11-2022

Tree-based Unidirectional Neural Networks for Low-Power Computer Vision

Abhinav Goel

Caleb Tung

Nick Eliopoulos

Amy Wang

Jamie C. Davis

See next page for additional authors

Authors

Abhinav Goel, Caleb Tung, Nick Eliopoulos, Amy Wang, Jamie C. Davis, George K. Thiruvathukal, and Yung-Hisang Lu

This article has been accepted for publication in IEEE Design & Test. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2022.3217016

Tree-based Unidirectional Neural Networks for Low-Power Computer Vision

Abhinav Goel, Caleb Tung, Nick Eliopoulos, Amy Wang*, James C. Davis,
George K. Thiruvathukal†, and Yung-Hsiang Lu
Purdue University, School of Electrical and Computer Engineering, West Lafayette, IN, USA
*West Lafayette Junior-Senior High School, West Lafayette, IN, USA
†Loyola University Chicago, Department of Computer Science, Chicago, IL, USA

***Abstract*—This article describes the novel Tree-based Unidirectional Neural Network (TRUNK) architecture. This architecture improves computer vision efficiency by using a hierarchy of multiple shallow Convolutional Neural Networks (CNNs), instead of a single very deep CNN. We demonstrate this architecture's versatility in performing different computer vision tasks efficiently on embedded devices. Across various computer vision tasks, the TRUNK architecture consumes 65% less energy and requires 50% less memory than representative low-power CNN architectures, e.g., MobileNet v2, when deployed on the NVIDIA Jetson Nano.**

I. Introduction

Convolutional Neural Networks (CNNs) have led to significant breakthroughs in many computer vision tasks [1]. The high accuracy of CNNs on computer vision is mainly attributed to their ability to train billions of parameters for learning complex functions [2]. The trend in computer vision research is to improve accuracy by using more resources — making CNNs deeper, wider, and more strongly connected [2]. Thus, the accuracy gains come with high energy consumption, memory, and computation overheads.

The state-of-the-art CNNs require several seconds to run on most embedded devices, e.g., Raspberry Pi [3]. To use such CNNs to process data captured by cameras on embedded devices, the computation is often offloaded to the cloud. However, many applications cannot be offloaded, e.g., computer vision deployed on drones in areas without high-speed networks. Privacy concerns also limit the applicability of cloud-based solutions [3].

Most existing CNNs like ResNet [4] use large *monolithic architectures* as seen in Fig. 1(a). Such architectures contain a single CNN to identify every feature associated with all categories to make decisions. To understand the shortcomings of monolithic architectures, consider the image classification problem: assign a single label from a set of categories to every input image. These CNNs require a large number of layers to extract the features associated with every category. However, when classifying a single image, only a small fraction of the CNN activations have non-zero values [5]. Since all the CNN operations have to be performed, there are many redundant operations. These redundancies decrease the efficiency of CNNs considerably.

Our work develops Tree-based Unidirectional Neural Network (TRUNK), a new CNN architecture that improves efficiency [6, 7, 8]. Instead of a single very deep CNN, multiple shallow CNNs in the form of a tree work together to perform computer vision tasks. TRUNK finds the *similarity* between different categories. Similar categories are grouped into *clusters*. Similar clusters are then grouped to form a tree. The shallow CNNs at every node of TRUNK classify between different clusters. Fig. 1(b) illustrates the TRUNK architecture, where the categories are *cat*, *dog*, etc. During inference, each image is first processed by the root CNN. Once a cluster is selected by the root, another CNN further classifies among the children of the chosen cluster. This process continues until one of the leaves of the tree are reached. The CNNs associated with other clusters are not used during the inference of that image, thus avoiding redundant arithmetic and memory operations.

Hierarchical computer vision techniques can be categorized as (a) ensemble or (b) divide-and-conquer. Existing ensemble techniques combine the output of multiple large CNNs to increase accuracy at the expense of efficiency [9]. Existing divide-and-conquer techniques improve efficiency, but result in significant accuracy losses [10]. This article presents methods to combine visual similarities with Neural Architecture Search to construct divide-and-conquer hierarchies that achieve high efficiency and high accuracy.

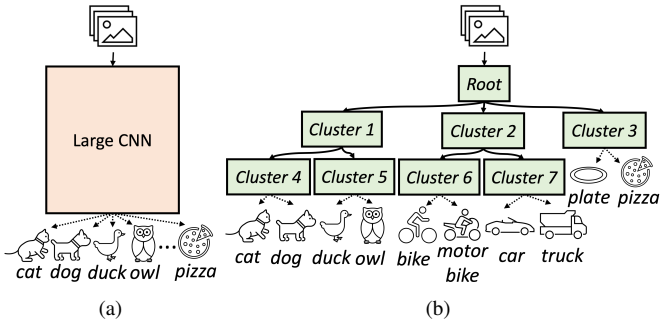


Fig. 1: (a) Existing CNN Architectures: A single monolithic architecture classifies images into their categories. (b) Proposed TRUNK: The input image is processed incrementally using small CNNs. After detecting the type of images, finer classifications are made.

II. Tree-based Unidirectional Neural Networks

This section describes the proposed Tree-based Unidirectional Neural Network (TRUNK) architecture. We use image classification to explain the architecture in more detail.

As seen in Fig. 1(b), the proposed architecture contains multiple small CNNs in the form of a tree. Note that (1) each input follows a single root-leaf path, and (2) the output of the parent is the input to the child, ensuring that the operations performed by a CNN are not repeated multiple times. Therefore, each root-leaf path of TRUNK acts like an independent CNN with several layers (divided into small CNNs), leading to an architecture with fewer redundant operations.

A. Properties Required for TRUNK

Before TRUNK can be used, the hierarchy structure must be selected. We highlight two properties that TRUNK hierarchies must satisfy to be both accurate and efficient.

Property 1: *The hierarchy structure should perform easy classifications near the root, and hard classifications near the leaves*. By doing so, the difficult classifications are performed after more layers have processed the input to extract informative features. Generally, such hierarchies can achieve high accuracy even when using smaller CNNs for high efficiency.

When forming the clusters in TRUNK, we can use two main types of similarity metrics: (a) *semantic*: objects are linked to one another using conceptual and lexical relations e.g., cats and dogs as animals, and (b) *visual*: objects are linked based on their appearances e.g., pizzas and plates because of their circular shape. Our experiments find that visual and semantic similarities do not always overlap [6].

When using semantic similarities, visually similar categories like *plates* and *pizzas* may belong to different clusters. In this case, the CNNs close to the root face the difficult task of distinguishing between *pizzas* and *plates*. To perform such operations accurately, larger and inefficient CNNs may be required. In comparison, with visual similarities, the CNNs near the root encounter a relatively easy task. They distinguish between clusters of visually similar categories. The more difficult classifications between the visually similar categories within clusters are performed farther from the root.

Property 2: *The hierarchy should have an intermediate structure, that is neither too wide or too tall.* Different visual similarity metrics may result in different hierarchies with varying structures. Each hierarchy structure provides a different accuracy-efficiency tradeoff. For a given hardware configuration and accuracy requirement, TRUNK hierarchies should have a hierarchy with appropriate width and height.

Consider a tall-and-narrow hierarchy, with multiple nodes in each root-leaf path and few children under each node. This hierarchy uses smaller CNNs at each node because each CNN only classifies between a small number of clusters (thus, few children). Although the CNNs perform relatively easy tasks, they usually do not obtain 100% accuracy. As a result, the error in each level of the hierarchy gets compounded, resulting in lower overall accuracy. On the other extreme, a short-and-wide hierarchy has many children at each node. To classify more children accurately, larger and more complex CNNs are required. Such CNNs may resemble the existing monolithic CNNs that TRUNK aims to replace. If short-and-wide hierarchies use large CNNs at each node, then TRUNK can achieve high accuracy, at the cost of lower efficiency.

B. Constructing Efficient and Accurate TRUNK Hierarchies

We now present a method to build TRUNK hierarchies that follow these properties. Our work finds that most existing visual similarity metrics require extensive manual fine-tuning for different datasets, or have constraints on the number of possible clusters, or are inconsistent in reporting the similarity between objects. Thus, we first develop a novel visual similarity metric that can solve these problems. We then vary the parameters of this visual similarity metric to control the hierarchy structure.

1) Confusion Between Categories: Our work [6] develops a new visual similarity metric called the Averaged Softmax Likelihood (ASL). The softmax layer is the CNN's output layer that assigns the prediction confidence to each possible output. Analyzing a CNN's softmax outputs helps us identify the categories that are often confused by the CNN. This CNN confusion is a measure of the visual similarity between categories. The greater the confusion between categories, the more visually similar they are.

The use of ASL can be understood with the example in Fig. 2. Suppose *horse* and *cow* are two categories in the training data. Eqt. (1) describes how the ASL is computed: the term $\text{softmax}_C(H)$ denotes the value obtained at the output (softmax) layer of the CNN corresponding to the object *cow*, when the input actually contains a *horse*. $|H|$ represents the number of input samples labeled as *horses*. Eqt. (1) is the CNN's average output for object category *cow*, when the inputs are *horses*. A large $L_C(H)$ implies that the CNN often mispredicts (with high confidence) that images of *horses* are *cows*. In other words, a larger $L_C(H)$ implies greater confusion between categories.

$$L_C(H) = \frac{\sum_H \text{softmax}_C(H)}{|H|} \tag{1}$$

The ASL technique described so far can find similarities between categories only for image classification datasets (each image has only one object); it cannot be used for images containing multiple objects (e.g., object counting and detection

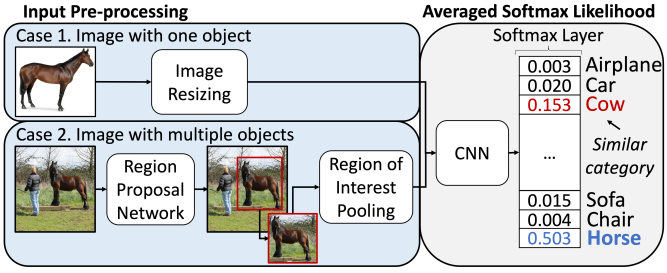


Fig. 2: Workflow to use the Averaged Softmax Likelihood to automatically identify clusters of visually similar categories.

tasks). In a subsequent study [7], we find that a Region Proposal Network (RPN) can be used to extend the ASL similarity metric to images with multiple objects. RPNs are small CNNs that propose regions-of-interest (RoIs): areas in images that may contain objects. By doing so, the RPN isolates each object in the image. Since CNNs accept only fixed sized images, we use a technique called RoI-Pooling to resize the RoIs without distorting their features. ASL can now be used with the labeled and resized RoIs to find the similarity between categories. The workflow for using ASL for images with a single object and for images with multiple objects is depicted in Fig. 2. Details about the implementation of ASL are available in our prior publications [6, 7].

2) Varying Similarity Metric Parameters to Control Hierarchy Structure: The similarity metric parameter is a measure of the strictness of the similarity metric. The similarity metric parameter decides how similar categories should be to get grouped into a single cluster. By tuning the parameter to increase the strictness, the similarity metric enforces small intra-cluster distances and large inter-cluster distances. In such cases, small clusters are formed only between highly similar categories, resulting in a short-and-wide hierarchy. On the other hand, tuning the parameter to reduce the similarity metric strictness will make the similarity metric group many categories to form fewer but larger clusters. This leads to a tall-and-narrow hierarchy.

By changing the CNN architecture used to compute ASL, we can tune the similarity metric parameter. The CNN's confusion is used to identify clusters of similar categories, e.g., *horse* and *cow* in Fig. 2. Larger and more complex CNNs are more accurate, and hence are less confused between categories. Using ASL with larger CNNs tunes the parameter to increase the strictness of the visual similarity metric, and consequently makes the hierarchy shorter and wider.

We use an architecture search technique that progressively grows the CNN, until it finds an architecture until the similarity metric parameter is tuned appropriately. In the following section, we describe how to find appropriate similarity metric parameters for different application requirements. Using this technique, TRUNK is constructed in a root-down fashion. First, a CNN architecture is found at the root. Then ASL groups categories and finds the first level of children. This process continues to grow the TRUNK. Finally, back-propagation is used to train the root CNN to classify between its children. For each newly formed child node, the process is repeated, continuing until all categories have been placed as leaves in the hierarchy. The algorithm to build TRUNK using this technique is beyond the scope of this article and is available in our prior publication [6].

C. Adaptation for Different Hardware Configurations

In this section, we describe how to control the attributes of TRUNK to meet different hardware constraints and accuracy requirements. Edge devices have different computing resources. The following examples present different scenarios where different TRUNK architectures may be beneficial. (a) On a powerful edge device equipped with a GPU (e.g., NVIDIA Jetson Nano with 4GB GPU memory), it may be acceptable to use a shorter-wider hierarchy to increase the overall accuracy at the expense of an increased computing workload due to the larger CNNs at each node. (b) On resource-scarce IoT devices with limited memory and no GPU (e.g., ARM Cortex M with 500KB memory), the small CNNs in a taller-narrower hierarchy may be required. (c) If TRUNK is unable to achieve the target accuracy on a given device, then an alternate device with more memory may be required. The larger memory would be able to accommodate larger CNNs, thus allowing a more accurate shorter-wider (also less efficient) hierarchy.

TABLE I: Symbols reference.

Symbol	Definition
P	Similarity metric parameter
F	Average fan-out at each node
A	Average accuracy on each CNN
S	Average workload size on each CNN
L	Average hierarchy path length
A_T	Overall TRUNK accuracy
W_T	Overall TRUNK workload size

To understand how to adapt TRUNK for varying requirements, we build two mathematical models that describe the TRUNK accuracy (A_T) and TRUNK workload size (W_T). The overall workload size represents the overall memory requirement, number of operations, or energy consumption. For simplicity, our experiments measure the overall memory requirement of TRUNK. Since the CNN's number of operations, energy consumption, and memory requirement are related, the same mathematical model and analysis can be extended for the other metrics [2, 3]. These models are built using five key attributes (listed in TABLE I) that impact TRUNK's performance.

The overall TRUNK accuracy (A_T) depends on the average CNN accuracy (A) and the average path length (L). The following example presents a simple analysis. Suppose A is 95% and there are 100 images in the test set. At the root, we can expect that 95/100 images are classified correctly. At the next level of the hierarchy, 95% of these 95 images are correctly classified again. For an entire hierarchy with depth, L, the expected overall test accuracy is modeled by eqt. (2).

$$A_T = A^L \tag{2}$$

Here, we see that TRUNK's accuracy decreases as the depth of the hierarchy increases. When L is large, $A \rightarrow 1.0$ is needed to achieve high accuracy. For example, to achieve $A_T > 0.95$, with $L = 2$, A must exceed 0.97. However, prior research shows that in order to achieve $A \rightarrow 1.0$, the CNN model, S, needs to increase significantly [2]. In most cases, we can say that A and S are positively correlated. Thus, increasing A may not be always be useful. We may need to decrease L.

L can be controlled by varying the similarity metric parameter (P) and the average fan-out (F) at each node of the hierarchy. P is a measure that decides if categories are similar enough to be grouped into a cluster. A large P leads to the formation of many small clusters of highly similar categories.

This article has been accepted for publication in IEEE Design & Test. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2022.3217016

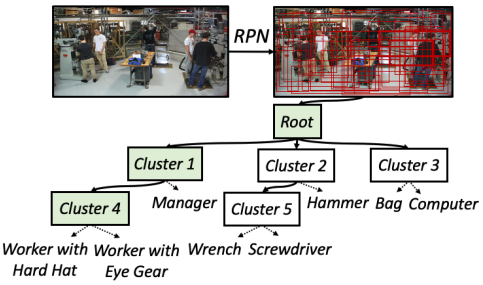
This results in a short-and-wide hierarchy with a large fan-out (F). A smaller P reduces the strictness of the similarity metrics and forms fewer but larger clusters, resulting in a tall-and-narrow hierarchy. In tall hierarchies, the paths are longer and hence we can see that P and F are positively correlated with one another, and they are negatively correlated with L.

The TRUNK overall workload size (W_T) can also be modeled. Thus, worst-case W_T is the sum of the individual CNN workloads along the longest root-leaf path. Thus, W_T depends on L and the average workload size of each CNN, S; and is modeled by eqt. (3). Our analysis uses the memory requirement of TRUNK as a proxy for the workload size, because prior research shows that the CNN number of operations, energy consumption, and memory requirement are related [2, 3].

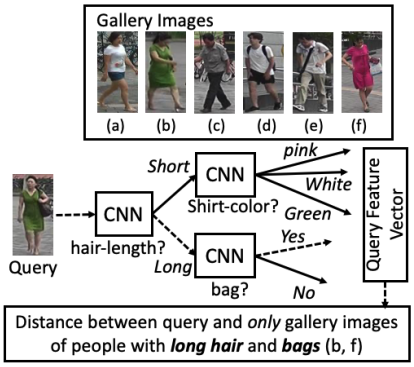
$$W_T = S \times L \tag{3}$$

Similar to L, S can also be controlled by varying P and F. As P and F increase, the problem at each CNN becomes larger and more complex. A large F resembles the existing monolithic CNNs like ResNet [4], and requires large CNNs for high accuracy. Thus, as F increases the CNN workload size (S) needs to increase to maintain the same accuracy A.

An ideal TRUNK hierarchy increases A_T and simultaneously decreases W_T. To increase A_T the hierarchy needs a small depth (L) and consequently a large fan-out (F) (L and F are negatively correlated). A large F requires large CNNs (S) to ensure no change in the accuracy at each node (A). This in turn increases W_T significantly. Thus, using these models, the TRUNK attributes can be tuned to achieve the desired tradeoff after considering the hardware constraints and accuracy requirements. We describe experiments in Section IV to validate these relationships.



(a)



(b)

Fig. 3: TRUNK used for different computer vision applications. (a) Object Counting: When attempting to count the number of workers wearing hard hats, only the RoIs classified into Cluster 1 and subsequently Cluster 4 are processed. All other RoIs are discarded to improve efficiency. (b) Object re-identification: Each small CNN identifies an attribute in the query image. With TRUNK we only rank gallery images with the same attributes as the query. Image source: Market-1501 dataset.

III. TRUNK for Efficient Computer Vision Applications

In this section, we use the object counting and re-identification (reID) problems to show how Tree-based Unidirectional Neural Network (TRUNK) can solve different computer vision tasks beyond image classification. In the object counting problem, the goal is to report the number of occurrences of a queried object category in an image with multiple objects. To avoid counting the same object multiple times, object counting is commonly combined with object reID. In the object reID problem, the goal is to identify if an image contains an object that has been seen before (possibly from a different angle or camera).

A. Object Counting

Existing object counters are based on object detectors. Most techniques use Region Proposal Networks (RPNs) to propose regions-of-interest (RoIs) in an image [11]. These methods then process all the RoIs with large CNNs to find all objects; finally, the occurrences of the queried object are counted. This process is highly redundant, because if we are only interested in counting *workers wearing hard hats* in an image, the computation involved in using large CNNs to detect every single object is not required.

TRUNK can perform object counting more efficiently than existing techniques, as seen in Fig. 3(a). A RPN is used to find RoIs (shown with red bounding boxes) in the image. Instead of processing each RoI with a large CNN, when using TRUNK, the RoIs are processed by the small CNN at the root. Only the RoIs that get classified onto the root-leaf path that contains the queried object (e.g., *workers wearing hard hats*) are processed by the next CNN. The other RoIs are not processed further, thus allowing us to increase efficiency.

B. Object Re-identification

Most existing reID techniques use large CNNs to extract features from the query image. The Euclidean Distance is used to compare these features with the features of every gallery image. The gallery images are ranked based on their distance from the query image. This typical approach performs many redundant operations because query images need not be compared with every gallery image. For example, the query image in Fig. 3(b) (a person with long hair and a bag) could be compared only to other people with long hair and bags — gallery images (b) and (f).

When using TRUNK for object reID, each small CNN of the hierarchy extracts features from the query and routes the query among its subsequent branches. Fig. 3(b) illustrates this approach. The query image of a pedestrian is processed by the root CNN to determine if the person has *long hair*. After the first attribute identification, the gallery reduces to images (a), (b), and (f). The next CNN continues to process the image and identifies if the person is *carrying a bag*. This attribute identification reduces the gallery to images (b) and (f). This process continues until a leaf CNN is reached. The features from the leaf CNN are used to perform comparisons with the remaining gallery images to re-identify the person. Because each node specializes in processing images with specific attributes, small, efficient CNNs can be used to obtain high accuracy.

IV. Experimental Evaluation

We conduct experiments to evaluate the TRUNK architecture for three computer vision applications: image classification, object counting, and object re-identification. We implement TRUNK using PyTorch. More details about the experimental setup and backbone architectures are available in our prior publications [6, 7, 8], and links to the prototypes.

TABLE II compares the test error of the proposed approach with existing techniques: hierarchical clustering, semantic similarities, and random grouping. Using ASL with NAS, TRUNK has the lowest error when the model sizes are the same.

TABLE III compares TRUNK with representative energy-efficient techniques: MobileNet v2 [1] for image classification on the CIFAR-10 dataset, YOLOv3 [12] for object counting on the PASCAL-VOC dataset, and ResNet50 [4] for object reID on the Market-1501 dataset. These techniques are existing (non-hierarchical) architectures that consume the least energy. The techniques are evaluated in terms of test error, memory requirement (in MB), number of arithmetic operations, and energy consumption when deployed on a NVIDIA Jetson Nano. The test error for classification, counting, and reID are measured as 1− test accuracy, Root Mean Squared Error [7], and Rank-1 error [8] metrics, respectively. For TRUNK, we report the worst case memory, operations, and energy requirement per image, i.e., the sum of the values along the longest root-leaf path. Experiments in our prior publications show that TRUNK can be used to improve efficiency on larger datasets, e.g., ImageNet, MS COCO, and VRAI [6]. On the larger datasets, TRUNK requires 70% less energy and memory, but achieves ∼3% lower accuracy.

TABLE II: Comparison of test error with different hierarchy construction techniques. Blue font indicates best result.

	Classification CIFAR-10		Counting PASCAL-VOC		reID Market-1501	
	Clustering	TRUNK	Semantic	TRUNK	Random	TRUNK
Test Error	0.231	0.079	2.560	1.800	0.212	0.115

TABLE III: Comparison of TRUNK with existing techniques. Blue font indicates best result for each application and metric.

	Classification CIFAR-10		Counting PASCAL-VOC		reID Market-1501	
	MobileNet	TRUNK	YOLOv3	TRUNK	ResNet	TRUNK
Test Error	0.060	0.079	1.610	1.800	0.128	0.115
Mem. (MB)	8.80	0.80	248.00	16.00	103.00	14.00
#Ops ($\times 10^6$)	100	28	141,000	44,000	3,882	808
Energy (J/img)	5.50	1.05	162.00	8.10	21.63	2.70

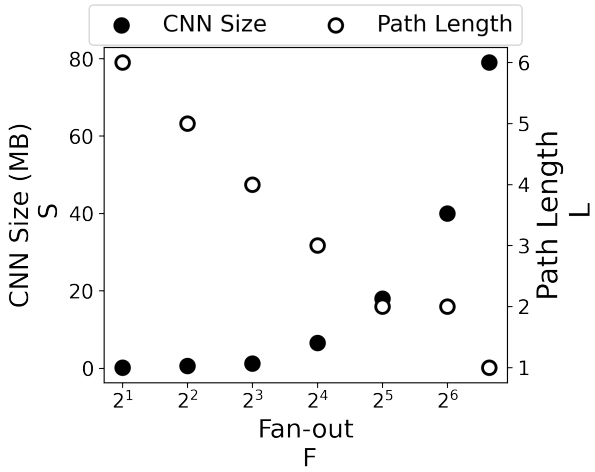


Fig. 4: As the fan-out (F) increases, the required workload size at one node (S) increases super-linearly and the hierarchy path length (L) decreases. Note that F is plotted on a logarithmic scale.

TRUNK consistently requires less memory and arithmetic operations, thus indicating higher efficiency. The performance gains come with a marginal loss in accuracy when compared with the state-of-the-art. Our experiments on the popular NVIDIA Jetson Nano reveals that the energy consumption and processing time of TRUNK is 80% to 95% lower than existing techniques across different computer vision applications and datasets. Our prior publications contain more experiments that analyze the impact of similarity metrics, thermal throttling, and embedded device hardware on overall performance [6, 7, 8]. Due to space constraints, we do not tabulate all the results [13].

Fig. 4 experimentally validates the mathematical models presented in eqt. (2) and (3). As mentioned in Section II-C, the overall workload size represents the overall memory requirement, number of operations, or energy consumption. For simplicity, our experiments measure the overall memory requirement of TRUNK. We construct hierarchies with varying fan-out (F) and plot the average CNN workload size (S) and hierarchy path length (L). For the CIFAR-100 dataset, we find that F is negatively correlated with L; depicted in Fig. 4 (white circles). The black circles in Fig. 4) show that as F increases the S grows super-linearly. Thus, these experiments show that decreasing L to increase accuracy (A_T) increases the workload

size (W_T) significantly. Thus, we use eqt. (2) and (3) to find appropriate hierarchy structures.

V. Discussion and Conclusion

In this article, we describe the Tree-based Unidirectional Neural Network (TRUNK) architecture that organizes multiple small CNNs in the form of a hierarchy. We present two important properties that are required by TRUNK to ensure high efficiency with high accuracy. Using those properties we develop a method to combine our novel visual similarity metric with neural architecture search to define hierarchy structures. We then build mathematical models that help adapt the TRUNK hierarchy structure to meet different hardware constraints and accuracy requirements. Finally, we also show the versatility of TRUNK to accurately perform different computer vision problems accurately on embedded devices. Our experiments confirm that TRUNK improves the deployability of computer vision systems.

This article highlights the application of TRUNK in three selected computer vision tasks. Looking forward, TRUNK can improve the efficiency of any deep learning task that benefits from a smaller search space, which is often the case in the real-world, where we are looking for very specific categories. The hierarchy structure effectively reduces the problem size, thus allowing smaller CNNs to complete the same task with fewer computational resources. Furthermore, this approach can be useful in other application scenarios. (a) Incremental learning: where new categories are discovered incrementally over time. When using TRUNK, only a small subset of CNNs need to be re-trained to accommodate a new object category. The other CNNs can be left unchanged. (b) Imbalanced datasets: some categories appear more often than others. The more frequently occurring objects can be placed closer to the root, thus allowing the objects to get classified faster.

The TRUNK approach represents a promising step for processing visual data on embedded devices. Future research could improve the utility of TRUNK by increasing its accuracy and versatility, without sacrificing efficiency.

Abhinav Goel received the Ph.D. degree from the Elmore Family School of Electrical and Computer Engineering at Purdue University in May 2022. His primary research focus is on efficient and low-power computer vision systems.

Caleb Tung is a doctoral student in the Elmore Family School of Electrical and Computer Engineering at Purdue University. Caleb's research is on energy-efficient computer vision on embedded devices.

Nick Eliopoulos is a doctoral student in the Elmore Family School of Electrical and Computer Engineering at Purdue University. Nick's research interest is in real-time computer vision and remote sensing.

Amy Wang is a high school student at West Lafayette Junior-Senior High School. She is interested in pursuing research in a computer science field.

James C. Davis is an assistant professor in the Elmore Family School of Electrical and Computer Engineering at Purdue University. His research is in empirical software engineering, focused on factors influencing the security of cyber- and cyber-physical systems.

George K. Thiruvathukal is a professor and chairperson in the Department of Computer Science of Loyola University Chicago. His research interests include parallel computing, software engineering, and computer vision.

Yung-Hsiang Lu is a professor in the Elmore Family School of Electrical and Computer Engineering at Purdue University. He is a fellow of the IEEE and distinguished scientist of the ACM.

References

[1] A. G. Howard et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". In: *2017 arXiv:1704.04861*.

[2] S. Bianco et al. "Benchmark Analysis of Representative Deep Neural Network Architectures". In: *2018 IEEE Access*.

[3] G K. Thiruvathukal et al. *Low-Power Computer Vision: Improve the Efficiency of Artificial Intelligence*. CRC Press, 2022.

[4] K. He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE CVPR*.

[5] S. Han et al. "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding". In: *2015 arXiv:1510.00149*.

[6] A. Goel et al. "Modular Neural Networks for Low-Power Image Classification on Embedded Devices". In:

[7] A. Goel et al. "Low-Power Object Counting with Hierarchical Neural Networks". In: *2020 ACM/IEEE ISLPED*.

[8] A. Goel et al. "Low-Power Multi-Camera Object Re-Identification using Hierarchical Neural Networks". In: *2021 IEEE/ACM ISLPED*.

[9] Z. Yan et al. "HD-CNN: Hierarchical Deep Convolutional Neural Networks for Large Scale Visual Recognition". In: *2015 IEEE ICCV*.

[10] X. Zhu et al. "B-CNN: Branch Convolutional Neural Network for Hierarchical Classification". In: *2017 arXiv:1709.09890*.

[11] S. Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *2016 arXiv:1506.01497*.

[12] J Redmon et al. "YOLOv3: An Incremental Improvement". In: *2018 arXiv:1804.02767*.

[13] URL: https://github.com/abhinavgoel95/TRUNK.