MT-DyNN: Multi-Teacher Distilled Dynamic Neural Network for Instance-Adaptive Detection in Autonomous Driving

Hang Shen, *Member, IEEE*, Qi Liu, Yuanyi Wang, Tianjing Wang, *Member, IEEE*, and Guangwei Bai

***Abstract*— Multi-object detection in autonomous driving faces challenges due to multi-scale entities, diverse streetscapes, and limited computational resources. To address these challenges, we present MT-DyNN, a Multi-Teacher knowledge-distilled Dynamic Neural Network framework for instance-adaptive detection, optimizing detection accuracy and inference cost in autonomous driving. The framework's student network comprises a customizable multi-branch residual detection network and a lightweight policy network. The former efficiently extracts multi-scale features in parallel without altering receptive fields, while the latter, depending on curriculum learning, captures task-relevant features and dynamically generates routing vectors to guide the activation or deactivation of residual blocks according to image instance complexity. The framework's teacher network employs a soft-voting strategy to consolidate knowledge from multiple pre-trained teacher models, providing consistent guidance to the student. Within this distillation paradigm, the policy network's routing search space is gradually refined, and the policy and detection networks are jointly fine-tuned to optimize the alignment between routing decisions and feature extraction. Experimental results on CIFAR and ImageNet demonstrate that compared to early exiting and stochastic depth methods, MT-DyNN achieves higher accuracy at the same inference cost and reduces the cost by 50% and 59% at comparable accuracy levels. The generated routing maintains channel sparsity across diverse scenarios.**

***Index Terms*— Dynamic neural network, autonomous driving, instance diversity, multi-teacher knowledge distillation, policy network, multi-branch residual network.**

I. Introduction

MOBILE platforms like autonomous vehicles make stringent demands on the real-time efficiency of neural networks for automated scene perception and analysis. Autonomous vehicle sensor suites generate over 1Gbps of images and videos, requiring rapid processing to support time-critical operations such as path planning and collision avoidance [1], [2]. However, the limited onboard computing resources prevent conventional neural network models from meeting the strict latency requirements of autonomous driving. These challenges render complex models impractical for deployment on resource-constrained platforms [3], [4]. Consequently, there is a pressing need for lightweight and adaptive neural architectures capable of providing real-time scene understanding within the resource limitations of autonomous vehicles, ensuring safe and efficient operation.

In autonomous driving, model compression techniques, including pruning [5], [6], [7], weight quantization [8], [9], [10], and knowledge distillation [11], [12], [13] are commonly employed to improve model efficiency by reducing parameters or computational costs. However, these methods typically rely on static network architectures, limiting their adaptability to dynamic road conditions, including varying object types, complexities, and clarity. Pruning permanently removes neurons and convolutions that may appear redundant but are often crucial for handling complex scenarios, such as occlusions or cluttered environments. Quantization improves efficiency by reducing parameter precision but may compromise fine-grained features needed for subtle object distinctions. Among these techniques, knowledge distillation strikes a better balance between accuracy and efficiency by transferring knowledge from a larger teacher model to a compact student model. However, its reliance on a static student architecture restricts its ability to adapt to varying input complexities.

To address the limitations of static architectures, dynamic neural networks (DyNNs) [14] introduce flexibility by selectively performing computations tailored to diverse image instances, enabling efficient and adaptive inference. Early exiting [15], [16] allows predictions at intermediate layers when confidence thresholds are met, but its fixed exit points limit adaptability to varying instance complexities, reducing effectiveness in dynamic environments. In contrast, skip-layer methods [17], [18] can adjust network routing based on input characteristics, allocating minimal resources to simple inputs while reserving deeper layers for complex cases. For example, as illustrated in Fig. 1, simple instances, such as isolated pedestrians in well-lit urban environments, may achieve accurate detection with shallow embeddings, whereas complex instances, like crowded commercial districts with architectural occlusions and dense pedestrian flow, require deeper embed-

Received 8 January 2024; revised 25 November 2024 and 15 February 2025; accepted 2 March 2025. Date of publication 14 March 2025; date of current version 5 May 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 61502230 and Grant 61501224, in part by the Natural Science Foundation of Jiangsu Province under Grant BK20201357, and in part by the Six Talent Peaks Project in Jiangsu Province under Grant RJFW-020. The Associate Editor for this article was M. Brackstone. *(Corresponding author: Hang Shen.)*

The authors are with the College of Computer and Information Engineering (College of Artificial Intelligence), Nanjing Tech University, Nanjing 211816, China (e-mail: hshen@njtech.edu.cn; qiliu@njtech.edu.cn; 18644024103@163.com; wangtianjing@njtech.edu.cn; bai@njtech.edu.cn).

Digital Object Identifier 10.1109/TITS.2025.3548467

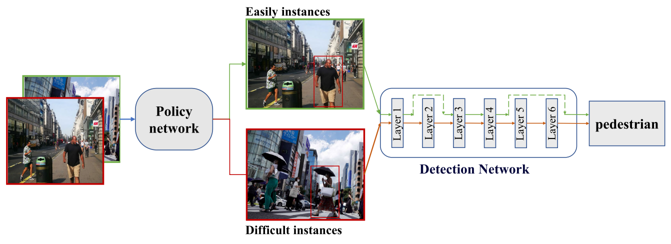


Fig. 1. Skip-layer dynamic reasoning for simple and complex instances.

dings to capture intricate features. This adaptability helps skip-layer methods balance speed and accuracy, optimizing resource utilization in autonomous navigating with frequently changing conditions. However, their full potential remains untapped in adapting computations to task-specific requirements and incorporating structured guidance during inference.

Integrating knowledge distillation with skip-layer DyNNs unlocks the potential of dynamic inference for autonomous driving multi-object detection. By strategically activating layers according to image instance complexity and scene variability, the student network leverages rich feature representations from larger teacher models, combining flexibility and efficiency to improve detection performance.

A. Challenging Issues and Related Works

Although promising, the joint design of knowledge distillation and skip-layer DyNNs for dynamic inference in resource-constrained vehicular environments poses unique challenges.

1) Instance-Adaptive Inference: Autonomous vehicles require balancing accuracy and real-time performance due to multi-scale traffic entities and dynamic driving scenes. DyNNs address this challenge by adapting computation based on input complexity, often leveraging multi-exit and early exiting strategies. BranchyNet [19] pioneered this approach with side-branch classifiers, enabling high-confidence samples to exit early and reducing computational costs. Building on this, MESS [20] optimizes exit configurations for specific device capabilities, while MAMO [21] integrates exit selection, model partitioning, and resource allocation to enhance performance. LECO [22] further refines early exiting with neural architecture search. eDeepSave [23] addresses mobile edge handovers by reallocating resources dynamically, while ClassyNet [16] introduces class-aware exits for prioritizing critical tasks in a resource-limited environment. In contrast, skip-layer methods offer greater flexibility. SkipNet [18] and Conv-AIG [24] use binary decisions to bypass residual blocks and reduce computational costs. Wang et al. [25] proposed a dual dynamic inference framework combining layer and channel skipping, which reduces computational costs for simple samples. While these methods offer high adaptability, challenges remain in achieving robust decision-making stability and consistently optimizing detection performance across diverse scenarios.

2) Lightweight and Multi-Scale Design: Lightweight network designs are essential for overcoming the resource constraints of autonomous vehicles. These models typically employ depthwise separable convolutions and channel shuffling techniques to minimize computational costs. MobileNet [26] achieves this by decomposing standard convolutions into depthwise and pointwise operations, while ShuffleNet [27] improves information flow through its channel shuffle mechanism. However, these models rely on fixed architectures, limiting their adaptability to the dynamic demands of vehicular networks. In contrast, DyNNs like skip-layer-based residual neural network (ResNet) [17] and recurrent neural networks (RNNs) [28] adjust computational paths or layer counts based on input complexity, optimizing resource allocation in changing environments. The multi-branch Inception architecture [29] compresses channels using 1×1 convolutions, reducing complexity while supporting multi-scale feature extraction. Recent work [30] extends this concept with an adaptive DyNN for aerial drone images, which uses flight height and camera tilt to estimate object scale and adjust convolution rates, avoiding explicit distance measurements. In complex autonomous driving conditions, further exploration is needed to combine multi-scale with lightweight, dynamic computation, ensuring efficiency and adaptability across diverse environments.

3) Multi-Teacher Consistent Guidance: DyNNs with skipping mechanisms depend on a policy network to regulate the detection network's activation depth, balancing computational efficiency and detection accuracy. Lightweight designs, while efficient, often result in reduced accuracy and unstable routing decisions. Multi-teacher collaborative optimization offers a promising solution to these challenges by leveraging diverse teacher expertise. Route-constrained optimization [31] leverages intermediate states from the teacher's optimization path, improving student training stability and accuracy. Similarly, TC3KD [32] adopts a learning approach where the teacher and student collaboratively define the training sequence, addressing instability through dynamic weighting. Traditional frameworks often rely on fixed or equal teacher weights [33], [34], which limit their ability to address conflicts among teacher outputs, resulting in suboptimal guidance. To address this issue, Yuan et al. designed a reinforcement learning-based dynamic teacher weighting approach [35], which adaptively assigns weights based on instance-specific requirements, enhancing decision-making efficiency. DMTKD-SP [36] incorporates confidence-based weight allocation to evaluate teacher expertise across logical forms, fostering effective collaboration. IsMt-KD [12] refines this with instance-specific grading, dynamically adjusting teacher contributions. MT4MTL-KD [37] highlights advanced multi-teacher collaboration by distilling knowledge across multiple tasks and incorporating context-aware features. Despite progress, challenges remain in resolving conflicts among teacher outputs with evolving visual instances. Adaptive team knowledge transfer needs further research.

B. Contributions and Organization

In response to the challenges and constraints outlined above, we propose MT-DyNN, a Multi-Teacher knowledge-distilled Dynamic Neural Network for instance-adaptive detection in autonomous driving, designed to balance accuracy and real-time performance in resource-constrained vehicular environments through dynamic routing and multi-teacher knowledge distillation, while adapting to diverse driving conditions. The key contributions are three folded:

- *Soft-Voting Multi-Teacher Knowledge Transfer*: A soft-voting mechanism dynamically selects high-confidence teacher models from parallel predictions and fuses their outputs to guide a dynamic student model, enabling task-specific adaptation in dynamic scenarios;
- *Dynamic Student Model with Multi-Branch Routing*: The student network incorporates a CL-based policy network and a lightweight multi-branch residual detection network. The policy network generates routing vectors tailored to instance complexity, while a reward function integrates the similarity between student predictions and teacher soft labels to optimize routing and detection. The detection network, built with Inception blocks and residual connections, balances reasoning efficiency and detection precision by reducing parameters without compromising the receptive field;
- *Route Training and Unified Fine-Tuning*: Under multi-teacher guidance, the policy network's routing search space is progressively narrowed, establishing initial routing capabilities, while the policy and detection networks are jointly fine-tuned to address misalignment between routing decisions and feature extraction, mitigating accuracy loss from shortcut paths.

Experiments on the CIFAR[1] and ImageNet[2] datasets validate the superiority of the proposed method. In particular, our experiments aim to answer the following questions:

- Under the same inference cost (accuracy level), does the proposed approach improve accuracy (reduce inference costs) compared to early exiting, stochastic depth, and pruning strategies?
- Can the MT-DyNN's routing strategy maintain sparsity in model channels when handling road conditions of varying image instance complexities?

The remainder of this paper is organized as follows. Section II introduces the MT-DyNN architecture, including the multi-teacher network, the detection and policy networks, and the joint fine-tuning framework. Section III provides quantitative experiments and performance evaluation. Section IV concludes this work and discusses future directions.

II. Proposed Solution

This section explains the rationale and implementation of the proposed MT-DyNN. As illustrated in Fig. 2, the framework consists of a multi-teacher network and a student network, which includes both a policy and a detection network. Multiple pre-trained teacher models collaboratively guide the parameter optimization of the student model, promoting alignment between the policy and detection networks.

[1]https://www.cs.toronto.edu/ kriz/cifar.html
[2]https://www.image-net.org/download.php

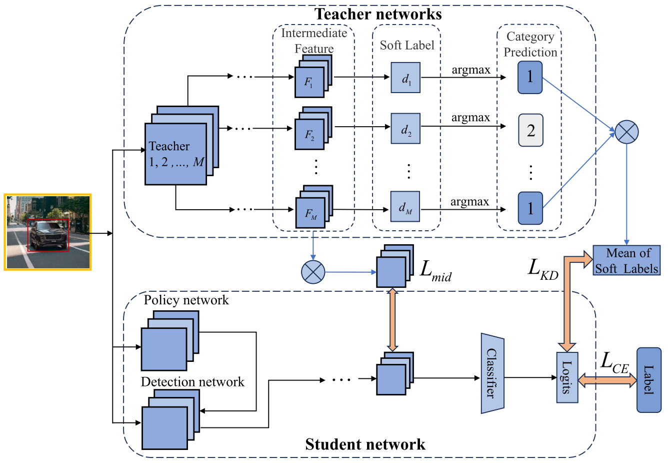


Fig. 2. MT-DyNN architecture.

This integrated knowledge transfer aims to improve the framework's adaptability to diverse and extensive data distributions. Once training is complete, the student model is deployed on autonomous vehicles for dynamic inference.

A. Soft-Voting Multi-Teacher Knowledge Transfer

In the proposed framework, multiple teacher models specialize in distinct detection tasks, such as general object detection, traffic signs, pedestrians, and traffic lights. Each teacher model is pre-trained extensively on its specific task to maximize recognition capabilities. These teacher models provide soft labels to guide the student model but are not further trained during the student training process, thereby avoiding additional computation or coordination costs.

To ensure consistent predictions among multiple teacher models, we employ a soft-voting mechanism that minimizes conflicts through parallel prediction and representative teacher selection. During student model training, teacher models process sensor-captured images concurrently, each extracting task-specific features and making predictions. Teachers generate high-confidence predictions in their areas of expertise, while their confidence is lower in non-specialized domains. Teachers are ranked by their prediction confidence, and the category with the highest confidence is chosen as the primary prediction. When a consensus is reached among the teacher models, those with the highest confidence are selected as the representative teacher. The prediction distributions from these consensus-based teachers are then fused using soft-voting to guide the student model, ensuring stable and efficient knowledge transfer.

Let C be the number of detection categories. For a given category c, the logit outputs of teacher model m and the student model are represented by $z_{m,c}$ and z_c. The corresponding predicted probabilities are represented as

$$\sigma\left(z_{m,c}\right) = \frac{\exp\left(z_{m,c}/\tau\right)}{\sum_{c'=1}^{C} \exp\left(z_{m,c'}/\tau\right)} \tag{1}$$

and

$$\sigma\left(z_c\right) = \frac{\exp\left(z_c/\tau\right)}{\sum_{c'=1}^{C} \exp\left(z_{c'}/\tau\right)} \tag{2}$$

where τ represents the distillation temperature. For an image x, teacher model m generates a probability distribution $d_m = [\sigma(z_{m,1}), \sigma(z_{m,2}), \ldots, \sigma(z_{m,C})]$, which serves as the weight for soft-voting. Denoted M the number of teacher models. Suppose category c is predicted by the majority of N ($\leq M$) teacher models (e.g., 1 in the Category Prediction box with a blue background in Fig. 2), while disregarding teachers that selected other categories (shown in the same box with a gray background). The set of teacher models that predict category c is determined as

$$\mathcal{M}_c = \{m \in \{1, 2, \ldots, M\} \mid \sigma(z_{m,c}) = \max_{c' \in \{1,2,\ldots,C\}} \sigma(z_{m,c'})\}.$$

The soft-voting mean of the high-level knowledge from the N teacher models can be computed as

$$v_c = \frac{1}{N} \sum_{m \in \mathcal{M}_c} d_m. \tag{3}$$

Based on (3), the cross-entropy loss between the aggregated soft labels from the teacher models and the student labels is computed as

$$\mathcal{L}_{KD} = -\sum_{c=1}^{C} v_c \log(\sigma(z_c)). \tag{4}$$

Intermediate layer features from the teacher models can also be used for concept-based distillation to transfer richer information [38]. The feature space embeddings of the teacher and student models are denoted as

$$\mathcal{L}_{mid} = \left\| F - \frac{1}{M} \sum_{m=1}^{M} F_m \right\|_2^2 \tag{5}$$

where F_m and F represent the intermediate layer features of teacher model m and the student model, respectively. The feature representation of the student model aims to approximate the mean feature representation across all teacher models. Additionally, the student model is required to learn from the ground truth labels, and the standard cross-entropy loss is introduced to optimize the training objective

$$\mathcal{L}_{CE} = -\sum_{c=1}^{C} y_c \log(\sigma(z_c)). \tag{6}$$

The overall loss is quantified as a weighted combination of (4), (5), and (6), given by

$$\mathcal{L} = \mathcal{L}_{CE} + \beta_1 \mathcal{L}_{KD} + \beta_2 \mathcal{L}_{mid} \tag{7}$$

where β_1 and β_2 balance the knowledge distillation loss and the standard cross-entropy loss.

B. Dynamic Student Model With Multi-Branch Routing

The student network comprises a CL-based policy network and a multi-branch residual detection network. The former achieves adaptive decisions and generalization through consolidated routing strategies from multiple teachers. The latter learns precise recognition from collective teacher classification cues. The components are detailed as follows.

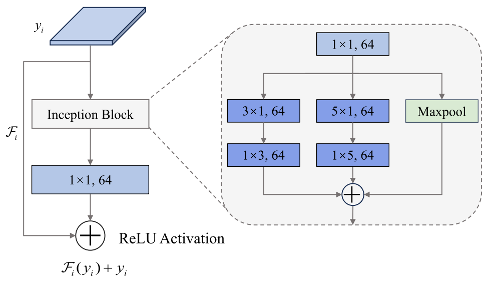


Fig. 3. Multi-branch residual block structure.

1) Multi-Branch Residual Detection Network: We design a novel dynamic multi-branch residual network that incorporates multiple parallel paths within the residual structure. Fig. 3 illustrates a multi-branch residual block composed of Inception blocks and residual connections. To reduce computational complexity, 1×1 convolutional kernels are used to decrease the channel dimension of the feature maps. A multi-branch structure is employed to extract multi-scale features, where the first branch replaces the 3×3 convolutional kernel of traditional residual networks with 3×1 and 1×3 convolutional kernels, reducing the total parameter count while maintaining the same receptive field. The second branch provides a 5×5 convolutional kernel, while the third branch connects directly to a pooling layer. The feature maps from all branches are summed to obtain an aggregated feature matrix, which is then further dimensionally transformed by an 1×1 convolution. This matrix is added to the downsampled original feature matrix through a shortcut connection in the residual network to produce the output. When y_i is fed into the ith multi-branch residual block, the output of this block is $y_{i+1} = \mathcal{F}_i(y_i) + y_i$, which serves as the input for the next residual block.

The proposed multi-branch residual block adopts a lighter-weight Inception block, followed by a 1×1 convolution layer (referred to as the filter expansion layer). This layer increases the dimensionality of the feature maps to achieve deep alignment with the input features before residual summation. This design enables the detection network to effectively capture multi-scale visual information, from vehicle contours to road signs, enhancing its ability to extract diverse and fine-grained features. Additionally, parallel processing is leveraged to improve computational efficiency in model inference, facilitating real-time performance in object detection tasks. The detection network is initialized following standard supervised training, establishing initial routing capabilities.

In a residual network, skipping a residual block does not significantly impact accuracy. Even with the removal of some residual blocks, low-dimensional feature information can still be partially retained [39]. Unlike single-path static networks (e.g., AlexNet [40] and VGGNet [41]), the proposed detection network offers multiple selectable paths. Inspired by this, we adjust the detection network architecture by selecting paths. Specifically, under the guidance of the policy vector, the multi-branch residual network achieves dynamic routing via shortcut connections in the residual structure. When the residual block receives a "skip" command, the convolutional kernels within the residual block do not participate in infer-

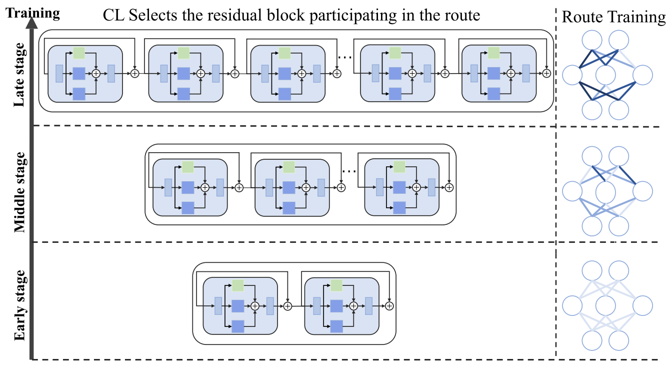


Fig. 4. CL-based route training.

ence, effectively bypassing the block. Thus, we have $y_{i+1} = y_i$. Nevertheless, skipping too many blocks or selecting blocks inappropriately during inference inevitably increases the risk of classification errors. Dynamically selecting an optimal combination of residual blocks based on instance attributes (i.e., dynamic routing) is essential for enhancing the model's real-time performance and adaptability.

2) CL-Based Policy Network: The policy network helps the pretrained detection network locate the optimal subset of residual blocks for accurate classification with the fewest blocks. A detection network with K residual blocks has 2^K on/off options for the blocks, corresponding to $2^K - 1$ optional routes. Thus, the dimension of the strategy search space grows exponentially with the number of residual blocks. Under high dimensionality, randomly initializing the strategy search makes policy network training difficult or even non-convergent.

To mitigate this issue, a CL-based lightweight policy network is introduced that progressively determines the action sequence through staged search with a scheduling mechanism. During training, a scheduling variable h controls the number of residual blocks involved in learning. Initially, h is set to 1, meaning only the last residual block is trained. As training progresses, h increases by 1 every T training epochs, following

$$h = \min\left\{1 + \left\lfloor \frac{e}{T} \right\rfloor, K\right\}, \tag{8}$$

where e represents the current training epoch, and T regulates the rate at which h increases. This scheduling ensures that during the early stages of training, only a small number of residual blocks are involved, while the model progressively incorporates more blocks until all K blocks are utilized. The scheduling function follows a monotonically increasing process, which facilitates a smooth transition from simpler to more complex network structures. At each training epoch, the policy network learns only the activation strategies of the last h residual blocks, while the first $K - h$ residual blocks remain active. This approach not only accelerates model convergence but also reduces training costs. Based on the input features, the policy network evaluates and optimizes each block's on/off status, effectively assessing its utility. After CL training, the policy network can flexibly identify and skip redundant residual blocks to accommodate instance diversity.

As shown in Fig. 4, the CL scheme begins model training with easy samples, gradually progressing to more challenging ones as training advances. Throughout the process, training samples are assigned dynamic weights based on their complexity—initially emphasizing simpler samples and gradually shifting focus to more complex ones. To formalize the curriculum-based sample selection process, let $P(z)$ denote the original data distribution. The weight for each sample in the eth training round is then denoted as $W_\lambda(z)$, where $\lambda = \frac{e}{E}$ represents the normalized training step, with E being the total number of training steps, and $0 \leq \lambda \leq 1$. $W_\lambda(z)$ is constrained within the range $0 \leq W_\lambda(z) \leq 1$. The sample distribution at normalized training epoch λ is given by

$$Q_\lambda(z) \propto W_\lambda(z) P(z), \forall z \tag{9}$$

with the constraint that $\int Q_\lambda(z)\,dz = 1$. When $\lambda = 1$, (9) simplifies to $Q_1(z) = P(z)$ since $W_1(z) = 1$.

Two constraints are introduced to ensure a smooth and gradual transition from simpler to more complex tasks during training. The first is a monotonic increase in information entropy, which requires the entropy of $Q_\lambda(z)$ to increase as training progresses. Specifically, the entropy at a later step $\lambda + \varepsilon$ must be less than the entropy at step λ, i.e.,

$$H(Q_\lambda) > H(Q_{\lambda+\varepsilon}), \forall \varepsilon > 0, \tag{10}$$

ensuring that the sample distribution becomes more diverse over time. The second constraint ensures a monotonic non-decrease in sample weights. This means that $W_\lambda(z)$ assigned to each sample should not decrease as training progresses, reinforcing the emphasis on more complex samples without diminishing their importance, expressed as

$$W_{\lambda+\varepsilon}(z) \geq W_\lambda(z), \forall z, \varepsilon > 0. \tag{11}$$

Accordingly, the CL weight function is expressed as

$$W_\lambda(z) = (1 - \lambda) \exp\left(-\frac{h(z)}{K}\right) + \lambda, \tag{12}$$

where $\frac{h(z)}{K}$ represents the proportion of activated residual blocks when processing sample z, serving as an indirect measure of its complexity. This weighting scheme ensures that simple samples (with few activated blocks) are prioritized in the early stages of training, while complex samples (with more activated blocks) are given focus later on. This aligns with the CL's core principles.

C. Routing Training and Unified Fine-Tuning

The policy network generates routing vectors under multi-teacher guidance to determine detection network routing, while detection classifications and teacher cross-entropy losses back-propagate to reinforce policy decisions. Although CL equips the policy network with diverse routing knowledge, selecting shortcut paths may reduce detection accuracy compared to full model inference. This limitation motivates a joint fine-tuning scheme to better align routing decisions with feature extraction. Unlike the earlier training stage, which emphasizes policy learning with fixed detection parameters, the fine-tuning phase updates both networks simultaneously, mitigating accuracy loss from shortcut paths and enhancing overall performance.

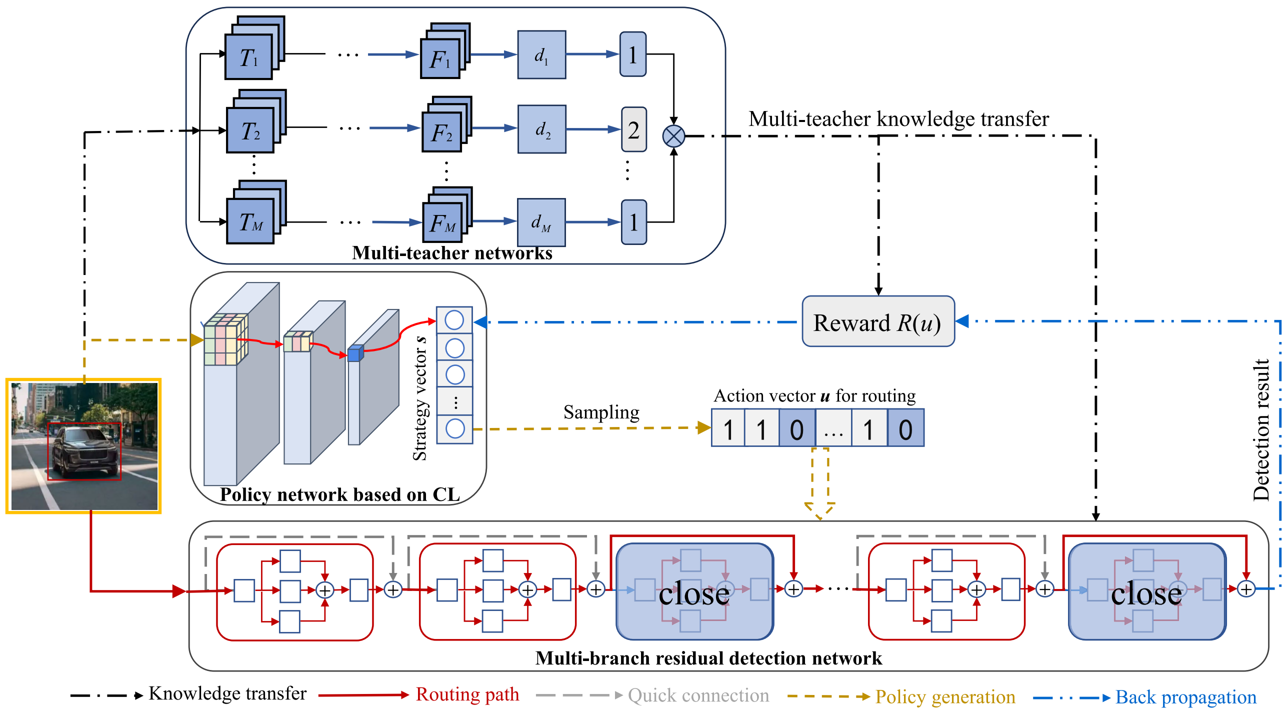


Fig. 5. Multi-teacher-assisted route training and joint fine-tuning framework.

As illustrated in Fig. 5, the key route training and fine-tuning workflow comprises:

- *Knowledge transfer*: Multiple teacher networks collaboratively guide the parameter tuning of the policy and detection networks, providing integrated cues tailored to each input instance.
- *Policy generation*: The CL-enabled policy network extracts input features and outputs routing vectors, with route training involving the policy, detection, and teacher networks.
- *Quick connection*: Guided by policy vectors, the multi-branch residual detection network leverages skip connections for dynamic leapfrogging.
- *Dynamic inference*: Residual blocks are selectively activated or bypassed based on the dynamic routing policy.
- *Unified fine-tuning*: The detection network fine-tunes itself under multi-teacher supervision, providing classification results and cross-entropy loss to the reward function for CL. These rewards are back-propagated to the policy network, aligning routing decisions more closely with feature extraction for improved synchronization.

Unlike stochastic depth [39], where residual block selection is random, block selection is controlled by the policy network in our method to improve alignment with detection instances. For an input image, the policy network outputs all routing decisions for the residual network at once, essentially forming a single-step Markov decision process given the input state. Let $s_k \in [0, 1]$ be the kth element in policy vector $\boldsymbol{s}$, and its value represents the probability that the residual block k is turned on. Given an image x and a pre-trained detection network with K multi-branch residual blocks, the strategy for residual block selection can be defined as a K-dimensional Bernoulli distribution, expressed as

$$\pi_W(u|x) = \prod_{k=1}^{K} s_k^{u_k}(1-s_k)^{1-u_k} \tag{13}$$

where the policy network is denoted as $f(x; W)$, a function of the input image, x, and weights, W. The output of x processed by an activation function $\sigma(x) = (1 + \mathrm{e}^{-x})^{-1}$ is expressed as

$$\boldsymbol{s} = f(x; W). \tag{14}$$

A lightweight ResNet-8 model is used to construct the policy network, with its inference overhead primarily determined by the number of convolutions. Due to the small number of convolutions, the policy network contributes only 8% of the total inference cost. According to (14), the policy network generates an action vector $\boldsymbol{u}$, which dictates the participation of residual blocks during inference. Let the binary variable u_k represent the k-th element in $\boldsymbol{u}$, where $u_k = 1$ indicates activation and $u_k = 0$ indicates deactivation of the k-th residual block. This action vector acts as a soft attention mechanism, guiding the activation ratio of the detection network.

We develop a novel reward function to quantify the benefits of action vector $\boldsymbol{u}$ and guide the policy network towards high accuracy and low-cost routing strategies. Based on (7), the similarity between the student network predictions and the soft labels generated by teacher networks is quantified as

$$p = 1 - \tanh \mathcal{L}. \tag{15}$$

The route training aims to improve (15) and reduce detection network route length (i.e., activated blocks). Accordingly, the reward function is formalized as

$$R(u) = p\left(1 - \left(\frac{1}{K}\sum_{k=1}^{K} u_k\right)^2\right) - (1-p)\gamma \tag{16}$$

where $\frac{1}{K}\sum_{k=1}^{K} u_k$ represents the proportion of activated residual blocks. With correct predictions, a shorter route leads to higher positive rewards, encouraging the policy network to skip more residual blocks. Hyperparameter γ serves as a baseline anchor to balance immediate and future rewards, while the reward function dynamically adjusts to the environment's diversity. Normally, utilizing more residual blocks improves detection accuracy at the cost of lower inference speed. The expectation of rewards is defined as

$$J \triangleq E_{u\sim\pi_W}[R(\boldsymbol{u})]. \tag{17}$$

We train a policy network that maximizes (17), whose yielded policy vectors determine the blocks conducting forward propagation in the detection network. The classification results based on such routes are used for reward calculation.

Policy samples are collected from a K-dimensional Bernoulli distribution. For $u_k \in \{0, 1\}$, the policy gradient is expressed as

$$\begin{aligned}\nabla_W J &= E[R(\boldsymbol{u})\nabla_W \log \pi_W(\boldsymbol{u}|x)]\\ &= E[R(\boldsymbol{u})\nabla_W \log \prod_{k=1}^{K} s_k^{u_k}(1-s_k)^{1-u_k}]\\ &= E[R(\boldsymbol{u})\nabla_W \sum_{k=1}^{K} \log[s_k u_k + (1-s_k)(1-u_k)]]. \end{aligned} \tag{18}$$

In mini-batches, Monte Carlo sampling obtains the expected gradient of (18). These gradient estimations are unbiased. A self-critical baseline [42] is applied in (18) to reduce variance. Accordingly, we reexpress (18) as

$$E[(R(\boldsymbol{u}) - R(\tilde{\boldsymbol{u}}))\,\nabla_w \sum_{k=1}^{K} \log[s_k u_k + (1-s_k)(1-u_k)]], \tag{19}$$

where $\tilde{\boldsymbol{u}}$ refers to the most likely routing under the current policy, with $u_k = 1$ if and only if $s_k \in (0.5, 1)$, otherwise, $u_k = 0$. To encourage the policy network to explore further, we introduce a weighting factor α and update (14) to

$$\boldsymbol{s} = \alpha \cdot \boldsymbol{s} + (1-\alpha)\cdot(1-\boldsymbol{s}), \tag{20}$$

with $\alpha > 1$ for sampling diversified policy vectors.

The route training and joint fine-tuning were performed as outlined in Algorithm II-C, with multiple teacher networks collaboratively guiding the optimization of both policy and detection networks to balance detection accuracy and inference speed. During the route initial phase, the policy network sets the first $K - h$ elements in $\boldsymbol{s}$ to 1, incrementally increasing h to facilitate the route training on the detection network based on CL (lines 5–23). During the training, the total loss (corresponding to 7) and the reward value in CL (corresponding to 16) are jointly updated, driving both towards synchronized convergence. Once routing patterns are progressively explored, the model transitions to joint fine-tuning, leveraging integrated knowledge from the teacher networks to align the policy network with the detection network further (lines 24–27).

Algorithm 1 Multi-Teacher-Distilled Route Training and Fine-Tuning Algorithm

```
Require: Image dataset; $M$ pre-trained teacher models
 1: Randomly initialize policy network $f$ with weight $W$;
 2: Set the CL and joint fine-tuning batches $B_1$ and $B_2$;
 3: Set $\alpha$, $\tau$, $\beta_1$, and $\beta_2$.
 4: Preprocess the images;
 5: for $h = 1$ to $B_1$ do
 6:     $s \leftarrow f(x; W)$;
 7:     $\boldsymbol{s} \leftarrow \alpha\boldsymbol{s} + (1-\alpha)(1-\boldsymbol{s})$;
 8:     if $h < K$ then
 9:         $s[1 : K - h] = 1$;
10:     end if
11:     Sample vector $\boldsymbol{u}$ from the Bernoulli distribution;
12:     Perform dynamic inference on the detection network
        based on $\boldsymbol{u}$;
13:     for $m = 1$ to $M$ do
14:         Obtain $\sigma(z_m)$ with $\tau$ according to (1);
15:     end for
16:     Find majority category $c$ and collect teacher set $\mathcal{M}_c$
17:     Derive soft-voting mean $v_c$ according to (3);
18:     Obtain $\sigma(z)$ with $\tau$ according to (2);
19:     Obtain $\mathcal{L}_{KD}$, $\mathcal{L}_{mid}$ and $\mathcal{L}_{CE}$ according to (4)-(6);
20:     Aggregate overall loss with parameters $\beta_1$ and $\beta_2$ to
        balance $\mathcal{L}_{KD}$ and $\mathcal{L}_{mid}$ according to (7);
21:     Calculate reward according to (16);
22:     Update the backpropagation gradient based on (19);
23: end for
24: for $h = 1$ to $B_2$ do
25:     Update detection network parameters based on (7);
26:     Update policy network parameters based on (19);
27: end for
```

III. Experiment Design and Result Analysis

To evaluate the proposed method's detection and inference performance across varying image complexities and object instances, we utilized three benchmark datasets: CIFAR-10, CIFAR-100, and ImageNet. The CIFAR datasets each consisted of 60,000 32 × 32 RGB images, divided into 50,000 for training and 10,000 for testing. ImageNet, with 1.2 million training images spanning 1,000 categories, included a validation set of 50,000 images for evaluating top-1 accuracy. Model training was conducted on a high-performance server equipped with a 13th Gen Intel Core i9-13900K processor, an MSI PRO Z690-P D4 motherboard, 4 × 32GB DDR5 3600MHz memory, and a 1TB Samsung NVMe M.2 SSD. The system also featured two NVIDIA RTX 4090 24GB GPUs, 128GB of RAM, and 6TB of high-speed storage.

The proposed solution was implemented in PyTorch, utilizing the Adam optimizer for training. α, in (20) was set to 0.8, and the temperature, τ, was uniformly set to 4 across all methods for consistency. β_1 and β_2, in (7) were set to 1 and

TABLE I
STUDENT MODEL COMPRESSION AND CLASSIFICATION ACCURACY

Model	Params	Comp.	CIFAR-10	CIFAR-100
Resnet110	7.1M	$\times$21.9	93.21%	73.40%
VGG-19	156M	$\times$1	92.94%	73.22%
DenseNet121	28.3M	$\times$5.5	95.30%	78.84%
Student	**3.2M**	**$\times$48.7**	**93.60%**	**73.62%**

TABLE II
IMPACT OF TEACHER NUMBERS ON STUDENT ACCURACY

Teacher model(s)	Student accuracy
No Teacher	91.3%
ResNet110	92.5%
ResNet110+VGG19	93.2%
ResNet110+VGG19+DenseNet121	93.6%

50. T, in (8) was set 3. The learning rate was initialized at 1×10^{-4}, with a batch size of 2048 for policy network CL training. During joint training, the learning rate was adjusted to 1×10^{-5}, and the batch size was reduced to 256.

Two types of the multi-branch residual detection network, referred to as MRDN-15 and MRDN-54, were constructed, inspired by the Inception15 and Inception54 architectures, and included comparable convolution counts to ResNet50 and ResNet110 [43]. These networks, featuring 15 and 54 multi-branch residual blocks, respectively, served as baseline static models to validate the effectiveness of DyNN routing.

A. Effectiveness of Multi-Teacher Distillation

Fig. 6(a) illustrates the accuracy convergence over epochs. Compared to no distillation and equal-weight distillation, the proposed soft-voting method achieved faster convergence with higher accuracy and exhibited minimal fluctuations, demonstrating superior stability. Fig. 6(b) presents the reward value convergence over epochs, where the proposed method displayed a faster convergence during early training stages and reached higher reward values in fewer epochs than the other methods. These results indicate that the proposed method enabled more accurate predictions with shorter inference paths. For subsequent experiments, the multi-teacher model with the voting mechanism was adopted as the default configuration.

To analyze model compression and accuracy improvements achieved by multi-teacher distillation, ResNet110 [43], VGG-19 [41], and DenseNet121 [44] were used to construct the teacher ensemble, guiding the training of the student network (a multi-branch residual network based on Inception15). Table I summarizes the parameter sizes and classification accuracy of each model on CIFAR-10 and CIFAR-100. Using the parameter-heavy VGG-19 as a benchmark, we evaluated the compression rates of other models. Despite having only 3.2M parameters, the student model achieved an impressive accuracy of 93.60% on CIFAR-10 and 73.62% on CIFAR-100, surpassing ResNet110 and VGG-19, and performing comparably to DenseNet121. These results highlighted the effectiveness of the proposed multi-teacher distillation framework in achieving competitive accuracy with significantly reduced model complexity.

Table II shows the impact of ensemble size on student model classification. Accuracy positively correlates with more teachers, albeit with diminishing returns. The gap is 0.7% from single to dual-teacher and 0.4% from dual to triple guidance, indicating marginal gains. The student sees less incremental accuracy gains from later teachers because of redundancy, model capacity bottlenecks, and only consensus knowledge being effectively retained while negative and niche transfers are filtered out. Feature representations learned by the student on CIFAR-100 are further visualized in Fig. 7. Unlike no-teacher and single-teacher, multi-teacher guidance produces more compact class-specific clusters with clearer decision boundaries among categories.

B. Effectiveness of DyNN Routing Strategies

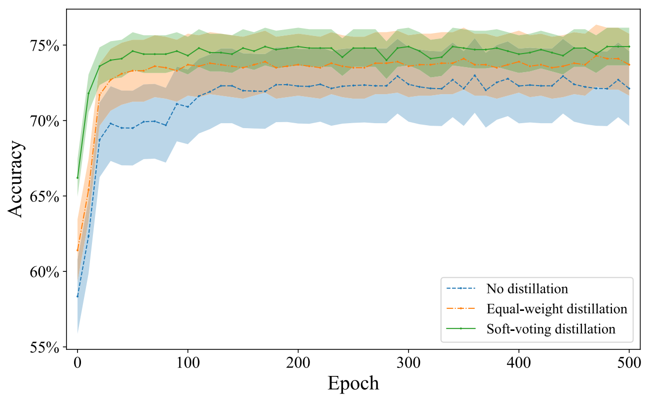
The second experiment compared MT-DyNN with early exiting [28] and stochastic depth [39]. Table III presents the accuracy and average route length results for MRDN-15 and MRDN-54 on CIFAR-10 and CIFAR-100. The average number of utilized residual blocks, denoted as L, represented the route length determined by the policy network. To ensure a fair comparison, early exiting and stochastic depth models were configured to use the ceiling of L, i.e., $\lceil L \rceil$, setting an upper bound on their detection capabilities relative to the proposed method. For early exiting, the first $\lceil L \rceil$ blocks remained activated, while for stochastic depth, $\lceil L \rceil$ blocks were randomly selected for activation.

On CIFAR-10, MRDN-15 with CL-trained dynamic routing achieved an average route length of 9.4 and a classification accuracy of 89.3%, outperforming early exiting and stochastic depth by 72.7% and 68.8%, respectively. Notably, in MRDN-54, nearly 15% of images required fewer than 10 blocks, with some utilizing fewer than 3. These results demonstrated that the proposed CL method improved classification accuracy and significantly reduced inference costs. Static pruning and early exiting lacked the granularity to achieve such dynamic tuning.

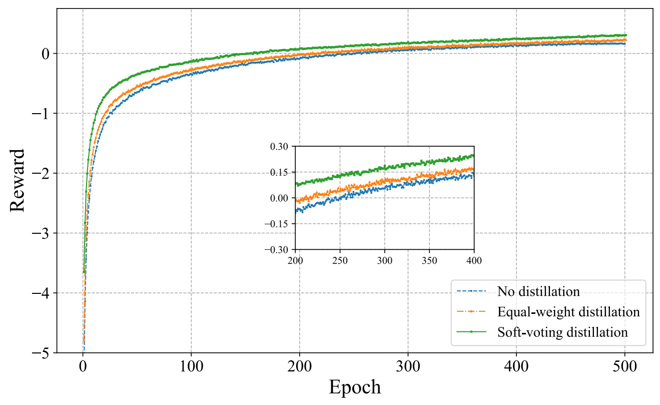
Joint fine-tuning further enhanced performance. On CIFAR-10, fine-tuned MRDN-15 and MRDN-54 improved classification accuracy by 4.3% and 17.3%, respectively, compared to models relying solely on CL. Additionally, average route lengths were reduced by 2.5 and 3.2 blocks. These results underscored the effectiveness of the joint fine-tuning approach in boosting accuracy and optimizing routing efficiency.

The developed policy network generates a complete routing vector in a single step without requiring intermediate outputs during inference, minimizing execution overhead. To validate this, the proposed dynamic routing strategy was compared to the baseline per-step policy inference method [28] (referred to as Single), which uses conventional reinforcement learning for policy training. For fairness, all methods were configured with the same residual blocks to compare inference speeds at identical accuracy levels.

Table IV summarizes the average inference latencies and speedup ratios on CIFAR-10. When achieving the same accuracy as MRDN-15, the proposed method improved classification speed by 16.3% over full network reasoning (named

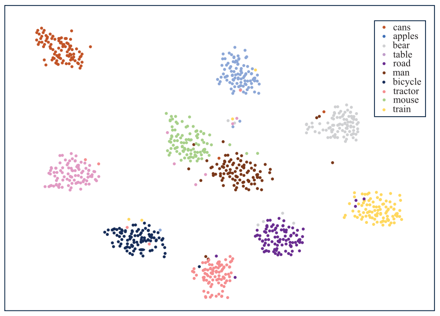


(a) Accuracy with error bands.

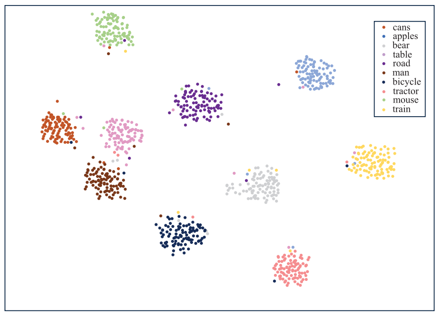


(b) Convergence curve of the reward function.

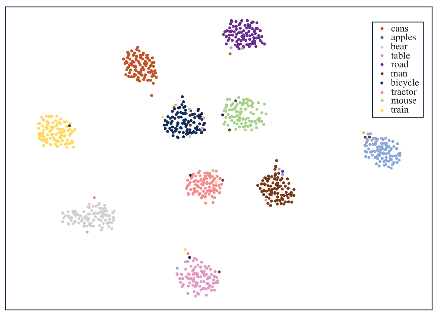
Fig. 6. Convergence curve of detection accuracy and the reward function.



(a) No teacher.



(b) Single-teacher.



(c) Dual-teacher.

Fig. 7. Feature visualization of MRDN-15-based student model on CIFAR-100.

TABLE III
Influence of Routing Strategies on Classification Accuracy

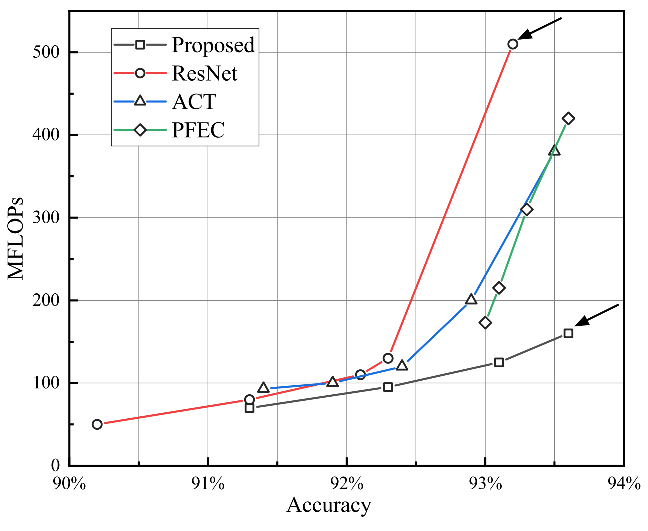
		CIFAR-10				CIFAR-100			
		No joint-training		Joint-training		No joint-training		Joint Training	
		Accuracy	Average route length	Accuracy	Average route length	Accuracy	Average route length	Accuracy	Average route length
MRDN-15	Early exiting	16.6%	10	84.3%	7	23.3%	13	66.5%	14
	Stochastic depth	20.5%	10	88.9%	7	38.3%	13	67.6%	14
	Proposed	**89.3%**	**9.4**	**93.6%**	**6.9**	**58.3%**	**12.4**	**73.6%**	**13.1**
	Full-Net	92.3%	15	92.3%	15	69.3%	15	69.3%	15
MRDN-54	Early exiting	13.3%	21	71.3%	17	63.5%	50	57.9%	31
	Stochastic depth	14.5%	21	90.1%	17	66.3%	50	68.4%	31
	Proposed	**77.4%**	**20.1**	**94.7%**	**16.9**	**72.1%**	**49.1**	**74.9%**	**30.2**
	Full-Net	93.2%	54	93.2%	54	72.2%	54	72.2%	54

Full-Net). In contrast, Single reduced classification speed by 28.7% due to its per-step inference, which introduced extra computations and resulted in a negative speedup. These comparisons verified the significant speedup achieved by the proposed one-shot routing vector generation.

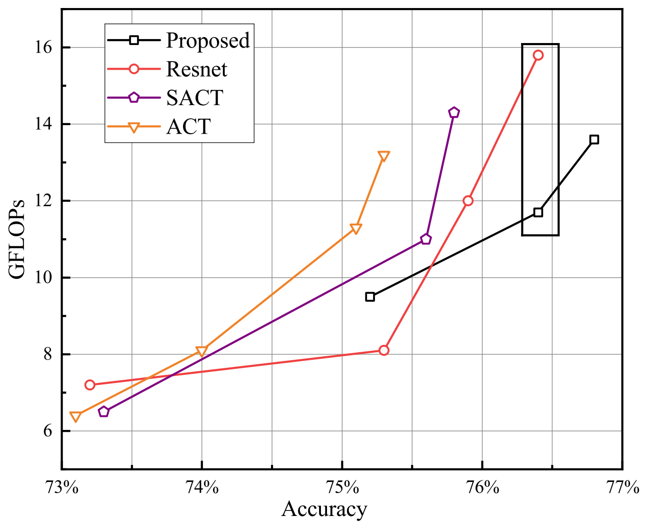
C. Dynamic Inference Acceleration Performance Analysis

The third experiment tuned the hyperparameter γ in (9) to balance routing lengths and detection accuracy, identifying optimal trade-off points under different detection requirements. Each point on the curves in Fig. 8 represented a set of model parameters under a given γ. The average floating-point operations (FLOPs) incurred during image classification on the test set were used to evaluate model complexity. For a comprehensive and fair comparison of acceleration performance, the following three baseline algorithms were included:

- ACT [28]: An early exiting method that terminates inference once confidence thresholds are met.
- SACT [28]: An extension of ACT that incorporates gate functions, enabling adaptive inference depths for different image regions (e.g., backgrounds, edges, or contours).



(a) CIFAR dataset.



(b) ImageNet dataset.

Fig. 8. Classification accuracy vs. average FLOPs across datasets.

TABLE IV
INFLUENCE OF ROUTING STRATEGIES ON INFERENCE COST

		Inference delay (ms)	Inference acceleration
MRDN-15	Full-Net	7.71	N/A
	Single	9.92	-28.7%
	Proposed	**6.45**	**16.3%**
MRDN-54	Full-Net	24.1	N/A
	Single	29.1	-20.7%
	Proposed	**10.5**	**56.4%**

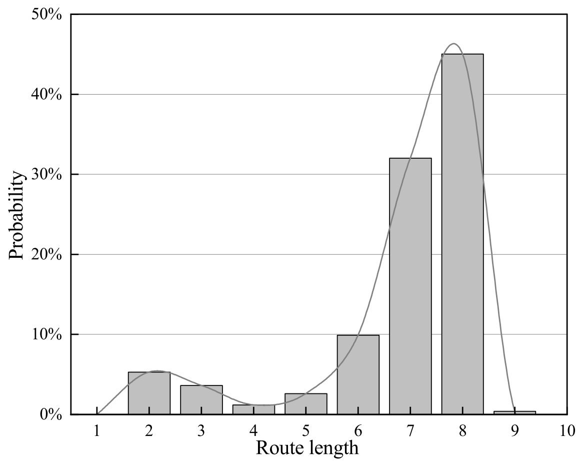


Fig. 9. Probability distribution of route length.

- PFEC [45]: A static network pruning method designed to accelerate convolutions by removing less critical parameters.

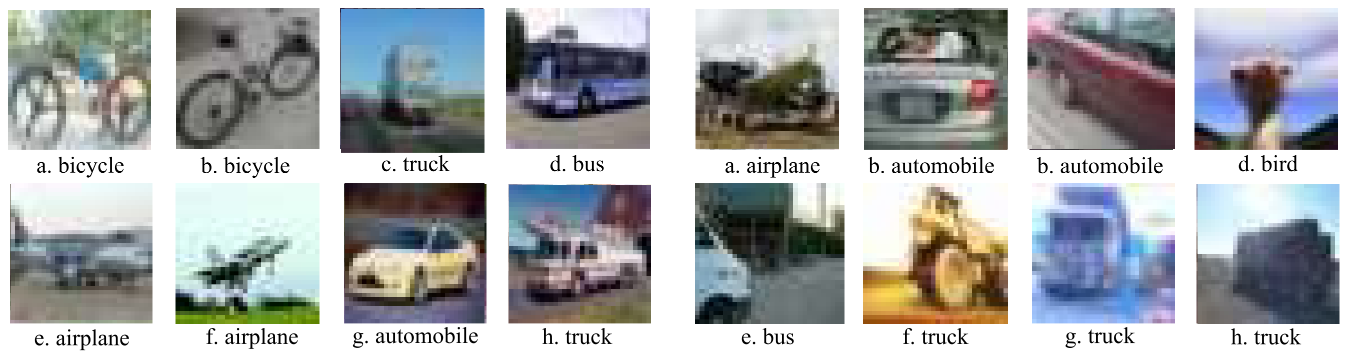
Fig. 8(a) shows the average FLOPs and detection accuracy of different methods on CIFAR. Compared to ResNet-110, the best model from our method (indicated by the arrow) improved accuracy by 0.4% (93.6% vs. 93.2%) while reducing FLOPs by 69% on average (1.6×10^8 vs. 5.1×10^8). Our method required only 48% of the FLOPs used by ACT to achieve the same 93% accuracy. At the same 93.6% accuracy, the proposed method reduced FLOPs by 62% compared to PFEC. Additionally, PFEC could be integrated with the proposed framework for further convolutional acceleration.

On ImageNet, as shown in Fig. 8(b), our method surpassed ResNet-110 in classification accuracy (76.9% vs. 76.3%) while reducing inference cost by 14% (1.36×10^{10} vs. 1.58×10^{10} FLOPs). Even with slightly lower accuracy (see the rectangle in the figure), our framework's inference performance matched that of the full ResNet-110 while reducing computational costs by 26% (1.17×10^{10} vs. 1.58×10^{10} FLOPs). Achieving a 26% inference speedup without compromising accuracy is notable. For instance, in a high-accuracy image recognition service receiving 1 billion API calls daily, the proposed solution could save 1200 GPU hours on a single P6000 GPU (0.024s/image).

D. Influence of Instance Complexity on Dynamic Inference

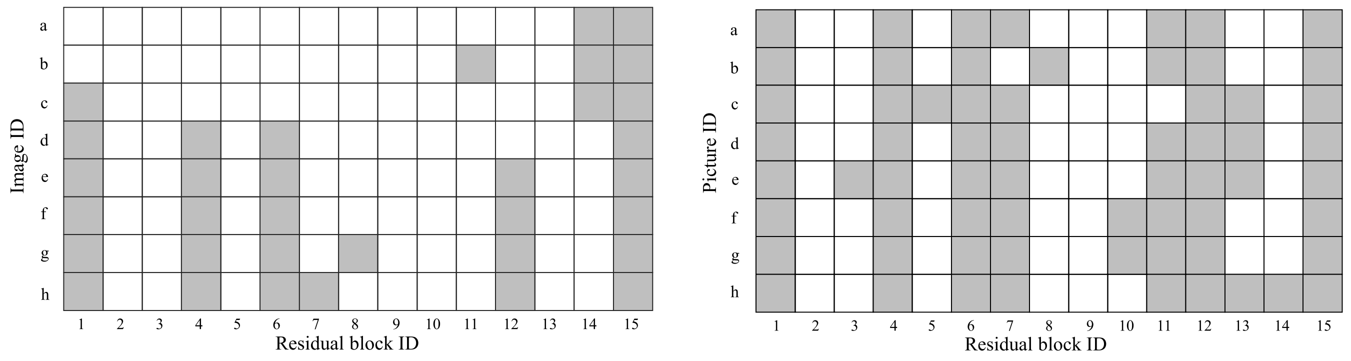
The experimental results demonstrate that the computational costs of the proposed method vary significantly across images of different complexities. Images with salient features typically require fewer convolutional embeddings than more complex or atypical images. This subsection examines the relationship between image complexity and routing lengths to better understand the impact of instance complexity on dynamic inference. To visualize this, the FLOPs of 10,000 test instances were collected, and the routing length utilized by each instance was recorded. Fig. 9 shows the probability density of routing lengths. With a detection network comprising 15 multi-branch residual blocks, test instances used an average of 6.8 blocks.

Instances below the average routing length of 7 were classified as the low-FLOP group, while those equal to or exceeding 7 formed the high-FLOP group. Fig. 10 displays eight representative images from these two groups, along with their ground-truth labels. The differences between the groups are visually distinct: images in the low-FLOP group have clear and complete features, making objects easily recognizable.



(a) Low FLOPs. (b) High FLOPs.

Fig. 10. Representative examples of low and high FLOPs from CIFAR-10.



(a) Low FLOPs. (b) High FLOPs.

Fig. 11. Visualization for short route examples for low FLOPs and high FLOPs from CIFAR-10.

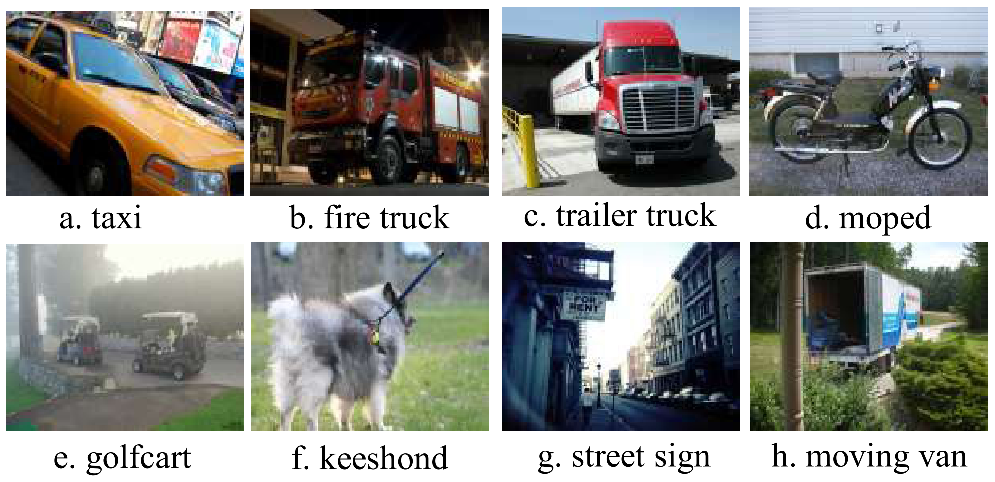


Fig. 12. Representative image examples from ImageNet.

In contrast, images in the high-FLOP group exhibit incomplete contours or atypical traits that hinder object identification. For example, image b lacks intact car contours, showing only the rear, while image c displays only the latter half. Other high-FLOP examples have blurred outlines or low contrast with backgrounds, increasing the risk of misclassification.

To further investigate, the policy network was fed the two image groups in Fig. 10, and the generated routing vectors were visualized in Fig. 11. The horizontal and vertical axes represent residual blocks and image indices, with grey indicating active blocks and white indicating skipped blocks. Three key observations emerge:

- *Consistent routing within categories*: Routing strategies are similar for image instances of the same category, suggesting that features for each category are stored in analogous filters.
- *Complexity-dependent routing lengths*: The low-FLOP group utilizes fewer blocks than the high-FLOP group, supporting the hypothesis that image complexity corre-

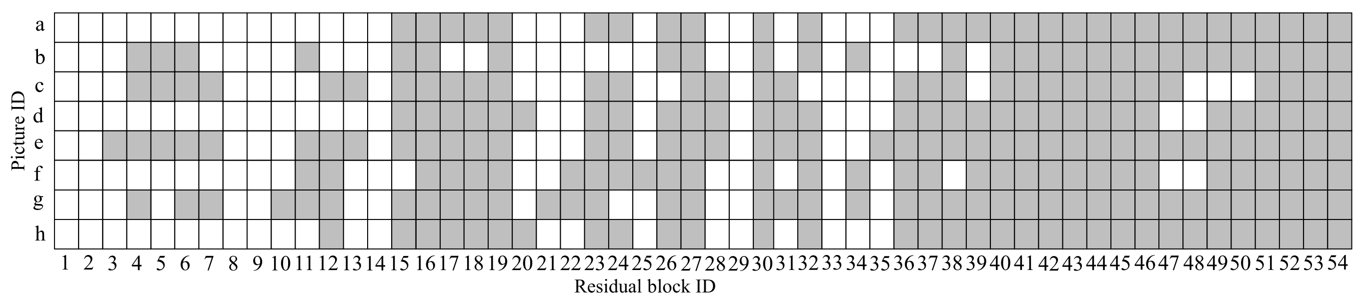


Fig. 13. Visualization for long route examples on ImageNet.

lates positively with routing lengths, leading to heavier inference loads for more ambiguous images.

- *Efficiency for distinguishable images*: Images with salient features rely on short routes, while ambiguous or blurry images require longer routes for accurate identification.

The analysis was extended to ImageNet, which contains a richer variety of image classes and higher resolutions than CIFAR. Larger models are needed to encapsulate intricate parameters and maintain accuracy. Fig. 12 displays eight representative examples from ImageNet, processed using the MRDN-54 model. Routing strategies for these examples are visualized in Fig. 13. Comparing node activations between Figs. 11 and 13, the dynamic network demonstrates improved versatility in adapting to varying instance complexities. It generates appropriate routing strategies tailored to both simple and complex images, balancing accuracy and computational cost. Notably, the presence of "routing blank" highlights the sparsity of active channels under the proposed dynamic routing strategy. This sparsity significantly reduces FLOPs, as illustrated in Fig. 8, where the policy network strategically limits the number of participating convolutions, enhancing efficiency without compromising classification performance.

In real-world autonomous driving scenarios, a vehicle can be equipped with detection networks of varying depths, relying on its capabilities. It can either deploy a specific detection network or have multiple networks of different depths. For the latter case, if combined with gating or similar switching mechanisms, the autonomous driving system can adapt its processing capabilities to differentiated situational requirements, providing fine-grained detection performance while managing computational limitations. Regardless of the detection network configuration, it can be integrated into the proposed multi-teacher framework.

IV. Conclusion

We have introduced MT-DyNN, a multi-teacher knowledge distillation-enhanced DyNN framework designed to enable real-time multi-object detection on resource-limited autonomous vehicles. The detection network dynamically activates or deactivates its multi-branch residual blocks based on routing vectors generated by the policy network, balancing inference cost and accuracy to address diverse demands. A multi-teacher-supervised routing training and fine-tuning scheme were developed to enhance the alignment between the detection and policy networks. Experimental results on CIFAR and ImageNet demonstrate that the proposed DyNN routing approach maintains channel sparsity across images of varying complexities, highlighting its efficacy and adaptability. Under the same inference cost (accuracy level), MT-DyNN outperforms mainstream baselines such as early exiting, stochastic depth, and pruning in detection accuracy (reasoning costs).MT-DyNN also allows both teacher and student networks to be replaced as needed. With further customization, MT-DyNN could streamline detection frameworks and enhance resource-constrained systems in autonomous vehicles, drones, and low-earth orbit satellites.

Future work will explore interpretability to understand the correlation between DyNN routing and image characteristics, further unlocking the potential of dynamic inference and multi-teacher collaboration.

References

[1] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge computing for autonomous driving: Opportunities and challenges," *Proc. IEEE*, vol. 107, no. 8, pp. 1697–1716, Aug. 2019.

[2] H. Shen, Y. Tian, T. Wang, and G. Bai, "Slicing-based task offloading in space-air-ground integrated vehicular networks," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 4009–4024, May 2024.

[3] J. J. M. Ople et al., "Controllable model compression for roadside camera depth estimation," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 12, pp. 15478–15485, Dec. 2023.

[4] B. Kutukcu, S. Baidya, and S. Dey, "SLEXNet: Adaptive inference using slimmable early exit neural networks," *ACM Trans. Embedded Comput. Syst.*, vol. 23, no. 6, pp. 1–29, Aug. 2024.

[5] F. Malandrino, G. D. Giacomo, A. Karamzade, M. Levorato, and C. F. Chiasserini, "Tuning DNN model compression to resource and data availability in cooperative training," *IEEE/ACM Trans. Netw.*, vol. 32, no. 2, pp. 1600–1615, Oct. 2024.

[6] H. Xu, M. Guo, N. Nedjah, J. Zhang, and P. Li, "Vehicle and pedestrian detection algorithm based on lightweight YOLOv3-promote and semi-precision acceleration," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 10, pp. 19760–19771, Oct. 2022.

[7] Y. Zhang and N. M. Freris, "Adaptive filter pruning via sensitivity feedback," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 8, pp. 10996–11008, Aug. 2024.

[8] K. Feng, Z. Chen, F. Gao, Z. Wang, L. Xu, and W. Lin, "Post-training quantization for vision transformer in transformed domain," in *Proc. IEEE Int. Conf. Multimedia Expo*, Jul. 2023, pp. 1457–1462.

[9] B. Jacob et al., "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, May 2018, pp. 2704–2713.

[10] Y. Li, Y. Guo, M. Alazab, S. Chen, C. Shen, and K. Yu, "Joint optimal quantization and aggregation of federated learning scheme in VANETs," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 10, pp. 19852–19863, Oct. 2022.

[11] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.

[12] W. Li, J. Wang, T. Ren, F. Li, J. Zhang, and Z. Wu, "Learning accurate, speedy, lightweight CNNs via instance-specific multi-teacher knowledge distillation for distracted driver posture identification," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 10, pp. 17922–17935, Oct. 2022.

[13] S. An, Q. Liao, Z. Lu, and J.-H. Xue, "Efficient semantic segmentation via self-attention and self-distillation," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 15256–15266, Sep. 2022.

[14] Y. Han et al., "Dynamic neural networks: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 11, pp. 7436–7456, Oct. 2021.

[15] N. P. Ghanathe and S. D. Wilton, "T-RECX: Tiny-resource efficient convolutional neural networks with early-exit," in *Proc. ACM Int. Conf. Comput. Frontiers*, Jan. 2022, pp. 123–133.

[16] M. Ayyat, T. Nadeem, and B. Krawczyk, "ClassyNet: Class-aware early exit neural networks for edge devices," *IEEE Internet Things J.*, vol. 11, no. 9, pp. 15113–15127, Sep. 2024.

[17] Z. Wu et al., "BlockDrop: Dynamic inference paths in residual networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 8817–8826.

[18] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez, "SkipNet: Learning dynamic routing in convolutional networks," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 409–424.

[19] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "BranchyNet: Fast inference via early exiting from deep neural networks," in *Proc. 23rd Int. Conf. Pattern Recognit. (ICPR)*, 2016, pp. 2464–2469.

[20] A. Kouris, S. I. Venieris, S. Laskaridis, and N. Lane, "Multi-exit semantic segmentation networks," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 330–349.

[21] F. Dong et al., "Multi-exit DNN inference acceleration based on multi-dimensional optimization for edge intelligence," *IEEE Trans. Mobile Comput.*, vol. 22, no. 9, pp. 5389–5405, Sep. 2022.

[22] J. Zhang, M. R. Tan, P. Dai, and W. Zhu, "LECO: Improving early exiting via learned exits and comparison-based exiting mechanism," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, Jan. 2023, pp. 298–309.

[23] W. Ju, D. Yuan, W. Bao, L. Ge, and B. B. Zhou, "eDeepSave: Saving DNN inference using early exit during handovers in mobile edge environment," *ACM Trans. Sensor Netw.*, vol. 17, no. 3, pp. 1–28, Jun. 2021.

[24] A. Veit and S. Belongie, "Convolutional networks with adaptive inference graphs," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 3–18.

[25] Y. Wang et al., "Dual dynamic inference: Enabling more efficient, adaptive, and controllable deep inference," *IEEE J. Sel. Topics Signal Process.*, vol. 14, no. 4, pp. 623–633, May 2020.

[26] A. G. Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[27] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6848–6856.

[28] M. Figurnov et al., "Spatially adaptive computation time for residual networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1790–1799.

[29] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-ResNet and the impact of residual connections on learning," in *Proc. AAAI Conf. Artif. Intell.*, Feb. 2017, vol. 31, no. 1, pp. 1–7.

[30] Y. Liu, H. Shen, T. Wang, and G. Bai, "Vehicle counting in drone images: An adaptive method with spatial attention and multiscale receptive fields," *ETRI J.*, vol. 47, no. 1, pp. 7–19, May 2025.

[31] X. Jin et al., "Knowledge distillation via route constrained optimization," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1345–1354.

[32] C. Wang, K. Yang, S. Zhang, G. Huang, and S. Song, "TC3KD: Knowledge distillation via teacher–student cooperative curriculum customization," *Neurocomputing*, vol. 508, pp. 284–292, Jul. 2022.

[33] M.-C. Wu and C. Chiu, "Multi-teacher knowledge distillation for compressed video action recognition based on deep learning," *J. Syst. Archit.*, vol. 103, Dec. 2019, Art. no. 101695.

[34] Z. Yang, L. Shou, M. Gong, W. Lin, and D. Jiang, "Model compression with two-stage multi-teacher knowledge distillation for Web question answering system," in *Proc. 13th Int. Conf. Web Search Data Mining*, Jan. 2020, pp. 690–698.

[35] F. Yuan et al., "Reinforced multi-teacher selection for knowledge distillation," in *Proc. AAAI Conf. Artif. Intell.*, Feb. 2021, vol. 35, no. 16, pp. 14284–14291.

[36] A. Zou et al., "Dynamic multi teacher knowledge distillation for semantic parsing in KBQA," *Expert Syst. Appl.*, vol. 263, Nov. 2024, Art. no. 125599.

[37] S. Gui, Z. Wang, C. Ji-Xiang, X. Zhou, C. Zhang, and Y. Cao, "MT4MTL-KD: A multi-teacher knowledge distillation framework for triplet recognition," *IEEE Trans. Med. Imag.*, vol. 43, no. 4, pp. 1628–1639, Dec. 2023.

[38] S. You, C. Xu, C. Xu, and D. Tao, "Learning from multiple teacher networks," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Disc. Data Mining*, 2017, pp. 1285–1294.

[39] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 646–661.

[40] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[41] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–14.

[42] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel, "Self-critical sequence training for image captioning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 7008–7024.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[44] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput Vis. Pattern Recognit.*, Jun. 2017, pp. 4700–4708.

[45] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–13.

Hang Shen (Member, IEEE) received the Ph.D. degree (Hons.) in computer science from Nanjing University of Science and Technology, Nanjing, China, in 2015. He was a full-time Post-Doctoral Fellow with the Broadband Communications Research (BBCR) Laboratory, Electrical and Computer Engineering Department, University of Waterloo, Waterloo, ON, Canada, from 2018 to 2019. He is currently an Associate Professor with the Department of Computer Science and Technology, Nanjing Tech University, Nanjing. His research interests include object detection and scene perception in autonomous driving, V2X communication networks, and blockchain for cybersecurity. He was a TPC Member of the 2024 IEEE International Conference on High Performance Computing and Communications (HPCC) and the 2021 Annual International Conference on Privacy, Security and Trust (PST). He serves as an Associate Editor for *Journal of Information Processing Systems*, *Frontiers in Blockchain*, and IEEE ACCESS, and was a Guest Editor of *Peer-to-Peer Networking and Applications*. He is an Executive Committee Member of the ACM Nanjing Chapter and a CCF Senior Member.

Qi Liu received the B.Eng. degree in computer science from Shenyang Ligong University, Shenyang, China, in 2023. He is currently pursuing the M.Eng. degree in computer science with Nanjing Tech University, Nanjing, China. His research interests include dynamic neural networks in autonomous driving object detection and multimodal large language models for cybersecurity.

Yuanyi Wang received the B.Eng. degree in vehicle engineering from Jilin University, Jilin, China, and the M.Eng. degree in computer science from Nanjing Tech University, Nanjing, China. His research interests include dynamic neural networks and knowledge distillation techniques for object detection in autonomous and connected vehicles.

Tianjing Wang (Member, IEEE) received the B.Sc. degree in mathematics from Nanjing Normal University in 2000, the M.Sc. degree in mathematics from Nanjing University, in 2002, and the Ph.D. degree in signal and information system from Nanjing University of Posts and Telecommunications (NUPT), in 2009. From 2011 to 2013, she was a Post-Doctoral Fellow with the School of Electronic Science and Engineering, NUPT. She was a Visiting Scholar with the Electrical and Computer Engineering Department, State University of New York at Stony Brook, from 2013 to 2014. She is currently an Associate Professor with the Communication Engineering Department, Nanjing Tech University. Her research interests include dynamic neural networks and multimodal learning for autonomous driving. She is a member of CCF.

Guangwei Bai received the B.Eng. and M.Eng. degrees in computer engineering from Xi'an Jiaotong University, Xi'an, China, in 1983 and 1986, respectively, and the Ph.D. degree in computer science from the University of Hamburg, Hamburg, Germany, in 1999. From 1999 to 2001, he was a Research Scientist with German National Research Center for Information Technology, Germany. In 2001, he joined the University of Calgary, Calgary, AB, Canada, as a Research Associate. Since 2005, he has been with Nanjing Tech University, Nanjing, China, as a Professor of computer science. From October to December 2010, he was a Visiting Professor with the Electrical and Computer Engineering Department, University of Waterloo, Waterloo, ON, Canada. He has authored and co-authored more than 80 peer-reviewed papers in international journals and conferences. His research interests include architecture and protocol design for future networks, multimedia networking, and blockchain for cybersecurity and privacy. He is an ACM Member and a CCF Distinguished Member.