Stochastic Functional Verification of DNN Design through Progressive Virtual Dataset Generation

Jinhang Choi*, Kevin M. Irick†, Justin Hardin*, Weichao Qiu‡, Alan Yuille‡, Jack Sampson*, and Vijaykrishnan Narayanan*
*School of Electrical Engineering and Computer Science, Pennsylvania State University, University Park, PA 16802, USA
†SiliconScapes LLC, USA
‡Department of Computer Science, Johns Hopkins University, Baltimore, MD 21218, USA
*{jpc5731, jdh5706, sampson, vijay}@cse.psu.edu, †kevin.irick@siliconscapes.net, ‡{wqiu7, alan.yuille}@jhu.edu

***Abstract*—Deep Neural Networks have emerged as state-of-the-art solutions for complex intelligence problems. DNNs derive their predictive power by learning from millions of training examples in either a supervised or semi-supervised fashion. As such, a critical aspect of the DNN system design procedure is the collection of large annotated training datasets that exhibit high coverage of the problem space. While data synthesis and annotation techniques have been proposed to mitigate the burden of acquiring large datasets, these methods do not quantify the usefulness of each generated dataset and its subsequent impact on training effort. In this work we establish parallels between the autonomous design of DNNs for machine vision applications and the task of functionally verifying a hardware design. Similar to automatic test vector generation, we propose a technique that progressively generates training datasets using virtual synthetic models. Furthermore, we propose an automated DNN design framework that jointly tries to stochastically maximize training coverage while minimizing the number of training and validation cycles utilizing insights from functional verification.**

Keywords*—Neural networks, Design automation, Virtual prototyping

I. Introduction

Following the substantial success that deep learning has shown for solving many complex real-world problem domains including machine vision, natural language processing, drug discovery, and intelligent recommendation systems [1], many systems that exploit Deep Neural Network (DNN) models have been proposed and implemented [2], [3], [4], [5]. DNN-based circuit/system design employs a *training procedure*, in which the parameters of the DNN model are learned and validated with respect to a large number of problem-specific training examples. For example, DNN models such as [6], [7], [8] beat competing computer vision approaches in the Imagenet Large Scale Visual Recognition Challenges (ILSVRC) by exploiting training on more than 1.2 million labeled images [9].

The primary impediments to utilize DNN approaches are *a*) the acquisition and annotation of large representative training datasets, and *b*) the time and resources required to train the DNN to achieve the desired performance. Massive data acquisition is difficult in many application domains because environments suitable for data collection are either small in number and/or too impractical or dangerous for access. Even if a large dataset can be collected, manual annotation can be prohibitively time consuming and error prone. One technique to reduce the effort of acquiring and labeling large datasets is to start with a relatively small dataset and to randomly permute the samples to produce a larger dataset. In the case of DNNs for machine vision, the common types of transformations include rotating, skewing, cropping, noising, blurring, and adding occlusions. Since each DNN model has its own specific semantic functionalities, not all transformations will have equal impact in making a larger representative dataset. The limitation of the approach can be illustrated by paralleling the use of unconstrained randomly generated test vectors for the functional verification of an electronic design component. While every new test vector certainly increases the verification effort, each new test vector is not guaranteed to test a relevant design functionality that has not been sufficiently covered. It is important to note that presenting a Design Under Test (DUT) with redundant test vectors only increases the test time but has a benign effect on the design itself. Contrarily, training a DNN with overly redundant data can lead to over-fitting, which reduces the predictive generality of the DNN [10].

Methods such as semi-supervised learning [11] and data synthesis exploiting DNN models [12] have been proposed to resolve difficulties in acquiring training datasets. Semi-supervised learning creates a bag of representative features based on a small set of annotations, which automatically augments data labels for the entire unlabeled dataset. Data synthesis learns to generate artificial training samples by extracting rules from real data samples. However, if the real dataset is insufficiently sized or does not exhibit enough configurable parameters, the synthesis process may not generate realistic synthetic examples. Moreover, relying exclusively on data synthesis to accurately cluster a large numbers of unlabeled data samples can lead to noisy and incomplete datasets.

Rapid advances in hyper-realistic game engines and virtual world authoring and rendering tools have presented an opportunity to use high quality 3D virtual assets to create training data for vision recognition tasks [13], [14]. Almost all visual appearances of many real-life object types can be rapidly generated by tuning viewpoint perspective, distance, deformation, occlusion and lighting. Moreover, there is a vast and growing collection of commercially available 3D object models that leverage the latest advances in high-fidelity graphic rendering. Even with the ability to generate arbitrary numbers of realistic data samples, the question remains of how to systematically train a DNN in a timely fashion with only the most relevant subset training samples.

We propose an automated DNN design flow inspired by the established and well-understood functional verification methodologies used in the electronic hardware design process. Our framework is suitable for inclusion in existing DNN frameworks [15], [16], [17] to increase DNN-based design efficiency. As a case study, we present training two different DNN models for two vision recognition tasks in grocery envi-

978-1-5386-4881-0/18/$31.00 ©2018 IEEE

Algorithm 1 Generate training dataset $\mathcal{A}$

Require: $n > 0, \{\Theta_i\}_0^{n-1} \neq \varnothing, \Omega \neq \varnothing, \xi \in (0, 1]$
Ensure: $\mathcal{A} \neq \varnothing$
  $\mathcal{A} \leftarrow \varnothing$
  $i \leftarrow 0$
  for $i < n$ **do**
    $\mathcal{A}_{new} \leftarrow \texttt{GenDataSet}(\Theta_i)$
    if $\texttt{Similarity}(\mathcal{A}^*, \mathcal{A}^*_{new}; \Psi_{model}) < \xi$ **then**
      $\mathcal{A} \leftarrow \mathcal{A} \cup \mathcal{A}_{new}$
      $\Psi_{model} \leftarrow \texttt{TrainSystem}(\mathcal{A}, \Omega)$
    else
      continue
    end if
    $i \leftarrow i + 1$
  end for

ronments[1]: grocery item localization using *SharpMask* [20], and grocery item classification using *GoogLeNet* [7]. Any application-specific system design exploiting DNN model is incomplete until we finish its training procedure. Training DNN models for grocery environments is particularly challenging because *a*) there are no publicly available grocery item image datasets and *b*) it is impractical to gather a large and diverse set of training images. As such, we trained and validated the two DNN models on synthetic imagery created from a virtual grocery store based on the Unreal Engine 4 (UE4) platform [21] and tested on images collected from real grocery items.

II. Stochastic Functional Verification with Progressive Synthetic Dataset Generation

Algorithm 1 depicts our proposed scheme for generating training datasets. A given virtual environment, Θ_i at phase i, constitutes a new candidate dataset, $\mathcal{A}_{new}$, to be added to the target training dataset, $\mathcal{A}$. To determine whether or not $\mathcal{A}$ already includes $\mathcal{A}_{new}$, we compare validation subsets of the two training datasets, $\mathcal{A}^*$ and $\mathcal{A}^*_{new}$, against the target model system, Ψ_{model}. As a comparison metric, we use cosine similarity in the vector space of task-specific objective quantity. For instance, in object detection, the comparison metric could be class-wise average precision. In the initial state, since there is no Ψ_{model} and $\mathcal{A}$ to check similarity, we set the test result to 0 as a special case. If a similarity is weaker than the predefined level, ξ, indicating that $\mathcal{A}_{new}$ contains novel data that has not been considered before, then we append $\mathcal{A}_{new}$ to $\mathcal{A}$, and train Ψ_{model} from the updated dataset based on model design, Ω.

Denoting a DNN under verification (DUV) by Ψ_{model}, each dataset generation phase represents a cycle of *stochastic functional verification*. Monitoring similarity level ξ against Ψ_{model} triggers DUV self-checking while monitoring stimuli $\mathcal{A}^*_{new}$ generated in virtual test pattern $\mathcal{A}_{new}$. A similarity checker queries scoreboard $\mathcal{A}^*$ containing design parameters in Θ_i. If similarity does not converge in a verification loop, we have to modify the model design Ω, which is analogous to bug rate saturation in the established functional verification process [22]. As a result, the final synthetic dataset is capable of training and validating a target DNN system model by exploring virtual design parameters deterministically. However, consideration of all design parameters results in combinatorial explosion. Instead, we target a reasonable training space coverage and focus on permuting the virtual environment parameters that are most relevant to the vision recognition task.

[1]The use of computer vision for adding intelligence in grocery environments has garnered significant commercial attention [18], [19].

(a) Front view (b) Isometric view (c) Oblique view

Fig. 1: visual perspectives of shopping shelves in real world

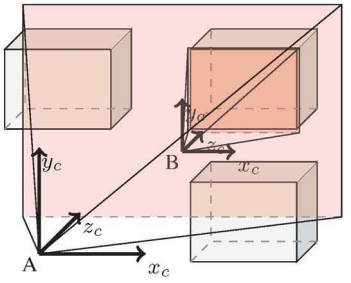


Fig. 2: The impact of viewpoint on projections. A distant viewpoint provides diverse visual perspectives from relatively relocated objects (A: oblique view). A close viewpoint focuses on object's texture details (B: front view). The red screens of two viewpoint projections have the same image resolution.

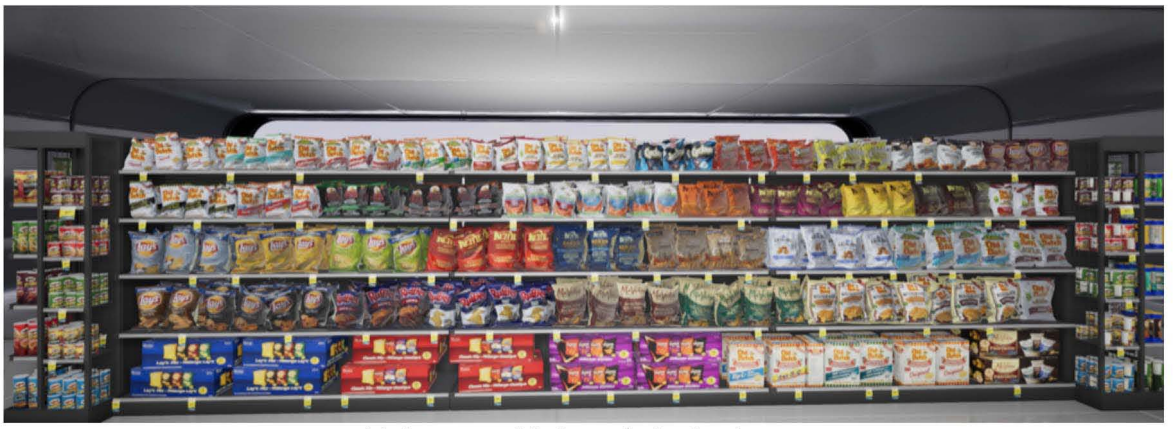
III. Coverage of Virtual Training Sets

Fig. 1 depicts how geometric shapes and their texture patterns change with respect to viewpoint due to image projection into a 2D image plane. 3D-2D geometric projection is based on the following equation:

$$\begin{bmatrix} \mathbf{u}' \\ \mathbf{v}' \\ \mathbf{w}' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{w} \\ 1 \end{bmatrix} \quad (1)$$

such that the final 2D coordinate in an image corresponds to $(u, v) = (\mathbf{u}'/\mathbf{w}', \mathbf{v}'/\mathbf{w}')$, where a_{ij} is the conversion from a 2D shape to pixel coordinates, f is focal length for 3D-2D projection, and r_{ij} and t_{axis} stand for the perspective angle and location in 3D, respectively. As described in (1), the effect of visual perspective can be achieved by changing either *a*) the perspective angle and location, or *b*) item displacement. In other words, placement of an item with horizontal or vertical positional variation provides diverse visual perspectives within a single viewpoint. Therefore, we do not have to explore many viewpoints, but rather some representative viewpoints by repetitively placing the same item at different locations. However, Fig. 2 depicts how a viewpoint projection interacts with a finite image resolution by sacrificing texture details depending on the distance between the viewpoint and items. Therefore, it is required to address a number of distance-dependent viewpoints by comparing pattern variation versus pattern details.

978-1-5386-4881-0/18/$31.00 ©2018 IEEE



(a) A grocery aisle in synthetic virtual space

(b) projection at $base$ position

(c) projection at $base + 784cm$

(d) projection at $base + 1,208cm$

Fig. 3: Manipulating design space parameters in virtual environment (a) generates validation stimuli: (b), (c), (d).

A real-life object can have an exponential number of shape and texture appearances due to deformation from applied forces and partial occlusion by itself and other objects. Additionally, the light sources can create glare that introduces texture occlusion or distortion on the object facet [23]. We can control and render many such object appearances by deploying a physical modeling-based graphic engine. Fig. 3a depicts an example of a grocery aisle defined in UE4. From pseudo-randomly placing the objects in terms of rotation and translation, a virtual synthetic environment such as UE4 makes it easy to achieve realistic individual test stimulus in the automation process. For example, Fig. 3b to 3d describe vertically dispersed object identities to relatively relocate the viewpoints so that we can capture diverse shapes of an object in a viewpoint. Partial occlusion, as well as lateral facet variation, are realized in a natural manner based on geometry.

IV. Design Space Exploration Strategy

While keeping environmental effects, such as light sources and object placements, constant on the UnrealCV virtual reality control framework [13], we manipulate the viewpoint and three different projection properties in the virtual space configuration, Θ_i, for each dataset generation phase. To be specific, we assume viewpoint space ranging from an oblique view to a detailed front facing view, where the effective vertical viewing angle is larger than 20°. This assumption restricts viewpoint positions in the virtual environment to a representative space as shown in Fig. 4. Additionally, we filtered out stimuli not satisfying occlusion ratio, aspect side ratio, and absolute 2D area when an image projection is generated.

A. Object localization with SharpMask

Object localization has the objective of identifying the exact position and shape of objects of interest. As a case study, we limit $n = 1, \xi = 1$, and present a near-front viewpoint to extract training/validation stimuli. Shaded green masks shown in Fig. 3b to 3d are some examples of the exact labeling according to 2D projections. To target grocery item detection, we consider 8 object categories: chips (lays, kettle, barbaras, and doritos), chip boxes, canned chips, soft drink bottles, and soft drink boxes. In addition, we validate data sample only if its projection presents more than 65% of the sample's entire area before occlusion, less than 2.5:1 in aspect side ratio, and larger than $8,100 (= 90 \times 90)$ pixels in projected 2D area. Each data sample is annotated as described in Fig. 3b to 3d. While repeating horizontal $2cm$ viewpoint displacement in a fixed viewpoint angle, we generate $1,000$ 2D image projections which are composed of $26,242$ stimuli. The labeled synthetic dataset is then used to train a DNN-based object region proposal generator, *SharpMask* [20]. To monitor model over-fitting, hold-out validation is conducted with a 9:1 ratio of training to validation stimuli ($\mathcal{A} - \mathcal{A}^* : \mathcal{A}^* = 9 : 1$). Fig. 5a and 5b demonstrate that representations trained on synthetic image datasets can lead to a well-established DNN model that localizes real world images. There is no intervention of augmenting any labeled real image data to obtain the DNN model.

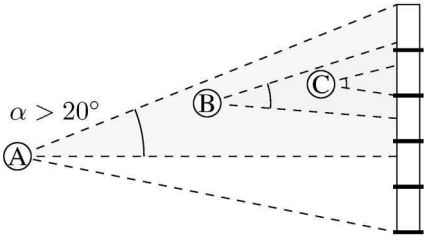
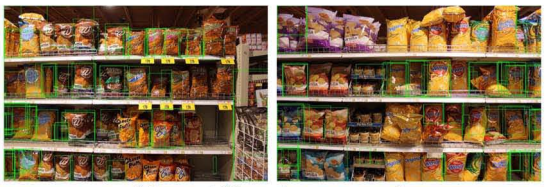


Fig. 4: Three representative viewpoints of human eye looking at a grocery aisle: a distant viewpoint for visual perspective variations (A), an intermediate viewpoint (B), and a close viewpoint for texture detail variations (C).

978-1-5386-4881-0/18/$31.00 ©2018 IEEE



(a) top-100 region proposals

(b) top-500 region proposals

Fig. 5: Test results of *SharpMask* against real images: synthetic image-based DNN training can be deployed at suggesting region of interests in a real world. Picture (a) and (b) are taken from *Wegmans* aisle.

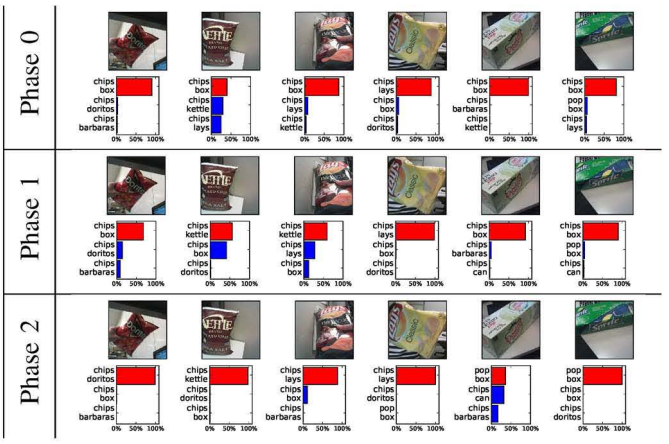


Fig. 6: Top-3 inference changes of *GoogLeNet* against real images: synthetic image-based DNN training can identify objects regardless of visual variations as we progress viewpoint-wise design space exploration.

B. *Object classification with GoogLeNet*

In contrast to localization, object classification requires a deeper understanding of both shape and texture details to assign an object class label to an image under consideration. As explained in Section III, stimuli are biased toward coarse shapes in a distant oblique viewpoint, and stimuli are concentrated on texture pattern details in close detail viewpoints. To take account of the visual impacts of viewpoints, we limit $n = 4, \xi = 0.95$, and provide the virtual synthetic space with four different viewpoints. Since we only adjust viewpoint for each data generation phase in this experiment, dataset similarity is close to 1. However, depending on changes in parameters such as light source and object occlusion ratio, the similarity is more likely to approach 0. In such circumstances, the dataset similarity plays an important role to specify how much we employ divergence in dataset distribution.

To train *GoogLeNet* [7], we generate $11,461$ training and validation stimuli from $1,300$ 2D image projections at phase 0; $43,748$ stimuli from other $2,600$ projections at phase 1; $13,217$ stimuli from the other $6,500$ projections at phase 2; and $10,809$ stimuli from the other $1,000$ projections at phase 3. The iterative generation of a new dataset $\mathcal{A}$ is inclusive of the previous phase's dataset. Since similarity evaluation approaches 1.00 at phase 3 as shown in Table I, we freeze dataset integration at phase 2. We again choose hold-out validation with a ratio of 9:1. As a result, the number of training/validation stimuli for a grocery item dataset corresponds to $10,178/1,283$ data samples at phase 0, $49,001/6,208$ data samples at phase 1, and $60,749/7,677$ data samples at phase 2, respectively. Fig. 6 depicts inference test results against images of real world objects after training *GoogLeNet* on the virtual synthetic dataset; again, we do not use any real image dataset in the training procedure. In generation phase 0, the synthetic dataset does not include texture pattern details but only representative colors as well as rich visual perspective-dependent shapes due to viewpoint-wise object projections. Therefore, *GoogLeNet* at Phase 0 can only differentiate (potato-chip) bag and box shapes. In phase 1, as finer texture patterns are added to the dataset, *GoogLeNet* starts classifying object class entities. Finally, with support of texture details in phase 2, all test objects are correctly matched with their corresponding classes. Due to the fixed layout of the virtual shelves, objects placed at corners do not equally contribute to the number of synthetic data samples, which leads to a comparatively low confidence level for ginger ale box inferences.

TABLE I: Similarity between synthetic dataset $\mathcal{A}^*$ and $\mathcal{A}^*_{new}$

$\mathcal{A}^*_{new}$	$\mathcal{A}^*$				Viewpoint
	Phase -	Phase 0	Phase 0+1	Phase 0+1+2	
Phase 0	**0**	1.00	1.00	1.00	Fig. 4-A
Phase 1	-	**0.87**	1.00	1.00	Fig. 4-B
Phase 2	-	-	**0.90**	1.00	Fig. 4-C
Phase 3	-	-	-	**1.00**[a]	Between Fig. 4-B and C

a) Dataset integration at phase 3 is skipped because of $\xi = 0.95$.

If we had a fully annotated real image dataset, the number of training cycles would decrease as such a dataset naturally covers the variations of our different training phases. However, as stated in [9], clean annotation becomes impossible with billions of real images. Even though our proposed stochastic functional verification does not perfectly guarantee validation against untested real world scenarios, it gives us a high fidelity of optimizing the design space that we have to consider as shown in Fig. 5 and 6.

V. Conclusion

In this paper, we propose a stochastic functional verification approach to the design of DNN-based systems. The design flow leverages the ability to automatically generate and annotate training datasets to systematically train and evaluate a DNN. As a consequence, we can escalate system design exploration with an iterative training procedure in a closed automation loop. Unfortunately, while traversal can be fully automated, the dataset design space remains exponential. However, our case studies demonstrate that parametric manipulation guides us to introduce well-designed synthetic datasets that are capable of effecting accurate inference on real-world test cases. In this context, we are working with an additive training strategy to narrow down the system design space either deterministically or heuristically. Augmenting multi-modality to the virtual design environment is a key direction of our future work.

Acknowledgment

This work was supported in part by NSF Expeditions in Computing Program: Visual Cortex on Silicon CCF 1317560.

978-1-5386-4881-0/18/$31.00 ©2018 IEEE

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[2] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun, and O. Temam, "DaDianNao: A Machine-Learning Supercomputer," in *Proceedings of 47th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2014, pp. 609–622.

[3] S. K. Esser, P. A. Merolla, J. V. Arthur, A. S. Cassidy, R. Appuswamy, A. Andreopoulos, D. J. Berg, J. L. McKinstry, T. Melano, D. R. Barch, C. di Nolfo, P. Datta, A. Amir, B. Taba, M. D. Flickner, and D. S. Modha, "Convolutional networks for fast, energy-efficient neuromorphic computing," *Proceedings of the National Academy of Sciences (PNAS)*, vol. 113, no. 41, pp. 11 441–11 446, 2016.

[4] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "EIE: Efficient Inference Engine on Compressed Deep Neural Network," in *Proceedings of the 43rd ACM Annual International Symposium on Computer Architecture (ISCA)*, 2016, pp. 243–254.

[5] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P.-l. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. H. Yoon, "In-Datacenter Performance Analysis of a Tensor Processing Unit," in *Proceedings of the 44th ACM Annual International Symposium on Computer Architecture (ISCA)*, 2017, pp. 1–12.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25 (NIPS)*, 2012, pp. 1097–1105.

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper With Convolutions," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770–778.

[9] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[10] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research (JMLR)*, vol. 15, pp. 1929–1958, 2014.

[11] S. Hong, H. Noh, and B. Han, "Decoupled Deep Neural Network for Semi-supervised Semantic Segmentation," in *Advances in Neural Information Processing Systems 28 (NIPS)*, 2015, pp. 1495–1503.

[12] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning From Simulated and Unsupervised Images Through Adversarial Training," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017.

[13] W. Qiu and A. Yuille, "UnrealCV: Connecting Computer Vision to Unreal Engine," in *Computer Vision – ECCV 2016 Workshops*, 2016, pp. 909–916.

[14] S. Qiao, W. Shen, W. Qiu, C. Liu, and A. Yuille, "ScaleNet: Guiding Object Proposal Generation in Supermarkets and Beyond," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017.

[15] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional Architecture for Fast Feature Embedding," in *Proceedings of the 22nd ACM International Conference on Multimedia*, New York, NY, USA, 2014, pp. 675–678.

[16] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[17] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A Matlab-like Environment for Machine Learning," in *BigLearn, NIPS Workshop*, 2011.

[18] Amazon Go Store, https://www.amazon.com/b?node=16008589011.

[19] Bossa Nova Robotics, http://www.bossanova.com/.

[20] P. O. Pinheiro, T. Lin, R. Collobert, and P. Dollár, "Learning to Refine Object Segments," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2016, pp. 75–91.

[21] Epic Games, "Unreal Engine 4," 2006. [Online]. Available: https://www.unrealengine.com

[22] B. Wile, J. C. Goss, and W. Roesner, *Comprehensive Functional Verification: The complete industry cycle*. Morgan Kaufmann, 2005.

[23] P. A. Zientara, J. Choi, J. Sampson, and V. Narayanan, "Drones as Collaborative Sensors for Image Recognition," in *Proceedings of IEEE International Conference on Consumer Electronics (ICCE)*, Las Vegas, USA, Jan. 2018.

978-1-5386-4881-0/18/$31.00 ©2018 IEEE