity.

IV. EXPERIMENTS AND EVALUATION

A. Datasets

The three popular datasets CIFAR10, CIFAR100, and DVS128 Gesture are used to assess our NAS technique. Often used as image classification benchmarks, CIFAR-10 and CIFAR-100 are static datasets consisting of 10 and 100 object classes, respectively, comprising low-resolution (32x32) RGB images. The event-based dataset DVS128 Gesture, which records gestures using a dynamic vision sensor (DVS), is instead used for the gesture recognition task. It includes asynchronous event streams captured from eleven dynamic hand movements. This dataset is perfectly suited for SNNs since it is conceived for neuromorphic computing and event-based processing.

Dataset	Method	Search space structure	Timesteps	Accuracy	Search time	SAR
CIFAR10 SpikeNas [34	SNASNet [33]	2 cells 5 operations	5	91.83%	2h 49min	0.12
	SpikeNas [34]	2 cells 2 operations	5	93.18 %	29s	0.08
	LightSNN (ours)	2 cells 3 operations	5	93%	2min 44s	0.09
CIFAR100 SpikeNas [3	SNASNet [33]	2 cells 5 operations	5	72.36%	2h 2min	0.12
	SpikeNas [34]	2 cells 3 operations	5	45.77%	3min 3s	0.12
	LightSNN (ours)	2 cells 3 operations	5	70.44%	5min 58s	0.13
DVS128Gesture SpikeNas [34]	SNASNet [33]	2 cells 5 operations	16	89.93%	11h 29min	0.13
	SpikeNas [34]	2 cells 3 operations	16	87.84%	9min 55s	0.05
	LightSNN (ours)	2 cells 3 operations	16	94.44 %	6min 54s	0.07

Table 2: Performance comparison of NAS methods for different datasets.

B. Hyperparameters

For the search phase, weights are randomly initialized with the Kaiming Initialization [39], and the search batch size is set to 32. Different search batch sizes could have been investigated but were not tested in this work. For a 300-epoch training phase, the surrogate gradient method is run to enable backpropagation in SNNs [14], the batch size was set to 64, with 0.2 as the learning rate with a cosine-annealing learning rate schedule. We used the vanilla SGD optimizer with a momentum of 0.9 and weight decay of 0.0005. The algorithm was implemented using Pytorch and SpikingJelly libraries and executed on an Nvidia A40 GPU.

C. Results

Using the three aforementioned datasets and computing setup, we make a targeted comparative analysis of our approach, LightSNN, and state-of-the-art methods based on the SNASNet framework. Rather than comparing to all techniques using diverse frameworks, we focus on a more direct and relevant assessment within the SNASNet-based search space.

To assess the effectiveness of our work, we consider three criteria: accuracy, search time, and spiking activity rate (SAR). The accuracy informs us about how well the model could learn and generalize to test data. The search time evaluates the efficiency of the proposed NAS framework and confirms its rapidity. Finally, the SAR quantifies the frequency of spike generation within the network, directly corresponding to the energy consumption of an SNN [40]. Specifically, the SAR is used as a proxy metric for sparsity, which generally results in lightweight and energy-efficient SNN models [37]. The SAR metric is here evaluated by dividing the total number of spikes by the total number of neurons multiplied by the total number of time steps.

Compared to SNASNet, lightSNN achieves significantly higher accuracy on CIFAR-10 and DVS128Gesture and comparable accuracy on CIFAR-100, while substantially reducing search time across all datasets. It also yields lower sparsity on CIFAR-10 and DVS128Gesture and maintains similar sparsity on CIFAR-100. Although lightSNN incurs a modest runtime penalty relative to SpikeNAS on CIFAR-10 and CIFAR100, due to its search method, it achieves substantially higher

accuracy on CIFAR-100 and nearly matches SpikeNAS on CIFAR10 at equivalent sparsity. Conversely, on DVS128Gesture, lightSNN not only outperforms SpikeNAS but also runs faster while maintaining similar sparsity.

We highlight our model's exceptional performance on the DVS128Gesture benchmark, underscoring its suitability as a prime candidate for event-based data classification.

V. CONCLUSION

In this work, we presented a new network architecture search framework to identify energy-efficient spiking neural network architectures. Building on previous research and, in particular, on the SNASNet algorithm, we propose a series of improvements with the following purposes: *i*) being able to explore a larger portion of solutions with respect to what SNASNet does (enlarging the search space), *ii*) reducing the search time for the final architecture, and *iii*) reducing the final model complexity in terms of number of connections and generated spikes. The findings are promising, showing a significant reduction in the time required for the architecture search to complete, alongside considerable enhancements in accuracy, complexity, and sparsity of the resulting models.

REFERENCES

- I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.
- [2] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural networks*, vol. 10, no. 9, pp. 1659–1671, 1997.
- [3] W. Gerstner and W. M. Kistler, Spiking Neuron Models: Single Neurons, Populations, Plasticity. Cambridge University Press, 2002.
- [4] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C.-K. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y.-H. Weng, A. Wild, Y. Yang, and H. Wang, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.
- [5] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G.-J. Nam, B. Taba, M. Beakes, B. Brezzo, J. B. Kuang, R. Manohar, W. P. Risk, B. Jackson, and D. S. Modha, "Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1537–1557, 2015.
- [6] Y. Liu, Z. Qin, and G. Y. Li, "Energy-efficient distributed spiking neural network for wireless edge intelligence," *IEEE Transactions on Wireless Communications*, vol. 23, no. 9, pp. 10683–10697, 2024.

- [7] T. Borsos, M. Condoluci, M. Daoutis, P. Hága, and A. Veres, "Resilience analysis of distributed wireless spiking neural networks," in 2022 IEEE Wireless Communications and Networking Conference (WCNC), pp. 2375–2380, 2022.
- [8] S. K. R. Singanamalla and C.-T. Lin, "Spiking neural network for augmenting electroencephalographic data for brain computer interfaces," *Frontiers in Neuroscience*, vol. 15, 2021.
- [9] E. Kim and Y. Kim, "Exploring the potential of spiking neural networks in biomedical applications: advantages, limitations, and future perspectives," *Biomedical Engineering Letters*, vol. 14, no. 5, pp. 967–980, 2024.
- [10] W. Guo, M. E. Fouda, A. M. Eltawil, and K. N. Salama, "Neural coding in spiking neural networks: A comparative study for robust neuromorphic systems," *Frontiers in Neuroscience*, vol. 15, p. 638474, 2021
- [11] N. Rathi and K. Roy, "Diet-snn: A low-latency spiking neural network with direct input encoding and leakage and threshold optimization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 6, pp. 3174–3182, 2021.
- [12] S. Wang, T. H. Cheng, and M.-H. Lim, "Ltmd: learning improvement of spiking neural networks with learnable thresholding neurons and moderate dropout," *Advances in Neural Information Processing Systems*, vol. 35, pp. 28350–28362, 2022.
- [13] T. Bu, W. Fang, J. Ding, P. Dai, Z. Yu, and T. Huang, "Optimal annsnn conversion for high-accuracy and ultra-low-latency spiking neural networks," arXiv preprint arXiv:2303.04347, 2023.
- [14] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 51–63, 2019.
- [15] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, "Long short-term memory and learning-to-learn in networks of spiking neurons," *Advances in neural information processing systems*, vol. 31, 2018.
- [16] W. Fang, Z. Yu, Y. Chen, T. Masquelier, T. Huang, and Y. Tian, "Incorporating learnable membrane time constant to enhance learning of spiking neural networks," in *Proceedings of the IEEE/CVF international* conference on computer vision (ICCV), pp. 2661–2671, 2021.
- [17] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, X. Chen, and X. Wang, "A comprehensive survey of neural architecture search: Challenges and solutions," ACM Computing Surveys (CSUR), vol. 54, no. 4, pp. 1–34, 2021.
- [18] X. Zhang, H. Xu, H. Mo, J. Tan, C. Yang, L. Wang, and W. Ren, "Denas: Densely connected neural architecture search for semantic image segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 13956–13967, 2021.
- [19] Y. Weng, T. Zhou, Y. Li, and X. Qiu, "Nas-unet: Neural architecture search for medical image segmentation," *IEEE access*, vol. 7, pp. 44247–44257, 2019.
- [20] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei, "Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, (Long Beach, CA), pp. 82–92, 2019.
- [21] X. Wang, T. Xiang, C. Zhang, Y. Song, D. Liu, H. Huang, and W. Cai, "Bix-nas: Searching efficient bi-directional architecture for medical image segmentation," in *Medical Image Computing and Computer Assisted Intervention–MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part I 24*, pp. 229–238, Springer, 2021.
- [22] B. Yan, H. Peng, K. Wu, D. Wang, J. Fu, and H. Lu, "Lighttrack: Finding lightweight neural networks for object tracking via one-shot architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15180–15189, 2021.
- [23] Y. Chen, T. Yang, X. Zhang, G. Meng, X. Xiao, and J. Sun, "Detnas: Backbone search for object detection," Advances in neural information processing systems, vol. 32, 2019.
- [24] J. Guo, K. Han, Y. Wang, C. Zhang, Z. Yang, H. Wu, X. Chen, and C. Xu, "Hit-detector: Hierarchical trinity architecture search for object detection," in *Proceedings of the IEEE/CVF conference on computer* vision and pattern recognition (CVPR), pp. 11405–11414, 2020.

- [25] C. Jiang, H. Xu, W. Zhang, X. Liang, and Z. Li, "Sp-nas: Serial-to-parallel backbone search for object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 11863–11872, 2020.
- [26] J. Kim, J. Wang, S. Kim, and Y. Lee, "Evolved speech-transformer: Applying neural architecture search to end-to-end automatic speech recognition.," in *Interspeech*, pp. 1788–1792, 2020.
- [27] B. Chen, P. Li, C. Li, B. Li, L. Bai, C. Lin, M. Sun, J. Yan, and W. Ouyang, "Glit: Neural architecture search for global and local image transformer," in *Proceedings of the IEEE/CVF International Conference* on Computer Vision (ICCV), pp. 12–21, 2021.
- [28] Q. Zhou, K. Sheng, X. Zheng, K. Li, X. Sun, Y. Tian, J. Chen, and R. Ji, "Training-free transformer architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (New Orleans, Louisiana), pp. 10894–10903, 2022.
- [29] J. Yan, Q. Liu, M. Zhang, L. Feng, D. Ma, H. Li, and G. Pan, "Efficient spiking neural network design via neural architecture search," *Neural Networks*, vol. 173, p. 106172, 2024.
- [30] B. Na, J. Mok, S. Park, D. Lee, H. Choe, and S. Yoon, "Autosnn: Towards energy-efficient spiking neural networks," in *International Conference on Machine Learning (ICML)*, (Baltimore MD, USA), pp. 16253–16269, PMLR, 2022.
- [31] W. Pan, F. Zhao, Z. Zhao, and Y. Zeng, "Brain-inspired evolutionary architectures for spiking neural networks," *IEEE Transactions on Arti*ficial Intelligence, 2024.
- [32] T. Li, J. Zhang, K. Bao, Y. Liang, Y. Li, and Y. Zheng, "Autost: Efficient neural architecture search for spatio-temporal prediction," in Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD), pp. 794–802, 2020.
- [33] Y. Kim, Y. Li, H. Park, Y. Venkatesha, and P. Panda, "Neural architecture search for spiking neural networks," in *European conference on computer vision (ECCV)*, (Tel Aviv, Israel), pp. 36–56, Springer, 2022.
- [34] R. V. W. Putra and M. Shafique, "Spikenas: A fast memory-aware neural architecture search framework for spiking neural network systems," arXiv preprint arXiv:2402.11322, 2024.
- [35] N. Brunel and M. C. Van Rossum, "Lapicque's 1907 paper: from frogs to integrate-and-fire," *Biological cybernetics*, vol. 97, no. 5, pp. 337– 339, 2007.
- [36] W. Chen, X. Gong, and Z. Wang, "Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective," arXiv preprint arXiv:2102.11535, 2021.
- [37] M. Yao, H. Zhang, G. Zhao, X. Zhang, D. Wang, G. Cao, and G. Li, "Sparser spiking activity can be better: Feature refine-and-mask spiking neural network for event-based visual recognition," *Neural Networks*, vol. 166, pp. 410–423, 2023.
- [38] R. Muthukumar and J. Sulam, "Sparsity-aware generalization theory for deep neural networks," ArXiv, vol. abs/2307.00426, 2023.
- [39] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in Proceedings of the IEEE international conference on computer vision (ICCV), (Santiago, Chile), pp. 1026–1034, 2015.
- [40] Z. Yan, Z. Bai, and W.-F. Wong, "Reconsidering the energy efficiency of spiking neural networks," arXiv preprint arXiv:2409.08290, 2024.