Cascading IDK Classifiers to Accelerate Object Recognition While Preserving Accuracy

Ishrak Jahan Ratul
Texas State University
ishrakratul@txstate.edu

Zhishan Guo
North Carolina State University
zguo32@ncsu.edu

Kecheng Yang
Texas State University
yangk@txstate.edu

***Abstract*—Real-time object recognition on edge devices with constrained computing resources involves a trade-off between computational workload and classification accuracy. Existing classifier models are individual classifiers that are typically designed for either fast inference with reduced accuracy or high accuracy with significant computational cost. A recently proposed concept, called IDK (which stands for "I don't know") classifiers, enables cascading multiple existing classifiers to achieve high accuracy while significantly reducing average inference time. In this work, we compose IDK classifier cascades for the Tiny ImageNet dataset. Each input is processed sequentially through classifiers—from faster, less accurate ones to slower, more accurate ones. When an upstream classifier returns a high-confidence prediction, downstream models are skipped, improving average inference time. Our experiments demonstrate that IDK classifier cascades can reduce average computation time per inference while maintaining high classification accuracy compared to state-of-the-art individual models.**

***Index Terms*—DNN, IDK classifier, neural networks inference, classifier cascade, PyTorch, edge device.**

I. Introduction

Recent advances in machine learning (ML), particularly deep learning, have significantly improved object detection and data processing in real-time. However, these improvements often come with increased computational costs and higher latency, making it challenging to meet the demands of real-time applications [9]. In object recognition, classification is a fundamental task typically performed by deep neural networks (DNNs). Although these models provide high accuracy, their computational complexity often leads to slower inference times, which becomes a bottleneck in applications like video processing. For example a 60 frames per second video, where each frame must be processed in under 16 milliseconds.

However, lightweight classifiers offer faster inference speeds [10], making them suitable for high-throughput or resource-constrained scenarios. But, they generally lack the ability to handle complex or ambiguous inputs, leading to reduced classification accuracy. This trade-off between speed and accuracy becomes even more critical in edge devices such as autonomous drones, robots, and embedded systems, where computing resource, memory, and power are limited [15, 20]. Rather than choosing between fast or accurate models, we explore a more adaptive approach: Can we combine them in a way that leverages their strengths while mitigating their weaknesses? This motivates the use of the IDK classifier cascade framework. In this framework, multiple classifiers, from lightweight to complex, are arranged in a sequence. Each classifier is equipped with a confidence threshold: if a prediction exceeds this threshold, it is accepted; otherwise, the model defers by outputting an IDK, passing the input to the next classifier in the cascade.

This cascade structure allows lightweight models to quickly handle easy inputs, while more complex models are reserved for difficult cases. By allowing early classifiers to handle a majority of the input load and escalating only uncertain predictions, the IDK cascade reduces the average inference time while preserving high overall accuracy. This makes it especially valuable for real-time tasks such as autonomous driving and video surveillance [5], where both low latency and quality of decision are essential.

Contribution. In this work, we present a practical IDK cascade framework to optimize average inference time and classification accuracy. We adapt pre-trained DNNs to build IDK cascades. These cascades are built by sequencing models from fastest to most accurate, passing inputs until a confident prediction is made. We introduce a confidence based thresholding for decision making in each classifier. We evaluate our cascades on the NVIDIA Jetson AGX Orin platform [8]. Experimental results demonstrate that our cascades significantly reduce average inference time while maintaining high accuracy.

II. IDK Cascade

A. Background and Related Work

The IDK cascade framework builds on the foundational work of Baruah et al. [3, 4, 5], which emphasized the need for efficient and accurate classification in real-time, safety-critical systems. Their studies showed that using complex models for all inputs is inefficient, while lightweight models, though fast, often lack sufficient accuracy [19]. To address this, the concept of IDK classifiers was introduced [5] and later extended [1]. These classifiers output an IDK decision when confidence is low, enabling a cascade structure that processes inputs through increasingly complex models until a confident prediction is made. This approach balances speed and accuracy by allocating computation based on input complexity.

An IDK cascade is thus a sequential classification pipeline where each model either accepts or defers an input. Here the lightweight models handle most inputs efficiently, while deeper models are used only when necessary. Unlike prior work that focused on theory and scheduling, our study implements and evaluates practical IDK cascades, demonstrating

This work is supported in part by NSF grants CNS-2104181 and CMMI-2246672.

significant reductions in average inference time with minimal impact on accuracy which can be suitable for real-time, resource-constrained environments.

B. Decision Rule

Each classifier in the IDK cascade operates based on a confidence threshold. Let C_i be the i-th classifier and x the input. The classifier's confidence score, $P(C_i(x))$, represents the likelihood that its prediction is correct. If this score exceeds a threshold τ_i, the classifier outputs a predicted class y_i; otherwise, it returns "IDK" and forwards the input to the next classifier. This rule allows lightweight classifiers to handle easy inputs, escalating harder ones to more accurate models [3].

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

C. Our Considerations

Our IDK cascade uses high-performing CNN and transformer-based models for object recognition, selected based on benchmark accuracy [2]. To ensure reliability, we performed a systematic threshold search to assign each model an optimal confidence level, minimizing misclassifications by escalating only uncertain inputs. Inputs are processed sequentially (batch size = 1), reflecting real-time constraints on edge devices like autonomous vehicles [13]. These systems must balance multiple concurrent tasks (e.g., LiDAR, radar, and vision), so avoiding batch or parallel execution helps preserve compute resources for critical operations.

III. Methodology

A. Setup

Our IDK cascade framework targets real-time classification on edge platforms. We deployed it on the NVIDIA Jetson AGX Orin developer kit, equipped with a 2048-core Ampere GPU, 64 Tensor cores, and dual NVDLA accelerators [8].

Experiments were conducted using PyTorch. We evaluated our models on the Tiny ImageNet dataset [12], which includes 200 object classes, each with 500 training and 50 validation images. We used the original validation set (10,000 images) as our test set and split the training data into 90% training and 10% validation. This setup ensured evaluation on unseen data, preventing data leakage. Our main objective was to build a low-average inference time, high-accuracy classification system suitable for edge deployment.

B. Fine-Tuning and Classifier Selection

To construct an efficient IDK cascade, we fine-tuned several state-of-the-art object recognition models like AlexNet [11], ResNet18 [6], VGG16 [16], Inception V3 [17], SqueezeNet [7], EfficientNet [18], and Swin Transformer [14] on Tiny ImageNet dataset. All models were pretrained on ImageNet and adapted by modifying their final layers for 200-class output. Training was conducted using PyTorch's Distributed Data Parallel (DDP) on a dual NVIDIA RTX A4000 GPU server, with cross-entropy loss and SGD optimizer (learning rate = 0.001, momentum = 0.9). Early stopping was applied to prevent overfitting, and generalization was enhanced using learning rate scheduling, data augmentation, and dropout.

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 I: Performance Metrics of Classifiers after Fine Tuning

Classifier selection for building IDK cascade was based on a balance between accuracy and average inference time. AlexNet (59.19%, 1.798 ms) was chosen for its low latency, ideal for early-stage classification. VGG16 (66.43%, 3.800 ms) and ResNet18 (72.41%, 5.467 ms) offered moderate to high accuracy with reasonable speed, making them suitable for intermediate stages. EfficientNet (79.45%, 17.830 ms) and Swin Transformer (89.56%, 45.428 ms) were placed later in the cascade due to their high accuracy and increasing computational cost. Models like Inception V3 (45.90%, 23.700 ms) and SqueezeNet (40.88%, 5.055 ms) were excluded due to poor accuracy-to-latency trade-offs.

C. Confidence Thresholding and Cascade Synthesis

To build the IDK cascade, we determined confidence thresholds τ for each classifier using validation data. Predictions above τ were accepted; otherwise, produces IDK output. This thresholding minimizes misclassifications while reducing computational cost by deferring only uncertain cases.

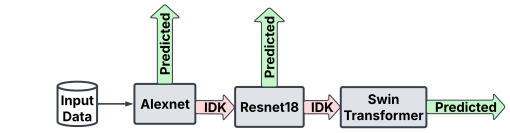
The accuracy for a given threshold τ was computed as:

$$A(\tau) = \frac{\sum_{x \in \text{correct}} I(C(x) > \tau)}{\sum_{x \in \text{correct}} I(C(x) > \tau) + \sum_{x \in \text{incorrect}} I(C(x) > \tau)} \quad (2)$$

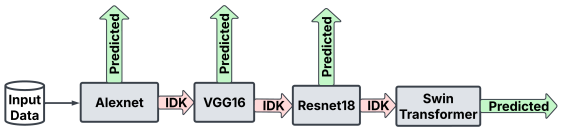
where $I(C(x) > \tau)$ is an indicator function returning 1 when the classifier's confidence for input x exceeds τ, and 0 otherwise. The algorithm sorts all validation samples by confidence in decreasing order and computes the accuracy $A(\tau)$ over samples with confidence above a given threshold τ. This accuracy is referred to as the classifier's *precision*, consistent with prior work [1]. Samples with confidence below τ are classified as IDK. Therefore, precision represents the accuracy over the subset of predictions the classifier chooses to accept. For each desired precision level, an appropriate confidence threshold can be selected accordingly. Classifiers were arranged by increasing average inference time from AlexNet to Swin Transformer so that faster models handled simpler inputs. We evaluated four different cascades shown in (Fig. 1 (a)–(d)).

IV. Results and Evaluation

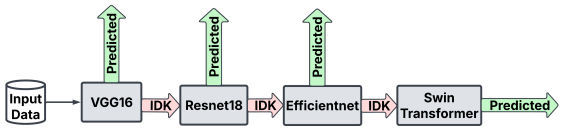
Individual Classifier Performance. Table I shows a clear trade-off between average inference time and accuracy. As model complexity increases, so does accuracy with higher



(a) IDK Cascade 1



(b) IDK Cascade 2



(c) IDK Cascade 3

(d) IDK Cascade 4

Fig. 1: IDK Cascade Structures

inference time. AlexNet is fastest (1.798 ms) but least accurate (59.57%), while Swin Transformer offers the best accuracy (89.56%) at a much higher cost (45.428 ms). VGG16 (66.43%) and ResNet18 (71.98%) provide moderate trade-offs, with ResNet18 showing the best balance for real-time tasks. EfficientNet improves accuracy to 79.22% but requires 17.830 ms. These results confirm that lightweight models are fast but less reliable, while complex models are accurate but expensive for time-sensitive applications.

Performance of IDK Cascades. Fig. 2 and Fig. 3 demonstrate the effectiveness of the IDK cascade framework in balancing accuracy and inference time. Cascade 1 (at 91% precision) relies mostly on lightweight models and achieves 83.14% accuracy at 17.612 ms. As more complex models are introduced, performance improves incrementally: Cascade 2 (at precision 93%) reaches 85.09% at 20.686 ms, Cascade 3 (at precision 96%) achieves 87.96% at 26.785 ms, and Cascade 4 (at 98% precision) reaches near Swin Transformer's accuracy with 89.35% while reducing average inference time by 27% (33.126 ms vs. 45.428 ms).

Notably, heavier models such as EfficientNet and Swin Transformer shows greater average inference time due to their complex and input-sensitive computations [13]. In contrast, lightweight models like AlexNet and ResNet18 maintain low average inference time. The IDK cascades inherit this low average inference time by processing most inputs early, escalating only when necessary. Cascade 3, offers a significant balance between speed and accuracy, while Cascade 4 demonstrates that near state-of-the-art accuracy can be achieved without running expensive models on every input. These findings validate that input-dependent computation via cascading is not only efficient but also practical for edge scenarios with lower average inference time demand.

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

TABLE II: Image Distribution in IDK Cascade 4 among 10000 Test Data

Impact of Different Thresholds for a Cascade. Threshold tuning is critical. Lower thresholds yield faster inference but reduce accuracy. Higher thresholds increase accuracy but escalate more inputs to complex classifiers. Table II shows how image distribution shifts with precision. Cascade 3 (95% precision) achieves 86.85% accuracy at just 8.681 which can benefit low-latency demands. Cascade 4 (98%) offers 89.35% accuracy at 23.812 ms, nearly 50% faster than Swin Transformer. Dynamic threshold tuning could further improve adaptability in changing conditions. While individual classifier precision remains high (92–95%), overall cascade accuracy is slightly lower due to harder inputs reaching deeper stages of in the cascade executing all upstream classifiers [1].

Comparison with State-of-the-Art. Compared to Swin Transformer, IDK cascades are significantly faster with minimal accuracy loss. Cascade 1 is 2.57× faster (83.14% acuracy), Cascade 2 is 2.2× faster (85.09% accuracy), Cascade 3 is 1.7× faster with 87.96% accuracy, and Cascade 4 nearly matches Swin's 89.56% accuracy with a 27% reduction in average inference time. The hierarchical structure ensures low average inference time with acceptable accuracy loss which is critical for real-time systems. Moreover, cascades are scalable, model-agnostic, and easy to deploy on platforms like NVIDIA Jetson AGX Orin. While they trade a small drop in accuracy for large latency gains, this can be a practical trade-off in edge AI.

V. Conclusion

In this work, we propose the IDK Cascade, a framework that balances speed and accuracy for real-time AI applications. Fast models are efficient but less reliable, while accurate models are computationally expensive. Our cascade dynamically adjusts computation based on input difficulty: each input is first processed by a lightweight model and escalated to deeper models only if prediction confidence is low. This reduces average inference time while maintaining high accuracy, making it ideal for edge devices with limited resources. Unlike single-model systems, the IDK Cascade uses confidence-based decisions to process simple inputs quickly and reserve complex models for harder cases. It offers a scalable, adaptable solution suitable for autonomous systems, real-time surveillance, and industrial automation.

References

[1] Tarek Abdelzaher, Kunal Agrawal, Sanjoy Baruah, Alan Burns, Robert I Davis, Zhishan Guo, and Yigong Hu. Scheduling idk classifiers with arbitrary dependences to minimize the expected time to successful classification. *Real-Time Systems*, 59(3):348–407, 2023.

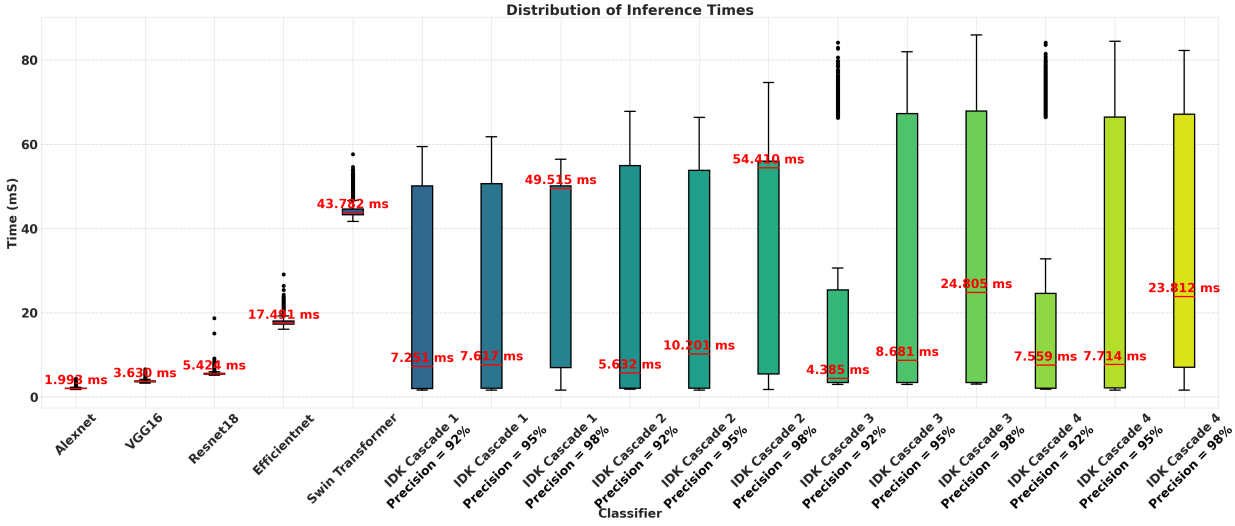


Fig. 2: Distribution of Inference Times varying the Precision

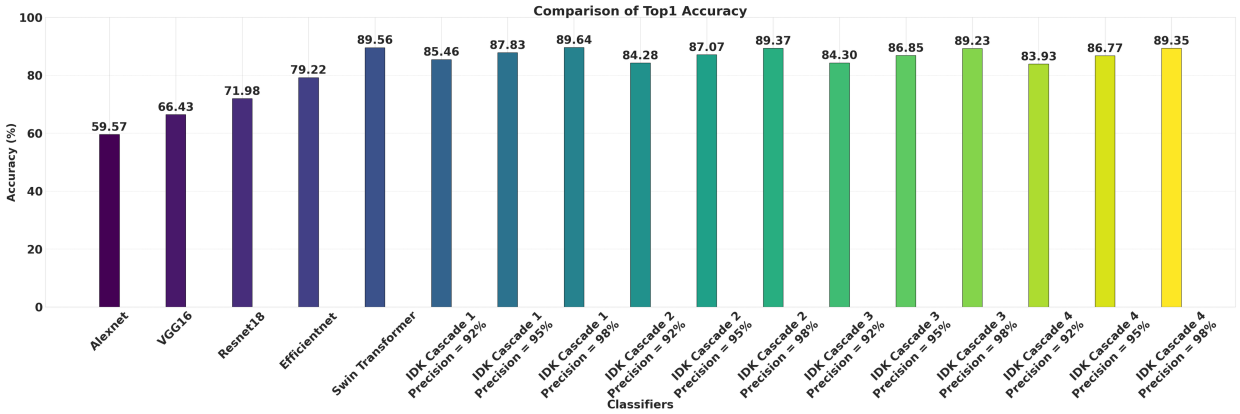


Fig. 3: Comparison of Test Accuracy

[2] Ayoub Benali Amjoud and Mustapha Amrouch. Object detection using deep learning, cnns and vision transformers: A review. *IEEE Access*, 11:35479–35516, 2023.

[3] Sanjoy Baruah. Real-time scheduling of multistage idk-cascades. In *2021 IEEE 24th International Symposium on Real-Time Distributed Computing (ISORC)*, pages 79–85. IEEE, 2021.

[4] Sanjoy Baruah, Alan Burns, Robert I Davis, and Yue Wu. Optimally ordering idk classifiers subject to deadlines. *Real-Time Systems*, 59(1):1–34, 2023.

[5] Sanjoy Baruah, Alan Burns, and Robert Ian Davis. Optimal synthesis of robust idk classifier cascades. 22(5s), September 2023.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[7] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5mb model size, 2016.

[8] Leela 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] Krishna Kavi, Robert Akl, and Ali Hurson. *Real-Time Systems: An Introduction and the State-of-the-Art*, pages 2369–2377. John Wiley & Sons, Ltd, 2009.

[10] Namho Kim, Seongjae Lee, Seungmin Kim, and Sung-Min Park. Automated arrhythmia classification system: Proof-of-concept with lightweight model on an ultra-edge device. *IEEE Access*, 12:150546–150563, 2024.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017.

[12] Yann Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

[13] Shaoshan Liu, Liangkai Liu, Jie Tang, Bo Yu, Yifan Wang, and Weisong Shi. Edge computing for autonomous driving: Opportunities and challenges. *Proceedings of the IEEE*, 107(8):1697–1716, 2019.

[14] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021.

[15] Seyed Yahya Nikouei, Yu Chen, Sejun Song, Ronghua Xu, Baek-Young Choi, and Timothy R. Faughnan. Real-time human detection as an edge service enabled by a lightweight cnn. In *2018 IEEE International Conference on Edge Computing (EDGE)*, pages 125–129, 2018.

[16] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[17] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.

[18] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

[19] Xin Wang, Yujia Luo, Daniel Crankshaw, Alexey Tumanov, Fisher Yu, and Joseph E Gonzalez. Idk cascades: Fast deep learning by learning not to overthink. *arXiv preprint arXiv:1706.00885*, 2017.

[20] Yulin Wang, Yizeng Han, Chaofei Wang, Shiji Song, Qi Tian, and Gao Huang. Computation-efficient deep learning for computer vision: A survey. *Cybernetics and Intelligence*, pages 1–24, 2024.