

A Novel Design of Adaptive and Hierarchical Convolutional Neural Networks using Partial Reconfiguration on FPGA

Mohammad Farhadi
Arizona State University
Tempe, AZ, USA
mfarhadi@asu.edu

Mehdi Ghasemi
Arizona State University
Tempe, AZ, USA
mghasem1@asu.edu

Yezhou Yang
Arizona State University
Tempe, AZ, USA
yz.yang@asu.edu

Abstract—Nowadays most research in visual recognition using Convolutional Neural Networks (CNNs) follows the “deeper model with deeper confidence” belief to gain a higher recognition accuracy. At the same time, deeper model brings heavier computation. On the other hand, for a large chunk of recognition challenges, a system can classify images correctly using simple models or so-called shallow networks. Moreover, the implementation of CNNs faces with the size, weight, and energy constraints on the embedded devices. In this paper, we implement the adaptive switching between shallow and deep networks to reach the highest throughput on a resource-constrained MPSoC with CPU and FPGA. To this end, we develop and present a novel architecture for the CNNs where a gate makes the decision whether using the deeper model is beneficial or not. Due to resource limitation on FPGA, the idea of partial reconfiguration has been used to accommodate deep CNNs on the FPGA resources. We report experimental results on CIFAR-10, CIFAR-100, and SVHN datasets to validate our approach. Using confidence metric as the decision making factor, only 69.8%, 71.8%, and 43.8% of the computation in the deepest network is done for CIFAR-10, CIFAR-100, and SVHN while it can maintain the desired accuracy with the throughput of around 400 images per second for SVHN dataset. <https://github.com/mfarhadi/AHCNN>.

I. INTRODUCTION

Recently, Convolutional Neural Networks (CNNs)-based methods achieve great success in image classification [23] and object detection [30] tasks. The success leads the researchers to explore deeper models such as ResNet[16] (152 layers), and these models yield high recognition accuracy. The “secret” sauce of success for these deeper and deeper CNNs models are stacking repetitive layers and increasing the number of model parameters. This practice is possible while the applications are running in big data centers or infrastructures with high performance processing capabilities. However, these complex models are not suitable for real-time and embedded systems due to low energy constraints and limited computing resources.

The aforementioned concern triggers various approaches, such as by the alignment of memory and SIMD (Single instruction, multiple data) operations to boost matrix operations (93% Top-5 accuracy) [12], specific hardware (FPGA) solutions (86.66% Top-5 accuracy) [27], network compression (89.10% Top-5 accuracy) [13] or using cloud computing (network latency should be considered) [6]. These approaches indeed can reduce the energy consumption, but they fail to

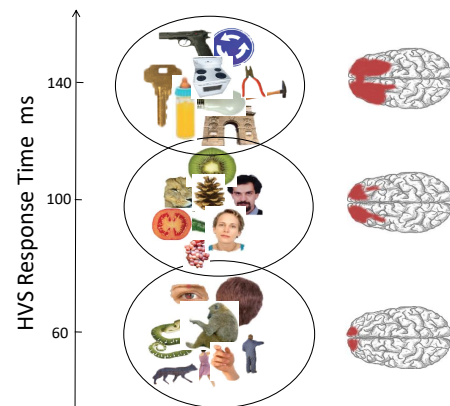


Fig. 1: Stimulus materials, fMRI brain coverage, and significant MEG-fMRI fusion results over different objects and background [8].

retain recognition accuracy while a system faces critical situations. In other words, they reduce the computation overload by trading a large chunk of recognition accuracy off from the state-of-the-art performances which is more than 96% Top-5 accuracy at the moment.

On the other hand, recent research suggests that the human vision system (HVS) has two stages for conducting visual classification: 1) a shallow primary stage and 2) a decision layer to pick a further processing pathway [31]. The study also supports the theory that the structure of the object representation in the HVS influences the decision layer during visual classification. Results from another research that is conducted in the field of neuroscience [7] showed that the response time of our HVS given an specific image as the stimuli differs a lot based on the category that the image belongs to. These results again suggest that HVS has a decision system which controls our processing resources assigned for each image. The study [8] shows the brain behavior corresponding to the images from different categories (in Fig. 1). From the fMRI imaging, researchers have speculation that for some input images, only a “shallow” part of our HVS is utilized while for other categories they invoke a “deeper” processing.

Following these insights and observations, we design and implement a feedback procedure to determine the early exit from the model. The core part of this procedure is an engine that accesses an image and predicts how accurate a certain model will perform. Our proposed model, besides classifying images, has an extra output which is designed to provide an evaluation on how well a model will perform. Then, our system relies on this evaluation to decide whether classifying this image with the deeper model will be beneficial or not. We implement a gate operation which takes the evaluation and it has an adjustable tolerance threshold for decision making. For example, in autonomous driving scenario, if the class “human” appears in the top 5 or 10 results from the shallow model, the system can adaptively decrease the decision layer threshold and yields a more accurate prediction using deeper models. Thus, our feedback procedure optimizes the resource usage by controlling the type and amount of images being sent to the deeper model.

In this paper, we report the implementation of the feedback procedure for determining the path of inference in the CNN based on the confidence level factor. The method has been implemented on an MPSoC (Pynq-Z1) with an ARM CPU and FPGA. We propose and implement the idea of partial reconfiguration in the FPGA to map the quantized CNN on the FPGA resources. We further demonstrate it using ResNet CNNs [16] on CIFAR-10, CIFAR-100, and SVHN datasets. The experimental results show that on average only 69.8%, 71.8%, and 43.8% of computation on the deepest network needs to be utilized for CIFAR-10, CIFAR-100, and SVHN benchmarking datasets, to maintain a comparably high recognition performance.

II. RELATED WORK

The last few years in the field of Deep Learning has laid the foundation for major advancements in visual recognition systems, ranging from object recognition [23], [17], action recognition [24], to scene recognition [41]. Significant improvements in recognition accuracies allow a wide range of science fiction ideas materialized, resulting in economic and societal benefits with AI applications such as autonomous vehicles [5], intelligent IoT systems [34], industrial robots, service robots and intelligent health care systems [29], [19].

The increasing number of real world applications require their corresponding visual recognition engine to not only recognize well, but also actively and effectively adjust its computational resources to handle the ever-changing physical world situations that the systems will face. The seminal work of the cascaded classification of Viola and Jones [37] represents the line of studies on cost-sensitive classification. The essence of their work is to treat classification as a cascaded process that contains control layers deciding the exit points where the system is confident in its current inference. Following the similar line of work, more recently, [25], [32] have proposed cascading CNNs structures to reduce the computational cost by reducing the structural complexities of CNNs.

Another thrust of work has focused on reducing the resource consumption of CNNs or other types of neural networks through various techniques of compressing the network structures [2], [14], [15], [28]. Network pruning is one of the well-studied approaches which removes unnecessary nodes and edges from network, to compress model and gaining inference speedup [14], [39], [18]. However, [15] pointed out that using the standard GPU implementation, the speedup is hard to achieve due to the lack of high degrees of exploitable regularity and computation intensity in the resulting network with sparse connections.

The use of adaptive structures is a relatively newer approach which decides how to further process the image [32], [42], [35], [3]. Teerapittayanon et al [35] proposed an adaptive model to allow early exit based on the entropy of model output which is called Branchy-Net. By adding sub-outputs to the model, Branchy-Net checks the entropy of model output and if the entropy is low enough, terminates the procedure. By doing this, Branchy-Net achieved 2x speedups at the inference time [35]. However, Branchy-Net spends a considerable amount of time to evaluate the early output [4]; it does not have a clear procedure to select the location of early branches, and it is changing the structure of original model to have an early exit. In response to the mentioned issue, Bollukbasi et al [4] proposed an adaptive method which adopts the Branchy-Net idea and stacks several models such as AlexNet [23] and ResNet [17]. This model still suffers from the overhead time of evaluating the model’s early output. Another study in [38], proposed a method based on a decision gate to skip some of the blocks in ResNet structure. The decision gates include convolution and fully connected layers which are trained using reinforcement learning. These decision gates are not suitable for shallow CNN models such as ResNet-18 [16].

The implementation of CNNs on FPGAs has been studied from the literature to certain extent. More specifically, *Binary-Eye* in [20] has presented an implementation of binary neural networks on FPGA. The presented implementation can be used in IoT and distributed systems where the stream of images for a camera needs to be processed. A framework called *FINN* has been also presented in [36] for the inference of binarized neural networks. The mentioned implementation does not adopt the partial reconfiguration to address the limitation of resources on FPGA.

Dynamic partial reconfiguration has been done in the relevant literature in [1], [21]. In [1], the authors have implemented the reconfiguration steps in a Zynq 7000 FPGA but do not implement CNN architectures. Dynamic reconfiguration has been done in [21] for the CNNs on the Pynq board. In the mentioned work, they have stated that the implementation of CNN using reconfiguration at each layer is expensive.

In this paper, we have implemented the idea of adaptive switching between shallow and deep networks on FPGA platform using partial reconfiguration to reduce the amount of needed computation. The confidence level was observed to be the most efficient factor to switch in comparison with the methods presented in Skip-Net [38] and Branchy-Net [35].

III. ADAPTIVE AND HIERARCHICAL CNNs

The key module of our proposed Adaptive and Hierarchical convolutional neural networks (AH-CNN) model is a feedback procedure which is designed to comprehensively evaluate the classification procedure. More specifically, AH-CNN consists of three parts: 1) a **shallow part** which is a light-weight CNN model; 2) a **decision layer** which evaluates shallow part's performance and makes a decision; and 3) a **deep part** which is a deep CNN with a high inference accuracy. As mentioned in Section I, the overall objective of our dynamic system is to obtain the highest possible recognition accuracy during critical time instances while maintaining a satisfiable performance using the shallow part during non-critical moments. Following this intuition, we put forward a mechanism with a combination of a shallow model, feedback procedure and a deep model, which has a flexible structure at the same time. This mechanism can achieve the same high recognition accuracy as other very deep networks by partially reconfiguring the hardware structure. Thus, an intelligent agent equipped with the AH-CNN can adaptively adjust its model structure to maintain a balance between the expected classification accuracy and the model complexity. This procedure can be applied repetitively and has several decision layers. In the following section, we will describe the details of the AH-CNN architecture.

A. AH-CNN Architecture

The authors in [40] showed that the preceding layers in deep neural networks respond to class-agnostic low-level features, while the rear layers extract more specific high-level features. Objects of certain categories can be classified solely by the low-level features but for the images of other categories, we need more specific high-level features, and deeper layers are needed to extract them. Thus, we design our architecture to have three modules: the shallow part, the deep part and a decision layer. Hence, the proposed AH-CNN with a design of an adaptive and hierarchical structure, can yield different behaviors based on the input image characteristics. We will describe the three mentioned modules in the following.

Shallow Part: In this work, the FPGA is loaded with the shallow part first. This part can be applied to the input tensor without any reconfiguration cost and classifies all input images and it outputs two results: 1) a predicted label $y = j$ and 2) a confidence value ($P(y = j|X_i) = \text{softmax}(z_j) = \exp(z_j) / \sum_k \exp(z_k)$, where z is the output of fully connected layer over the input image X_i) which will be later used in the feedback procedure.

Deep Part: This part is the next group of convolution layers which should be loaded on FPGA. Due to the transfer and configuration time, loading the new part on the FPGA is expensive. This group of convolution layers is responsible to extract more specific high-level features and detect the images which are misclassified by the shallow part. This part will be applied over the output of the last convolution in the shallow part to reach higher confidence.

Decision Layer: This part of AH-CNN takes the shallow part's outputs and makes a decision to whether activate the

deep part, or simply terminate further processing and take the shallow part's result as the overall model output. This layer has a feedback procedure to make the network behavior decision by evaluating the shallow part.

To this end, the decision layer currently yields a binary behavior based on three factors: 1) the confidence value from the shallow part; 2) the priority of the object classes; 3) the overall expected classification accuracy (which is obtained by validating the model over the data set). The binary behavior either activates the deep part or takes the shallow part's classification output as the overall model's output.

Algorithm 1 AH-CNN: Inference Phase

Require: Input image X_i , Desired accuracy Λ , Number of early branches N_i , High priority classes S_{HP} .

```

while  $X_i$  do
  while  $N_i$  do
    Assign proper  $\Gamma$  based on  $\Lambda$ 
     $\beta, ShOutput \leftarrow ForwardPropagate(X_i, Shallow)$ 
    if  $S_{HP}$  appear in ShOutput Top- $n$  then
       $\Gamma = \Gamma + \Theta$ 
    end if
    if  $\beta \leq \Gamma$  then
      Load deep part on FPGA
       $Output \leftarrow ForwardPropagate((ShOutput, Deep)$ 
    else
       $Output \leftarrow ShOutput$ 
    end if
  end while
end while

```

Algorithm 1 shows the AH-CNN processing procedure in the inference phase. The decision layer first checks the top- n classification results from the shallow part's classification vector. If a label from the high priority set (S_{HP}) exists, there is a higher probability that the input needs further processing. Next, the decision layer checks the current expected classification accuracy, which will affect the fraction of all the input images that need further processing. Finally, the model checks the shallow part's Confidence value. The interpretation of the confidence value yields a feedback procedure. The priority of the object classes and the overall expected classification accuracy are then considered to tune a threshold value to compare with the confidence value, which we refer to it as the trigger point later.

The most critical element of the feedback procedure of AH-CNN is the trigger point Γ . After feed-forwarding each image over the shallow part, the decision layer gets the confidence value β and compares it with the assigned threshold Γ . If β does not reach Γ , it means that the shallow part has less confidence than our system's tolerance over the input image and further processing is needed to gain a higher expected accuracy. As a consequence, the decision layer load and activates the deep part. The value of the trigger point can be actively adapted according to the real-world situations. In cases that we do not need a high accuracy, we can decrease the trigger point value. In cases that the member of S_{HP} appear in the top- n outputs, we can increase the trigger value (Γ) by Θ to expect a higher classification accuracy over that image. The trigger point makes our model innately adaptive.

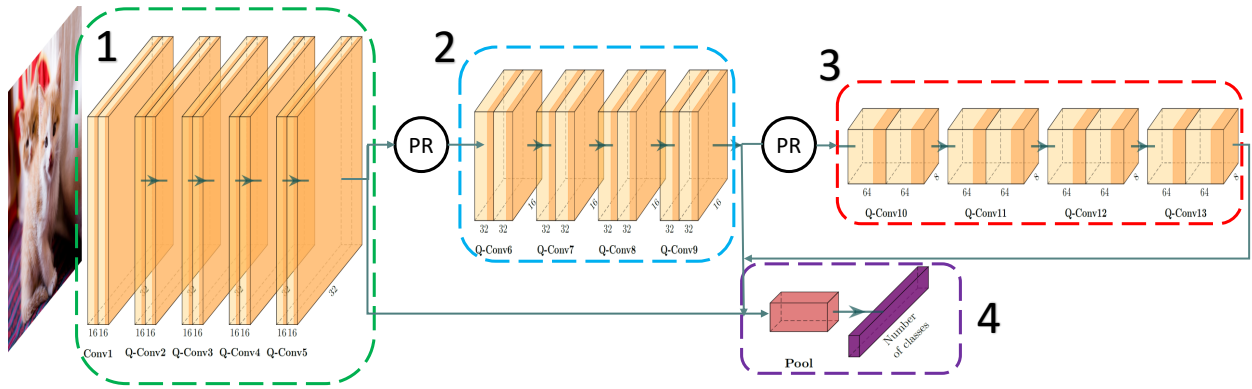


Fig. 2: The scheme of CNN implementation on FPGA using dynamic reconfiguration and adaptive feedback. The adaptive feedback makes the decision to classify the image or apply the next stack of convolution layers based on the output confidences computed from each part.

We discuss how to set a proper trigger point as well as its range in Section. V-B.

IV. IMPLEMENTATION ON FPGA

The overall scheme of implementation on the FPGA is depicted in Figure 2. The convolution layers in the CNN are based on the ResNet CNN structure. The whole CNN is divided into three parts which are numbered in the figure. The output of each part can be used as the input for the pooling layer in part 4. There is a Partial Reconfiguration unit labeled as *PR* which changes the bitstream file on the FPGA when necessary. The reason for partial reconfiguration is to save the LUT area on the FPGA and address the limitation of computational resources.

In order to implement the CNN on FPGA, a quantized version of CNN has been used which is popular in the FPGA community [36]. In this network, the weights are binary and the activation data are five bits (quantized bits). Even using this quantization method and binary values, an acceptable accuracy of classification can be obtained which is shown in the experimental result section.

Batch processing has been used to improve the overall throughput of the system. During batch processing, the re-configuration overhead of changing the bitstream files would be considered for all the images that are going to be processed in the network. Therefore, the overhead of reconfiguration would be negligible when calculating the inference time for one image on average.

V. TRAINING PHASE

Both the shallow and the deep part aim to classify images with the best possible performance that can be achieved individually. Consequently, the feedback procedure should not have any influence over the shallow part's classification performance. We train both the deep part and the shallow part using the stochastic gradient mini-batch [9]. Also, the mean and range of trigger point value are needed to be learned from the training data. In the following sections, we first introduce

the overall model learning procedure in Section. V-A, and then report our training details in Section. V-B.

A. Learning Procedure

In the first stage, all parts are trained jointly over training set S_T and validated over validation set S_V . In each epoch, the accuracy of all parts are evaluated over the validation. The model with the highest accuracy over the deepest part will be selected as the best model due to reaching the best possible accuracy at critical inference time.

Identifying the trigger point: Following the aforementioned design, the shallow part after feed-forwarding each input image has a confidence value over the output belief vector. To have an evaluation over this value and its range, we feed all images from S_T into shallow part and collect the confidence values. The calculated mean C_{Mean} and the standard deviation C_{Std} over these values are used to control the expected classification accuracy of the AH-CNN.

B. Model Training Details

Initializing: We first adopt the ResNet-18 model as the base model, where each of the blocks in this model is considered as a separate classification module. We added a pooling and a fully connected layer for each part. Xavier initialization [11] is used for having proper initial weights to propagate the signals precisely.

Defining the loss function: For a classification task, the cross entropy is mostly used as loss function. Here, we have several parts which get their input from previous layer and have independent classification layer output. Consequently, these parts should be trained jointly. The objective function can be formulated as

$$L(\hat{y}, y; \theta) = \sum_N L(\hat{y}_n, y; \theta),$$

where

$$L(\hat{y}, y_n; \theta) = -\frac{1}{\zeta_{S_T}} \sum_k y_n^k \log f(x_k; \theta),$$

and N denotes the total number of classification modules, x_k the input images, ζ the set of all possible labels and $f(\theta)$ denotes the whole model.

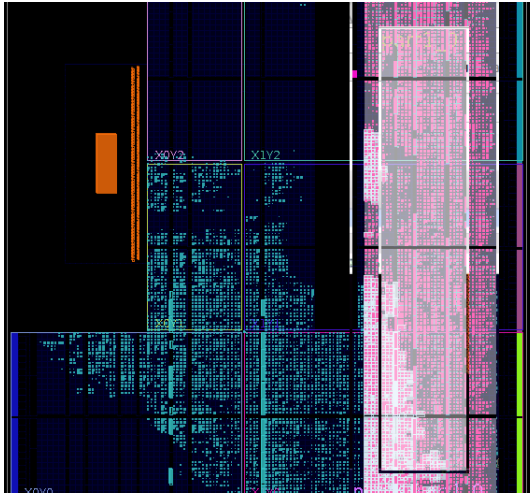


Fig. 3: Layout of the reconfigurable design.

VI. EXPERIMENTS

The theoretical framework we have presented suggests two hypotheses that deserve empirical tests: 1) AH-CNN can perform visual classification with much higher efficiency while maintaining the accuracy; and 2) Deep CNN models can be executed on a resource-constrained FPGA using partial reconfiguration. To validate these two hypotheses, we implement AH-CNN on Xilinx Zynq-7000 and evaluate on the CIFAR-10, CIFAR-100 [22] and SVHN [26] datasets. We implemented the AH-CNN as described in Sec. III where all convolution parts were implemented as separate hardware IP cores. We utilize Vivado HLS to synthesis the IP cores [10]. The training procedure was performed using the PyTorch framework.

A. Implementation

We select the PYNQ-Z1 to perform our evaluations. This board consists of a Xilinx Zynq-7000 ZC7020 and a dual-core ARM A9 processor. Images were loaded to our convolution IP cores through a Direct Memory Access (DMA) IP core.

We adopt the Resnet-18 [17] as the base model. Due to limited available LUTs on this board, the network was broken into three parts. All parts consist of a group of convolution layers, pooling and fully connected layer. To reduce the reconfiguration time, we remove the last pooling and fully connected layer and create a new part which will be shared. Figure 2 shows an overview of our model. Part 1 is the shallowest model of this architecture. Parts 2-3 are the deeper blocks for extracting more features. Part 4 is the common one among all. Table I shows the resources needed for each part and total available resources on FPGA.

	Part 1	Part 2	Part 3	Part 4	Total
BRAM	81	91	96	31	280
DSP	120	96	96	24	220
FF	15672	16647	34069	9908	106400

TABLE I: Available resources on the Zynq XC7Z020, in comparison to used resources by convolution parts.

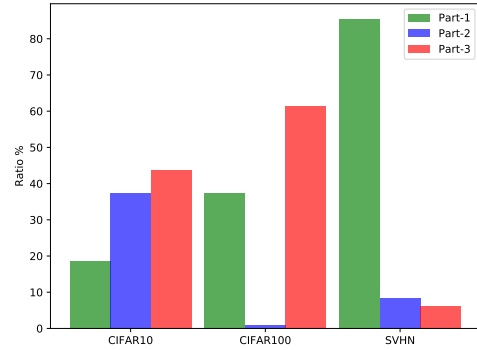


Fig. 4: The stop ratio for each part on CIFAR-10, CIFAR-100, and SVHN dataset.

As shown previously, the total hardware resources needed for the whole architecture is more than available resource over the target device. Moreover, there are shared modules over all convolution parts such as Part 4, DMA, etc. Consequently, we applied Dynamic Partial Reconfiguration in order to reduce the reconfiguration time by just changing the convolution parts and keeping the shared modules. Fig. 3 shows the layout of our implementation. The reconfigurable area is shown by purple and the fixed ports on the FPGA by white.

The resulting partial parts have all the same size of 2.4 MB and the size of main bitstream is 4 MB.

Training: The training part was carried out using PyTorch framework. We implemented special quantized convolution layer and fully connected layer with 1-bit weight and 5-bit activation. The initial learning rate is set to be 0.01 and it was decreased by a factor of 10 in every 20 epochs. Training continues until 100 epochs with a mini-batch size of 256.

Feedback Evaluation: The aforementioned procedure in section. V-B is followed to estimate the confidence value. The mean and the standard deviation of all the confidence values were achieved after the various parts were collected over S_T .

B. Overall Evaluation

We choose the CIFAR10, CIFAR-100, and SVHN validation sets in the overall AH-CNN model testing. Here, we evaluate the partial reconfiguration approach. Also, we compare three selection methods: 1) our proposed feedback procedure; 2) SkipNet method [38]; and 3) an entropy-based method [4].

Partial Reconfiguration: We have three accelerator IPs to reconfigure which are connected to the ARM processor through AXI interface, clocked at 100 MHz. The AXI channel and partial reconfiguration module is controlled by a Python script. We have also implemented a CPU version of AH-CNN architecture which runs on an ARM chip at 666 MHz. Table II shows the measurements of partial reconfiguration, FPGA and CPU execution time. As the reconfiguration region is same for all IPs, The reconfiguration time is always the same. By using batch processing (batch=512), the throughput of our system is ≈ 160 image per second while applying all parts to the images. This is 32 times faster than the CPU implementation.

Table III shows the accuracy that can be achieved by applying each IP of convolutions to the input stream. It is clear

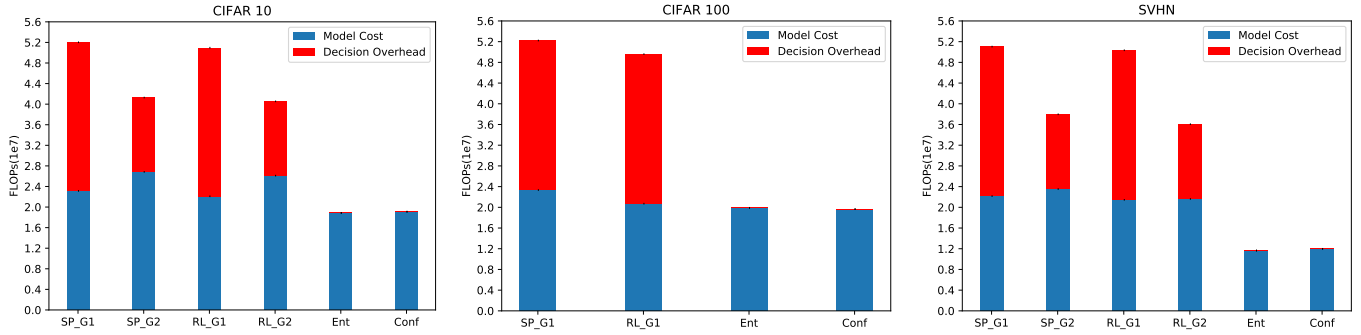


Fig. 5: Computation reduction of Entropy (Ent), Confidence (Conf), SkipNet+SP (SP) and SkipNet+HRL+SP (RL) with feed-forward gates (G1 has two convolution layers, G2 has one convolution layer) while preserving the full network accuracy. The computation cost includes the computation of decision method. We are able to reduce the computation costs by $\approx 30\%$, $\approx 27\%$ and $\approx 57\%$ on the CIFAR-10, CIFAR-100, and SVHN data using confidence decision method compared to the base model. Since the feed-forward gates are more expensive, SkipNet is not a suitable method for the scope of the study.

Bitstream	FPGA Config Time	FPGA Execution Time	CPU Execution Time	FLOPS
Part 1	38-42 ms	2 ms	98 ms	10.24M
Part 2	38-42 ms	2 ms	57 ms	8.6M
Part 3	38-42 ms	2 ms	49 ms	8.5M

TABLE II: Performance evaluation on different parts of the design.

	CIFAR10	CIFAR100		SVHN
		Top1	Top5	
Part 1	70.95	42.26	72.14	80.35
Part 2	80.57	52.23	80.25	91.24
Part 3	86.27	56.60	83.46	94.62

TABLE III: Top-1 accuracy of the HLS optimized IP-cores.

that the system can reach to the higher accuracy by extracting more feature using deeper layers. Also, a significant portion of images can be classified correctly without using deep layers.

Feedback Procedure: Initially, we explore the trigger point by collecting the confidence of each AH-CNN branch. AH-CNN model achieves 85.4%, 55.4%, 94.2% Top-1 validation accuracy over CIFAR10, CIFAR-100, and SVHN respectively. In Fig. 4, we also report the portion of images classified by each branch. Due to the simplicity of the feedback procedure, this method has the lowest overhead.

SkipNet [38]: In this method, instead of selecting images by our feedback procedure, decision layer selects images using a gate consisting of convolution and fully connected layers. We adopt two different gates and two training methods proposed by [38] to evaluate our method. These gates show desirable performance over large CNN models. However they do not have the same performance over models such as ResNet-18 or ResNet-38. For each decision, one or two convolution layers and a fully connected layer should be applied to the stream.

Entropy Selection [4]: This method uses the entropy of the shallow part's output to decide whether the input image needs further processing or not [4]. The work [4] implemented two

variants: two-stacked model (AlexNet [23] and -50 [17] and three-stacked model (AlexNet, GoogleNet [33] and ResNet-50). Due to calculating the entropy of the output vector at each branch, this method is more expensive than the feedback procedure.

Fig. 5 depicts the computation reduction by applying the different decision procedures. We observe that by just considering the confidence, the model outperforms the SkipNet gates. SkipNet gates not only are so expensive but also are not as successful as other methods in our case study. The confidence and entropy selection have the same results however the confidence method has less computation cost. The confidence selection method decreased the computation to 69.8%, 71.8%, 43.8% of the base model in CIFAR-10, CIFAR-100, and SVHN respectively. Also, the throughput of model reaches to 268, 217, and 408 images per second.

VII. CONCLUSION

In this paper, we proposed a new approach to run heavy neural networks on FPGAs with constrained resources. We stacked various shallow and deep models yielding an adaptive and hierarchical structure for quantized neural networks. We conducted experiments on CIFAR-10, CIFAR-100 and SVHN, and empirically validated that AH-CNN maintains a similarly low inference time as the shallow models while achieving the high recognition accuracy of the deep model on image classification tasks. The flexible nature of this hierarchical method makes it suitable for applications that need adaptive behavior towards dynamic priority change over object categories, such as an agent with active perception.

Acknowledgments: The National Science Foundation under the Robust Intelligence Program (1750082), and the IoT Innovation (I-square) fund provided by ASU Fulton Schools of Engineering are gratefully acknowledged. We also acknowledge NVIDIA and Xilinx for the donation of GPUs and FPGAs.

REFERENCES

- [1] M. Al Kadi, P. Rudolph, D. Gohringer, and M. Hubner. Dynamic and partial reconfiguration of zynq 7000 under linux. In *2013 International Conference on Reconfigurable Computing and FPGAs (ReConFig)*, pages 1–5. IEEE, 2013.
- [2] J. Ba and R. Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662, 2014.
- [3] E. Bengio, P.-L. Bacon, J. Pineau, and D. Precup. Conditional computation in neural networks for faster models. *arXiv preprint arXiv:1511.06297*, 2015.
- [4] T. Bolukbasi, J. Wang, O. Dekel, and V. Saligrama. Adaptive neural networks for efficient inference. In *International Conference on Machine Learning*, pages 527–536, 2017.
- [5] C. Chen, A. Seff, A. Kornhauser, and J. Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2722–2730, 2015.
- [6] T. Y.-h. Chen. Glimpse : Continuous , Real-Time Object Recognition on Mobile Devices Categories and Subject Descriptors. *SenSys '15 Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pages 155–168, 2015.
- [7] R. M. Cichy, D. Pantazis, and A. Oliva. Resolving human object recognition in space and time. *Nature Neuroscience*, 17(3):455–462, 2014.
- [8] R. M. Cichy, D. Pantazis, and A. Oliva. Similarity-based fusion of meg and fmri reveals spatio-temporal dynamics in human cortex during visual object recognition. *Cerebral Cortex*, 26(8):3563–3579, 2016.
- [9] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le, et al. Large scale distributed deep networks. In *Advances in neural information processing systems*, pages 1223–1231, 2012.
- [10] T. Feist. Vivado design suite. 2012.
- [11] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.
- [12] Y. Gong, L. Liu, M. Yang, and L. Bourdev. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*, 2014.
- [13] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.
- [14] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In *Proceedings of the International Conference on Learning Representations*, 2016.
- [15] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, pages 1135–1143, 2015.
- [16] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
- [17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.
- [18] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.
- [19] M. Izadyazdanabadi, E. Belykh, M. Mooney, N. Martirosyan, J. Eschbacher, P. Nakaji, M. C. Preul, and Y. Yang. Convolutional neural networks: Ensemble modeling, fine-tuning and unsupervised semantic localization. *arXiv preprint arXiv:1709.03028*, 2017.
- [20] P. Jokic, S. Emery, and L. Benini. Binaryeye: A 20 kfps streaming camera system on fpga with real-time on-device image recognition using binary neural networks. In *2018 IEEE 13th International Symposium on Industrial Embedded Systems (SIES)*, pages 1–7. IEEE, 2018.
- [21] F. Kästner, B. Janßen, F. Kautz, M. Hübner, and G. Corradi. Hardware/software codesign for convolutional neural networks exploiting dynamic partial reconfiguration on pynq. In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 154–161. IEEE, 2018.
- [22] A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.
- [23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, pages 1–9, 2012.
- [24] C. Lea, R. Vidal, and G. D. Hager. Learning convolutional action primitives for fine-grained action recognition. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1642–1649. IEEE, 2016.
- [25] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua. A convolutional neural network cascade for face detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5325–5334, 2015.
- [26] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011.
- [27] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song, et al. Going deeper with embedded fpga platform for convolutional neural network. In *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 26–35. ACM, 2016.
- [28] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.
- [29] D. Ravi, C. Wong, F. Deligianni, M. Berthelot, J. Andreu-Perez, B. Lo, and G.-Z. Yang. Deep learning for health informatics. *IEEE journal of biomedical and health informatics*, 21(1):4–21, 2017.
- [30] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016.
- [31] J. B. Ritchie, D. A. Tovar, and T. A. Carlson. Emerging Object Representations in the Visual System Predict Reaction Times for Categorization. *PLoS Computational Biology*, 11(6):1–18, 2015.
- [32] H. Shen, S. Han, M. Philipose, and A. Krishnamurthy. Fast video classification via adaptive cascading of deep models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [33] 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 the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.
- [34] J. Tang, D. Sun, S. Liu, and J.-L. Gaudiot. Enabling deep learning on iot devices. *Computer*, 50(10):92–96, 2017.
- [35] S. Teerapittayanon, B. McDanel, and H. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pages 2464–2469. IEEE, 2016.
- [36] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers. Finn: A framework for fast, scalable binarized neural network inference. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 65–74. ACM, 2017.
- [37] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages 1–I. IEEE, 2001.
- [38] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 409–424, 2018.
- [39] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2074–2082, 2016.
- [40] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8689 LNCS(PART 1):818–833, 2014.
- [41] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495, 2014.
- [42] H.-Y. Zhou, B.-B. Gao, and J. Wu. Adaptive feeding: Achieving fast and accurate detections by adaptively combining object detectors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3505–3513, 2017.