

CARN: Complexity-Aware Routing Network for Efficient and Adaptive Inference

Rebati Gaire
 University of Illinois Chicago
 Chicago, Illinois
 rrgaire@uic.edu

Arman Roohi
 University of Illinois Chicago
 Chicago, Illinois
 aroohi@uic.edu

Abstract

Deep neural networks (DNNs) have achieved remarkable success across various domains, yet their rigid, static computation graphs lead to significant inefficiencies in real-world deployment. Standard architectures allocate equal computational resources to all inputs, disregarding their inherent complexity, which results in unnecessary computation for simple samples and suboptimal processing for complex ones. To address this, we propose the Complexity-Aware Routing Network (CARN), a novel framework that dynamically adjusts computational pathways based on input complexity. CARN integrates a self-supervised complexity estimation module that quantifies input difficulty using confidence, entropy, and computational cost, guiding a neural network-based routing mechanism to optimally assign task modules. The model is trained using a routing loss function that balances assignment accuracy and computational efficiency, mitigating expert starvation while preserving specialization. Extensive experiments on CIFAR-10, CIFAR-100, and Tiny-ImageNet demonstrate that CARN achieves up to $4\times$ reduction in computational cost and over $10\times$ reduction in parameter movement while maintaining high accuracy compared to state-of-the-art static models. The code and pre-trained models are made available at <https://github.com/rrgaire/CARN> for reproducibility and further research.

1. Introduction

Over the past two decades, deep neural networks (DNNs) have expanded their impact across various domains, including natural language processing [14], computer vision [5], drug discovery [32], and multimodal learning [34]. Despite these advancements, the architectural design of modern neural networks has largely converged on a small set of foundational principles, focusing primarily on scaling up models in terms of parameters and data [25]. While this

scale-driven paradigm has yielded state-of-the-art results, it has also exacerbated computational inefficiencies, increasing inference time, energy consumption, and deployment costs. As models continue to grow in size and complexity, the need for adaptive computation mechanisms becomes critical for ensuring efficiency and scalability.

A fundamental limitation of conventional neural networks is their rigid, static computation graph. Whether convolutional, recurrent, or transformer-based, most networks follow a sequential execution of operations, where each input undergoes the same computational process regardless of its inherent complexity. This fixed computation paradigm leads to inefficiencies: simple samples are overprocessed, while complex ones may not receive the required computational resources. Standard optimization techniques, such as model pruning [7, 33], quantization [21], and knowledge distillation [9, 18], provide partial solutions by statically reducing model complexity but fail to introduce adaptive computation tailored to individual samples.

In contrast, modern computing systems optimize execution dynamically. Hardware components selectively activate only the necessary resources at a given moment, and software systems execute conditional branches based on input-specific requirements. Inspired by this, recent research in neural networks has explored dynamic computation paradigms, such as conditional execution [25], mixture-of-experts (MoEs) [19, 27], and early-exit models [30, 31]. Early-exit architectures introduce auxiliary classifiers at intermediate layers, allowing simple samples to exit early, thereby reducing computation and inference time. On the other hand, MoEs leverage multiple specialized subnetworks (experts) and a gating mechanism to dynamically route inputs to the most appropriate expert, offering a balance between scalability and computational efficiency.

At the heart of conditional computing lies routing [16, 35], the decision-making process that determines the computational path an input follows. The effectiveness of this routing mechanism is critical, as it directly influences both model efficiency and specialization. Existing routing strate-

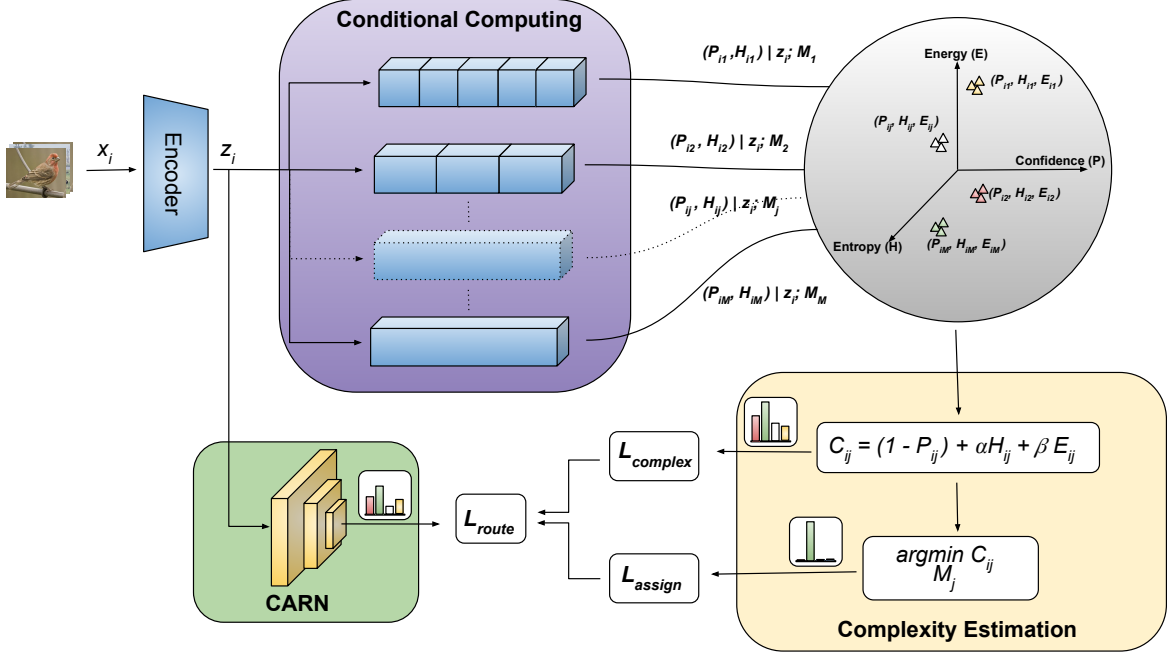


Figure 1. Overview of the Complexity-Aware Routing Network (CARN) framework. The input image (x_i) is processed through multiple task modules (M_j) to estimate its complexity score (C_{ij}) with respect to each module based on confidence (P_{ij}), entropy (H_{ij}), and computational cost (E_{ij}). The Complexity Estimation module assigns the task module with the minimum complexity score (M_i^*) as the optimal choice. The CARN is trained using a combination of assignment loss (L_{assign}) and complexity loss (L_{complex}) to balance performance and computational cost.

gies predominantly depend on heuristic-based or manually designed mechanisms, often resulting in inefficient expert utilization and expert starvation [17, 19, 25]. Moreover, these methods lack a principled framework for quantifying sample characteristics to guide expert selection, leading to suboptimal routing decisions and poor resource allocation. Additionally, most approaches focus solely on task performance without explicitly accounting for computational efficiency, preventing an optimal trade-off between accuracy and resource consumption.

To address these challenges, we introduce a novel framework, Complexity-Aware Routing Network (CARN), that integrates self-supervised complexity estimation with a neural network-based routing mechanism to enable more adaptive and efficient inference. Unlike conventional approaches, CARN explicitly models sample complexity and leverages it to guide dynamic routing decisions. Through this approach, CARN achieves up to $4\times$ reduction in computational cost and over $10\times$ reduction in parameter movement while maintaining high accuracy across multiple benchmarks. Our contributions can be summarized as follows:

1. We introduce a self-supervised complexity estimation framework that quantifies input complexity, integrating confidence, entropy, and energy cost to guide routing decisions.

2. We design CARN, a neural network-based routing mechanism, which dynamically assigns samples to computational pathways based on their estimated complexity.
3. We propose a novel routing loss function that jointly optimizes routing accuracy and computational efficiency, mitigating expert starvation and improving model specialization.
4. We conduct extensive experimental evaluations across diverse classification tasks, demonstrating superior accuracy-efficiency trade-offs, significantly reducing computations, inference time, and parameter movement while maintaining or improving task performance.

2. Related Works

Conditional computation has emerged as a critical approach to improving the efficiency and scalability of deep neural networks by selectively activating parts of a model based on input characteristics [1, 25, 26]. While early-exit networks attempt to reduce inference cost by allowing confident predictions to bypass deeper layers [3, 31], typically rely on heuristic confidence thresholds, which can be unreliable for ambiguous samples and may lead to misclassification [24, 25]. Mixture-of-Experts (MoEs), widely used in large-scale vision and language models [8, 22], distribute compu-

tation across specialized subnetworks, enhancing scalability but often suffering from expert imbalance, where certain experts dominate while others remain underutilized [26, 36]. Token selection [2] and layer skipping [6] mechanisms further refine computational efficiency by pruning unimportant tokens or dynamically adjusting depth, but they frequently lack a principled method for quantifying sample characteristics, leading to suboptimal routing [35]. Despite these advancements, most existing approaches prioritize performance optimization without explicitly incorporating computational and energy constraints [11], making them inefficient for real-world adaptive inference.

Routing strategies play a critical role in conditional computing by determining how inputs traverse computational pathways [16, 25]. Fixed routing approaches predefine expert assignments based on prior knowledge or metadata, ensuring deterministic selection but lacking adaptability to unseen data distributions [20]. While simple and efficient, these approaches struggle to generalize in dynamic environments where input complexity varies significantly. Learned routing methods introduce trainable gating functions to determine expert selection dynamically [23, 26, 35]. Hard routing approaches [23] make discrete module selection decisions, activating only a subset of experts per input. While this sparsifies model activation and reduces inference costs, it often leads to module collapse, where only a few experts are frequently utilized, diminishing model diversity [25]. Soft routing, in contrast, assigns continuous selection weights to modules, making it trainable via gradient descent but at the cost of increased computational overhead [8]. Furthermore, most learned routing approaches optimize for task performance rather than computational efficiency, leading to excessive parameter activation and unnecessary floating-point operations [36]. Hypernetwork-based routing offers greater adaptability by generating task-specific parameters, but the added computational burden makes it impractical for large-scale applications [25]. Most of these routing approaches fail to incorporate a structured method for sample complexity estimation, leading to inefficient expert selection and redundant computation [8].

In contrast, our work introduces CARN, a framework that explicitly quantifies input complexity to guide routing decisions. Unlike existing approaches, CARN employs a self-supervised complexity estimation module to assess input difficulty, ensuring that routing decisions are both adaptive and computationally efficient. By incorporating a neural network-based routing mechanism, CARN optimally distributes computational effort while preventing expert imbalance.

3. Methodology

Our Complexity-Aware Routing Network (CARN) framework is designed to optimize adaptive routing by leveraging

a novel self-supervised complexity estimation mechanism. This framework consists of two primary components: (1) a complexity estimation module that quantifies the difficulty of each sample based on pre-trained task module predictions and (2) a CARN model trained with a routing loss to dynamically assign inputs to optimal task modules. By jointly optimizing both components, our approach enables computational efficiency while maintaining high predictive accuracy. An overview of the framework is provided in Figure 1.

3.1. Self-Supervised Complexity Estimation

The complexity of an input sample is determined by its prediction confidence, entropy, and computational cost, all derived from pre-trained task modules. The complexity score guides the routing decisions by evaluating these factors. Given an input sample $\mathbf{x}_i$, the complexity score for task module $\mathcal{M}_j$ is defined as:

$$C_{ij} = (1 - \mathcal{P}_j(y^* | \mathbf{x}_i)) + \alpha H_j(\mathbf{x}_i) + \beta E_j(\mathbf{x}_i) \quad (1)$$

where:

Prediction Confidence $(\mathcal{P}_j(y^* | \mathbf{x}_i))$ denotes probability assigned to the ground-truth class y^* by $\mathcal{M}_j$ for sample $\mathbf{x}_i$. Lower confidence indicates greater uncertainty, suggesting that the sample is more complex for $\mathcal{M}_j$.

Prediction Entropy $H_j(\mathbf{x}_i)$ captures the uncertainty in the full probability distribution over all classes:

$$H_j(\mathbf{x}_i) = - \sum_{k=1}^K \mathcal{P}_j(y_k | \mathbf{x}_i) \log \mathcal{P}_j(y_k | \mathbf{x}_i) \quad (2)$$

where K is the number of classes, and $\mathcal{P}_j(y_k | \mathbf{x}_i)$ is the prediction confidence of $\mathcal{M}_j$ for the k^{th} class. A higher entropy score suggests greater ambiguity in predictions, requiring more computational effort. The weight α modulates the contribution of entropy to the overall complexity score.

Computational Cost $E_j(\mathbf{x}_i)$ represents the computational resources required to process $\mathbf{x}_i$ with module $\mathcal{M}_j$. It can be measured in terms of floating-point operations (FLOPs), latency, and/or memory consumption. The weight β modulates its impact, where higher values prioritize energy-efficient models.

The optimal task module for a given sample $\mathbf{x}_i$ is selected as:

$$\mathcal{M}_i^* = \arg \min_{\mathcal{M}_j} C_{ij} \quad (3)$$

where $\mathcal{M}_i^*$ serves as a *pseudo-label* for sample $\mathbf{x}_i$ to supervise CARN training. This approach ensures that routing

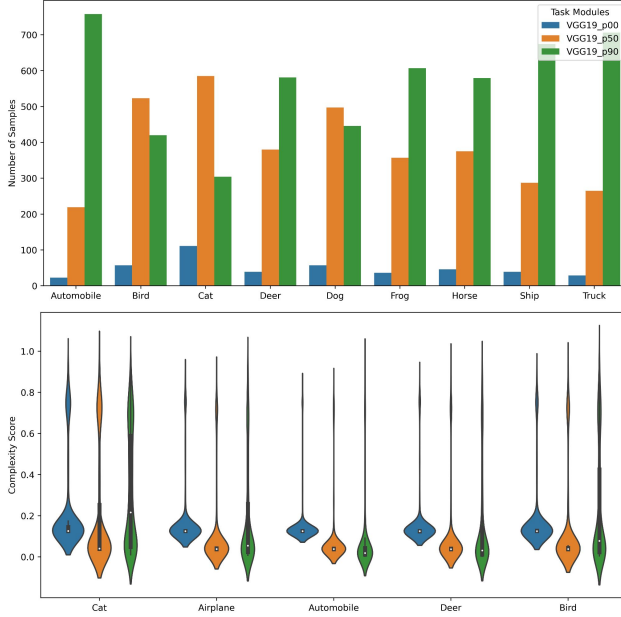


Figure 2. Task Module Assignment and Complexity Score Distribution. Top: Bar graph showing the number of CIFAR-10 samples from each class assigned to different VGG19-based task modules. Bottom: Violin plot illustrating the distribution of complexity scores for samples within a selected CIFAR-10 class across the task modules.

decisions are guided by a structured complexity estimation framework, where complexity is derived from the performance characteristics of task modules, rather than relying on heuristic-based assignments.

An ablation study is conducted to examine the impact of α and β on overall task performance and computational cost in Section 4.4.1.

3.2. Complexity-aware Routing

A lightweight routing network, CARN, is trained to predict the optimal task module based on extracted feature representations. The CARN is implemented as a fully connected neural network comprising a pooling layer followed by two dense layers. Given an input $\mathbf{x}_i$, the CARN outputs a probability distribution over the available task modules:

$$\mathbf{p} = \text{softmax}(\mathcal{R}(\mathbf{z}_i)), \quad (4)$$

where $\mathbf{z}_i$ is the feature representation from the early layers of the backbone network or encoder for $\mathbf{x}_i$, and $\mathcal{R}$ is the adaptive routing model CARN. The probability distribution $\mathbf{p} = \{p_{i1}, p_{i2}, \dots, p_{iM}\}$ represents the probability of selecting each task module $\mathcal{M}_j$ for sample $\mathbf{x}_i$.

3.2.1. Training Objectives

The routing mechanism is optimized with a loss function designed to balance task assignment accuracy and computational efficiency:

$$L_{\text{route}} = L_{\text{assign}} + \lambda L_{\text{complex}} \quad (5)$$

where L_{assign} ensures correct sample-to-module assignments, and L_{complex} encourages efficient routing by minimizing unnecessary computation.

Task Assignment Loss

$$L_{\text{assign}} = - \sum_{i=1}^N \sum_{j=1}^M \mathcal{M}_{ij}^* \log \mathbf{p}_{ij} \quad (6)$$

where $\mathcal{M}_{ij}^*$ is a one-hot encoded ground truth label indicating the correct task module for sample $\mathbf{x}_i$. And $\mathbf{p}_{ij}$ is the probability predicted by the CARN for assigning $\mathbf{x}_i$ to module $\mathcal{M}_j$. This is the standard categorical cross-entropy loss, which enforces the correct assignment of each input sample $\mathbf{x}_i$ to its optimal task module $\mathcal{M}_i^*$.

Complexity Loss

$$L_{\text{complex}} = \sum_{i=1}^N \sum_{j=1}^M \mathbf{p}_{ij} (C_{ij} + \gamma \log \mathbf{p}_{ij}) \quad (7)$$

This loss consists of two key terms:

- **Complexity Minimization Term** $\mathbf{p}_{ij} C_{ij}$: Encourages the routing network to assign samples to the most efficient task module by minimizing computational complexity.
- **Exploration Term** $\gamma \mathbf{p}_{ij} \log \mathbf{p}_{ij}$: Regularizes the assignment probability distribution, preventing premature convergence to a narrow set of modules. This ensures that routing decisions remain flexible in early training stages before settling into an optimal strategy.

The combined effect of these terms allows CARN to learn an optimal balance between minimizing computational overhead and ensuring correct module assignments. The training dynamics of CARN are illustrated in Fig. 1. The impact of these losses on training convergence, task performance, and computational cost is further analyzed in the Section 4.4.2.

3.2.2. Training and Inference

The training process follows a structured self-supervised approach, using the complexity scores as pseudo-labels to update CARN. The overall procedure is outlined in Algorithm 1. For each input sample $\mathbf{x}_i$, the shared feature extractor $\mathcal{F}$ generates a latent representation $\mathbf{z}_i$, which is used to compute complexity scores C_{ij} across all pre-trained task modules $\{\mathcal{M}_j\}_{j=1}^M$. These scores are derived using the formulation in Equation 1, which combines prediction confidence, entropy, and estimated energy cost. The task module with the lowest complexity score is selected as the pseudo-label $\mathcal{M}_i^*$, guiding the target output for CARN. The routing

network $\mathcal{R}$, described in Section 3.2, then predicts a soft probability distribution $\mathbf{p}$ over task modules. The loss is computed using the routing objective defined in Equation 5, and the parameters of $\mathcal{R}$ are optimized using stochastic gradient descent with a cosine annealing learning rate schedule. All task modules remain fixed during this process, ensuring efficient and stable training.

At inference time, as described in Algorithm 2, a test sample $\mathbf{x}$ is passed through the same feature extractor $\mathcal{F}$ to obtain its latent embedding $\mathbf{z}$. The trained $\mathcal{R}$ outputs routing probabilities, and the task module with the highest probability $\mathcal{M}^*$ is selected for final prediction $\hat{y}$. This inference strategy enables dynamic, input-dependent module selection, where simpler samples are routed to lightweight models, and only complex samples are processed by high-capacity task modules.

Algorithm 1 CARN Training

Require: Dataset $\mathcal{D}$, pre-trained task modules $\{\mathcal{M}_j\}_{j=1}^M$,
feature extractor $\mathcal{F}$, CARN $\mathcal{R}$

```

1: while not converged do
2:    $(\mathbf{x}_i, y_i) \sim \mathcal{D}$  // Sample input
3:    $\mathbf{z}_i \leftarrow \mathcal{F}(\mathbf{x}_i)$  // Feature extraction
4:    $C_{ij} \leftarrow C_{ij}(\mathbf{z}_i)$  // Complexity scores (Eq. 1)
5:    $\mathcal{M}_i^* \leftarrow \arg \min_{\mathcal{M}_j} C_{ij}$  // Assign pseudo-label
6:    $\mathbf{p} \leftarrow \text{softmax}(\mathcal{R}(\mathbf{z}_i))$  // Routing probabilities
7:    $L \leftarrow L_{\text{route}}(\mathbf{p}, \mathcal{M}_i^*, C_{ij})$  // Routing loss (Eq. 5)
8:    $\theta_{\mathcal{R}} \leftarrow \theta_{\mathcal{R}} - \eta \nabla_{\theta_{\mathcal{R}}} L$  // Gradient update
9: end while

```

Algorithm 2 CARN Inference

Require: Sample $\mathbf{x}$, feature extractor $\mathcal{F}$, trained CARN $\mathcal{R}$, task modules $\{\mathcal{M}_j\}_{j=1}^M$

$$\begin{array}{ll}
1: \mathbf{z} \leftarrow \mathcal{F}(\mathbf{x}) & // \text{Feature extraction} \\
2: \mathbf{p} \leftarrow \text{softmax}(\mathcal{R}(\mathbf{z})) & // \text{Routing probabilities} \\
3: \mathcal{M}^* \leftarrow \arg \max_{M_j} \mathbf{p} & // \text{Select task module} \\
4: \hat{y} \leftarrow \mathcal{M}^*(\mathbf{z}) & // \text{Final prediction}
\end{array}$$

4. Experiments

4.1. Experimental Setup

Datasets: We use three of the standard, diverse, and challenging benchmarks for image classification, CIFAR-10 [15], CIFAR-100 [15], and Tiny-ImageNet [4], to ensure robust evaluation across varying levels of difficulty. CIFAR-10 consists of 60,000 color images of size 32×32 across 10 classes. CIFAR-100 shares the same structure as CIFAR-10 but contains 100 classes, increasing the complexity and requiring more fine-grained classification. Tiny-ImageNet is a subset of ImageNet with 200 classes, each having 500 training and 50 validation images of size 64×64 , providing a more complex and large-scale evaluation setting.

Baselines: We evaluate our proposed Complexity-Aware Routing Network (CARN) against VGG19 [28], ResNet50 [12], and DenseNet121 [13], which were chosen for their architectural diversity. VGG19 represents deep convolutional networks, ResNet50 introduces residual learning for stable gradient flow, and DenseNet121 leverages feature reuse through dense connectivity.

To create task-specific modules, we systematically prune each architecture at 50% and 90% using a structured pruning technique [7], producing three variants per backbone: p00 (unpruned), p50 (50% pruned), and p90 (90% pruned). These pruned models form the candidate task modules used by our CARN framework.

Implementation Details: After obtaining the candidate task modules, we generate pseudo-labels to train the CARN. Specifically, we pass each data sample through these candidate task modules and generate their complexity labels based on the computed complexity score as described in Section 3.1. The CARN is then trained using the proposed loss functions using Algorithm 1 with λ and γ set to 1 and 0.001, respectively. The learning process employs a learning rate of $\eta = 0.001$, with the SGD optimizer and a cosine annealing scheduler. For computing the complexity score, the hyperparameters are set to $\alpha = 0.6, \beta = 0.2$.

All hyperparameters used for training the framework are determined via grid search. The best-performing CARN model is selected based on its validation set performance, and the final results are reported using its inference performance on the test set.

4.2. Evaluation Metrics

Performance is evaluated using four key metrics: Task Accuracy (%), which measures classification accuracy on the test set; Average FLOPs (Avg-FLOPs), representing the mean floating-point operations per sample to quantify computational cost; Average Parameter Movement (Avg-Param), capturing the total volume of parameters transferred from memory to processing units per sample, a crucial efficiency bottleneck in modern hardware [10, 29]; and Average Speedup (Speedup), defined as the ratio of FLOPs in the unpruned model to the evaluated model, where higher values indicate improved computational efficiency while preserving accuracy.

4.3. Results and Analysis

Figure 2 illustrates the task module assignment and complexity score distribution, showing the number of CIFAR-10 samples from each class assigned to different VGG19-based task modules and the distribution of complexity scores for samples across the task modules. Notably, the distribution of complexity scores exhibits clear separation across the task modules, highlighting their varying capacity

Table 1. Quantitative comparison of CARN across CIFAR-10, CIFAR-100, and Tiny-ImageNet datasets using metrics: Task Accuracy (Accuracy), Average FLOPs (Avg-FLOPs), Average Parameter Movement (Avg-Param), and Average Speedup (Speedup).

Architecture	Task Module	CIFAR-10				CIFAR-100				Tiny-ImageNet			
		Accuracy (%)	Avg-FLOPs (1e8)	Avg-Param (1e6)	Speedup	Accuracy (%)	Avg-FLOPs (1e8)	Avg-Param (1e6)	Speedup	Accuracy (%)	Avg-FLOPs (1e8)	Avg-Param (1e6)	Speedup
VGG19	VGG_p00	93.31	7.99	20.04	1.00	71.71	7.99	20.08	1.00	48.98	7.99	20.13	1.00
	VGG_p50	91.36	3.55	5.28	2.25	68.70	4.98	10.03	1.60	44.14	3.55	5.33	2.25
	VGG_p90	86.91	2.03	0.48	3.94	62.11	2.49	1.87	3.20	43.23	2.49	1.89	3.20
	CARN (Ours)	91.25	2.04	0.51	3.92	68.22	2.58	2.21	3.09	44.04	2.77	2.84	2.88
ResNet50	ResNet_p00	94.91	26.09	23.52	1.00	76.07	26.09	23.70	1.00	57.80	26.10	23.91	1.00
	ResNet_p50	93.58	10.22	6.10	2.55	74.52	15.31	11.78	1.70	54.48	10.22	6.29	2.55
	ResNet_p90	85.53	4.78	0.46	5.45	71.43	6.40	2.12	4.07	52.94	6.40	2.18	4.07
	CARN (Ours)	93.34	12.52	8.83	2.08	73.07	6.50	2.23	4.01	54.75	7.90	3.83	3.30
DenseNet121	DenseNet_p00	95.07	17.96	6.95	1.00	78.24	17.96	7.04	1.00	55.37	17.96	7.15	1.00
	DenseNet_p50	93.89	10.50	2.09	1.71	75.14	10.50	2.14	1.71	50.80	10.50	2.19	1.71
	DenseNet_p90	88.39	7.79	0.45	2.30	49.81	7.79	0.46	2.30	47.65	8.57	0.95	2.09
	CARN (Ours)	93.70	7.80	0.46	2.30	75.23	8.08	0.65	2.22	50.92	8.84	1.12	2.03

to handle input samples of different difficulty levels. The streamlined, computationally efficient modules tend to assign lower complexity scores to a majority of the input data, suggesting that these models are well-suited for processing a significant portion of the dataset with minimal resource overhead. In contrast, only a relatively small fraction of samples are inherently more complex, necessitating the use of larger, more expressive models to achieve accurate predictions.

The results in Table 1 demonstrate the effectiveness of CARN in significantly reducing computational cost while maintaining high classification accuracy. Across all datasets, CARN achieves up to $4\times$ reduction in average FLOPs per sample while preserving accuracy within a minimal margin compared to the unpruned models. Notably, on CIFAR-100 with ResNet50, CARN reduces average FLOPs from 2.61B to just 0.65B (75.08% reduction), achieving a $4.01\times$ speedup while maintaining an accuracy of 73.07%.

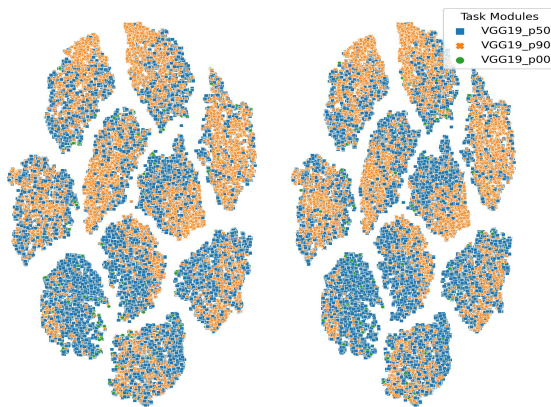


Figure 3. t-SNE visualization of sample routing in CARN using VGG19-based task modules. Left: Estimated complexity pseudo-labels for CIFAR-10 samples. Right: Predicted task modules for those samples.

Similar trends hold across Tiny-ImageNet and CIFAR-10, where CARN achieves over $3\times$ speedup with minimal accuracy drop.

Beyond computation, CARN also minimizes parameter movement, a key bottleneck in real-world deployment. For example, on CIFAR-100 with ResNet50, parameter movement is reduced from 23.70M to 2.23M, over a $10\times$ reduction, demonstrating CARN’s efficiency in reducing memory access overhead. This improvement directly translates to faster inference and lower energy consumption, making CARN a compelling choice for resource-constrained environments. The ability to dynamically route samples to appropriate model complexities ensures optimal trade-offs between computational efficiency and accuracy, making CARN an effective and scalable solution for adaptive deep learning inference.

Furthermore, the t-SNE visualizations in Figure 3 validate the correctness of CARN’s routing mechanism. The left plot represents the optimal task module assignments derived from minimum complexity scores, while the right plot showcases CARN’s predicted assignments. The strong correspondence between these distributions highlights that CARN effectively learns to route samples to the most suitable task module, preserving efficiency without sacrificing classification performance. This demonstrates that our framework can successfully infer complexity-aware routing decisions without explicit supervision, reinforcing its adaptability and impact in optimizing inference.

4.4. Ablation Study

4.4.1. Effects of α and β

To assess the influence of the routing parameters α and β , we trained the router network under various configurations and evaluated task accuracy alongside computational cost in FLOPs. As defined in Equation 1, α regulates the weight of entropy in the complexity score, guiding the router toward more confident and accurate models. Conversely, β introduces a penalty on computationally expensive models, enforcing a bias toward efficiency.

Empirical results reveal a fundamental trade-off: increasing α prioritizes task accuracy by favoring larger models, while increasing β reduces computational overhead by selecting smaller, less complex models. This interplay enables fine-grained control over the efficiency-performance spectrum, making it adaptable to different resource constraints.

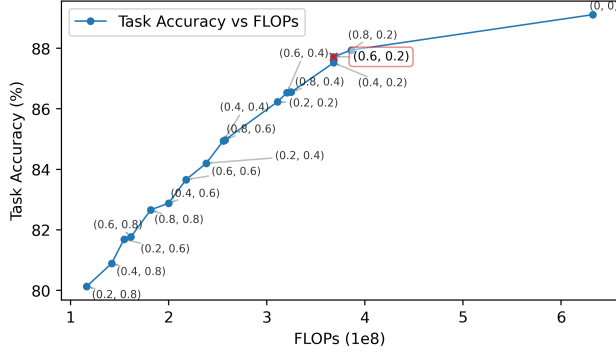


Figure 4. Trade-off between Performance and FLOPs for various values of $(\alpha$ and $\beta)$. The optimal trade-off is highlighted in the red box.

To identify an optimal operating point, we plotted the Pareto frontier of accuracy versus computational cost, as shown in Figure 4. Using the elbow selection method, we determined that $\alpha = 0.6$ and $\beta = 0.2$ offer the best trade-off, achieving 87.72% accuracy while reducing FLOPs by 42% compared to the unconstrained setting. Beyond this point, further increasing β results in significant accuracy degradation, while a higher α leads to diminishing accuracy improvements at a substantial computational cost. This selection ensures an optimal balance between accuracy and efficiency, making CARN suitable for deployment in resource-limited environments.

4.4.2. Effects of Loss Functions

To evaluate the impact of incorporating the complexity-aware loss function, we compare models trained with only the L_{assign} loss against those trained with the combined loss ($L_{assign} + \lambda L_{complex}$). As shown in Figure 5, the addition of $L_{complex}$ leads to a notable improvement in CARN accuracy, achieving a higher convergence rate and overall performance across epochs.

Beyond accuracy improvements, the $L_{complex}$ significantly reduces computational cost. The right plot in Figure 5 illustrates that while task accuracy remains consistently high, the FLOPs required for inference decrease substantially, highlighting the efficiency gains from optimized sample routing. This reduction is attributed to the complexity loss components, which promote more efficient task allocation by penalizing high-confidence, low-entropy, but computationally expensive paths. Consequently, the model

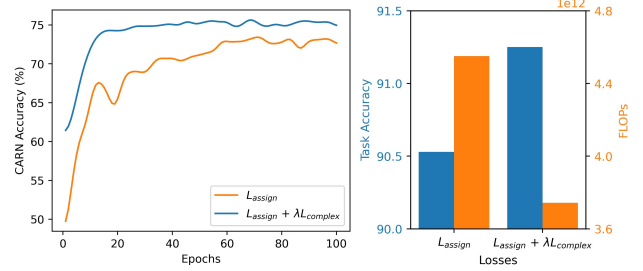


Figure 5. Impact of Complexity Loss on CARN Performance on CIFAR-10 dataset using VGG19-based task modules. Left: Improved training convergence. Right: Significant reduction in computations while maintaining high task accuracy.

achieves a balance between accuracy and efficiency, demonstrating the effectiveness of our proposed loss formulation.

5. Conclusion

In this work, we introduced CARN, a novel framework for adaptive inference that dynamically adjusts computational pathways based on input complexity. Unlike conventional static architectures that allocate uniform computational resources irrespective of sample difficulty, CARN leverages a self-supervised complexity estimation module to guide routing decisions, integrating confidence, entropy, and computational cost to determine the optimal task module for each input. This principled complexity-aware routing mechanism enables significant computational savings while maintaining competitive task performance.

Through extensive experiments on CIFAR-10, CIFAR-100, and Tiny-ImageNet, we demonstrated that CARN achieves up to $4\times$ reduction in computational cost and over $10\times$ reduction in parameter movement, all while preserving accuracy within a negligible margin of baseline static models. Beyond computation, CARN provides a scalable and generalizable approach to dynamic inference, effectively mitigating expert starvation while enhancing model specialization. The proposed routing loss function ensures optimal trade-offs between assignment accuracy and computational efficiency, paving the way for more robust and adaptive neural network architectures. Future research directions include extending CARN to more diverse architectures, such as transformer-based models, and exploring its applicability in multimodal and real-time decision-making systems.

Acknowledgment

This work is supported in part by the National Science Foundation under Grant No. 2504839.

References

- [1] Emmanuel Bengio, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup. Conditional computation in neural networks for faster models. *arXiv preprint arXiv:1511.06297*, 2015. 2
- [2] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*, 2022. 3
- [3] Xinshi Chen, Hanjun Dai, Yu Li, Xin Gao, and Le Song. Learning to stop while learning to predict. In *International conference on machine learning*, pages 1520–1530. PMLR, 2020. 2
- [4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 5
- [5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1
- [6] Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, et al. Layer-skip: Enabling early exit inference and self-speculative decoding. *arXiv preprint arXiv:2404.16710*, 2024. 3
- [7] Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. Depgraph: Towards any structural pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16091–16101, 2023. 1, 5
- [8] William Fedus, Jeff Dean, and Barret Zoph. A review of sparse expert models in deep learning. *arXiv preprint arXiv:2209.01667*, 2022. 2, 3
- [9] Rebati Gaire, Sepehr Tabrizchi, and Arman Roohi. Encode: Enhancing compressed deep learning models through feature—distillation and informative sample selection. In *2023 International Conference on Machine Learning and Applications (ICMLA)*, pages 633–638. IEEE, 2023. 1
- [10] Rebati Gaire, Sepehr Tabrizchi, Deniz Najafi, Shaahin Angizi, and Arman Roohi. Deco: Dynamic energy-aware compression and optimization for in-memory neural networks. In *2024 IEEE 67th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 1441–1445. IEEE, 2024. 5
- [11] Rebati Gaire, Sepehr Tabrizchi, and Arman Roohi. Resource-efficient adaptive-network inference framework with knowledge distillation-based unified learning. In *2024 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 508–513. IEEE, 2024. 3
- [12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5
- [13] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 5
- [14] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024. 1
- [15] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 5
- [16] Tianlin Liu, Mathieu Blondel, Carlos Riquelme, and Joan Puigcerver. Routers in vision mixture of experts: An empirical study. *arXiv preprint arXiv:2401.15969*, 2024. 1, 3
- [17] Sarthak Mittal, Yoshua Bengio, and Guillaume Lajoie. Is a modular architecture enough? *Advances in Neural Information Processing Systems*, 35:28747–28760, 2022. 2
- [18] Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeibi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. Compact language models via pruning and knowledge distillation. *Advances in Neural Information Processing Systems*, 37:41076–41102, 2024. 1
- [19] Jonas Pfeiffer, Sebastian Ruder, Ivan Vulić, and Edoardo Maria Ponti. Modular deep learning. *arXiv preprint arXiv:2302.11529*, 2023. 1, 2
- [20] Janarthanan Rajendran, Aravind Srinivas, Mitesh M Khapra, P Prasanna, and Balaraman Ravindran. Attend, adapt and transfer: Attentive deep architecture for adaptive transfer from multiple sources in the same domain. *arXiv preprint arXiv:1510.02879*, 2015. 3
- [21] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016. 1
- [22] Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34:8583–8595, 2021. 2
- [23] Clemens Rosenbaum, Ignacio Cases, Matthew Riemer, and Tim Klinger. Routing networks and the challenges of modular and compositional computation. *arXiv preprint arXiv:1904.12774*, 2019. 3
- [24] Simone Sarti, Eugenio Lomurno, and Matteo Matteucci. Anticipate, ensemble and prune: Improving convolutional neural networks via aggregated early exits. *Procedia Computer Science*, 222:519–528, 2023. 2
- [25] Simone Scardapane, Alessandro Baiocchi, Alessio Devoto, Valerio Marsocci, Pasquale Minervini, and Jary Pomponi. Conditional computation in neural networks: Principles and research trends. *Intelligenza Artificiale*, 18(1):175–190, 2024. 1, 2, 3
- [26] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarczyk, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017. 2, 3
- [27] Tianxiao Shen, Myle Ott, Michael Auli, and Marc’Aurelio Ranzato. Mixture models for diverse machine translation:

- Tricks of the trade. In *International conference on machine learning*, pages 5719–5728. PMLR, 2019. [1](#)
- [28] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. [5](#)
- [29] Sepehr Tabrizchi, Rebati Gaire, Mehrdad Morsali, Maximilian Liehr, Nathaniel Cady, Shaahin Angizi, and Arman Roohi. Apris: Approximate processing reram in-sensor architecture enabling artificial-intelligence-powered edge. *IEEE Transactions on Emerging Topics in Computing*, 2024. [5](#)
- [30] Shengkun Tang, Yaqing Wang, Zhenglun Kong, Tianchi Zhang, Yao Li, Caiwen Ding, Yanzhi Wang, Yi Liang, and Dongkuan Xu. You need multiple exiting: Dynamic early exiting for accelerating unified vision language model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10781–10791, 2023. [1](#)
- [31] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd international conference on pattern recognition (ICPR)*, pages 2464–2469. IEEE, 2016. [1](#), [2](#)
- [32] Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, et al. Scientific discovery in the age of artificial intelligence. *Nature*, 620(7972):47–60, 2023. [1](#)
- [33] Lu Yu and Wei Xiang. X-pruner: explainable pruning for vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 24355–24363, 2023. [1](#)
- [34] Mingyang Zhou, Luwei Zhou, Shuohang Wang, Yu Cheng, Linjie Li, Zhou Yu, and Jingjing Liu. Uc2: Universal cross-lingual cross-modal vision-and-language pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4155–4165, 2021. [1](#)
- [35] Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022. [1](#), [3](#)
- [36] Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022. [3](#)