Journal of Systems Architecture 175 (2026) 103753

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Accuracy-Aware IDK Cascades for Real-Time Object Classification at the Edge

Ishrak Jahan Ratul [a,*], Zhishan Guo [b], Kecheng Yang [a]

[a] Department of Computer Science, Texas State University, San Marcos, 78666, TX, USA
[b] Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, 27606, NC, USA

ARTICLE INFO

Keywords:
IDK classifier
Neural networks inference
Autonomous systems
Classifier cascade
Edge device

ABSTRACT

Real-time object classification on edge devices with constrained computing resources involves a trade-off between computation workload and classification accuracy. Existing classifier models are typically developed either for fast inference with compromised accuracy or for high accuracy with heavy computational cost. A recently proposed concept, called IDK ("I don't know") classifiers, enables cascading multiple existing classifiers to achieve high accuracy while reducing average inference time.

In this work, we compose IDK classifier cascades by fine-tuning existing models. We use the Tiny ImageNet and CIFAR-100 datasets with modified final layers for 200- and 100-class classification, respectively. We select a confidence threshold for each model to declare either a successful classification or an IDK decision. With a cascade of IDK classifiers, each input can be examined by more than one classifier, ordered from faster, less accurate ones to slower, more accurate ones. When an upstream classifier returns a class with high confidence, downstream classifiers are not executed, improving inference time. More accurate downstream classifiers are needed when upstream ones lack confidence.

Our experimental results demonstrate that IDK classifier cascades reduce average inference time while maintaining high classification accuracy, making them suitable for real-time AI applications on edge devices.

1. Introduction

Advances in machine learning (ML), particularly deep learning, have greatly impacted real-time object classification and data processing. Meanwhile, these developments often result in greater computational costs and higher average latency, posing challenges for systems that must satisfy real-time performance requirements [1].

In object recognition, classification is a vital task where ML or deep neural network (DNN) based classifiers are widely applied. Research on such classifiers has been mostly focused on achieving ever higher classification accuracy. The resulting complex classifiers, which have numerous layers and parameters designed to attain high accuracy, often become bottlenecks in real-time applications, slowing down the system and compromising its performance. A significant example is video streaming at 60 frames per second (FPS), which requires each frame to be processed within about 16 ms to compensate for any bottleneck.

Lightweight classifiers, on the other hand, are simpler models with fewer parameters and faster inference times that can handle large throughput but may fall short of the accuracy required for more complicated jobs. This trade-off between computing efficiency and accuracy is crucial for the system's stable performance [2].

Furthermore, at the user level, these classifiers are commonly deployed on edge devices, such as autonomous systems and robots. However, the resource constraints of edge devices exacerbate the challenge, requiring them to handle high computational demands with limited processing power, energy, and memory. This underscores the need for innovative techniques to balance execution time and accuracy in real-time systems [3,4].

The limitations of current classification models highlight the importance of an efficient approach. Lightweight classifiers excel at speed, making them ideal for simple cases with clear decision boundaries. However, they lack the ability to handle complex data, resulting in lower accuracy in difficult scenarios. However, high accuracy comes at the cost of much higher processing times and computational resource consumption from complex classifiers. This raises an important question: should lightweight classifiers be completely replaced with more accurate but slower models? Or is it possible to get a better solution by strategically combining the two types? Similar to the aforementioned video streaming system example, many frames could be accurately classified by lightweight models. For the fewer complex frames, a complex classifier may be required to ensure accuracy. This question can be extended to medium-sized models with moderate/intermediate classification accuracies — are they completely of no usage?

This insight motivates the development of an IDK (stands for "I Don't Know") classifier cascade framework that combines complex and lightweight classifiers to attain reasonable accuracy and computational

* Corresponding author.
E-mail address: ishrakratul@txstate.edu (I.J. Ratul).

https://doi.org/10.1016/j.sysarc.2026.103753
Received 14 August 2025; Received in revised form 24 January 2026; Accepted 23 February 2026
Available online 24 February 2026

efficiency. The IDK classifier cascade framework offers a systematic method for addressing the trade-off between computational efficiency and accuracy. In this framework, a classifier could normally produce a predicted class or output an IDK decision if the confidence of its prediction is low. An IDK cascade usually starts with a lightweight classifier that can handle the majority of instances efficiently. If the upstream classifier makes a high-confidence prediction, the result is accepted without activating any following downstream classifier in the cascade. However, if the confidence level falls below an acceptable confidence threshold, the upstream classifier would return IDK and the cascade would move forward to activate the next downstream classifier, which is expected to provide more accurate prediction with higher confidence as the cost of a high computation time. This process continues until a confident prediction is made or the last classifier in the cascade has been reached, which provides the highest accuracy but has the highest computational cost.

The IDK classifier cascade framework was proposed in prior research that emphasizes the trade-offs between model complexity, accuracy, and latency. In prior work, Baruah et al. [5] and Wang et al. [6] investigated the feasibility of IDK classifiers, which enable models to output an IDK decision when prediction confidence is low. This approach reduces unnecessary computations while increasing reliability by escalating uncertain cases to more robust classifiers. Extensions of this work, such as those by Abdelzaher et al. [7], concentrate on synthesizing cascades with arbitrary dependencies between classifiers while optimizing expected classification times under deadline constraints. Similarly, Wang et al. [6] propose using augmenting classifiers to measure uncertainty, emphasizing the utility of hybrid solutions that combine lightweight and complex models.

The IDK cascade structure has numerous key advantages. By using lightweight classifiers for simple inputs, the overall computational cost is decreased, leaving resource-intensive models for more complex scenarios. The architecture ensures that most inputs are processed quickly, allowing for responsiveness in real-time applications like video streaming and autonomous navigation. For complex inputs, the cascade effortlessly progresses to more powerful classifiers, ensuring accuracy without sacrificing efficiency. Furthermore, the framework can be adapted to specific applications by modifying the cascade's structure and confidence threshold in response to task needs and resource restrictions. Finally, in resource-constrained situations such as edge devices, the cascade makes optimal use of computational resources, allowing for high-performance inference without relying on cloud-based processing [5]. The IDK classifier cascade framework has significant implications for real-time systems in safety-critical and resource-constrained environments. In autonomous vehicles, where instantaneous decisions are crucial, the capacity to quickly process most inputs while allocating extensive computations for infrequent, complex situations improves both safety and efficiency. In video surveillance systems, lightweight classifiers process routine frames, whereas advanced models evaluate suspicious activity with higher accuracy. These applications illustrate how IDK facilitates optimal utilization of resources, ensuring prompt and reliable decisions.

Overall, the IDK classifier cascade framework brings a substantial advancement in enhancing real-time object recognition and data streaming. By strategically integrating lightweight and complex models, it achieves a balance between computational efficiency and accuracy, tackling significant challenges in contemporary real-time systems. The increasing demand for intelligent, resource-efficient systems necessitates the development and enhancement of hybrid approaches, which will be crucial in addressing the requirements of next-generation machine learning applications.

Contribution. This study presents a novel IDK classifier cascade framework designed to optimize accuracy and average inference time in real-time systems. Utilizing pre-trained deep neural network (DNN) models, we modify them such that they will output 'IDK' when the prediction confidence is below a certain threshold. The performance and total execution time of each classifier are evaluated, allowing for the development of a hierarchical cascade that favors lightweight classifiers for simpler cases and escalates more complex inputs to advanced classifiers. This design provides efficient utilization of computational resources while preserving higher prediction accuracy.

We implement and evaluate the IDK cascade framework on the NVIDIA Jetson AGX Orin platform to validate the practicality of our approach [8]. We adopt the Tiny Imagent dataset for the experiments [9]. The results of this evaluation demonstrate how well the cascades worked to lower average inference time while maintaining good accuracy. Our framework provides a scalable and effective solution for real-time applications by combining lightweight and complex classifiers, particularly in environments with limited resources and safety concerns. This contribution establishes the groundwork for future developments in hybrid decision-making frameworks for real-time machine learning systems.

Organization. The rest of this paper is organized as follows: Section 2 gives a background overview of IDK cascade. Section 3 describes the methodology of our research. Section 4 illustrates the results and evaluation, while Section 5 concludes our work.

2. IDK cascade

2.1. Related work

The IDK cascade framework is based on the foundational research of Baruah et al. [5,10,11], which emphasizes the necessity of enhancing efficiency and accuracy in real-time classification tasks for safety-critical systems. Their research highlights the inefficiencies of employing complex classifiers for all inputs, particularly in applications such as autonomous systems, where prompt decision making is essential. Lightweight classifiers, despite their computational efficiency, are inadequate for processing complex inputs, resulting in lower accuracy. In contrast, complex classifiers, while precise, involve a significant computational cost and are inappropriate for resource-limited settings, as evidenced by previous research [6].

The IDK classifier concept, presented by Baruah et al. [5] and subsequently improved by Abdelzaher et al. [7], incorporates a self-aware mechanism by which classifiers produce an IDK decision for predictions with low confidence. This facilitates cascading by postponing ambiguous classifications to later, more proficient classifiers, thereby ensuring computational efficiency for simpler instances while achieving high accuracy for complex scenarios. This hierarchical framework optimizes resource utilization and ensures adherence to deadlines in real-time tasks, as demonstrated in previous research on deadline-constrained cascade synthesis.

Preliminary results of this work were presented in [12], where we introduced a practical implementation of IDK classifier cascades. That study empirically demonstrated that cascading IDK-enabled deep neural networks can significantly accelerate inference while preserving accuracy.

Our work extends foundational research by implementing and evaluating a practical IDK cascade framework. Unlike prior studies that focus on theoretical scheduling, we empirically assess multiple IDK cascades, demonstrating significant inference speedup while maintaining high accuracy.

2.2. What is an IDK cascade

An IDK cascade is a structured framework for real-time classification that processes inputs sequentially using a series of classifiers. Each classifier either makes a confident prediction or returns an IDK result and passes the input to the next classifier in the sequence. This framework achieves a balance of computational efficiency and accuracy by using lightweight classifiers for simple inputs and complex classifiers for more difficult ones. Fig. 1 illustrates the working principle of an IDK cascade.

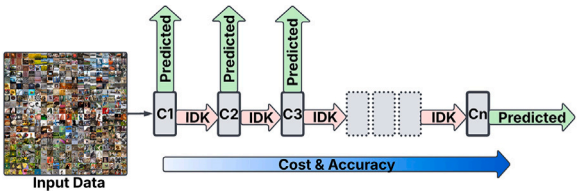


Fig. 1. An IDK cascade with classifier, C1, C2, C3, ...,Cn.

2.3. Decision rule

Each classifier in the cascade operates based on a confidence threshold. Let C_i represent the ith classifier in the cascade, and x denote an input. The confidence score $P(C_i(x))$ indicates the classifier's probability in predicting the correct class. If this confidence score exceeds a predefined threshold τ_i, the classifier outputs the predicted class y_i. Otherwise, it outputs the IDK class and passes x to the next classifier C_{i+1}. This can be denoted as:

$$\text{Output of } C_i(x) = \begin{cases} y_i, & \text{if } P(C_i(x)) \geq \tau_i, \\ \text{IDK}, & \text{if } P(C_i(x)) < \tau_i. \end{cases} \tag{1}$$

Here:

- $P(C_i(x))$: Confidence score of C_i for input x,
- τ_i: Confidence threshold for C_i,
- y_i: Predicted class by C_i.

This ensures that lightweight classifiers process most inputs quickly, escalating only less confident cases to more complex classifiers [10].

2.4. Expected inference time

The efficiency of the cascade is measured by its expected inference time T, which is the average time taken to classify an input. This is calculated as the weighted sum of the execution times t_i of each classifier C_i, where the weights represent the probabilities that an input reaches C_i:

$$T = \sum_{i=1}^{n} P_{\text{IDK},i-1} \cdot t_i, \tag{2}$$

where:

- n: Total number of classifiers in the cascade,
- $P_{\text{IDK},i-1}$: Probability that an input reaching C_i, derived from the cumulative IDK rates of preceding classifiers,
- t_i: Execution time of C_i.

The probability $P_{\text{IDK},i-1}$ is recursively calculated as:

$$P_{\text{IDK},i} = P_{\text{IDK},i-1} \cdot P(C_i(x) = \text{IDK}), \tag{3}$$

with $P_{\text{IDK},0} = 1$, as all inputs initially start at C_1 [6].

2.5. Optimization of confidence thresholds

Confidence thresholds τ_i play a important role in balancing false positives (unnecessary escalation to IDK) and false negatives (misclassifications). Optimizing these thresholds ensures that the cascade reduces inference time while maintaining accuracy. The optimization problem can be stated as follows:

$$\min_{\tau_1,\tau_2,\ldots,\tau_n} T \quad \text{subject to} \quad A \geq A_{\min}, \tag{4}$$

where $A_{\min}$ is the minimum acceptable accuracy for the cascade [7].

2.6. Workflow of an IDK cascade

1. **Input Processing:** The input x is initially evaluated by the lightweight classifier C_1. If $P(C_1(x)) \geq \tau_1$, the output is the predicted class y_1. Otherwise, C_1 outputs IDK, and x is passed to C_2.
2. **Escalation:** Each subsequent classifier C_i processes the input x, and makes a decision based on the confidence threshold τ_i. This process repeats until a confident prediction is made or x reaches the final classifier.
3. **Final Classification:** The final classifier C_n ensures that all inputs are classified, even if all previous classifiers output IDK.

2.7. Advantages

- **Computational Efficiency:** Lightweight classifiers handle simpler cases, resulting in shorter average inference times.
- **Accuracy:** Escalating uncertain inputs to complex classifiers ensures acceptable high accuracy.
- **Adaptability:** Confidence thresholds can be adjusted for specific applications to balance efficiency and accuracy.
- **Real-Time Applicability:** The framework is suitable for resource-constrained systems, such as NVIDIA Jetson platforms, to ensure reliable real-time performance [5].

2.8. Our considerations

Our IDK cascade was designed to use state-of-the-art classifiers for object recognition tasks, including convolutional neural networks (CNNs) and transformer-based models. These architectures were chosen for their proven performance to achieve high accuracy on a wide range of object recognition benchmarks [13]. To ensure the cascade's reliability, we used a systematic threshold search technique to calculate the confidence threshold for each classifier. This step was critical for balancing false positives and false negatives while accepting confident predictions and escalating uncertain inputs to the next stage in the cascade.

We used a sequential classification approach to design the cascade, processing inputs frame by frame, which is equivalent batch size of one. This decision was influenced by the intended use of the cascade on resource-constrained edge devices. Edge devices, such as those used in self-driving cars, are frequently required to handle multiple real-time tasks at once. For example, these systems process data from a variety of sensors, including LiDAR and radar, which all require real-time computational resources. By avoiding batch processing or parallel execution, we ensure that system utilization remains efficient, allowing enough capacity for other critical tasks [14].

3. Methodology

In this section we present the experimental setup, the fine-tuning of the pre-trained models, and the selection of classifiers based on accuracy and inference time. We also define confidence thresholds for decision making and detail the cascade synthesis process.

3.1. Setup

Our IDK cascade framework was designed with the goal of improving real-time classification performance on edge platforms. We chose the NVIDIA Jetson AGX Orin developer kit with 2048-core NVIDIA Ampere GPU, 64 Tensor cores, and dual NVDLA accelerators, making it ideal for real-time tasks that require high computational efficiency and low power consumption [8].

PyTorch was part of our experimental software stack. For evaluation, we used the Tiny ImageNet dataset, a benchmark for multi-class object recognition tasks. This dataset consists of 200 classes, each with

500 training images and 50 validation images. Additionally, a validation set of 10,000 images was available [9]. For a better comparison of our result with the state of the art, we use the original validation set as the test set and divide the training set into 90% and 10% to obtain a new training and validation set. This ensured that our models were evaluated on previously unseen data, ensuring no data leakage in testing process. The primary goal of this study was to create a system capable of providing low-average inference time, high-accuracy classification.

In addition, we evaluated our approach on the CIFAR-100 dataset to assess the generalizability of the IDK cascade concept. CIFAR-100 contains 100 classes with 600 images per class and provides a complementary evaluation setting with lower-resolution images and a different class distribution.

3.2. Fine-tuning pretrained models

We fine-tuned a diverse set of widely used, state-of-the-art object recognition models to construct an efficient IDK cascade framework. For experiments on the Tiny ImageNet dataset, we evaluated AlexNet [15], ResNet18 [16], VGG16 [17], Inception V3 [18], SqueezeNet [19], EfficientNet [20], and Swin Transformer [21]. All models were pretrained on the ImageNet dataset, and their final classification layers were adapted to produce predictions for 200 classes corresponding to Tiny ImageNet. This ensured architectural diversity while maintaining a consistent training and evaluation pipeline.

For experiments on the CIFAR-100 dataset, we fine-tuned AlexNet, ResNet18, VGG16, SqueezeNet, EfficientNet, and Inception V3, while excluding the Swin Transformer. Instead, we incorporated ConvNeXt [22], a modernized convolutional architecture that has demonstrated competitive performance with vision transformers on large-scale image classification benchmarks. ConvNeXt preserves convolutional inductive biases while adopting contemporary design choices, making it a strong and widely used baseline for evaluating generalization across datasets with differing image resolutions and class distributions.

For fine tuning, we used the Distributed Data Parallel (DDP) paradigm to enable parallel training on multiple GPUs. This setup ensured optimal utilization of computational resources and significantly accelerated the training process. The training was carried out on a high performance server equipped with dual NVIDIA RTX A4000 GPUs. The loss function for training was the loss of cross-entropy, which was implemented with `nn.CrossEntropyLoss()`. Stochastic Gradient Descent (SGD) was used as the optimizer, with a learning rate of 0.001 and a momentum of 0.9, to ensure effective gradient updates for model convergence. The models were trained for 15 epochs, with early stopping to prevent overfitting and ensure peak performance. Early stopping tracked validation loss and stopped training when no significant improvement was seen over successive epochs.

To improve the models' generalization ability, we used advanced training strategies such as learning rate scheduling, data augmentation (random cropping, flipping, and color jittering), and dropout regularization.

3.3. Selecting classifiers for IDK cascade

We chose classifiers for the IDK cascade based on accuracy versus inference time. The goal was to find a balance by prioritizing higher accuracy models with shorter inference times. To achieve this, we empirically evaluated a set of fine-tuned classifiers and selected dataset-specific cascades based on the performance metrics reported in Tables 1 and 2.

Table 1
Performance metrics of classifiers after fine tuning (Tiny ImageNet).

Classifier	Top 1 Accuracy (%)	Average inference time (mS)
Alexnet	59.57	1.798
VGG 16	66.43	3.800
Squeezenet	40.88	5.055
Resnet 18	71.98	5.467
Efficientnet	79.22	17.830
Inception v3	45.90	23.700
Swin Transformer	89.56	45.428

Table 2
Performance metrics of classifiers after fine tuning (CIFAR-100).

Classifier	Top 1 Accuracy (%)	Average inference time (mS)
Alexnet	72.85	2.907
VGG 16	77.96	6.265
Squeezenet	70.19	7.665
Resnet 18	77.66	7.920
ConvNeXt	85.46	17.100
Efficientnet	85.75	27.324
Inception v3	72.24	35.938

3.3.1. Tiny ImageNet

Based on the data form Table 1, we chose AlexNet, VGG16, ResNet18, EfficientNet, and Swin Transformer for inclusion in the cascade.

AlexNet (59.19% accuracy, 1.798 ms inference time) was chosen because of its extremely low inference cost, making it an excellent candidate for the cascade's initial stage. VGG16 (66.43% accuracy, 3.800 ms inference time) performed moderately. ResNet18 (72.41% accuracy, 5.467 ms inference time) was chosen for its high accuracy while remaining computationally efficient. EfficientNet (79.45% accuracy, 17.830 ms inference time) produced a higher accuracy model with a reasonable inference latency, making it appropriate for later stages of the cascade. Finally, Swin Transformer (89.56% accuracy, 45.428 ms inference time) was selected as the cascade's final classifier because it achieved the highest accuracy while incurring a significant computational cost, making it a reliable fallback for difficult cases.

In contrast, Inception V3, and SqueezeNet were excluded from the cascade. Inception V3 (45.90% accuracy, 23.700 ms inference time) had a long inference time compared to its accuracy, making it unsuitable. SqueezeNet (40.88% accuracy, 5.055 ms inference time) was the least accurate of the models tested and did not justify its computational cost in the cascade structure.

3.3.2. CIFAR-100

Classifier selection for the CIFAR-100 dataset was guided by the performance metrics in Table 2. Based on this data, AlexNet, VGG16, and ConvNeXt were selected for inclusion in the CIFAR-100 cascade.

AlexNet achieved a top-1 accuracy of 72.85% with the lowest average inference time among all evaluated models (2.907 ms), making it an effective first-stage classifier for rapidly processing simpler inputs. VGG16 improved classification accuracy to 77.96% while maintaining a relatively low inference time of 6.265 ms, offering a favorable balance between accuracy gain and computational cost as an intermediate stage. ConvNeXt provided a substantial increase in accuracy, achieving 85.46% top-1 accuracy with an average inference time of 17.100 ms, and was therefore selected as the final stage of the cascade to handle the most difficult samples.

Other models were excluded from the CIFAR-100 cascade due to less favorable trade-offs. Although EfficientNet achieved a comparable accuracy of 85.75%, its significantly higher average inference time (27.324 ms) made it less suitable for inclusion. Inception V3 exhibited a high inference cost (35.938 ms) without a corresponding accuracy improvement, while SqueezeNet and ResNet18 offered lower accuracy (70.19% and 77.66%, respectively) without sufficient advantages to justify their inclusion.

3.4. Confidence threshold

A threshold selection strategy that maximized the trade-off between accuracy and computational efficiency was used to calculate the confidence threshold for each classifier in the IDK cascade. The goal was to construct a threshold value τ so that predictions with confidence scores above this threshold would be considered acceptable, while predictions with lower confidence would indicate IDK and escalated to the next classifier in the cascade. This procedure ensured that each classifier only contributed to the final choice when its confidence was high enough, reducing misclassifications and computational cost.

We used histogram plots of confidence values and distinguish between correct and incorrect classifications which are shown (only for Tiny ImageNet data, as we followed exactly the same approach for CIFAR-100 data) in Figs. 2, 3, 4, 5 and 6. This visualization helped to determine how well the confidence scores matched classification accuracy and provided an optimal threshold. A larger difference between the confidence distributions of the correct and incorrect classifications indicated a well-trained model, where high confidence predictions are more likely to be correct.

To determine the optimal confidence threshold, we used the validation data in our selected classifiers. The accuracy $A(\tau)$ for a given threshold τ was computed as:

$$A(\tau) = \frac{\sum_{x \in \text{correct}} I(C(x) > \tau)}{\sum_{x \in \text{correct}} I(C(x) > \tau) + \sum_{x \in \text{incorrect}} I(C(x) > \tau)} \quad (5)$$

where:

- $I(C(x) > \tau)$ is an indicator function that takes the value 1 if the confidence score of prediction $C(x)$ is greater than τ, and 0 otherwise.
- $x \in$ correct represents correctly classified samples.
- $x \in$ incorrect represents incorrectly classified samples.

This algorithm iterated all samples that are sorted by confidence values in decreasing order and evaluated the accuracy $A(\tau)$ among the subset of samples with confidence at or above a given confidence threshold. We call this accuracy over the more confident subset of samples only as the *precision* of this classifier to be consistent with the terminology in prior work [7]. Please note that the samples with confidence lower than the confidence threshold are to be classified as IDK. Therefore, the precision of a classifier is indeed its accuracy over the set of samples for which it does not classify as IDK. That is, for each classifier and for each desired precision, a confidence threshold can be determined accordingly.

This confidence-based thresholding mechanism ensured that each classifier in the IDK cascade only accepted predictions it was confident about. By imposing a minimum confidence requirement, the cascade effectively reduced misclassifications while efficiently escalating uncertain cases to more complex classifiers.

3.5. Synthesis of IDK cascade

The IDK cascade was created by carefully selecting and sequencing classifiers according to their inference time. The chosen models were arranged in a progressive cascade to prioritize computational efficiency. The sequence was chosen so that the fastest models was at the top, allowing for quick classification of simple inputs, while more complex models were positioned further down the cascade to handle more difficult cases.

Each classifier's confidence threshold was predefined, so a model would only accept a prediction if its confidence exceeded the threshold. Otherwise, it produces an IDK result and the input was forwarded to the next model in the cascade. This threshold-based approach optimized the balance between computational efficiency and prediction accuracy.

During inference, each input frame was processed sequentially through the cascade. Single frame inference which is equivalent to batch size of 1 was used to keep total system utilization lower as there might be some other tasks simultaneousy in a real-time-system. We designed and evaluated four different cascade configurations for the Tiny ImageNet data, as illustrated in Figs. 7(a), 7(b), 7(c), and 7(d). For the CIFAR-100 data we evaluated two cascade configurations, illustrated in Figs. 8(a) and 8(b). In each cascade, lightweight classifiers were positioned at the initial stages to handle simpler inputs efficiently, while more complex models were placed progressively deeper in the

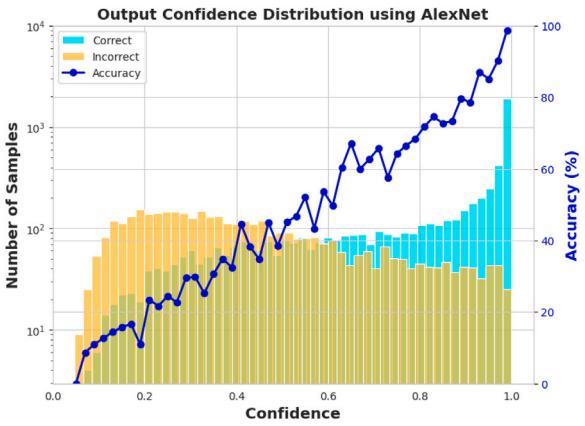


Fig. 2. Frequency distribution of output confidence in validation data by Alexnet (Tiny ImageNet).

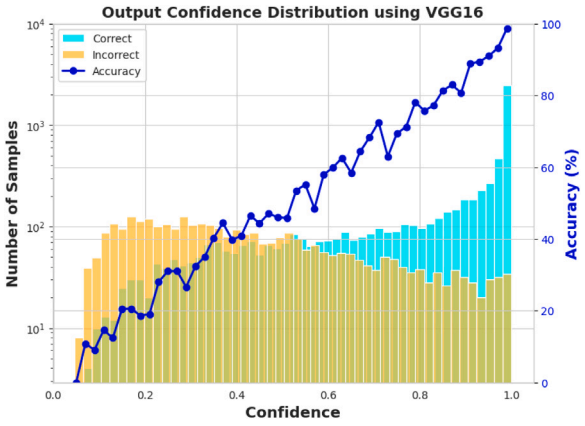


Fig. 3. Frequency distribution of output confidence in validation data by VGG16 (Tiny ImageNet).

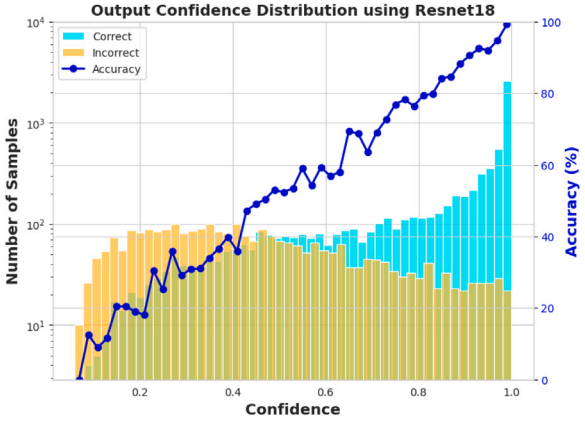


Fig. 4. Frequency distribution of output confidence in validation data by Resnet18 (Tiny ImageNet).

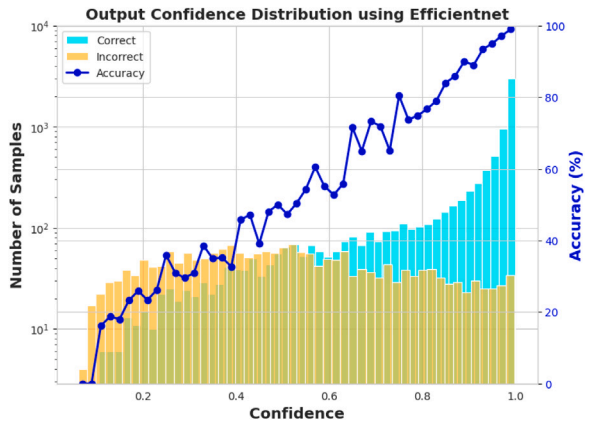


Fig. 5. Frequency distribution of output confidence in validation data by Efficinetnet (Tiny ImageNet).

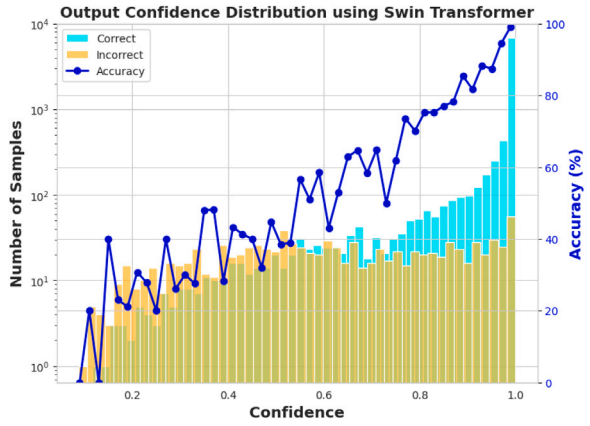


Fig. 6. Frequency distribution of output confidence in validation data by Swin Transformer (Tiny ImageNet).

hierarchy to process more challenging cases. Notably, the last classifier in each cascade did not require a confidence threshold, as all its predictions should be considered as final output.

To ensure efficient model execution, each classifier was loaded with pretrained weights, adjusted to match the dataset structure, and then set to the evaluation mode. The cascade significantly reduced overall inference costs because early-stage models confidently classified the majority of images, reducing the need for computationally expensive models.

We prioritized low-latency predictions by structuring the cascade with a progressive complexity strategy, escalating uncertain cases to more sophisticated models only when necessary. This design provided an optimal balance of speed and accuracy, making the IDK cascade ideal for real-time object recognition on edge devices.

4. Results and evaluation

This section we present the results analysis of our experiments. We discuss the impact of IDK cascades on classification accuracy, inference time, and overall efficiency.

Individual Classifier Performance. The individual classifier performance reported in Tables 1 and 2 provides insight into the accuracy versus inference time trade-offs across models and datasets. The results reveal a clear trend of higher model complexity that generally leads to improved classification accuracy, but at a significantly increased inference time.

On Tiny ImageNet (Table 1), AlexNet achieves 59.57% accuracy with a very low inference time of 1.798 ms, making it suitable for time-critical scenarios but limited in handling complex features. VGG16 improves accuracy to 66.43% at 3.800 ms, while ResNet18 offers a stronger balance with 71.98% accuracy at 5.467 ms. SqueezeNet (40.88%, 5.055 ms) and Inception V3 (45.90%, 23.700 ms) show less favorable accuracy–inference time trade-offs. More complex models provide higher accuracy at significantly higher cost: EfficientNet reaches 79.22% at 17.830 ms, and Swin Transformer achieves the best accuracy of 89.56% but with a much higher latency of 45.428 ms.

A similar trend appears on CIFAR-100 (Table 2). AlexNet provides fast inference (72.85%, 2.907 ms), while VGG16 (77.96%, 6.265 ms) and ResNet18 (77.66%, 7.920 ms) offer improved accuracy with moderate inference time. SqueezeNet (70.19%, 7.665 ms) and Inception V3 (72.24%, 35.938 ms) are less efficient relative to their cost. Among higher-capacity models, ConvNeXt achieves 85.46% accuracy at 17.100 ms, offering a strong balance, while EfficientNet slightly improves accuracy to 85.75% but with a higher inference time of 27.324 ms.

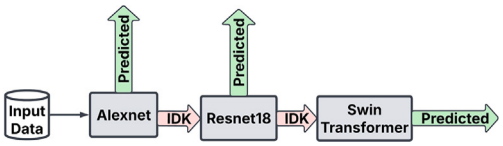
Overall, these findings highlight the intrinsic trade-off between accuracy and computational cost across both datasets. Lightweight models such as AlexNet, VGG16, and ResNet18 are better suited for faster inference where inference time is critical, while more complex architectures such as EfficientNet, ConvNeXt, and Swin Transformer are more appropriate for high-confidence decision stages where accuracy is prioritized over speed.

Performance of IDK Cascades. In Tiny ImageNet data Fig. 10 shows that the cascades achieve accuracy close to that of the most complex standalone model, while maintaining lower inference times, as illustrated in Fig. 9. This demonstrates the effectiveness of the cascade strategy in balancing predictive performance with inference time constraints.

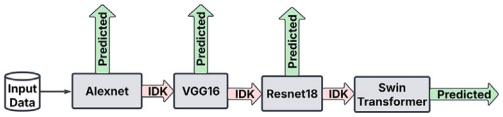
IDK Cascade 1 (precision = 91%), which predominantly relies on lightweight models such as AlexNet and ResNet18, achieves an accuracy of 83.14% with an average inference time of 17.612 ms. This indicates that a large proportion of inputs can be classified confidently using low-cost models, significantly reducing computational overhead. As the confidence threshold increases, IDK Cascade 2 (precision = 93%) improves accuracy to 85.09% with only a moderate increase in inference time to 20.686 ms, showing that selectively forwarding more uncertain inputs to EfficientNet refines predictions with limited latency impact. A more pronounced shift is observed in IDK Cascade 3 (precision = 96%), which achieves 87.96% accuracy at 26.785 ms, reflecting increased utilization of Swin Transformer for more ambiguous cases. Finally, IDK Cascade 4 (precision = 98%) reaches 89.35% accuracy, closely matching the standalone Swin Transformer accuracy of 89.56%, but with an inference time of 33.126 ms, representing a 27% reduction compared to Swin Transformer's standalone latency of 45.428 ms. This demonstrates that the cascade preserves high accuracy while substantially lowering average inference time, making it well suited for resource-constrained environments such as edge devices and embedded AI systems.

The CIFAR-100 results also show that the IDK cascades achieve accuracy close to high-complexity standalone models while significantly reducing inference time (Figs. 12 and 11).

For Cascade 1, a lower confidence threshold (precision=92%) allows most inputs to be processed by lightweight classifiers, achieving 83.98% accuracy with an average inference time of only 2.88 ms. This indicates that a substantial portion of CIFAR-100 samples can be classified without invoking deeper models, minimizing computational cost. Increasing the precision to 95% improves accuracy to 84.77% with a modest rise in latency to 4.06 ms, demonstrating that forwarding uncertain inputs to deeper stages yields accuracy gains with limited overhead. At a 98% precision, Cascade 1 reaches 85.33% accuracy but requires 20.36 ms, reflecting heavier reliance on ConvNeXt for difficult samples. Cascade 2 shows a similar pattern, where accuracy increases



(a) IDK cascade 1.

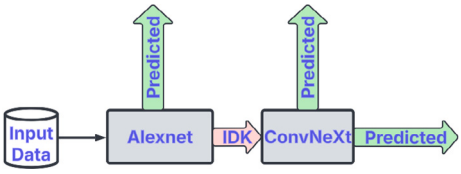


(b) IDK cascade 2.

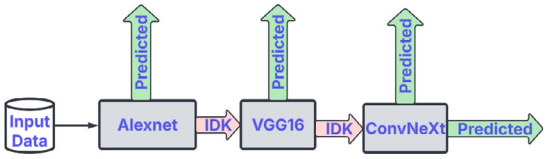
(c) IDK cascade 3.

(d) IDK cascade 4.

Fig. 7. IDK cascade structures (Tiny ImageNet Data).



(a) IDK cascade 1.



(b) IDK cascade 2.

Fig. 8. IDK cascade structures (CIFAR-100 Data).

from 83.08% at 92% precision (3.30 ms) to 85.46% at 98% precision (7.95 ms). Even at higher thresholds, both cascades maintain significantly lower latency than running ConvNeXt (17.1 ms) or EfficientNet (27.3 ms) independently, while achieving comparable accuracy.

An important observation is the adaptive behavior of cascades, where deeper models are engaged only when necessary, resulting in efficient use of complex classifiers. This is particularly beneficial for applications that require low-latency inference with an acceptable accuracy. The results also show that incremental accuracy gains come at progressively higher latency costs, emphasizing the need for careful threshold tuning to balance efficiency and accuracy. Moreover, the inference time variation shown in Figs. 9 and 11 indicates that deeper models introduce greater latency variability.

Inference Time Distribution. The distribution of inference times across individual classifiers and IDK cascades, as shown in Figs. 9 and 11, demonstrates the clear computational efficiency benefits of the cascading approach on both Tiny ImageNet and CIFAR-100. On Tiny ImageNet, lightweight models such as AlexNet and ResNet18 exhibit low variance in inference time with mean values of 1.798 ms and 5.467 ms, respectively, due to their relatively simple architectures and consistent execution behavior across inputs. In contrast, EfficientNet and Swin Transformer show greater variability in inference time, with mean values of 17.830 ms and 45.428 ms, respectively, reflecting their higher computational complexity and more extensive feature extraction mechanisms.

The IDK cascades significantly reduce overall inference time variance by invoking complex models only when necessary, allowing most inputs to be processed efficiently by lightweight classifiers.

A similar trend is observed on CIFAR-100. Lightweight models such as AlexNet (2.907 ms) and VGG16 (6.265 ms) demonstrate stable and low-latency behavior, while more complex models like ConvNeXt (17.100 ms) and EfficientNet (27.324 ms) exhibit higher latency and greater variability. The CIFAR-100 cascades maintain low average inference times at moderate confidence thresholds by classifying most samples in early stages.

Impact of Different Thresholds for a Cascade. The choice of confidence thresholds in the IDK classifier cascade plays a critical role in balancing inference speed and classification accuracy on both

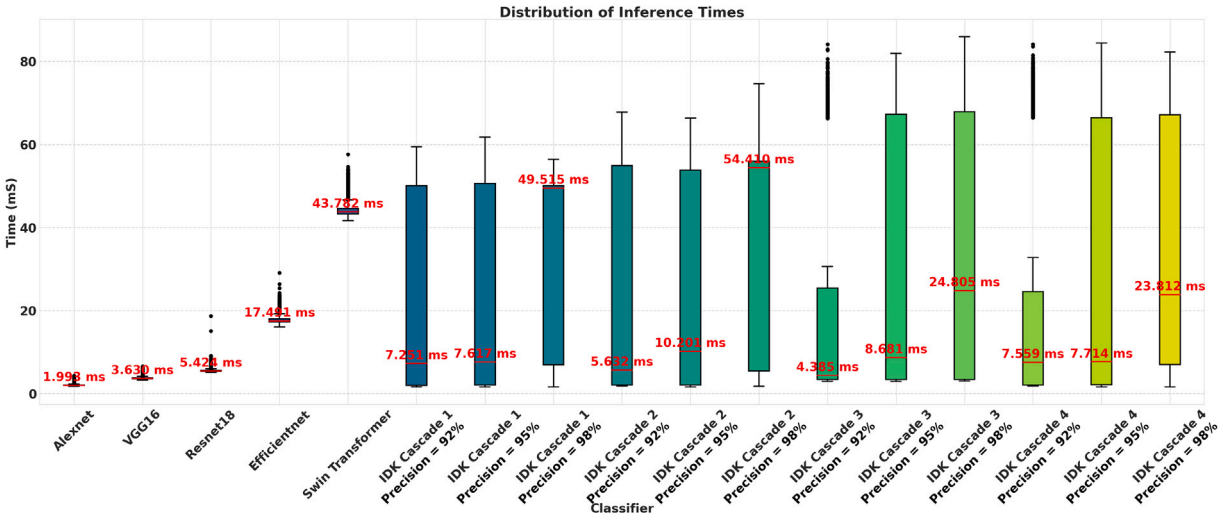


Fig. 9. Distribution of inference times varying the precision (Tiny ImageNet Data).

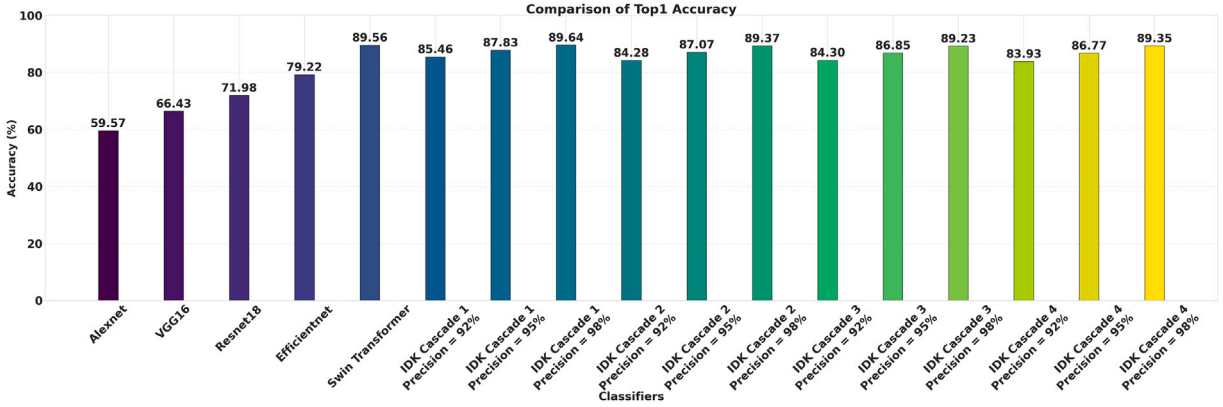


Fig. 10. Comparison of test accuracy (Tiny ImageNet Data).

Tiny ImageNet and CIFAR-100. As shown in Table 3 and reflected in the accuracy and latency trends in Figs. 9, 10, 11, and 12, lower confidence thresholds allow lightweight classifiers to handle the majority of inputs with minimal inference time. However, this comes at the cost of reduced accuracy, since fewer samples are escalated to deeper models that can extract more discriminative features. Increasing the confidence threshold pushes more uncertain inputs to later stages of the cascade, improving overall accuracy but incurring additional computational overhead.

On Tiny ImageNet, IDK Cascade 3 (precision = 95%) achieves a strong balance, reaching 86.85% accuracy with an average inference time of 8.681 ms. This makes it particularly suitable for scenarios requiring both responsiveness and reliable classification performance. A similar trade-off is observed on CIFAR-100. For example, Cascade 1 improves from 83.98% accuracy at a precision of 92% (2.88 ms) to 84.77% at precison 95% (4.06 ms), and further to 85.33% at 98% precision (20.36 ms). This demonstrates that moderate thresholds effectively control computational cost while still achieving high classification accuracy.

The findings suggest that dynamic confidence threshold tuning could further enhance efficiency, allowing cascades to adapt in real time based on input complexity or system resource constraints. An adaptive thresholding mechanism would enable automatic regulation of the accuracy–latency trade-off, optimizing performance under varying workload conditions.

It is also important to note that although individual classifiers operate at high precision levels (typically in the 92%–95% range), the overall cascade accuracy can remain below 90%. This may appear to contradict the observations in Sec. 6.6 and Table 14 of [7]. The reason is that easier samples are often resolved by earlier, less powerful classifiers, while later-stage models are exposed primarily to harder, more ambiguous inputs. As a result, the effective precision of deeper classifiers within the cascade may be lower than their standalone precision. This highlights that individual classifier precision alone is insufficient to predict overall cascade accuracy, and further analysis is needed to better model these interactions.

Comparison with State-of-the-Art. On Tiny ImageNet, IDK Cascade 1, which primarily relies on lightweight classifiers, achieves 83.14% accuracy and is 2.57× faster than the standalone Swin Transformer, reducing inference time from 45.428 ms to 17.612 ms. As deeper models are incorporated, IDK Cascade 2 achieves 85.09% accuracy with a 2.2× speedup, while IDK Cascade 3 reaches 87.96% accuracy at approximately 1.7× the speed of Swin Transformer. IDK Cascade 4, the most powerful configuration, achieves 89.35% accuracy, closely matching Swin Transformer's 89.56% accuracy but with a 27% reduction in inference time (33.126 ms versus 45.428 ms). These results confirm that cascades can preserve near state-of-the-art accuracy while substantially lowering computational cost.

A similar efficiency trend is observed on CIFAR-100. For example, at moderate thresholds, the cascades reach accuracy levels above 84% with inference times below 5 ms, compared to 17.100 ms for ConvNeXt and 27.324 ms for EfficientNet. Even at higher thresholds where ConvNeXt is engaged more frequently, the cascades maintain competitive

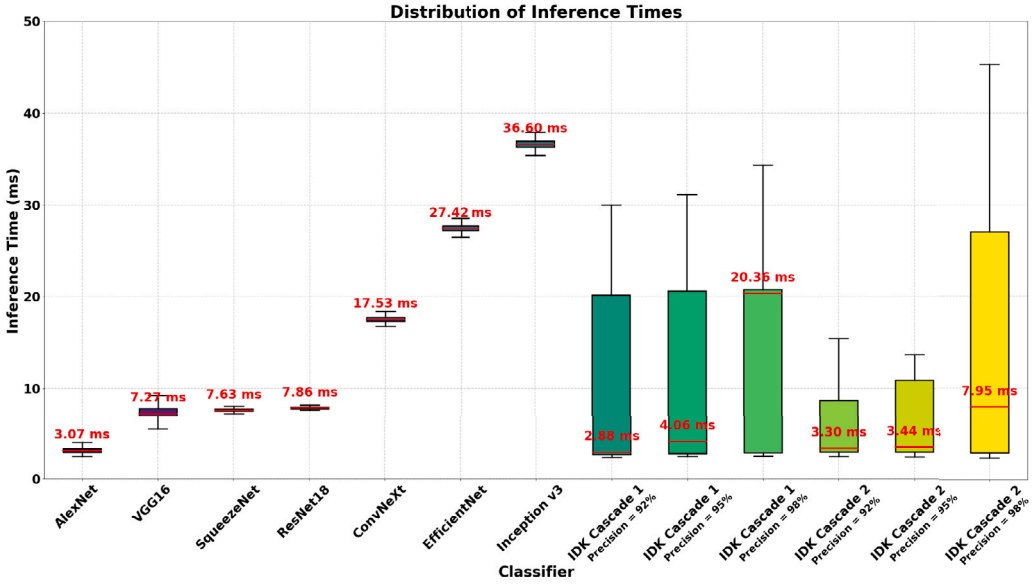


Fig. 11. Distribution of inference times varying the precision (CIFAR-100 Data).

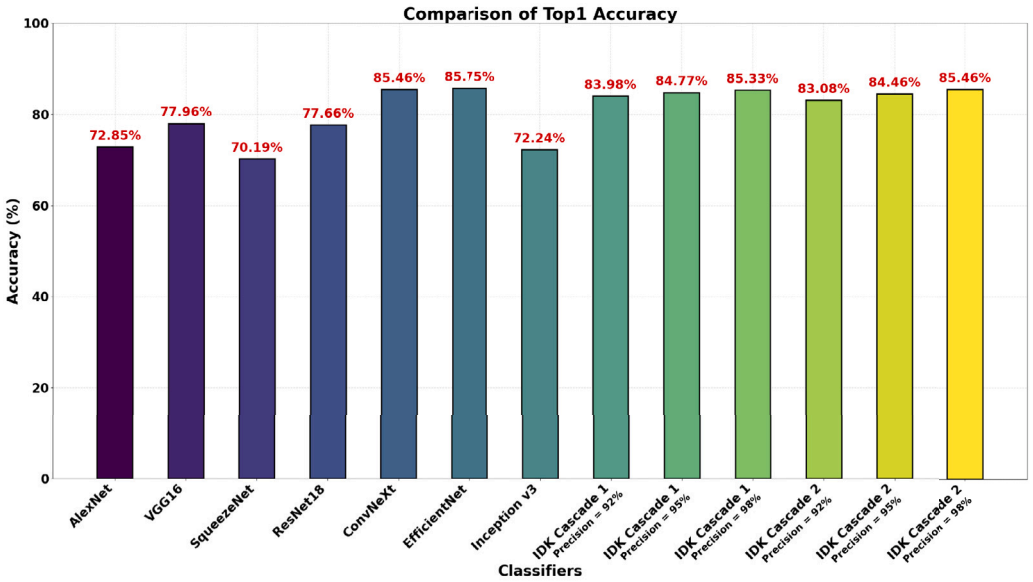


Fig. 12. Comparison of test accuracy (CIFAR-100 Data).

accuracy while still reducing average inference time compared to executing the deepest models on every input. This demonstrates that the cascading strategy generalizes well across datasets of different scales and complexities.

One of the primary conclusions from these findings is that IDK cascades achieve computational efficiency through hierarchical organization. Lightweight models handle the majority of inputs, while deeper models are engaged only when necessary, substantially reducing average inference time without sacrificing accuracy.

Additionally, the IDK Cascade framework is scalable across different accuracy requirements. By adjusting confidence thresholds, the cascades dynamically adapt to varying classification difficulty levels, making them well suited for deployment on edge computing platforms such as the NVIDIA Jetson AGX Orin, where computational resources are limited. The observed speedups relative to high-complexity models highlight how cascades make advanced deep learning architectures more practical for real-world, latency-sensitive scenarios.

We also observed that although the final classifier in the Tiny ImageNet cascades is the Swin Transformer with a standalone accuracy of 89.56%, none of the cascades fully reaches this value. This occurs because earlier, faster classifiers with slightly lower precision handle many inputs, including some borderline cases that could otherwise benefit from deeper processing. As a result, incorporating high-precision but lightweight classifiers slightly reduces the overall cascade accuracy.

The IDK cascades do not enforce a strict deadline for each individual sample. Instead, they are designed to minimize the average inference time across the entire test set. However, in real-world scenarios like autonomous driving, inputs often come in sequences that are temporally correlated, where similar or complex scenes can appear back to back. When that happens, several images in a row may need to go through the entire cascade, causing short bursts of higher latency and reducing the efficiency gains seen under typical conditions. This points to an important consideration for real-world deployment: while cascades perform well for average workloads, system designers may

Table 3
Image distribution in IDK cascade 4 among 10000 test data (Tiny ImageNet).

Precision	Alexnet	Resnet 18	Efficientnet	Swin Transformer
92%	3801	2694	1685	1820
95%	3120	2537	1678	2665
98%	2259	2037	1510	4194

need to use buffering, or adaptive thresholding to keep performance consistent.

Overall, while IDK cascades offer a strong trade-off between computational cost and accuracy, confidence threshold selection remains a key factor in achieving optimal performance. The scalability of cascades is not limited to only a few cascade configurations; we can easily incorporate any other model into the cascade without hurting it. The ability to dynamically route inputs based on model confidence opens opportunities for intelligent, context-aware AI systems that efficiently balance performance and resource usage.

5. Conclusion

In this work, we present the IDK Cascade and evaluate its performance in real-time AI applications, where achieving both speed and accuracy is challenging, faster models often lack reliability, while more accurate models come with higher computational costs. Our IDK cascade framework bridges this gap by intelligently deciding how much computation each input actually needs. Instead of running every input through a heavy deep learning model, our approach starts with lightweight classifiers and only escalates to more complex models when necessary. This way, we reduce average inference time while still ensuring accurate predictions, making AI inference on edge devices both efficient and reliable.

Unlike a single classifier, not every input requires the same amount of computation to achieve a confident prediction. By leveraging a confidence-based decision mechanism, the IDK cascade processes most inputs quickly, keeping computational costs low while preserving accuracy for more challenging cases. The results show that our framework provides a flexible trade-off to achieve higher accuracy at a greater computational cost or opt for a moderate yet acceptable accuracy at a significantly faster inference speed. Beyond just performance gains, our framework is scalable and flexible, meaning it can easily be extended to new models or tailored for different AI tasks. Whether it is for autonomous systems, industrial automation, or real-time surveillance, the IDK cascade provides a practical and efficient solution for AI applications that demand both speed and accuracy.

6. Future work

Future work will focus on developing more efficient algorithms for synthesizing IDK cascades. We intend to create an adaptive threshold selection approach.

We will also broaden our experiments to video data, looking into frame-wise decision propagation. Additionally, we want to include more system performance metrics.

CRediT authorship contribution statement

Ishrak Jahan Ratul: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology. **Zhishan Guo:** Writing – review & editing, Validation, Supervision. **Kecheng Yang:** Writing – review & editing, Validation, Supervision, Investigation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported in part by NSF grants CNS-2104181, CNS-2442078, CMMI-2246671, and CPS-2521121.

Data availability

All data used in this study are publicly available.

References

[1] K. Kavi, R. Akl, A. Hurson, Real-time systems: An introduction and the state-of-the-art, in: Wiley Encyclopedia of Computer Science and Engineering, John Wiley & Sons, Ltd, 2009, pp. 2369–2377.
[2] N. Kim, S. Lee, S. Kim, S.-M. Park, Automated arrhythmia classification system: Proof-of-concept with lightweight model on an ultra-edge device, IEEE Access 12 (2024) 150546–150563.
[3] S.Y. Nikouei, Y. Chen, S. Song, R. Xu, B.-Y. Choi, T.R. Faughnan, Real-time human detection as an edge service enabled by a lightweight CNN, in: 2018 IEEE International Conference on Edge Computing, EDGE, 2018, pp. 125–129.
[4] Y. Wang, Y. Han, C. Wang, S. Song, Q. Tian, G. Huang, Computation-efficient deep learning for computer vision: A survey, Cybern. Intell. (2024) 1–24.
[5] S. Baruah, A. Burns, R.I. Davis, Optimal synthesis of robust IDK classifier cascades, 22 (5s) (2023).
[6] X. Wang, Y. Luo, D. Crankshaw, A. Tumanov, F. Yu, J.E. Gonzalez, Idk cascades: Fast deep learning by learning not to overthink, 2017.
[7] T. Abdelzaher, K. Agrawal, S. Baruah, A. Burns, R.I. Davis, Z. Guo, Y. Hu, Scheduling IDK classifiers with arbitrary dependences to minimize the expected time to successful classification, Real-Time Syst. 59 (3) (2023) 348–407.
[8] L.S. Karumbunathan, NVIDIA Jetson AGX Orin Series. Online at https://www.nvidia.com/content/dam/en-zz/Solutions/gtcf21/jetson-orin/nvidia-jetson-agx-orin-technical-brief.pdf.
[9] J. Wu, Q. Zhang, G. Xu, Tiny ImageNet Challenge, Stanford University CS231n Course Project Report, 2017, https://cs231n.stanford.edu/reports/2017/pdfs/930.pdf, Accessed: 2026-02-24.
[10] S. Baruah, Real-time scheduling of multistage IDK-cascades, in: 2021 IEEE 24th International Symposium on Real-Time Distributed Computing, ISORC, IEEE, 2021, pp. 79–85.
[11] S. Baruah, A. Burns, R.I. Davis, Y. Wu, Optimally ordering IDK classifiers subject to deadlines, Real-Time Syst. 59 (1) (2023) 1–34.
[12] I.J. Ratul, Z. Guo, K. Yang, Cascading IDK classifiers to accelerate object recognition while preserving accuracy, in: 2025 IEEE 49th Annual Computers, Software, and Applications Conference, COMPSAC, 2025, pp. 1522–1525.
[13] A.B. Amjoud, M. Amrouch, Object detection using deep learning, CNNs and vision transformers: A review, IEEE Access 11 (2023) 35479–35516.
[14] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, W. Shi, Edge computing for autonomous driving: Opportunities and challenges, Proc. IEEE 107 (8) (2019) 1697–1716.
[15] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, Commun. ACM 60 (6) (2017) 84–90.
[16] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, 2015.
[17] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2015.
[18] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, 2015.
[19] F.N. Iandola, S. Han, M.W. Moskewicz, K. Ashraf, W.J. Dally, K. Keutzer, SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5mb model size, 2016.
[20] M. Tan, Q.V. Le, EfficientNet: Rethinking model scaling for convolutional neural networks, 2020.
[21] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, B. Guo, Swin transformer: Hierarchical vision transformer using shifted windows, 2021.
[22] Z. Liu, H. Mao, C. Wu, C. Feichtenhofer, T. Darrell, S. Xie, A ConvNet for the 2020s, 2022, CoRR.