ICNN: An Iterative Implementation of Convolutional Neural Networks to Enable Energy and Computational Complexity Aware Dynamic Approximation

Katayoun Neshatpour, Farnaz Behnia, Houman Homayoun, Avesta Sasan
Department of Computer and Electrical Engineering, George Mason University,
{kneshatp, fbehnia, hhomayou, asasan} @gmu.edu

***Abstract*—With Convolutional Neural Networks (CNN) becoming more of a commodity in the computer vision field, many have attempted to improve CNN in a bid to achieve better accuracy to a point that CNN accuracies have surpassed that of human's capabilities. However, with deeper networks, the number of computations and consequently the power needed per classification has grown considerably. In this paper, we propose Iterative CNN (ICNN) by reformulating the CNN from a single feed-forward network to a series of sequentially executed smaller networks. Each smaller network processes a sub-sample of input image, and features extracted from previous network, and enhances the classification accuracy. Upon reaching an acceptable classification confidence, ICNN immediately terminates. The proposed network architecture allows the CNN function to be dynamically approximated by creating the possibility of early termination and performing the classification with far fewer operations compared to a conventional CNN. Our results show that this iterative approach competes with the original larger networks in terms of accuracy while incurring far less computational complexity by detecting many images in early iterations.**

I. Introduction

Computer visions detection and prediction accuracy, credited to recent developments in the design of deep and modern Convolutional Neural Networks (CNN), and processing power provided by Graphical Processing Units (GPU) for training them, has improved significantly. However, many Neural Network algorithms and CNN as a part of this family, due to their deep networks and dense connectivity, are computationally intensive. For example, AlexNet [1], a CNN architecture that won the 2012 ImageNet visual recognition challenge, contains 650K neurons and 60M parameters which demand computational performance in the order of 0.8G-1.0G Floating Point Operations (FLOP)s per classification. Next generations of vision-based CNN algorithms have further improved the prediction accuracy; however, this is achieved via even deeper networks. VGG [2], GoogleNet [3] and ResNet [4] have improved the prediction accuracy via increasing the CNN depth from 8 in AlexNet to 19, 22, and 152 layers respectively, but still keeping the CNN a computationally intensive and power hungry solution.

Albeit higher performance requirements, there is a need to aggressively reduce the power consumption of these solutions as many desired platforms for vision-based applications are energy constrained. Adopting complex vision algorithms in many of mobile and hand-held, embedded systems and IoT applications will not be feasible if energy consumption barrier is not addressed. At the same time, many of the desired applications require real-time and short latency responses. Therefore, the optimization space involves Accuracy, Latency, Power and Area (ALPA). With this in mind, we propose a radically different approach from modern and deep CNN models; we reformulate the learning from a single feed-forward network to a series of smaller networks that are executed iteratively.

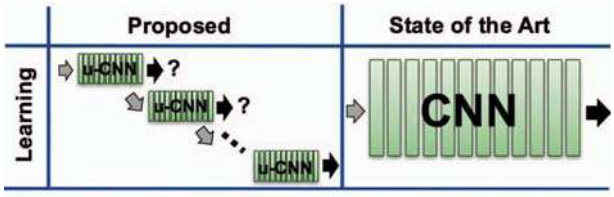


Fig. 1: ***Reformulating the CNN into an iterative solution***

Figure 1 illustrates a high-level abstraction of the proposed iterative CNN (ICNN). With iterative learning, each iteration processes a small set of sub-sampled input features and enhances the accuracy of the classification. The proposed ICNN model removes the need for a large neural network and constructs a learning model based on iterative execution of substantially smaller networks. In each iteration, by combining the processing results of the previous iteration with new features extracted from the sub-sampled input image, ICNNs classification accuracy is refined.

While CNN, as well as all other deep-learning networks, are inherently approximate, ICNN further exploits this approximate nature to reduce the large computational load of such networks with negligible effect on the performance. The proposed learning model improves the energy-efficiency of CNN processing by lowering the overall computational complexity, and by allowing early-termination (upon reaching a satisfactory confidence threshold). In addition, it prepares a fast initial classification and supports dynamic deadline-driven scheduling (by making the best classification decision within the available time budget) for real-time applications. Lower energy consumption, lower complexity and inherent support for deadline-driven applications make ICNN an attractive solution for resource-constrained and real-time devices.

II. Background

CNNs are constructed from multiple computational layers formed as Directed Acyclic Graph (DAG) [5], [6]. Each layer extracts an abstraction of data provided in the previous layer, called a feature map (fmap). Most common layers are Pooling (POOL), Convolution (CONV), and Fully Connected (FC).

978-3-9819263-0-9/DATE18/©2018 EDAA

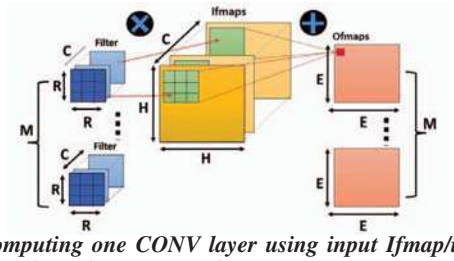


Fig. 2: ***Computing one CONV layer using input Ifmap/image and filters to produce the output (Ofmaps)***

In CONV layers, as illustrated in Figure 2, 2-D filters slide over the input images/feature-maps (*Ifmaps*) performing convolution operation to extract feature characteristics from local regions and generate output images/feature-maps (*Ofmaps*). Equation 1 explains how a CONV layer is computed, while equation 2 defines the constraints on used parameters.

$$Of[z][u][x][y] = Bias[u]+$$

$$\sum_{k=0}^{C-1}\sum_{i=0}^{R-1}\sum_{j=0}^{R-1} If[z][k][Ux+i][Uy+j] \times W[u][k][i][j] \quad (1)$$

$$0 \leq z \leq N; 0 \leq u \leq M; 0 \leq x, y \leq E; E = (H - R + U)/U \quad (2)$$

In this equations, Of and IF are *Ofmap* and *ifmap*, respectively. W and $Biaa$ are the filter weights and bias. N is the batch size of input images. The rest of parameters used in these equations are explained in Figure 2. Each filter generates a new *Ofmap* adding to the features extracted from the *Ifmaps*. Computation of CONV layer in popular CNNs accounts for more than 90% of the overall operations and requires a large amount of data movement and memory operations [7]. In addition, the large size of *Ifmap*, *Ofmap* and partial results that are generated during the CONV processing, increases the memory requirements for these architectures. After every CONV layer, a non-linear operation is applied to each *Ofmap* pixel to introduce non-linearity in the network. For example, Rectified Linear Unit (ReLU) is an operator that replaces all negative pixel values by zero. Other non-linear functions include *Tanh* and *Sigmoid* operators.

POOL layers perform down-sampling along the spatial dimensions of *Ifmaps* by partitioning them into a set of sub-regions and combining the values in each sub-region into a single value. Max-pooling and average-pooling are examples of POOL operators which use the maximum and the average values for each sub-region respectively. Next, FC layers combine all the neurons in the previous layer and connect them to every single neuron in the next layer. The outputs from the CONV and POOL layers represent high-level features of the input image, and FC layers fuse these features to generate a relational representation of these features with respect to each class in the classifier detection set. Finally, a Softmax classifier uses the outputs of the last FC layer to produce normalized class probabilities for various classes. Softmax classifier is a multi-class version of the binary logistic regression classifier, which produces un-normalized log probabilities for each class using cross-entropy loss.

III. Related Work

AlexNet [1] is one of the first works that deployed a deep convolutional neural network for image recognition and won ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) in 2012. The ILSVRC [8] is an object detection and image classification competition, which includes classification of images into 1000 different classes by training on 1.2 million labeled images. AlexNet architecture consists of 5 convolutional layers (with filter sizes of 11×11, 5×5 and 3×3), 3 fully connected layers. To enhance the accuracy, during the testing phase, AlexNet re-sizes each image to $3 \times 256 \times 256$ and takes five $3 \times 224 \times 224$ crops from the images (four corners and one center). Using the 5 crops and their horizontal reflections, the predictions of the Softmax layer for the ten images are averaged to yield the results. This technique increases the top-5 accuracy of AlexNet from 80% to 84%. With modifications to AlexNet (using 7×7 filter size instead of 11×11), ZF Net [9] wins the ILSVRC in 2013 with a top-5 accuracy of 88.8%.

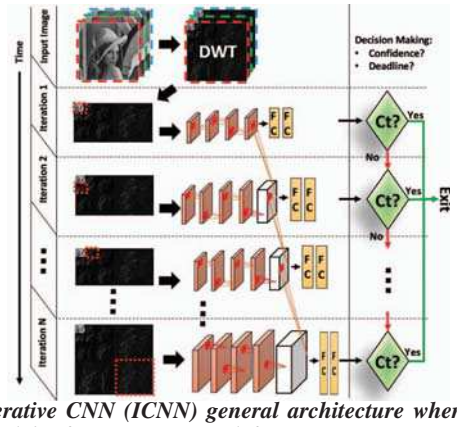


Fig. 3: ***Iterative CNN (ICNN) general architecture where each u-CNN is fed by features extracted from its previous u-CNN, and a DW sub-band generated from DWT transformation of the input image. Classification accuracy is checked at the end of each uCNN, based on which either ICNN is terminated, or next u-CNN is invoked.***

VGG [2] advocates the idea that going deeper with CNNs increases the accuracy [10]. It is proven that the effective receptive fields of 2 and 3 back-to-back convolutional layers with filter sizes of 3×3 are equivalent to the receptive field of convolutional layers with filter sizes of 5×5 and 7×7, respectively. With this idea, and by using 16 weight layers, VGG-16 achieves a top-5 accuracy of 92.5% by stacking 13 CONV layers with filter sizes of 3×3, and 3 FC layers. Rather than going deeper, GoogLeNet [3] introduces inception layer, in which, pooling layers and multiple convolutional layers with different kernel sizes process the same input. All the outputs are then concatenated allowing the model to take advantage of multi-level feature extraction from each input. For instance, it extracts general (3×3 and 5×5) and local (1×1) features at the same time. With 22 weight layers, GoogLeNet achieves a top-5 accuracy of 93.3% and wins ILSVRC 2014.

Finally, Microsoft ResNet [4] uses residual blocks in which, each input goes through a series of CONV-ReLu-CONV layers before being added to itself. The formulation of residual blocks is realized through shortcut connections [11], [12], which allows features to skip one or more CONV layer and be combined with other features at a later stage in the network. The authors show that these residual networks

are easier to optimize and they considerably benefit from increased depth. With 152 layers, ResNet wins the ILSVRC 2015 with a top-5 accuracy of 96.4%, the highest accuracy reported for the ImageNet challenge.

An overall analysis of the existing CNN architectures shows that the number of layers and the complexity of the CNNs have dramatically increased over time to enhance the accuracy. However, not all images need to go through such complex networks to yield satisfactory classification results. In this work, we propose an iterative architectural solution that breaks the large AlexNet CNN network into a sequence of smaller networks. Each smaller CNN, which is referred as Micro-CNN (u-CNN), processes a sub-sample of the input image and only proceeds to the next u-CNN stage if the classification confidence remains below a predefined threshold. This allows us to achieve a considerable reduction in computational complexity, and provide us with a route to terminate the process early if the classification under test reaches the desired confidence threshold upon termination of each u-CNN. For popularity and simplicity of AlexNet, we demonstrate our solution on this network. However, similar reformulation is applicable to other CNN architectures.

IV. Iterative Learning

State of the art DAG-based CNN networks are composed of a single feed-forward computational network, where the prediction is given and its confidence is determined after performing all necessary computations. This conventional model of learning has little or no regard for energy or power saving and is purely focused on improving the detection rate and the classification confidence. Our proposed reformulation is driven by the needs of resource-constrained vision applications for lowering the energy consumption and shortening the classification latency when deploying CNN solutions. In the proposed solution, a large CNN block is decomposed into many smaller networks (u-CNN in Figure 1), allowing iterative refinement and greater control over the execution of algorithm. Thus, not all images pass through all the u-CNNs; By monitoring the successive execution of u-CNN networks, a thresholding mechanism decides when to terminate the forward u-CNN traversal based on the current classification confidence of the images.

The proposed solution requires sub-sampling of input images into various sets for various rounds of computation. We propose the application of *Discrete Wavelet* sampling to decompose an input image into various input sets (sub-bands). The learning is then initiated using the first sub-sampled input set. Upon completion of first computational round (first u-CNN), the classification confidence is tested. If the confidence is unsatisfactory, it could be progressively increased by working on additional input samples (chosen from remaining sub-bands). Discrete Wavelet Transformation (DWT) provides the proposed learning algorithm with an attractive start point and unlike Fourier transform, in addition to frequency information, it also preserves temporal information of an image[13]. However, note that other sampling mechanisms could also be used for ICNN.

A high-level representation of envisioned iterative learning algorithm fed by DWT is illustrated in Figure 3. Each

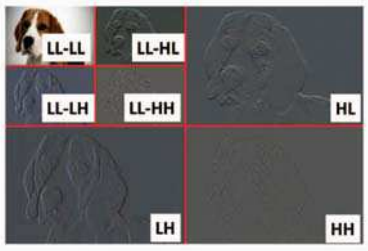


Fig. 4: ***sub-bands generated from a 2-level 2-dimensional Haar Discrete Wavelet Transformation (DWT) of an input image.***

iteration is a u-CNN, which takes a new DWT sub-band as its input and refines the confidence of learning network. DWT, being a convolutional filter, could be readily computed using processing elements (PE) in CNN processing engine of interest, or could be provided directly to CNN.

The iterative transformation of learning algorithm has many advantages: It could be terminated as soon as a u-CNN produces the desired confidence level. Further iterations could be avoided if the first u-CNN detection confidence is below a certain threshold signifying no contextually significant input. And confidence could be improved by moving to the next iteration, if the current measure of confidence remains between demarcated thresholds, aiding the rise or decline of classification confidence.

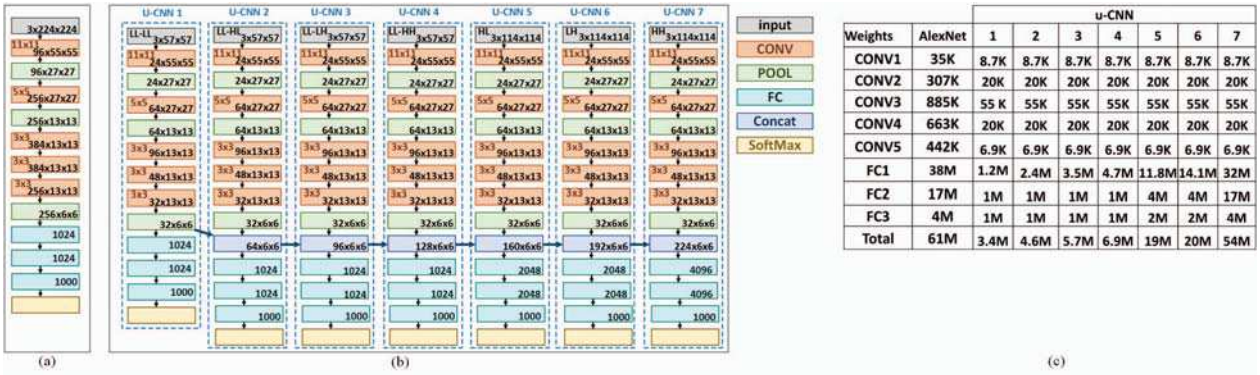
V. Training ICNN

In order to train the ICNN network, we deploy a top-down approach, in which, most of the ICNN training is done in one step. To achieve this, we train the last iteration of ICNN by initializing the weights for all the CONV layers in all u-CNNs and only the FC layers in the last iteration from a Gaussian distribution with zero mean and a standard deviation of 0.01. The training was started with a learning rate of 0.01. The learning rate was reduced by $2\times$ every 20-epoch until the learning rate was as low as 10^{-6} (by one-epoch, we refer to one pass of all the 1.2 million images in ImageNet; however, for data augmentation purposes, every few epochs the order of the images was modified, the image crops were altered and horizontal mirror of the images were utilized). Using a single GPU, this process takes 7-9 days.

To train the FC layers of earlier iterations, the weights in the CONV layers of these networks were initialized to those computed in the last iteration. Then, to keep these weights constant, the learning rate of them was set to zero. This allowed us to only train the FC layers of early iterations. The time required for the training of these FC layers ranged from a couple of hours for the 1-th u-CNN, to a day for the 6-th u-CNN (for an ICNN using 2-level DWT and thus 7 iterations). Please note that it is also possible to train u-CNNs sequentially to improve the accuracy of each u-CNN. This training scheme will be further explored in our future work.

VI. Implementation framework

The proposed ICNN was implemented to build a 1000-class image classifier for the ImageNet dataset. The ImageNet training and the validation archives include 1.2 million and 50K labeled RGB images respectively. The images are of varied spatial dimensions and labeled into 1K different classes.



		u-CNN						
Weights	AlexNet	1	2	3	4	5	6	7
CONV1	35K	8.7K	8.7K	8.7K	8.7K	8.7K	8.7K	8.7K
CONV2	307K	20K	20K	20K	20K	20K	20K	20K
CONV3	885K	55 K	55K	55K	55K	55K	55K	55K
CONV4	663K	20K	20K	20K	20K	20K	20K	20K
CONV5	442K	6.9K	6.9K	6.9K	6.9K	6.9K	6.9K	6.9K
FC1	38M	1.2M	2.4M	3.5M	4.7M	11.8M	14.1M	32M
FC2	17M	1M	1M	1M	1M	4M	4M	17M
FC3	4M	1M	1M	1M	1M	2M	2M	4M
Total	61M	3.4M	4.6M	5.7M	6.9M	19M	20M	54M

Fig. 5: ***The architecture of (a) original CNN (b) iterative CNN with 7 iterations. (The numbers in the boxes on left show the filter sizes for CONV layers and the numbers on right show the number and size of Ofmaps) (c) Comparing the number of required parameters for executing each CONV and FC layer in ICNN and AlexNet***

A common problem associated with training of complex CNN architectures is over-fitting. Over-fitting happens when the model fits too well to the training set, making it difficult for the model to generalize to new examples that were not in the training set. One of the solutions to the over-fitting problem is to increase the size of training data. For this purpose, following the methodology in [3], the horizontal mirror of each training image was added to the training set. Then, all training images were reshaped to $3 \times 256 \times 256$ and various $3 \times 228 \times 228$ crops were taken from each image.

Subsequently, a 2-level 2-dimensional DWT was captured for each channel of the RGB images through the Haar filter [14]. The resulting sub-bands are four RGB images of size $3 \times 57 \times 57$ in smaller sub-bands (corresponding to LL-LL, LL-LH, LL-HL, LL-HH), and three RGB images of $3 \times 114 \times 114$ in larger sub-bands (corresponding to HL, LH and HH). The sub-bands' naming convention is captured in Figure 4. The 2-level DWT is composed of 7 sub-bands allowing us to sample the input image 7 times and to build the ICNN with 7 iterations. To increase the number of iterations, the DWT depth can be increased or alternatively, DWT can be applied the other sub-bands (LH, HL, HH).

The sub-bands are stored in a lightning memory-mapped database (lmdb) format [15], which is the database of choice when using large datasets. For the pre-processing and storage of images, several Python package libraries, including lmdb, PyWavelets,OpenCV, were used. Moreover, a mean image was calculated for each sub-band based on the training data. The mean images are subtracted from each image for all the sub-bands to ensure a zero mean for every feature pixel.

AlexNet, was modified to be iterative and implemented in Caffe [16], a deep learning framework developed by Berkeley AI Research (BAIR). To train the network, Tesla K80 GPUs were deployed. Figure 5 shows the decomposition and reformulation of AlexNet into its ICNN representation. The iterative AlexNet includes 7 iterations. Each Concat layer fuses the *Ofmaps* of the last CONV layer in current iteration with the *Ofmaps* of the last CONV layer from the previous iteration. It should be noted that the number of *Ofmaps* which are processed at any given CONV layer in each u-CNN is considerably smaller than that of original AlexNet. Hence, the computational complexity of each u-CNN is considerably smaller than that of AlexNet.

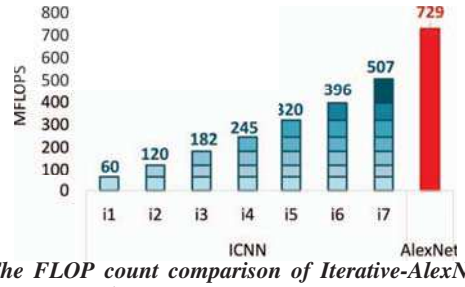
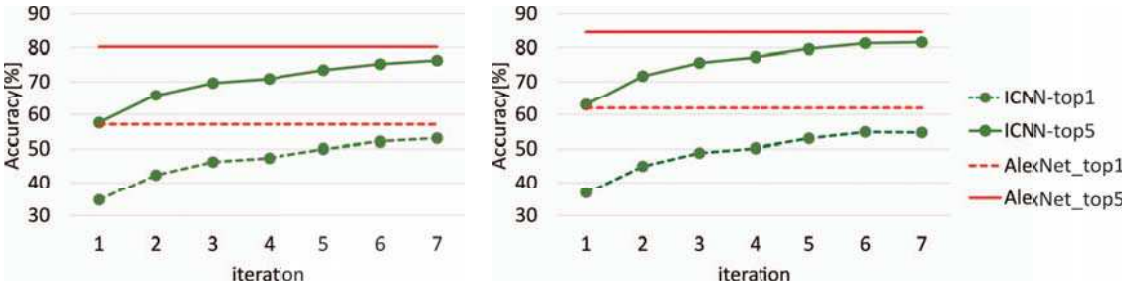


Fig. 6: ***The FLOP count comparison of Iterative-AlexNet (ICNN representation) vs AlexNet***

VII. Complexity Analysis

As mentioned in the previous section, the number of *Ofmaps* in each u-CNN is considerably reduced with respect to the original CNN. Figure 6 shows the number of Floating Point Operations (FLOP) for a forward pass of the iterative AlexNet per image at each iteration. The figure shows that, even if executed to the last iteration, ICNN still has a lower computational complexity (needing 30% fewer FLOPs) than the original AlexNet. On top of this, many images are detected at earlier iterations, removing the need to process subsequent u-CNNs. This further reduces the total FLOP count for a large number of images. More specifically, images detected in iterations 1, 2, 3, 4, 5 and 6 respectively require $12.2\times$, $6.1\times$, $4\times$, $3\times$, $2.3\times$ and $1.8\times$ fewer FLOPs when compared to the original AlexNet.

As stated previously, the computation intensive layers in AlexNet are CONV layers, and ICNN considerably reduces the required FLOP count. However, when it comes to the parameter count, in AlexNet, the largest number of parameters are associated with FC layers. This problem, as illustrated in Figure 5-c is exacerbated for the Iterative AlexNet, where FC layers are repeated for each u-CNN. Although the input *Ifmaps* to the FC layers in each u-CNN is smaller than that of AlexNet, repetition of FC layers increases the number of parameters for the iterative representation of AlexNet. To remedy this, the size of FC layers in u-CNNs could be reduced to smaller sizes (e.g. 1024 or 2048) at the expense of lower classification accuracy in these iterations. In addition, the size of parameters in these layers could be reduced to have a fewer number of bits. Note that this is an AlexNet model-specific problem and ICNN representation of other popular networks could experience significant reduction in both parameter and FLOP counts. For example, ICNN model



(a) (b)

Fig. 7: ***The top-1 and top-5 accuracy of AlexNet vs. Iterative AlexNet using (a) original image re-sized to*** $\mathbf{3 \times 228 \times 228}$ ***(b) the average of ten*** $\mathbf{3 \times 228 \times 228}$ ***crops from images re-sized to*** $\mathbf{3 \times 256 \times 256}$

of ResNet will require far fewer parameters for the final classification as it only has a single FC layer.

Additionally, for resource-contained applications not all the parameters of the network are kept in the memory. Assuming that the processing unit only keeps the parameters needed for the current CONV or FC layer in the memory, the maximum memory footprint for keeping network parameters in Iterative-AlexNet is reduced by 12%.

VIII. Results

For testing, the 50K images in the validation set of ImageNet were classified and the top-1 and top-5 scores are reported in Figure 7. The top-1 score is the percentage of the test images for which, the classifier gives the correct class the highest probability. The top-5 score is the percentage of the test images for which, the classifier includes the correct class among its top 5 guesses. To fit the test images into the network we re-sized them to $3 \times 228 \times 228$. Figure 7-a shows the results for re-sized images. Alternatively, to enhance the accuracy, a similar approach as in [1] was deployed. The images were re-sized to $3 \times 256 \times 256$ and five $3 \times 228 \times 228$ crops from the images (four corners and one center) were taken, as well as their horizontal reflections (ten crops in total), and the predictions of the Softmax layer for the ten crops was averaged. The results are illustrated in Figure 7-b. The figure shows that this approach increases the top-5 scores by up to 4%. Moreover, the top-5 accuracy of ICNN approaches that of AlexNet in the final iterations.

While the accumulated FLOP count of all iterations is lower than AlexNet, the decrease in the top-5 score of the last iteration is about 2%. For effectively using the proposed iterative approach and making intelligent termination decisions, we need to set confidence goals for the classification. For instance, while iteration 1 yields an accumulated probability of 80% for the top-5 class for a test image, it might yield a probability of 40% for the top-5 class of another test image. Needless to say, the prediction confidence for the first image is significantly higher than the second image. ICNN, in this case, terminates the CNN processing after the first iteration for the first image and moves forward to the next iteration for the second image.

To exploit the iterative CNN for reducing computational complexity, for each image in the validation set (50K images), ICNN starts the classification with the first iteration and calculates the classification confidence. Classification confidence is calculated by adding up the probabilities of the top-5 classes. Subsequently, the classification confidence is compared to the desired Confidence Threshold (C_t), and if lower, ICNN moves forward to the next iteration. ICNN continues this process until the classification confidence is higher that C_t. In this approach, various images are detected in different iterations. Moreover, some of the images never reach a classification confidence above C_t. For these images, the results of the last iteration are used.

Figure 8 shows the number of images classified in each iteration and their top-5 score for various C_t values. Since ICNN is proposed to reduce the computational complexity of CNN, only one image was used for classification (rather than averaging 10 different crops). In Figure 8, the newly detected images in each iteration and low-confidence images detected in the last iteration are depicted under the labels *new* and *remaining*, respectively. For the *remaining* images, the top-5 classification accuracy is significantly lower than those with confidence values higher than the threshold. With increasing values of C_t the number of these *remaining* images increases, pushing the classifier toward the last iteration and thus increasing the total FLOP count.

Figure 8 shows that with each new iteration, the ICNN detects more images with high classification confidence. This is to be expected, as the last iterations combine a larger number of features in more complex architectures with a larger number of parameters, allowing a more accurate prediction model and thus classifying more images.

Moreover, by increasing the value of C_t, the number of images classified in early iterations decreases; however the classification accuracy (correct label within top-5) increases. In Figure 8 this is illustrated by comparing the difference in the heights of *Top-5* and *Detected* bars at each iteration, where a larger delta means larger miss-classification. More specifically, higher values of C_t enhances the accuracy at the expense of larger computation complexity, and lower C_t values reduce the complexity at the expense of lower classification accuracy. Thus, an intelligent selection of C_t maintains a trade-off between accuracy and computational complexity.

Note that the C_t doesn't have to be a fixed value across different u-CNN iterations and could be different for each iteration. In addition, it could be tuned at run-time to dynamically control the trade-off between accuracy and computational complexity. Access to such run-time control knob is extremely desirable and is a new concept in CNN

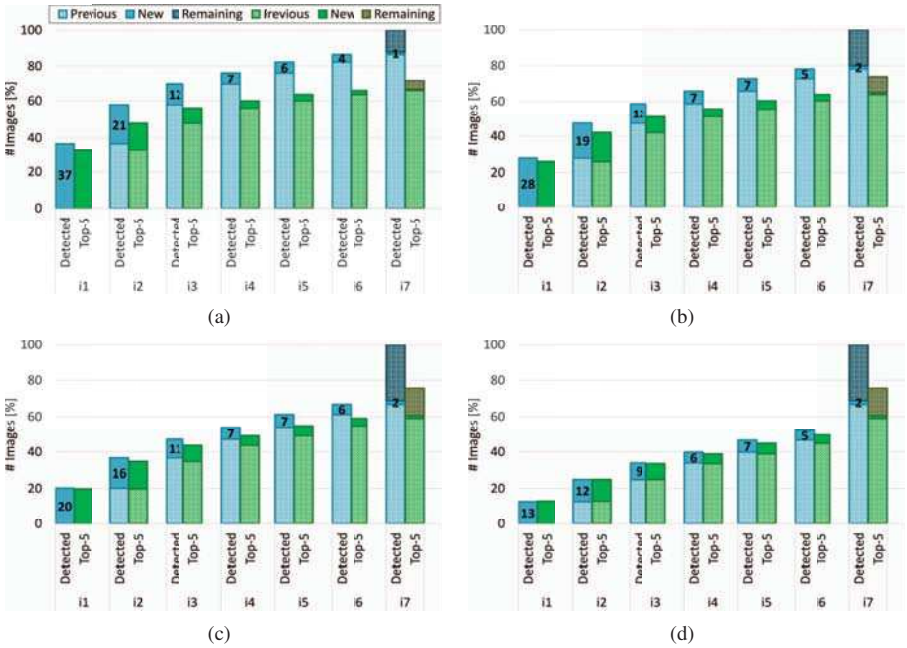


Fig. 8: ***Number of classified images and the number of classified images with a correct label in the top-5 probabilities for various confidence threshold (C_t) (a) $C_t = 0.6$ (b) $C_t = 0.7$ (a) $C_t = 0.8$ (d) $C_t = 0.9$***

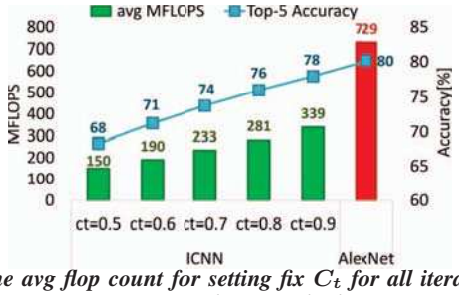


Fig. 9: ***The avg flop count for setting fix C_t for all iterations and detecting remaining images with original AlexNet***

networks which is made possible by ICNN. Figure 9 illustrates this trade-off where the overall accuracy of ICNN, as well as the average number of FLOPs required to process the 50K images in the validation-set changes with the selection of C_t values (C_t is fixed across all u-CNN layers). Interestingly, with a fixed confidence threshold of 0.9, the overall accuracy is the same as using the data from all the iterations to process all images (see Figure 7) while requiring only half the FLOPs. This trade-off and its applications will be studied and explained further in our future work.

IX. Conclusions

In this work, we proposed a novel iterative architectural solution (ICNN) that breaks a large CNN network (AlexNet) into a sequence of smaller (u-CNN) networks. Each smaller u-CNN processes a sub-sample of the input image and only proceeds to the next u-CNN stage if its classification confidence remains below a predefined threshold. Depending on how far the ICNN is executed (which u-CNN stage generates the final classification that satisfies the desired confidence threshold), the FLOP count required for classification is reduced between $2.8\times$ to $12.2\times$. Reduction in computational complexity results in a similar reduction in the classification time and energy required for classification. The lower energy consumption, lower computational complexity, inherent support for real-time deadline-driven applications, and the ability to dynamically tradeoff accuracy versus complexity by application of desired thresholding policies, makes the ICNN an attractive solution for resource-constrained mobile and real-time devices.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 1097–1105.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[3] C. Szegedy and et al., "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[4] K. He and et al., "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[5] M. Liu and et al., "Towards better analysis of deep convolutional neural networks," *IEEE transactions on visualization and computer graphics*, vol. 23, no. 1, pp. 91–100, 2017.

[6] Y. LeCun and et al., "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[7] Y.-H. Chen and et al., "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2017.

[8] J. Deng and et al., "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[9] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[10] K. He and et al., "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[11] C. M. Bishop, *Neural networks for pattern recognition*. Oxford university press, 1995.

[12] B. D. Ripley, *Pattern recognition and neural networks*. Cambridge university press, 2007.

[13] C. S. Burrus and et al., "Introduction to wavelets and wavelet transforms: a primer," 1997.

[14] S. Mallat, *A wavelet tour of signal processing*. Academic press, 1999.

[15] H. Ch and S. Corporation, "Lightning memory-mapped database manager (lmdb)," http://104.237.133.194/doc/.

[16] Y. Jia and et al., "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 675–678.