

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

Abhinav Goel, Caleb Tung, and Nick Eliopoulos

School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907 USA

Amy Wang

West Lafayette Junior-Senior High School
West Lafayette, IN 47906 USA

James C. Davis

School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907 USA

George K. Thiruvathukal

Department of Computer Science
Loyola University Chicago
Chicago, IL 60660 USA

Yung-Hsiang Lu

School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907 USA

Editor's notes:

Convolutional neural networks (CNNs) have paved paths for high-accuracy computer vision. This article presents a tree-based hierarchy of multiple shallow CNNs to enable their low-power implementations for embedded devices.

—Muhammad Shafique, New York University Abu Dhabi

■ **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

Digital Object Identifier 10.1109/MDAT.2022.3217016

Date of publication: 2 November 2022; date of current version: 24 April 2023.

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, for example, 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, for example, 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 Figure 1.

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 nonzero 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 a 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. Figure 1b illustrates the TRUNK architecture, where the categories are cat, dog, and so on. 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 is 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: 1) ensemble or 2) 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.

Tree-based unidirectional neural networks

This section describes the proposed TRUNK architecture. We use image classification to explain the architecture in more detail.

As seen in Figure 1b, 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.

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: 1) *semantic*: objects are linked to one another using conceptual and lexical relations, for example, cats and dogs as animals and 2) *visual*: objects are linked based on

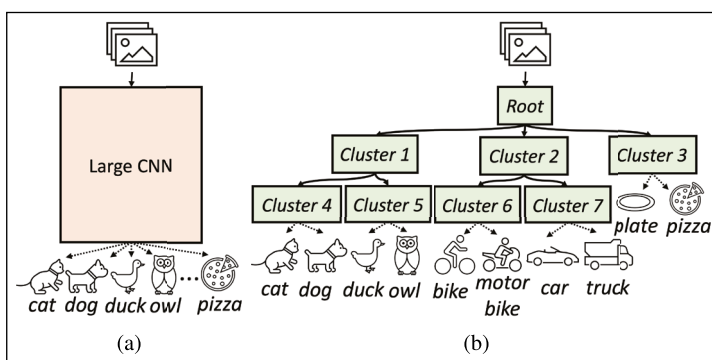


Figure 1. (a) Existing CNN architectures: A single monolithic architecture classifies images into their categories. (b) Proposed TRUNK: The input image is processed incrementally using small CNNs. After detecting the type of images, finer classifications are made.

their appearances, for example, 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 nor 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 a 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.

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 data sets, 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.

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 Figure 2. Suppose horse and cow are two categories in the training data. The following equation describes how the ASL is computed:

$$L_c(H) = \frac{\sum_H \text{softmax}_c(H)}{|H|}. \quad (1)$$

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. Equation (1) is the CNN's average output for the 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.

The ASL technique described so far can find similarities between categories only for image

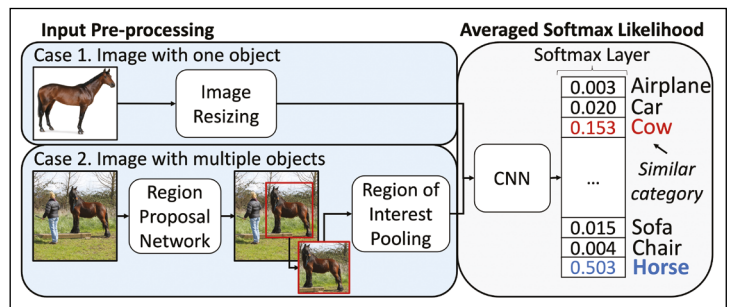


Figure 2. Workflow to use the ASL to automatically identify clusters of visually similar categories.

classification data sets (each image has only one object); it cannot be used for images containing multiple objects (e.g., object counting and detection 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 (Rols): 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 Rol-pooling to resize the Rols without distorting their features. ASL can now be used with the labeled and resized Rols to find the similarity between categories. The workflow for using ASL for images with a single object and images with multiple objects is depicted in Figure 2. Details about the implementation of ASL are available in our prior publications [6], [7].

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 intracluster distances and large intercluster 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, for example, horse and cow in Figure 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 categorize and find 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].

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.

- 1) On a powerful edge device equipped with a GPU (e.g., NVIDIA Jetson Nano with 4 GB 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.
- 2) On resource-scarce IoT devices with limited memory and no GPU (e.g., ARM Cortex M with 500 KB memory), the small CNNs in a taller-narrower hierarchy may be required.
- 3) 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.

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 1) 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 the following:

$$A_T = A^L. \quad (2)$$

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 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 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 results in a short-and-wide hierarchy with a large fanout (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, the 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

$$W_T = S \times L. \quad (3)$$

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].

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 the upcoming section to validate these relationships.

TRUNK for efficient computer vision applications

In this section, we use object counting and reidentification (reID) problems to show how 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).

Object counting

Existing object counters are based on object detectors. Most techniques use RPNs to propose Rols in an image [11]. These methods then process all the Rols with large CNNs to find all objects; finally, the

Table 1. 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

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 Figure 3a. An 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.

Object reidentification

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 Figure 3b (a person with long hair and a bag) could be compared only to other people with long hair and bags in 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. Figure 3b 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 reidentify the person. Because each node specializes in processing images with specific attributes, small, efficient CNNs can be used to obtain high accuracy.

Experimental evaluation

We conduct experiments to evaluate the TRUNK architecture for three computer vision applications: image classification, object counting, and

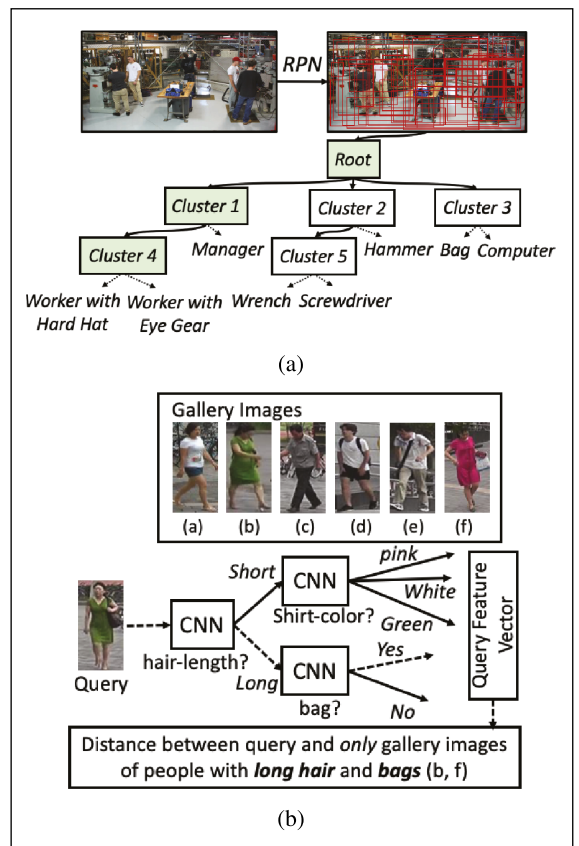


Figure 3. TRUNK used for different computer vision applications. (a) Object counting: When attempting to count the number of workers wearing hard hats, only the RoIs classified into Cluster 1 and subsequently Cluster 4 are processed. All other RoIs are discarded to improve efficiency. **(b) Object reID:** 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 data set.

object reidentification. 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 2 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 3 compares TRUNK with representative energy-efficient techniques: MobileNet v2

Table 2. Comparison of test error with different hierarchy construction techniques. Blue font indicates the 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 3. Comparison of TRUNK with existing techniques. Blue font indicates the 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

[1] for image classification on the CIFAR-10 data set, YOLOv3 [12] for object counting on the PASCAL-VOC data set, and ResNet50 [4] for object reID on the Market-1501 data set. These techniques are existing (nonhierarchical) 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 an NVIDIA Jetson Nano. The test error for classification, counting, and reID are measured as $1 - \text{test accuracy}$, 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, that is, 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 data sets, for example, ImageNet, MS COCO, and VRAI [6]. On the larger data sets, TRUNK requires 70% less energy and memory, but achieves ~3% lower accuracy.

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 reveal that the energy consumption and

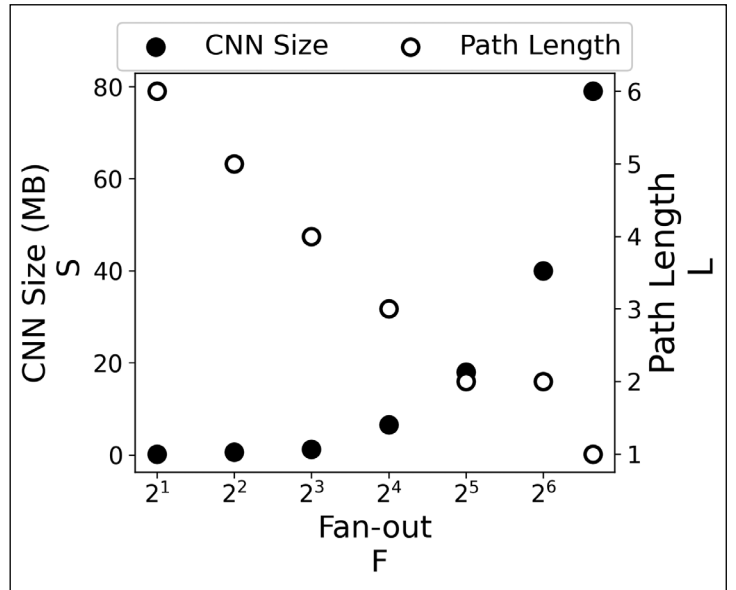


Figure 4. As the fan-out (F) increases, the required workload size at one node (S) increases superlinearly and the hierarchy path length (L) decreases. Note that F is plotted on a logarithmic scale.

processing time of TRUNK are 80%–95% lower than existing techniques across different computer vision applications and data sets. 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].

Figure 4 experimentally validates the mathematical models presented in (2) and (3). As mentioned in the previous section, 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 data set, we find that F is negatively correlated with L , as depicted in Figure 4 (white circles). The black circles in Figure 4 show that as F increases, S grows superlinearly. Thus, these experiments show that decreasing L to increase accuracy (A_T) increases the workload size (W_T) significantly. Thus, we use (2) and (3) to find appropriate hierarchy structures.

In this article, we describe the 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.

- 1) *Incremental learning*: Where new categories are discovered incrementally over time. When using TRUNK, only a small subset of CNNs need to be retrained to accommodate a new object category. The other CNNs can be left unchanged.
- 2) *Imbalanced data sets*: 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. ■

References

- [1] A. G. Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.
- [2] S. Bianco et al., "Benchmark analysis of representative deep neural network architectures," *IEEE Access*, vol. 6, pp. 64270–64277, 2018.
- [3] G. K. Thiruvathukal, *Low-Power Computer Vision: Improve the Efficiency of Artificial Intelligence*. Boca Raton, FL, USA: CRC Press, 2022.
- [4] K. He et al., "Deep residual learning for image recognition," in *Proc. IEEE CVPR*, Jun. 2016, pp. 770–778.
- [5] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, *arXiv:1510.00149*.
- [6] A. Goel et al., "Modular neural networks for low-power image classification on embedded devices," *ACM Trans. Design Autom. Electron. Syst.*, vol. 26, no. 1, pp. 1–35, Jan. 2021.
- [7] A. Goel et al., "Low-power object counting with hierarchical neural networks," in *Proc. ACM/IEEE ISLPED*, Aug. 2020, pp. 163–168.
- [8] A. Goel et al., "Low-power multi-camera object re-identification using hierarchical neural networks," in *Proc. IEEE/ACM ISLPED*, Jul. 2021, pp. 1–6.
- [9] Z. Yan et al., "HD-CNN: Hierarchical deep convolutional neural networks for large scale visual recognition," in *Proc. IEEE ICCV*, Dec. 2015, pp. 2740–2748.
- [10] X. Zhu and M. Bain, "B-CNN: Branch convolutional neural network for hierarchical classification," 2017, *arXiv:1709.09890*.
- [11] S. Ren et al., "Faster R-CNN: Towards real-time object detection with region proposal networks," 2015, *arXiv:1506.01497*.
- [12] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*.
- [13] *TRUNK Source Code*. Accessed: Feb. 9, 2022. [Online]. Available: <https://github.com/abhinavgoel95/TRUNK>

Abhinav Goel is interested in efficient and low-power computer vision systems. Goel has a Ph.D. from the Elmore Family School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA.

Caleb Tung is pursuing a Ph.D. in the Elmore Family School of Electrical and Computer Engineering at Purdue University, West Lafayette, IN 47907 USA. His research is on energy-efficient computer vision on embedded devices.

Nick Eliopoulos is pursuing a Ph.D. in the Elmore Family School of Electrical and Computer Engineering at Purdue University, West Lafayette,

IN 47907 USA. His 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, University, West Lafayette, IN 47907 USA. She is interested in pursuing research in computer science.

James C. Davis is an assistant professor in the Elmore Family School of Electrical and Computer Engineering at Purdue University, West Lafayette, IN 47907 USA. 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 a chairperson in the Department of Computer Science at Loyola University Chicago, Chicago, IL 60660

USA. 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, West Lafayette, IN 47907 USA. He is a Fellow of IEEE and a Distinguished Scientist of the ACM.

■ Direct questions and comments about this article to Abhinav Goel, School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA; goel39@purdue.edu.