Work-in-Progress: Utilizing Probabilistic Analysis to Fine-Tune Optimal IDK Cascades

Anh-Vu Nguyen[1], Albert M. K. Cheng[2], Thomas Carroll[2]

[1]Cy-Fair High School, Cypress, TX 77429, USA
[2]Department of Computer Science, University of Houston, Houston, TX 77004, USA

Abstract— **In an effort to reduce the runtime of classification algorithms, IDK (I Don't Know) cascades have been presented as an alternative to current classification models. These structures comprise of a "cascade" of classifiers that only categorizes an input if a classifier outputs a confidence level that exceeds a predetermined threshold. If it does not, it outputs the class "I don't know" and moves on to the next classifier in the cascade. However, these IDK cascades often reach their worst-case execution time, where classification is only completed by the final classifier. This paper aims to improve the static structure employed by these cascades, deploying a dynamic IDK framework that skips certain classifiers upon meeting specific conditions. A probabilistic analysis run on a previously validated optimal IDK cascade helped us identify these specific conditions, using the outputted confidence level of the first classifier to dictate if a cascade should run its middle classifiers. Ultimately, we generated a dynamic IDK cascade that ran up to 17% faster than its static counterpart.**

I. INTRODUCTION

Cyber-Physical Systems have seen a rise in Deep Learning model integration. More specifically, they have encountered an increase in the use of classification frameworks, with systems, such as autonomous vehicles, calling for accurate and timely models that meet high-performance expectations. Countless studies have been conducted to improve the accuracy of existing neural network-based classifiers, reaching a point where negligible improvements are traded for with significant increases in classification runtime [1]. Given the importance of safety, efficiency, and the advancement of artificial intelligence in real-time systems, the demand for fast and efficient classifiers has never been higher.

IDK Cascades present a solution to reduce the overall execution time of neural network-based classifiers. As introduced by [2], IDK Cascades are characterized by a set of classifiers that work in succession to yield results promptly; typically, the fastest and least accurate networks are first, with the more powerful and time-intensive models following. Upon encountering an input, if the first classifier does not return a class with a confidence level that exceeds a predefined threshold, it outputs "I Don't Know", and passes the input to the next classifier with its own confidence threshold. This process is repeated until a classification is made or a deterministic classifier, which outputs a classification regardless of its confidence level, is reached.

Synthesizing IDK Cascades has frequently been a topic of study, with algorithms commonly used to arrange classifiers to minimize both average and worst-case execution times. [1] presents a series of arbitrary dependencies acting as guidelines for the construction of different cascades. Despite the existence of many works similar to [1] dedicated to ordering classifiers [3] [4], they seldom break the linear structure of previously synthesized sequences, creating a circumstance where worst-case scenario is frequent.

In this paper, we fine-tune an optimal IDK cascade synthesized by [1] while preserving its capability for accurate classification. This experiment utilizes a probabilistic analysis to identify specific conditions in which abandoning the optimal cascade structure cuts down on average execution time and mitigates the worst-case runtime. Once we recognize these specific conditions, we use them to produce a Dynamic IDK Cascade that executes different orders of classifiers depending on whether certain circumstances are met. In effect, we obtain a shorter runtime interval without jeopardizing the classification accuracy of the IDK Cascade.

The remainder of the paper is organized as follows:

- Synthesizing an optimal IDK Cascade [1].
- A probabilistic analysis of an optimal IDK cascade against one with a structure similar to that of a dynamic cascade.
- An implementation of a dynamic cascade that follows the results found within the probabilistic analysis.
- A validation of the dynamic cascade using a subset of the ImageNet Large Scale Visual Recognition Challenge data set [7]

II. PROBABILISTIC ANALYSIS

The linear structure of an IDK cascade is simple: when a classifier does not meet a predetermined confidence threshold, it outputs the class "I don't know", and the input moves to the next classifier. However, reaching the end of the cascade suggests that the middle classifiers spent time calculating the "I don't know" class, and since no class was output, the runtime of these intermediate neural networks can be considered misspent. By preventing these middle classifiers from running under certain conditions, the overall execution time of an IDK cascade will be reduced.

A. Constructing An IDK Cascade

The strategies for generating a viable IDK Cascade have frequently been explored, with many optimal cascades existing in conjunction with varying dependencies. An IDK Cascade may be developed under two primary differences: whether the cascade is independent or dependent [6]. The former characterizes a cascade compiled with classifiers that

979-8-3315-4026-5/24/$31.00 ©2024 IEEE
DOI 10.1109/RTSS62706.2024.00046

receive inputs independent of each other. For example, the first classifier may analyze visual parameters, while the second may use audio inputs. Dependent IDK cascades denote a set of classifiers that all receive the same input, e.g., they all receive the same image file.

For simplicity, we used an IDK Cascade synthesized by Baruah et al. [1]. This cascade was ordered with the presumption that it would meet a specific classification threshold; in other words, it must be able to classify an arbitrarily chosen proportion of inputs successfully. After profiling a set of 4 ResNet classifiers [5], Baruah et al. developed an optimal dependent IDK cascade that meets a classification threshold of 0.65: ResNet-18, ResNet-34, ResNet-152 [1]. This cascade will be referred to as ABC, with A, B, and C, corresponding with each ResNet classifier respectively.

Baruah's ResNet cascade follows three specifications:

- The cascade can be considered dependent; ResNet models require a 224x224 image for classification, meaning all three classifiers must input the same sample.
- The numbers following each respective ResNet classifier indicate the number of layers the networks have, suggesting that the higher the number is, the more powerful and resource-intensive its execution.
- Each classifier uses its own default pre-trained weights for evaluation, ensuring as little variation between the ResNet networks as possible.

B. Setting Up a Probabilistic Comparison

To synthesize an IDK Cascade that works dynamically, skipping the middle classifiers when it deems beneficial, we must first identify the conditions in which the middle classifiers are more likely to output the class "I don't know". Upon recognizing these conditions, we will execute classifier B in our cascade if they are not met and move straight to classifier C when they are. Using probabilistic analysis, we better understand the likelihood of an "I don't know" class from classifier B.

Note that the only factors that may affect the "I don't know" rate of classifier B are the classes and confidence levels outputted by classifier A. The calculations by classifier C do not affect the outcome of B; therefore, we did not record its output in our analysis. It is imperative to realize that when the final classifier is reached, e.g., C in our case, both the first and last classifiers must be called to make a classification. In effect, the only way to save time in this manner is through omitting the execution of the middle classifier(s).

In our probabilistic analysis, we used two different models: the optimal ABC cascade found by Baruah et al. [1] and a less efficient cascade: AC. The latter was used to simulate the optimal cascade with the middle classifier B taken out, consisting of the same ResNet-18 and 152 models with their default pre-trained weights as described previously [5]. The analysis called for two measurements when classifier A outputted the class "I don't know": the confidence level of classifier A and the time required for each cascade to make a classification. The cases where Classifier A made a successful classification without passing it on to B were not documented, as this outcome has the quickest runtime and both cascades would output identical classes.

C. Obtaining Results

Using the 10,000-image "ImageNetV2-TopImages" [8] subset of the ImageNet Large Scale Visual Recognition Challenge dataset [7] by Schmidt et al., we executed both cascades. Both cascades produced the same classes with identical confidence levels for each input, except for a few misclassifications by classifier B in cascade ABC, which were classified differently by classifier C in AC.

As seen in Table I, the results revealed an association between lower confidence levels output by classifier A and a classification speed-up when "skipping classifier B". When A had a confidence level between 0.000 and 0.300, we received a 5-6 ms decrease in execution time per input. The table also shows that as the confidence level of A increases, we get diminishing returns saving time, even receiving an increase when reaching the 0.700-0.890 confidence range.

TABLE I
PROBABILISTIC COMPARISON

Confidence Range of Classifier A	ABC Avg Execution Time (ms)	AC Avg Execution Time (ms)	Time Saved (ms)
0.000-0.099	36.34	30.71	-5.63
0.100-0.199	36.40	30.02	-6.38
0.200-0.299	35.18	30.12	-5.06
0.300-0.399	33.73	30.34	-3.39
0.400-0.499	33.55	30.13	-3.41
0.500-0.599	32.83	29.90	-2.93
0.600-0.699	31.23	30.01	-1.22
0.700-0.799	29.38	30.10	+0.72
0.800-0.890	24.90	30.01	+5.11

Further analysis of the behavior of classifier B with respect to the confidence level of classifier A would assist in explaining the decrease in time for the AC ResNet cascade. A second investigation examined the probability of B making a successful classification when classifier A did not meet its confidence requirement. We use this to understand the likelihood of receiving an "I don't know" output for a specific confidence level outputted by A. We ran the same TopImages [8] dataset from ImageNet [7] through the cascade, keeping the pre-trained weights identical to the ones in the previous comparison. The results of this second investigation are shown in Figure 1.

The results indicate that classifier B would determine a higher proportion of inputs to be "I don't know" when classifier A had a confidence level of 0.0-0.3, reflecting the same diminishing results seen in Table I as the confidence range increases. The decreasing proportion of "I don't know" classes showcased in Figure 1 provides a reason for the behavior found in the probabilistic comparison: lower confidences in A correspond with a higher probability that B would output "I don't know". Since we would only save time skipping classifier B when it outputs an "I don't know" class,

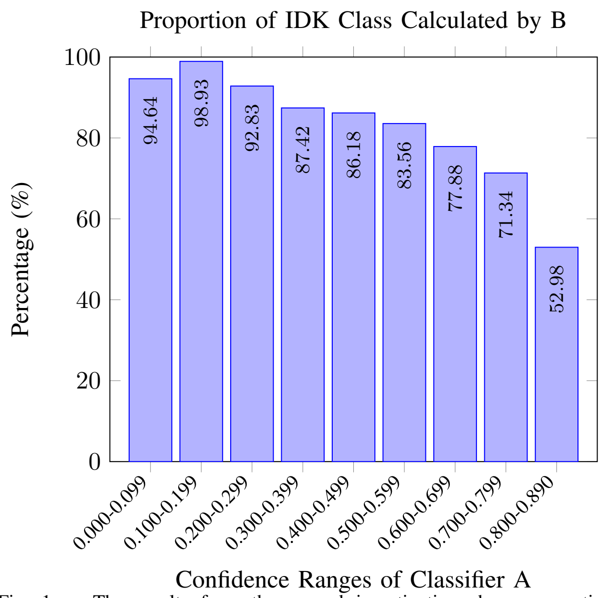

Fig. 1. The results from the second investigation show a negative correlation between higher classifier A confidence levels and lower rates of "I don't know" by classifier B.

this finding explains why the AC cascade's highest decrease in execution time resulted from when the confidence range of A was at its lower points.

III. Dynamic Cascades

The results found in the previous section suggest that the confidence level of classifier A can be used as a rough predictor of the output of classifier B. We take advantage of this fact by synthesizing a dynamic IDK cascade that decides to skip the middle classifier(s) when they are more likely to classify an input in the "I don't know" class (i.e. the first classifier labeling the input with a low confidence level).

A. Synthesis

The structure of these cascades is very similar to the ones discussed in section II.A. Both cascades start with the first classifier receiving the input and assigning a class if and only if its confidence threshold is met. However, as seen in Fig. 2, the dynamic cascade implements a minimum confidence threshold that, if not met, sends the input to a classifier further down the cascade, instead of the next one in line. This second confidence threshold is determined by the middle classifiers' probability of assigning a real class on an input with respect to the first classifier's outputted confidence level. The rest of the dynamic cascade would run similarly to the first, leaving the last classifier to output a class regardless of its confidence level.

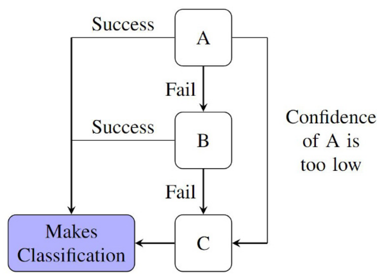

Fig. 2. Structure of Dynamic Cascade

This dynamic jumping strategy is only effective if the modified IDK cascade is dependent. Recall that a dependent classifier runs the same input as the classifier before it, and it is only able to make a more accurate classification with the trade-off of an increased execution time. This distinction is important for two primary reasons.

The first is the correlation between the first classifier's confidence level and the likelihood of the middle classifier outputting an "I don't know" class. If the two classifiers were independent, such as the first reading an image input and the second an audio sample, the confidence level of the first classifier could not be representative of what the second classifier would decide. The second reason concerns the accuracy level of the entire cascade. Dependent classifiers are typically ordered from least to most powerful, with the subsequent classifiers taking longer than preceding ones. Skipping the middle classifier would not hinder the accuracy of the dynamic cascade since the following classifier would have a higher accuracy rate.

With these requirements in mind, dynamic cascades will yield lower average execution times with negligible effect on the overhead of the system, since they minimize the time wasted by the middle classifier to reach an "I don't know" conclusion.

We created a dynamic IDK cascade using the optimal ABC cascade developed by Baruah et al. as a base [1]. To determine the threshold at which classifier A must output to skip classifier B, we performed a profiling phase highlighted in Table II, testing similar ranges as discussed in Section II. The profiling phase was run again with the TopImages [8] subset of ImageNet [7], with the base case mimicking the static structure of the original ABC cascade.

TABLE II
Profiling the Dynamic Cascade

Confidence Threshold for Skipping B Classifier	Average Execution Time per Input (ms)	Overall Classification Accuracy
Base Case	19.82	80.24%
<0.1	19.22	80.24%
<0.2	17.88	80.25%
<0.3	16.45	80.28%
<0.4	17.08	80.28%
<0.5	16.86	80.27%
<0.6	17.99	80.25%
<0.7	17.71	80.25%
<0.8	17.88	80.28%

The results in Table II indicate that the optimal confidence threshold for bypassing Classifier B occurs when Classifier A assigns a class with a confidence level of less than 0.300, with an average execution time of 16.45ms per input. A threshold with this range resulted in a 17% decrease in execution time per input when compared to the static ABC cascade. Moreover, the overall accuracy of the cascade remained relatively unchanged throughout the analysis, fluctuating by only a hundredth of a percent per run.

B. Validation

To validate the efficacy of our dynamic cascade, we tested two more 10,000-image datasets developed by Schmidt et al. [8]. Table III compares the static ABC cascade to the dynamic ABC cascade that skips classifier B when classifier A outputs a class at a confidence level between 0.000 and 0.300. The cascades ran on a NVIDIA GeForce GTX 1070 Ti for classification. As seen in the table, Threshold 0.7 saw an 11.50% decrease in average execution time for each input while the Matched-Frequency data set saw 13.76%.

TABLE III
STATIC VS DYNAMIC CASCADE

Dataset	Static		Dynamic		Percent Speed Up
	Avg Duration	Avg Accuracy	Avg Duration	Avg Accuracy	
TopImages	19.82 ms	80.24%	16.45 ms	80.28%	17.00%
Threshold 0.7	21.54 ms	75.12%	19.70 ms	75.12%	11.50%
Matched Frequency	22.74 ms	66.84%	19.61 ms	66.86%	13.76%

When comparing the static cascade to the dynamic one, there was little to no change in classification accuracy, with both cascades meeting the accuracy threshold outlined in [1]. The slight increase in accuracy for the TopImages and Matched Frequency datasets can be attributed to the structure of the dynamic cascade, deploying a more accurate classifier for the inputs that classifier B may have misidentified. In practice, the accuracy of a dynamic cascade should never fall below that of its static counterpart since the classifier executed in place of the skipped one would either make the same classification or an improved prediction.

Recall that the profiling phase for our dynamic cascade is executed considering only three classifiers. In the case of four or more classifiers, a similar profiling process can be carried out: analyzing the success probability of all succeeding classifiers against the confidence output of the first. Multiple thresholds would be present on the first classifier, dictating which classifier to jump to in the cascade. The IDK structure minimizes the inherent resource-extensive risk of this strategy, as an optimal IDK cascade typically contains less than five or six classifiers. A cascade that comprises any more would generally perform slower than its final classifier.

IV. RELATED WORKS

Sanjoy Baruah adopted a similar dynamic jumping technique in [9] across multiple IDK cascades. The decision to skip to a deterministic classifier was controlled by an execution deadline, that is, if a cascade would cause an execution to exceed a maximum duration that cascade would not be run. To the best of our knowledge, no method utilizes the probabilities of failing classifiers to determine a dynamic jump, as many recent works have been focusing on synthesizing only optimal cascades themselves.

V. CONCLUSION

This proof of concept study illustrates how skipping over the middle classifiers of a dependent IDK cascade can cut down on its average execution time without affecting the accuracy of the model. Probabilistic analysis shows an association between low first-classifier confidence levels and high second-classifier "I don't know" rates, identifying conditions in which skipping the middle-classifier yields an average runtime that is 10-17% quicker than the original cascade.

There are limitations to this study that may keep dynamic IDK cascades from reaching a maximum duration reduction. For example, for any set of potential classifiers, to verify that the confidence of A correlates to the chance of success in the following classifiers, an extensive profiling phase is required to build the threshold for classifier skipping since the assumptions are based on probabilities. In future works, the relationship between classifiers, i.e. the assumed rate of failure of one classifier based on another, should be quantified to formalize this assumption of dependency between potential classifiers. A second limitation is a lack of variety in our input data. The correlation between the first classifier's confidence level and the second classifier's output may not be as strong when considering audio inputs.

Along with mitigating the limitations presented, future works should expand on the conditions that call for dynamic jumping; implementing these separate conditions within a dynamic IDK cascade would bring down its execution duration. Presenting the dynamic model with a level of fault tolerance would also serve to increase performance expectations while maintaining its decrease in average runtime compared to static models. [3] and [4], already introduce the factors of robustness and fault tolerance, respectively, with formulated algorithms that produce more involved static cascades.

REFERENCES

[1] Abdelzaher, T., Agrawal, K., Baruah, S., Burns, A., Davis, R. I., Guo, Z., & Hu, Y. (2023). Scheduling IDK classifiers with arbitrary dependences to minimize the expected time to successful classification. *Real-Time Systems*, 59(3), 348–407. https://doi.org/10.1007/s11241-023-09395-0

[2] Wang, X., Luo, Y., Crankshaw, D., Tumanov, A., Yu, F., & Gonzalez, J. E. (2017, June 3). IDK Cascades: Fast Deep Learning by Learning not to Overthink. arXiv.org. https://arxiv.org/abs/1706.00885

[3] Baruah, S., Burns, A., & Davis, R. I. (2023). Optimal synthesis of robust IDK classifier cascades. ACM Transactions on Embedded Computing Systems, 22(5s), 1–26. https://doi.org/10.1145/3609129

[4] Baruah, S., Bate, I., Burns, A., & Davis, R. I., "Optimal Synthesis of Fault-Tolerant IDK Cascades for Real-Time Classification," 2024 IEEE 30th Real-Time and Embedded Technology and Applications Symposium (RTAS), Hong Kong, Hong Kong, 2024, pp. 29-41, doi: 10.1109/RTAS61025.2024.00011.

[5] He, K., Zhang, X., Ren, S., & Sun, J. (2015, December 10). Deep residual learning for image recognition. arXiv.org. https://arxiv.org/abs/1512.03385

[6] Baruah, S., Burns, A., and Wu, Y., (2021) Optimal Synthesis of IDK-Cascades. Proceedings of the 29th International Conference on Real-Time Networks and Systems (RTNS '21). Association for Computing Machinery, New York, NY, USA, 184–191. https://doi.org/10.1145/3453417.3453425

[7] Deng, J., Dong, W., Socher, R., Li, L. -J., Li, K., and Li, F. -F., "ImageNet: A large-scale hierarchical image database," 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 2009, pp. 248-255, doi: 10.1109/CVPR.2009.5206848.

[8] Recht, B., Roelofs, R., Schmidt, L., & Shankar, V., (2019). Do ImageNet classifiers generalize to ImageNet? arXiv (Cornell University). https://doi.org/10.48550/arxiv.1902.10811

[9] Baruah, S., "Real-Time Scheduling of Multistage IDK-Cascades," 2021 IEEE 24th International Symposium on Real-Time Distributed Computing (ISORC), Daegu, Korea (South), 2021, pp. 79-85, doi: 10.1109/ISORC52013.2021.00021.