Dynamic Spiking YOLO for Object Detection in Autonomous Vehicles

Dong-Gyun Kim†
Software Department
Kwangwoon University
Seoul, South Korea
ehdrbs4935@kw.ac.kr

Jeong-Yun Heo†
Electronic Convergence Engineering Department
Kwangwoon University
Seoul, South Korea
jylemon1128@kw.ac.kr

Jae-Han Lim*
Software Department
Kwangwoon University
Seoul, South Korea
ljhar@kw.ac.kr

***Abstract*—Accurate object detection is critical for safe autonomous driving. In addition to accuracy, energy efficiency is also important for deploying detection modules on vehicular edge computing systems, since high power consumption shortens battery lifetime. Although achieving both objectives is essential for practical deployment, existing spiking YOLO approaches provide only partial solutions. For example, converting all YOLO (You Only Look Once) layers into spiking layers can reduce energy consumption, but the lack of flexibility in conversion often degrades accuracy. Similarly, using long time steps improve accuracy by capturing more precise spike dynamics, but this substantially increases detection latency and energy usage. In this paper, we propose Dynamic Spiking YOLO (DS-YOLO), a new framework that dynamically adapts both YOLO's structure and its layer types to balance accuracy and energy efficiency. Instead of converting all layers, our method selectively transforms only those layers that are energy-efficient and accuracy-preserving, with the selection process guided by driving conditions. To the best of our knowledge, this is the first approach that enables dynamic adaptation of YOLO at both the structural and layer levels for autonomous vehicle perception. Experimental results show that DS-YOLO achieves detection accuracy comparable to YOLOv8 while reducing energy consumption by up to 40%.**

***Index Terms*—spiking YOLO, object detection, autonomous driving, energy efficiency**

I. Introduction

Object detection is a core technology in autonomous driving, enabling real-time recognition of surrounding objects and dynamic path adjustment [1]. To support such tasks, numerous frameworks based on Artificial Neural Networks (ANNs) have been proposed. Among them, the You Only Look Once (YOLO) series [2] is widely adopted. Although YOLO models provide high detection accuracy, their high computational complexity and power consumption limit deployment on vehicular edge computing systems, where excessive energy use reduces battery lifetime. To address this challenge, Spiking Neural Networks (SNNs) have been studied as an energy-efficient alternative to conventional ANNs [3]. Recent studies have attempted to combine the accuracy of YOLO-based detectors with the efficiency of SNNs through ANN-to-SNN conversion[1] [4] and direct training techniques[2] [5]–[8]. The first attempt, Spiking-YOLO, applied ANN-to-SNN conversion to YOLOv3 [4]. To reduce conversion errors, it used high firing rates and thousands of time steps, achieving high accuracy but negating the low-power advantage of SNNs due to increased latency and energy cost. Later works proposed surrogate gradient-based direct training, including EMS-YOLO [6] and Tr-Spiking-YOLO [5] based on YOLOv3, and Spike-YOLO [7] based on YOLOv8. These models reduced the number of time steps to fewer than 10. More recently, AMS-YOLO [8] operated with a single time step, but its performance remains insufficient for practical use. In general, direct-training models still fall short of the detection accuracy compared to ANN-based YOLO.

Thus, existing approaches face a trade-off between accuracy and energy efficiency: the rise in time steps improves accuracy but sacrifices latency and power consumption; reducing time steps saves energy but sacrifices accuracy. A key limitation is that most existing approaches indiscriminately convert all ANN layers into spiking layers, without considering their individual impact on accuracy and energy efficiency. These constraints hinder their applicability in real-world autonomous driving.

To overcome this challenge, we propose Dynamic Spiking YOLO (DS-YOLO), a framework that selectively converts only those ANN layers that improve energy efficiency while preserving accuracy, instead of converting the entire network. DS-YOLO partially converts a trained ANN into spiking layers and fine-tunes the hybrid model using direct training with fewer time steps. Furthermore, DS-YOLO dynamically adapts the selection of spiking layers based on driving conditions such as speed and traffic congestion, ensuring both stable detection performance and energy efficiency across diverse scenarios.

The main contributions of this paper are as follows:

- We propose DS-YOLO, a novel framework that selectively converts accuracy-preserving ANN layers into spiking layers.
- We design a mechanism that dynamically adapts the conversion process based on driving conditions (e.g., speed, traffic congestion) to maintain both accuracy and efficiency.
- We provide extensive evaluations on publicly available dataset. The results demonstrated that DS-YOLO achieves comparable to ANN-based YOLOv8 while reducing energy consumption by up to 40%.

†Equally contributed
*Corresponding author

[1]This approach trains a model as an ANN and then converts it into an SNN with the same structure and weights.

[2]Direct training replaces the non-differentiable spike function with a continuous surrogate function, enabling backpropagation.

II. Background and Related Work

A. You Only Look Once(YOLO)

YOLO is a representative object detection model that was first introduced in 2015 with version 1, and has since evolved through continuous development and active research, reaching version 11. In this study, we adopt YOLOv8 as the base model, as it has demonstrated both stability and high detection accuracy in numerous previous works [9]–[11]. The architecture of the YOLOv8 model is shown in Fig. 1. It consists of three main components: the `backbone`, which extracts features from the input image; the `neck`, which combines the extracted features at multiple scales; and the `head`, which predicts the location and class of each object using fused feature maps. In this study, we focus on the fact that each `Detect` block is responsible for objects of different sizes. The feature maps input to `Detect1`, `Detect2`, and `Detect3` blocks have resolutions of 80×80, 40×40, and 20×20, respectively. The higher-resolution feature maps capture fine details of small objects, while the lower-resolution maps primarily represent features of larger objects. In other words, `Detect1`, `Detect2`, and `Detect3` blocks are designed to detect small, medium, and large objects, respectively.

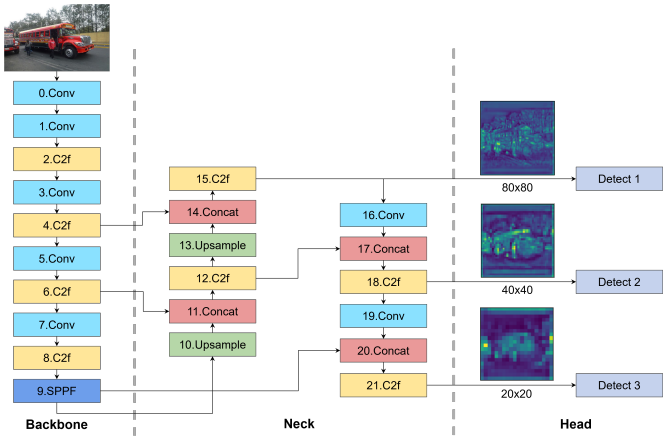

Fig. 1. YOLOv8 network architecture

B. Spiking Neural Network

SNN is a type of ANN that emulates the process of information transmission observed in biological neurons [3]. It uses spikes to represent discrete signals and includes spiking neurons that increase their membrane potential in proportion to incoming spikes, firing when the potential reaches a threshold. Synapses, through which spikes enter the post-synaptic neuron, are associated with synaptic weights that determine the membrane potential increase when a spike (typically valued at 1) is received. Due to this event-driven mechanism, SNN consumes significantly less energy than conventional time-driven ANN, as computations are only triggered by spike events.

Various types of spiking neurons have been developed based on this mechanism. Representative examples include the Integrate-and-Fire (IF) neuron [12], which accumulates spikes in a stepwise manner, and the Leaky-Integrate-and-Fire (LIF) neuron [13], where the membrane potential gradually decays over time. As illustrated in Fig. 2, the LIF neuron incorporates a "leaky" term into the IF model, resulting in a behavior more closely resembled that of biological neurons. Beyond their biological plausibility, SNN offers superior energy efficiency compared to conventional ANNs due to its use of binary spikes and accumulate (AC) operations instead of energy-intensive multiply-accumulate (MAC) operations.

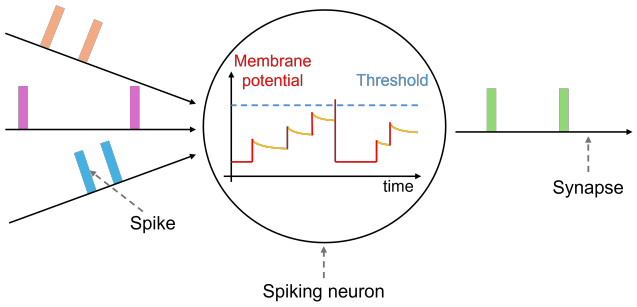

Fig. 2. Mechanism of leaky-integrate-and-fire neuron

III. Proposed Scheme

A. Overview

In this study, we propose an Dynamic Spiking YOLO (DS-YOLO), a new framework that dynamically selects between the ANN and spiking layers according to the driving context, rather than converting all layers to spiking layers. The proposed model consists of three core components: a macro-architecture composed of four sub-model architectures, a micro-architecture, and a method for determining the complexity of driving scenarios. **At the macro-architecture**, our model is based on the YOLOv8 framework and incorporates both the ANN and SNN layers in parallel within key structural blocks. Depending on the driving context, either the ANN or SNN pathway is selectively activated, as shown in Figure 3.

At the micro-architecture, we redesign the core components of YOLO—namely, the `Conv`, `C2f`, and `Detect` blocks using LIF neurons to construct modules compatible with SNN. **At the input image,** the level of traffic congestion is determined by measuring the degree of overlap between objects using the Intersection over Union (IoU) metric, as an on-board method to estimate congestion without relying on external infrastructure. This congestion metric directly affects the actual detection performance, and by utilizing this, clearly distinguishes the detection difficulty according to the level of congestion [14], [15].

B. Macro-Architecture

The overall structure of the proposed macro-architecture is shown in Figure 3. First, the model's `backbone` is entirely maintained as ANN layers because the `backbone` is responsible for extracting essential visual features from the input image. Through experimental validation on COCO dataset, we observed that replacing this component with SNN layers leads to significant information loss, with over 20% degradation in detection accuracy. Therefore, the `backbone` is excluded from the SNN-converted layers. In the following descriptions of sub-models, the `backbone` is omitted for clarity.

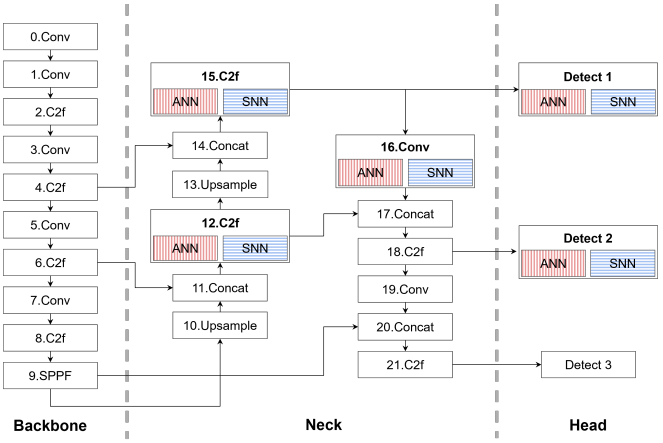

Fig. 3. Macro-architecture of DS-YOLO

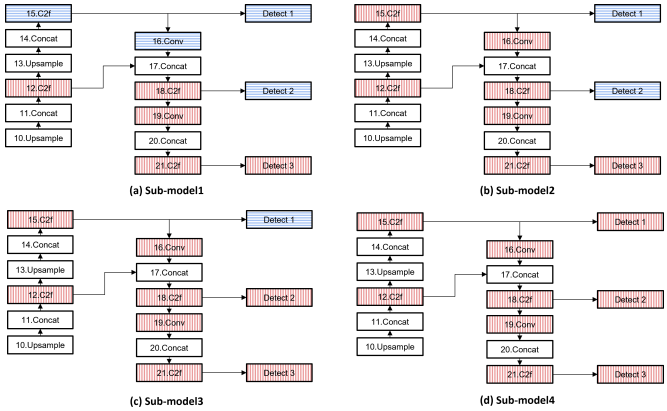

Fig. 4. Layer composition of sub-models in DS-YOLO

As shown in Fig. 4, we propose four sub-models. Each sub-model consists of a combination of ANN and SNN layers, and other computational blocks. They are visually distinguished by red vertical stripes (ANN), blue horizontal stripes (SNN), and no pattern (other blocks) in Fig. 4.

Sub-model1 has the highest proportion of SNN layers, with the `15.C2f`, `16.Conv`, `Detect1`, and `Detect2` blocks implemented as SNN. This configuration is used for detecting large objects and is suitable for low-speed, low-congestion driving scenarios in which objects in the image are relatively large and sparsely distributed.

Sub-model2 configures only the `Detect1` and `Detect2` blocks as SNN layers. By maintaining the floating-point feature map at a resolution of 80×80 from the `backbone` to the `neck`, this model retains more visual information about relatively small objects. Therefore, it is more suitable for detecting distant objects under faster and more complex driving conditions than sub-model1.

Sub-model3 uses SNN only in the `Detect1` block. Since the `Detect2` block processes feature maps with higher resolution than the `Detect3` block, this configuration enables more precise recognition of the shapes and boundaries of small and densely packed objects. As a result, it improves detection performance in faster and more congested driving environments compared to sub-model2. This is because high-speed scenarios require detecting objects that appear small in the camera view due to longer braking distances, while high-congested environments increase object overlap, reducing the visibility of objects in the image. Therefore, high-resolution features are required for accurate detection.

Sub-model4 corresponds to the original YOLO model, consisting entirely of ANN layers, and provides the highest detection accuracy among all configurations. This sub-model is designed for the most challenging scenarios, such as high-speed and highly congested driving environments, where objects are small and densely located.

We design the sub-models based on the observation that the resolution of the feature maps input to each Detect block in the YOLOv8 model differs, which in turn affects the sizes of objects that each block can effectively detect. The proposed sub-models consist of four variants, limited to intuitive and interpretable structures that clearly reflect the detection characteristics according to object size. In this paper, instead of using sub-models with various combinations of ANN and SNN layers, we adopt four simple sub-model structures to clearly demonstrate the performance differences of each sub-model according to object size. This approach also facilitates the analysis of the proposed model. In future work, we will explore the more diverse sub-models by considering various combinations of ANN and SNN layers to enhance performance.

C. Micro-Architecture

Fig. 5 illustrates the micro-architecture of the proposed model. In the original YOLO architecture, the `Conv` block adopts the `Conv(ANN)` structure, which consists of a sequential combination of `Conv2d`, `BatchNorm2d`, and the `SiLU` activation function. In contrast, the proposed `Conv(SNN)` block replaces the SiLU activation with a LIF neuron and places the LIF component at the front of the block. This modification enables spike-based information processing and allows the model to take advantage of event-driven computation.

In this study, we selectively replace the `Conv` blocks within specific modules of YOLO's macro-architecture—such as `C2f`, `Bottleneck`, `Conv`, and `Detect`—with the proposed `Conv(SNN)` blocks. For example, converting the `Detect` block results in the replacement of four `Conv` blocks with `Conv(SNN)` blocks. This selective conversion strategy maximizes energy efficiency while minimizing performance degradation.

D. Traffic Congestion

DS-YOLO selects appropriate sub-models based on vehicle speed and traffic congestion levels, as shown in Fig. 6. Although traffic congestion information can theoretically be obtained via Vehicle-to-Vehicle (V2V) and Vehicle-to-Pedestrian (V2P) communication technologies, real-time detection and response remain limited due to communication delays and unstable connectivity. Moreover, such data acquisition becomes unreliable in high-speed driving environments [16]. Therefore, an on-board method for estimating traffic congestion is required.

In this study, we assume a camera-based sensor setup

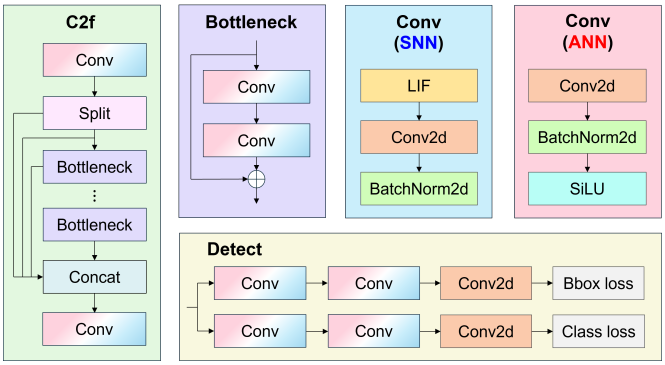

Fig. 5. Micro-architecture of DS-YOLO

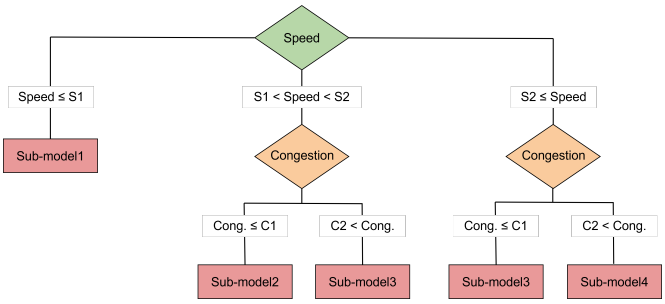

Fig. 6. Flowchart of sub-model selection

and define traffic congestion based on the degree of overlap between objects in a single frame using the Intersection over Union (IoU) metric. This approach can also be extended to other sensor modalities such as radar or LiDAR. The IoU is calculated as shown in Equation 1, where A and B represent arbitrary objects' bounding boxes. Here, $A \cap B$ denotes the area of overlap between the two bounding boxes, while $A \cup B$ indicates the total area covered by both boxes.

$$\mathrm{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{1}$$

The detailed computation of image-level congestion is presented in Equation 2. For each object B_i in a frame, we compute the IoU with all other objects B_j $(j \neq i)$ and take the maximum value. The final congestion score C is obtained by averaging these maximum values across all N objects in the image.

$$\mathrm{C} = \frac{1}{N} \sum_{i=1}^{N} \max_{j \neq i} \mathrm{IoU}(B_i, B_j) \tag{2}$$

IV. Experiment

A. Experimental Setting

1) Dataset: To evaluate the applicability of the proposed model in real-world autonomous driving environments, we used the nuImages dataset [17]. This dataset comprises real driving images collected under various conditions, making it suitable for assessing the performance of object detection models in practical scenarios.

Speed	Congestion Level
$\leq$ 30 km/h	Low
$\leq$ 30 km/h	High
30–70 km/h	Low
30–70 km/h	High
$\geq$ 70 km/h	Low

TABLE I. Scenario definition according to driving conditions

2) Driving scenarios: The experimental scenarios are defined based on vehicle speed and traffic congestion, as summarized in Table I. The speed of the vehicle is assumed to be measured by the speedometer on board, while traffic congestion is calculated using the method mentioned in Subsection III-D, without relying on external information. The speed is categorized into three ranges according to general traffic regulations: less than or equal to 30 km/h, between 30 km/h and 70 km/h, and greater than or equal to 70 km/h. Traffic congestion is binarized into 'low' and 'high' to facilitate a clear comparison of detection performance. Specifically, images with an average IoU below 0.3 are considered to have low congestion, while those with an average IoU above 0.7 are considered to be highly congested.

The reason behind the categorization of vehicle speed and congestion levels is as follows. Vehicle speed is divided based on commonly applied speed limits in real road environments; speed below 30 km/h correspond to school zones, 30∼70 km/h to urban driving, and above 70 km/h to highway driving. Congestion levels are binarized using threshold set at the bottom and top 30% of the congestion score distribution, enabling clearer observation of performance differences. This binary division further facilitates more interpretable comparisons across congestion conditions.

For each scenario, an appropriate sub-model is selected according to the decision flow presented in Figure 6. The speed thresholds used in the decision process, denoted `S1` and `S2`, are set at 30 km/h and 70 km/h, respectively, and the congestion thresholds, denoted `C1` and `C2`, are set at 0.3 and 0.7, respectively. If the speed is below 30 km/h, Sub-model 1 is selected regardless of congestion. Scenarios that involve high speed and high congestion are excluded from the experiments, as such conditions are rarely encountered in real-world driving environments. These threshold values can be adjusted according to specific autonomous driving environments and safety regulations, and future research should investigate optimal parameter selection in more detail.

3) Evaluation Metrics: To evaluate the object detection performance of the proposed model, we use the mean Average Precision at IoU threshold 50% (mAP@50). mAP is a widely adopted metric that comprehensively assesses an object detector's accuracy in both localization and classification. In particular, mAP@50 considers a prediction to be a true positive if the IoU between the predicted bounding box and the ground truth exceeds 50%.

In addition, we evaluate the energy efficiency of the model using energy consumption as a key performance metric. The energy consumption of each layer is calculated separately for the ANN and SNN layers using Eq. 3 and Eq. 4, respectively. These equations are commonly used to quantitatively assess energy consumption in the field of SNN [18]. In the equations

TABLE II. Comparison of models across speed and congestion scenarios

Speed / Congestion	Model	Params	Timestep	mAP@50 (%)	Energy (pJ)
$\leq$ 30km/h / $\leq$ 0.3	YOLOv8	3.2M	X	83.1	11.83
		11.2M	X	83.1	38.3
	EMS-YOLO	6.2M	4	81.7	6.94
		9.3M	4	84.4	8.48
	SpikeYOLO	13.2M	4	82.2	18.5
	Proposed DS-YOLO	3.2M	1	82.2	7.01
		11.2M	1	82.5	27.3
$\leq$ 30km/h / $\geq$ 0.7	YOLOv8	3.2M	X	61.1	11.83
		11.2M	X	61.8	38.3
	EMS-YOLO	6.2M	4	56.3	6.96
		9.3M	4	61.0	8.5
	SpikeYOLO	13.2M	4	60.8	18.5
	Proposed DS-YOLO	3.2M	1	60.0	7.03
		11.2M	1	60.8	27.3
30$\sim$70km/h / $\leq$ 0.3	YOLOv8	3.2M	X	72.2	11.83
		11.2M	X	74.9	38.3
	EMS-YOLO	6.2M	4	65.5	6.92
		9.3M	4	67.8	8.45
	SpikeYOLO	13.2M	4	72.0	18.4
	Proposed DS-YOLO	3.2M	1	71.2	8.02
		11.2M	1	74.0	30.8
30$\sim$70km/h / $\geq$ 0.7	YOLOv8	3.2M	X	63.8	11.83
		11.2M	X	67.2	38.3
	EMS-YOLO	6.2M	4	55.0	6.93
		9.3M	4	58.2	8.47
	SpikeYOLO	13.2M	4	64.0	18.5
	Proposed DS-YOLO	3.2M	1	63.4	9.1
		11.2M	1	66.3	33.6
$\geq$ 70km/h / $\leq$ 0.3	YOLOv8	3.2M	X	64.6	11.83
		11.2M	X	67.1	38.3
	EMS-YOLO	6.2M	4	57.0	6.92
		9.3M	4	59.9	8.44
	SpikeYOLO	13.2M	4	64.8	18.4
	Proposed DS-YOLO	3.2M	1	64.3	9.08
		11.2M	1	66.4	33.6

below, C_{in} and C_{out} denote the number of input and output channels of the layer, k^2 denotes the kernel size, and O^2 denotes the spatial size of the output feature map. In Eq. 4, fr denotes the average firing rate of LIF neurons, and T denotes the number of time steps. fr is defined as $fr = N_{\text{spike}}/T$, where N_{spike} is the total number of spikes generated in the LIF neuron layer of the `Conv(SNN)` block during T time steps.

4) Training & Inference: For training, the SGD method is applied using the Adam optimizer. A YOLOv8 model consisting of ANN layers is first pre-trained on 60,209 training samples. The resulting weights are then transferred to each sub-model, in which certain layers are replaced with SNN layers. Each sub-model is subsequently fine-tuned using the same 60,209 training samples. For inference, evaluation is performed on a dataset composed of 3,000 driving image samples corresponding to each scenario.

$$\mathrm{E}_{ANN} = O^2 \times C_{in} \times C_{out} \times k^2 \times E_{MAC} \tag{3}$$

$$\mathrm{E}_{SNN} = T \times fr \times O^2 \times C_{in} \times C_{out} \times k^2 \times E_{AC} \tag{4}$$

5) Baseline Models for Comparison: In this study, we compare the proposed DS-YOLO model with the original YOLOv8 and two recent SNN-based YOLO models: EMS-YOLO [6] and Spike-YOLO [7]. EMS-YOLO is based on YOLOv3, while Spike-YOLO is based on YOLOv8; Both models adopt the direct training method. Considering the real-time processing requirements of autonomous vehicles and the resource constraints of embedded environments, the experiments are conducted using only lightweight models with fewer than 15 million parameters. For YOLOv8, we adopt the nano model with 3.2M parameters and the small model with 11.2M parameters. The other baseline models used for comparison are also selected such that their parameter sizes do not exceed 15M.

B. Experimental Analysis

1) Comparison of proposed DS-YOLO with YOLOv8: DS-YOLO demonstrates a nearly identical performance to the upper-bound model YOLOv8, with only a slight decrease in accuracy of 0.3$\sim$1.1% in all scenarios. Despite this minimal decrease, the energy consumption of DS-YOLO shows significant improvement. DS-YOLO achieves 23.07$\sim$40.57% and 12.27$\sim$28.72% energy reductions over YOLOv8 across all scenarios for the 3.2M and 11.2M parameter models, respectively. This is because DS-YOLO reduces energy consumption while preserving detection accuracy by replacing relatively less critical layers with SNN layers depending on the driving scenarios.

2) Comparison of proposed DS-YOLO with Other Models: For a fair comparison, we evaluated the models by considering the number of parameters.

Small-model comparison: Our 3.2M parameter model is compared with EMS-YOLO (6.2M, 9.3M). Compared to EMS-YOLO (6.2M), our model consumes 0.07$\sim$2.17pJ more

energy across all scenarios. However, our model achieves 0.5∼8.4% higher detection accuracy, with performance gap widening as the speed exceeds 30 km/h. Compared to EMS-YOLO (9.3M), our model shows 1.0∼2.2% lower detection accuracy at speeds below 30 km/h. However, at speeds above 30 km/h, our model outperforms EMS-YOLO with 3.4∼ 5.2% higher detection accuracy. Although our model consumes more energy in this case, the difference remains within 1pJ.

Large-model comparison: Our 11.2M parameter model is compared with SpikeYOLO (13.2M). In terms of energy consumption, our model consumes 8.8∼15.2pJ more than SpikeYOLO, indicating lower energy efficiency. However, in terms of detection accuracy, our model performs slightly better, with a 0.0∼0.3% improvement at speeds below 30 km/h and a more noticeable 1.6∼2.3% improvement at speeds above 30 km/h.

As demonstrated by the results, although our model is less energy-efficient than the existing models and exhibits comparable detection performance under low-speed conditions, it achieves significantly better accuracy under high-speed scenarios. This is because the EMS-YOLO and SpikeYOLO models convert most of layers into spiking layers to improve energy efficiecy, which results in reduced detection performance for small objects. In contrast, DS-YOLO utilize a sub-model structure tailored to the size of the objects depending on the situation. Furthermore, DS-YOLO is shown to use only a single time step. Since fewer time steps reduce the number of iterations required to propagate spikes between layers, this enables faster detection and demonstrates that the model may be more suitable for real-time processing. These results demonstrate that, in the context of autonomous driving where maintaining detection accuracy is more critical than minimizing energy consumption, DS-YOLO is a more appropriate choice.

V. Conclusion

In this paper, we proposed DS-YOLO, a new framework that dynamically changed both YOLO's structure and its layer types for detection for autonomous vehicle scenarios. In DS-YOLO, only a portion of YOLO layers were dynamically selected and converted into spiking layers to achieve both accuracy and energy efficiency. Furthermore, we introduced a dynamic architecture that selected the suitable sub-model according to driving speed and traffic congestion. The experimental results showed that DS-YOLO achieved a detection accuracy comparable to YOLOv8 (within 1%) in all scenarios, while reducing energy consumption by up to 40.75%. Compared with recent SNN-based detectors, our model not only reduced the number of parameters but also demonstrated superior detection performance, with improvements of up to 8.4%. In future work, we will explore the development of the model to enable conversion with a wider variety of ANN/SNN layer combinations.

Acknowledgment

This work was partially sponsored by the National Research Foundation of Korea (NRF) (grant no.: RS-2023-00241628 and RS-2021-NR062896).

References

[1] X. Zhang, C. Fu, Y. Cui, L. Yi, Y. Sun, W. Wu, and X. Liu, "Corrdiff: Adaptive delay-aware detector with temporal cue inputs for real-time object detection," 2025.

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016.

[3] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[4] S. Kim, S. Park, B. Na, and S. Yoon, "Spiking-yolo: Spiking neural network for energy-efficient object detection," 2019.

[5] M. Yuan, C. Zhang, Z. Wang, H. Liu, G. Pan, and H. Tang, "Trainable spiking-yolo for low-latency and high-performance object detection," *Neural Networks*, vol. 172, p. 106092, 2024.

[6] Q. Su, Y. Chou, Y. Hu, J. Li, S. Mei, Z. Zhang, and G. Li, "Deep directly-trained spiking neural networks for object detection," 2023.

[7] X. Luo, M. Yao, Y. Chou, B. Xu, and G. Li, "Integer-valued training and spike-driven inference spiking neural network for high-performance and energy-efficient object detection," 2025.

[8] W. Li, J. Zhao, L. Su, N. Jiang, and Q. Hu, "Spiking neural networks for object detection based on integrating neuronal variants and self-attention mechanisms," *Applied Sciences*, vol. 14, no. 20, 2024.

[9] D. Reis, J. Kupec, J. Hong, and A. Daoudi, "Real-time flying object detection with yolov8," 2024.

[10] B. Khalili and A. W. Smyth, "Sod-yolov8—enhancing yolov8 for small object detection in aerial imagery and traffic scenes," *Sensors*, vol. 24, no. 19, 2024.

[11] R. Varghese and S. M., "Yolov8: A novel object detection algorithm with enhanced performance and robustness," in *2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS)*, pp. 1–6, 2024.

[12] L. Abbott, "Lapicque's introduction of the integrate-and-fire model neuron (1907)," *Brain Research Bulletin*, vol. 50, no. 5, pp. 303–304, 1999.

[13] D. Tal and E. L. Schwartz, "Computing with the leaky integrate-and-fire neuron: Logarithmic computation and multiplication," *Neural Computation*, vol. 9, pp. 305–318, 02 1997.

[14] C. Zhao, J. Wan, and A. B. Chan, "Density-based object detection in crowded scenes," 2025.

[15] M. S. Rana, A. Nibali, and Z. He, "Selection of object detections using overlap map predictions," *Neural Computing and Applications*, vol. 34, pp. 1–17, 06 2022.

[16] M. Ji, Q. Wu, P. Fan, N. Cheng, W. Chen, J. Wang, and K. B. Letaief, "Graph neural networks and deep reinforcement learning based resource allocation for v2x communications," 2025.

[17] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," 2020.

[18] P. Panda, A. Aketi, and K. Roy, "Towards scalable, efficient and accurate deep spiking neural networks with backward residual connections, stochastic softmax and hybridization," 2019.