Exploiting Energy-Accuracy Trade-off through Contextual Awareness in Multi-Stage Convolutional Neural Networks

Katayoun Neshatpour, Farnaz Behnia, Houman Homayoun, Avesta Sasan
Department of Computer and Electrical Engineering, George Mason University

Abstract— One of the promising solutions for energy-efficient CNNs is to break them down into multiple stages that are executed sequentially (MS-CNN). In this paper, we illustrate that unlike deep CNNs, MS-CNNs develop a form of contextual awareness of input data in initial stages, which could be used to dynamically change the structure and connectivity of such networks to reduce their computational complexity, making them a better fit for low-power and real-time systems. We suggest three run-time optimization policies, which are capable of exploring such contextual knowledge, and illustrate how the proposed policies construct a dynamic architecture suitable for a wide range of applications with varied accuracy requirements, resources, and time-budget, without further need for network re-training. Moreover, we propose variable and dynamic bit-length fixed-point conversion to further reduce the memory footprint of the MS-CNNs.

I. Introduction

Due to recent developments in the design of deep and modern Convolutional Neural Networks (CNN), and processing power provided by Graphical Processing Units (GPU) for their training, CNNs have become ubiquitous in applications including vision, speech recognition, and natural language processing. However, effective mapping of these models to low-power devices is extremely challenging, as such applications are computationally expensive and power hungry. In addition, many real-time applications require fast and deadline-driven computation to operate safely and correctly; for current learning models, this could be satisfied by adapting more parallelism and boosting the hardware performance, which is not possible in low power and resource constrained devices.

One of the promising solutions to reduce the computational complexity of deep CNNs is to break down these deep and complex networks into a series of smaller CNN networks that are executed sequentially [1], [2], [3], [4]. This approach provides a real-time tuning capability based on the accuracy requirements, availability of resources, and their constraining energy and time-budget. The existing Multi-Stage CNNs (MS-CNNs) differ in the structure of each sub-network and the relative connectivity of various sub-networks. However, all of them practice early-termination if classification confidence reaches a desired threshold in a given stage, skipping all future stages. It should be noted that, all the existing MS-CNNs suffer from substantial increase in parameter count.

In this paper, we explore various means of utilizing the classification outcome of each stage of MS-CNNs to reduce the computational complexity and number of required parameters in the subsequent stages. Some of the proposed policies evaluated in this paper include: (1) *Pruning the Classifier*: pruning the investigated classes by skipping computation of improbable classes that score a low confidence in a previous stage. (2) *Pruning the Filters*: skipping the computation of channels/filters that do not significantly contribute to the probable classes, and (3) *Adjusting Parameter Bit-length*: reducing the bit-length of the parameters in early stages or the bit-length of parameters corresponding with less probable classes. While various work focus on finding an optimized CNN implementation, we show that the proposed policies offer a flexible solution that fits a wide range of application with varied requirements through a single architecture with no need for fine-tuning and re-training.

II. Background and Related Work

Table I captures the detection accuracy of CNN architectures which have competed in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) in recent years. The ILSVRC [5] is an object detection competition for classification of images into 1000 different classes by training on 1.2 million labeled images. Table I shows that the number of layers and the complexity of the CNNs in terms of the number of FLOPs has dramatically increased over time to enhance the classification accuracy.

These state of the art CNN solutions are composed of a single feed-forward computational network, where the classification confidence is determined after the network is completely executed. With solely focusing on classification accuracy, these learning models have little or no regard for energy or power saving. To reduce the computational complexity and average energy consumed per classification, researchers investigated the possibility of performing classification in multiple phases using sequence of serially executed smaller networks [1][2][3][4], resulting in a new class of architecture that we refer to as Multi-Stage CNN.

In MS-CNNs, the overall CNN solution is decomposed into a sequence of smaller networks capable of classification, referred to as micro CNN (uCNN) [2]. MS-CNN makes it possible to practice early termination if classification confidence threshold is met before executing the entire network, which enables the MS-CNN classifier to speed up the the average classification time (depending on input data) by bypassing the remaining network and skipping its required computation.

In [1] a dynamic configuration allows the CNN to be partially or fully deployed based on classification confidence of partial network, but the memory footprint required to keep the intermediate features from partial networks is significant. Conditional Deep Learning Network (CDLN) is proposed in [3], in which only fully connected (FC) layers are added to the intermediate layers to produce early classification results. BranchyNet [4] proposes using additional convolution layers (CONV) at each exit point (branch) to enhance the performance. However, [3] and [4] are tested for 10-class data sets (MNIST [6], CIFAR-10) for up to 4 stages. On the other hand, in [2], iterative CNN (ICNN) is proposed for the 1000-class ImageNet for 7 stages, and input images is sampled into its

TABLE I: Specification of the existing CNN architectures.

	AlexNet[11]	ZFNet[12]	VGG[13]	GoogLeNet[14]	Resnet[15]
Top5 Accuracy	80.2%	88.8%	89.6%	89.9%	96.3%
layers	8	8	19	22	152
FLOPS	729M	663M	19.6G	1.5G	11.3G

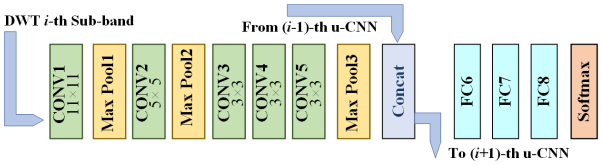


Fig. 1: The architecture of the i-th uCNN for the multi-stage AlexNet.

sub-bands using Discrete Wavelet Transform (DWT), thus the computation load is reduced by processing sub-bands with reduced dimensions in each sub-network.

A common drawback of all proposed MS-CNNs is the increased number of FC layers required for each stage of such networks. FC layers account for a large number of parameters (i.e., 96% in AlexNet), as a result MS-CNN suffers from significant increase in the parameter count. Moreover, none of the proposed multi-stage approaches use the contextual awareness from early stages, or exploit various termination strategies to optimize and tune their network structure or reduce their required parameter count dynamically.

In this paper, we use the contextual knowledge learned from an input image in early stages to provide hints to subsequent next stages to skip unnecessary computation and dynamic pruning of unnecessary parameters. Moreover, we provide a far more flexible and much richer set of features for trading off the accuracy vs computational complexity vs energy consumption vs classification time by proposing various pruning, thresholding and bit-length adjustment techniques.

It should be noted that other optimization approaches have been proposed to enhance the energy-efficiency of the CNN architectures including Tensor decomposition and low-rank optimization [7], [8], parameter compression and exploitation of sparsity [9], and precision reduction of weights and/or neurons by fixed-point tuning and quantization [10]. Not only are these approaches orthogonal and applicable to MS-CNNs, but also they can benefit from the contextual knowledge from the MS-CNN to further refine their approach to best optimize each stage. An example of this is studied in Sec. IV-C.

III. Implementation Framework

As a representative of MS-CNNs, we deploy the ICNN [2] as our experimental framework. A 7-stage MS-CNN was built and trained as suggested in [2]. For training we used Berkeley Caffe [16] and deployed Tesla K80 GPUs. The ImageNet training and the validation archives were used to train and network and validate the results.

Fig. 1 captures the structure of the i-th uCNN in MS-CNN. The overall MS-CNN is constructed by connecting 7 of such smaller networks. Each uCNN processes one sub-band of the DWT of the input image. Concat layer fuses the output feature maps (*Ofmap*) of the last CONV layer in the current stage with the *Ofmaps* of the last CONV layers from previous stage(s). Note that since the number of *Ofmaps* which are processed at any given CONV layer in each uCNN is considerably smaller than that of original AlexNet (see Table II), the FLOP count of the resulting MS-CNN is considerably smaller than that of AlexNet. However, due to the increased number of FC layers (one set per stage), the total parameter count is higher.

TABLE II: The configuration, i.e., number of output channels, parameters, FLOPs for each stage of multi-stage AlexNet.

stage	s1	s2	s3	s4	s5	s6	s7	AlexNet
CONV1	24	24	24	24	24	24	24	96
CONV2	64	64	64	64	64	64	64	256
CONV3	96	96	96	96	96	96	96	384
CONV4	48	48	48	48	48	48	48	384
CONV5	32	32	32	32	32	32	32	256
FC6	1024	1024	1024	1024	2048	2048	4096	4096
FC7	1024	1024	1024	1024	2048	2048	4096	4096
FC8	1000	1000	1000	1000	1000	1000	1000	4096
Parameters[M]	3.36	4.54	5.72	6.90	18.15	20.5	54.01	60.97
FLOPS[M]	59.57	60.75	61.93	63.11	74.36	76.72	110.22	728.27
Total	Parameter: 113M				FLOP: 506M			-

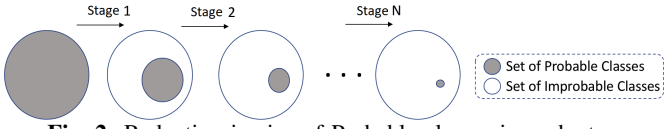


Fig. 2: Reduction in size of Probable classes in each stage

IV. Contextual Awareness in MS-CNN

Real-time application of deep learning algorithms is often hindered by high computational complexity and frequent memory accesses. Network pruning is a training-phase technique to solve this problem [17]; however, MS-CNNs enable us to apply the pruning at run-time. This capability is the result of having an early classification, that provides MS-CNN with a contextual knowledge that could be used in the next stages to skip or reduce computation without affecting the overall accuracy. As illustrated in Fig. 2, after executing each uCNN (each stage), the class probabilities are biased, dividing the classes into two set of probable and improbable classes. Note that after each stage the size of probable classes reduces as classification accuracy improves. Hence, the probability of classes could be used as an effective measure to reduce computation in the subsequent stages by avoiding computation related to improbable classes, or reducing the precision of computation for less probable classes.

To further discuss benefits of contextual awareness in MS-CNN, we introduce ***Prediction Rank*** **(PR)** as an indicator of prediction accuracy of a model for each image. Consider an image with class label $C[i]$. After executing a uCNN, all the class probabilities are sorted in descending order. Lets denote the location of a class $C[i]$ in the sorted array of class probabilities as its Prediction Rank. If statistical analysis of the data set for class $C[i]$ shows that PR of probable classes is always smaller than L in all stages, where $(1 \leq L \leq 1K)$, limiting the number of computed class probabilities in each uCNN to top-L of classes in previous stage instead of 1k, would have no impact on the probability of detection of class $C[i]$. On the other hand, if the number of computed classes (i.e., L), is chosen smaller than range of PR variation for class $C[i]$, then by pruning the computation for class α we may end up with misclassification. Expanding this to all classes, the probability of misclassification (MC) conditioned on pruning the classes to those with $PR \leq L$ is given by:

$$P(MC|P_{th} = L) = \sum_{i=1}^{1000} P(C[i])P(PR(C[i]) > L). \quad (1)$$

In this equation the P_{th} is the Pruning threshold, $C[i]$ is the i-th class, $PR(C[i])$ is the prediction rank for $C[i]$, and L is the chosen limit for pruning. For obtaining the probability $P(PR(C[i]) > L)$, we define the Decision function D as:

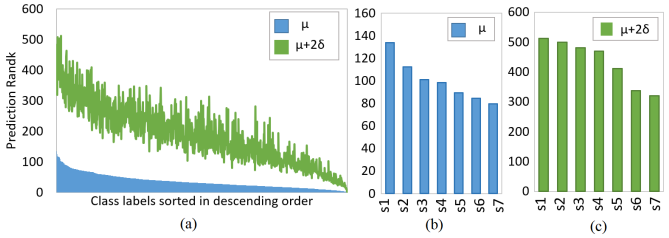


Fig. 3: prediction Range (a) PR_μ & $PR_\mu + 2 \times PR_\sigma$ for various classes at stage 1. (b) Maximum PR_μ among all classes at each stage (c) $PR_\mu + 2 \times PR_\sigma$ among all classes at each stage.

$$D(\alpha^{C[i]}(j), L) = \begin{cases} 1, \text{if} PR(\alpha^{C[i]}(j)) > L \\ 0, \text{else} \end{cases} \quad (2)$$

In this equation $\alpha^{C[i]}(j)$ is the j-th image member of class $C[i]$. If we define $S[i]$ as the number/size of images in the data set (or expected number of images in the batch) that belong to class $C[i]$, $P(PR(C[i]) > L)$ is computed as follows:

$$P(PR(C(i)) > L) = \frac{\sum_{j=1}^{S[i]} D(\alpha^{C[i]}(j), L)}{S[i]}. \quad (3)$$

By using equation 1 the pruning threshold L could be chosen to set the probability of misclassification due to pruning to a desired value. With higher value of L fewer classes are pruned, resulting in a higher classification accuracy. The value L could reduce from stage to stage, making the pruning more aggressive as the accuracy of classification increases.

Fig. 3-a shows the mean (PR_μ) and variation (PR_σ) of PR for all 1k classes sorted in descending order of PR_μ at the first stage. Assuming a normal distribution, 95% of images of each class will have a PR below $PR_\mu + 2 \times PR_\sigma$. In Fig. 3 $PR_\mu + 2 \times PR_\sigma$ of none of the classes exceeds 500, suggesting that by removing 50% of classes ($P_{th} = 500$) the prediction accuracy does not drop beyond 5% for any of the classes. Fig. 3-b and 3-c show the maximum of PR_μ and $PR_\mu + 2 \times PR_\sigma$ among all classes for all the stages. The maximum of $PR_\mu + 2 \times PR_\sigma$ is reduced as MS-CNN moves to the next stages, indicating that in subsequent stages, the pruning policy could be more aggressive. Note that based on the eliminated classes, the computational complexity of the next stages is reduced by pruning the neurons in the FC layers and/or the filters in the CONV layers which are highly correlated to classes with close to zero probabilities.

A. Pruning Neurons in FC Layers

By setting the Pruning Threshold (P_{th}) to value L in a given stage, top-L high probability classes are selected for the next stage and the rest of the classes are eliminated. The computation of FC layer for the eliminated classes could be skipped. This results in significant reduction in required FC computation and memory transfer of parameters.

B. Pruning Filters in CONV Layers

Visualization [12] and statistical analysis of data allows identifying the CONV filters which are closely associated with each class. By selecting top-L classes for the next stage, only CONV filters associated with these classes are needed, and the computation of other filters could be skipped.

C. Variable and Dynamic Bit-length Selection

There are two general approaches to convert a neural network to fixed point. One is to train a network with fixed-point constraints [18], [19]. While this approach yields high accuracy, it requires tight integration between the network design, training and implementation, which is not always feasible. The other approach is to convert a trained network into a fixed point model. This approach is more suited for applications where a pre-trained network is utilized with no access to the original training data [20].

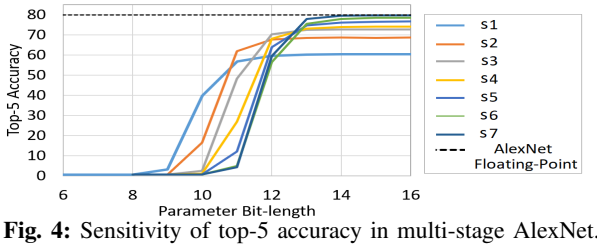


Fig. 4: Sensitivity of top-5 accuracy in multi-stage AlexNet.

TABLE III: Parameter size of variable bit-length conversion of multi-stage AlexNet vs. Single precision AlexNet.

Stage	s1	s2	s3	s4	s5	s6	s7	AlexNet
Param. Bit-length	11	11	12	13	14	15	16	32
Original Param. Size [MB]	13.4	18.2	22.9	27.6	72.6	82	216	244
Updated Param. Size [MB]	4.6	6.2	8.6	11.2	31.8	38.4	108	-

In [21] an exploration of optimal fixed point bit-length allocation across CNN layers shows that without further training, a minimum of 16 bit-length is required for the parameters of the CNN to yield results with an accuracy comparable to single-precision floating point. Similar story applies to the MS-CNN; however, in the early stages of MS-CNN the network is less sensitive to quantization errors induced by fixed-point transformation. In addition, in case of loss in accuracy, it could recover in the future stages. This allows the quantization to be applied more aggressively in early stages.

Fig. 4 shows the sensitivity of accuracy of MS-CNN to the bit-length of the parameters at each stage. At each stage the optimal bit-length is derived from a saturation point after which increased bit-length does not enhance the accuracy. As Fig. 4 shows, this saturation point is pushed to lower bit-lengths for earlier stages. Hence, for MS-CNN rather than selecting an optimal bit-length for the entire network, various sub-networks deploy different bit-lengths allowing further reduction in the number of parameters. Table III shows the size of parameters for various stages of multi-stage AlexNet and a single-precision AlexNet. While the number of parameters required for multi-stage AlexNet is 80% higher than original AlexNet, variable bit-length selection for multi-stage AlexNet reduces its parameter size by more than 15%.

In addition to per-stage bit-length adjustment, the contextual knowledge from early stages allows the precision of parameters corresponding to less probable classes to be reduced *dynamically*, allowing further flexibility in tuning MS-CNN to meet the requirements of larger set of applications.

V. Policies for Complexity-Accuracy Trade-off

The context-aware nature of MS-CNN creates opportunities for exploring the trade-off between computational complexity (that impacts run-time and energy per classification) versus classification accuracy. In this section we propose several policies to exploit these opportunities.

A. Variable Thresholding Policy ***(TP)***

A fixed thresholding policy for making early termination decision is used in [2], [3], [4]. Using this policy MS-CNNs

terminate as soon as a uCNN reaches desired classification confidence. The classification confidence is calculated by summing the probabilities of top-C (e.g. $C = 5$) suggested classes, which is compared to a predefined Detection Confidence-Threshold (D_{ct}). In this section, we explore the impact of choosing variable threshold values for each uCNN and investigate the resulting trade-off between CNN computational complexity versus classification-accuracy.

Algorithm 1 implements the dynamic TP policy. The $uCNN(i, C_L, img)$ function invokes the i-th uCNN with img as input, requesting classification result for all class labels in the $\bar{C_L}$ vector and produces a vector of probabilities $\bar{P_r}$, one for each of 1000 labels in $\bar{C_L}$. After each uCNN, the sum of top-5 probabilities ($\sum_{k=1}^{5} \bar{P_r}[k]$) is compared with Detection Confidence Threshold ($\bar{D_{ct}}[i]$), and if greater, the image is classified. Note that some of the images never reach a classification confidence above D_{ct}. For these images, the results of the classification in the last stage are used.

Algorithm 1 TP

```
for i = 1 to N − 1 do
    P̄_r ← uCNN(i, C̄_L, img);
    Sort_descending(P̄_r);
    if ((Σ_{k=1}^{5} P̄_r[k]) > D̄_ct[i]) then
        exit;
P̄_r ← uCNN(N, C̄_L, img);
```

B. Context-aware Pruning Policy (CAP)

1) Context-aware Pruning Policy for FC layer (CAP_{FC}):

Based on Sec. IV, Algorithm 2 proposes the implementation of CAP policy for FC neurons. In the first uCNN, C_L contains all 1000 labels. The $uCNN(i, C_L, img)$ function invokes the i-th uCNN. Subsequently, The less-probable classes are pruned based on pruning threshold stored in pruning policy vector $\bar{P_{th}}$. For instance, $\bar{P_{th}}[i] = 100$ results in only choosing the 100 labels and disables all other neurons in the FC layer of $uCNN(i + 1)$ associated with the eliminated labels.

Algorithm 2 CAP_{FC}

```
for i = 1 to N − 1 do
    P̄_r ← uCNN(i, C̄_L, img);
    Sort_descending(P̄_r);
    C̄_L ← C̄_L[1 : P̄_th[i]];
P̄_r ← uCNN(N, C̄_L, img);
```

Since the compute-intensive parts of the CNN architectures are the CONV layers, CAP_{FC} slightly reduces the computational complexity as it only affects the FC layers. However, it considerably reduces the number of weights needed to be moved to the memory, yielding a dynamic trade-off between accuracy and the required memory footprint.

2) Context-aware Pruning for CONV layer (CAP_{CONV}):

Visualization of CNN [12] and statistical analysis of labeled dataset makes it possible to identify and remove trained CONV filters that are closely associated with classification of low-probability classes. In order to enable the CAP_{CONV} pruning, as described in Algorithm 3, we analyzed the 1.2 million images in the training set of ImageNet data set; for each (stage, CONV layer, class) triple, we extracted a vector of accuracy-loss due to the removal of each filter in the given CONV layer. In Algorithm 3 the $\bar{Pre}(i, cnv, c)$ shows the pre-processing vector for the i-th stage, where cnv refers to the target CONV layer and c refers to class label. Thus, the removal of the filter associated with the first argument in $\bar{Pre}(i, cnv, c)$ has the least effect on overall accuracy of class c. $CONV_{lst}$ is the list of CONV layers targeted for filter pruning and rm denotes number of filters to be removed from each CONV layer.

Algorithm 3 CAP_{CONV}

```
Pre-Process:Obtain P̄re(i, cnv, c)
for i = 1 to N − 1 do
    P̄_r ← uCNN(i, C̄_L, img);
    Sort_descending(P̄_r);
    for cnv=1 in CONV_lst do
        Fltrem = {}
        for c = 1 to P̄_th[i] do
            Fltrem+ = P̄re(i, cnv, c)[1 : rm]
        Fltrem ← Maj(Fltrrem, rm))
        Update(uCNN, i , cnv, Fltrem)
P̄_r ← uCNN(N, C̄_L, img);
```

In the i-th stage, the filters least affecting the top-P_{th} classes are gathered in $Fltrem$ determining the candidate filters for removal. Subsequently, the majority function $Maj(Fltrrem, rm)$ returns rm most repeated arguments in $Fltrem$. This allows us to find the union of filters that least affect the top-$P_{th}[i]$ classes in the i-th stage. Subsequently, $Update(uCNN, i, cnv, Fltrem)$ updates the i-th uCNN by removing rm filters in $Fltrem$ from cnv-th CONV layer.

C. Pruning and Thresholding Hybrid Policy (PTHP)

The PTHP scheme takes advantage of both early termination in the thresholding scheme and the pruning of the FC layers in context-aware pruning policy. Algorithm 4 illustrates the PTHP policy. In the first uCNN the probability for all classes (all labels) is computed. In the while loop, if top-5 confidence is above the detection threshold $\bar{D_{ct}}[i]$, the classification is terminated. Otherwise, based on the value of a *Selection Threshold* $\bar{S_{th}}[i]$, a number of labels are selected for further processing in the next layer. The selected classes are the minimum number of labels with an accumulated probability of no less than $\bar{S_{th}}[i]$. In this algorithm $C\bar{C}_L$ is the shrunk version of $\bar{C_L}$ that only contains the labels of interest.

Algorithm 4 PTHP

```
P̄_r ← uCNN(1, C̄_L, img);
for i = 1 to N − 1 do
    Sort_descending(P̄_r);
    if ((Σ_{k=1}^{5} P̄_r[k]) > D̄_ct[i]) then
        exit;
    else
        Sum = 0, CC̄_L = [], label=1;
        while Sum < S̄_th[i] do
            Sum+ = P̄_r[label];
            CC̄_L[label] = C̄_L[label];
            label++;
        P̄_r ← uCNN(i + 1, CC̄_L, img);
```

VI. Implementation Results

For reporting the the accuracy values for the proposed polices we evaluated the 50K images in the validation set of ImageNet repository. The results are summarized next:

A. Variable Thresholding Policy

Fig. 5 shows the overall accuracy and average FLOP count for variable thresholding policy. Note that since not all images go through all the stages of ICNN, the FLOP count varies for each image. Thus, in Fig. 5 the average parameter and FLOP counts are reported over 50K images in the validation set. Moreover, to better highlight how the thresholding policy contributes to reducing computational complexity, the FLOP and parameter counts are devised relative to the last stage of

MS-CNN. Fig. 5-b shows the $\bar{D_{ct}}$ values for each stage of the studied thresholding policies. T1 to T10 are sorted based on FLOP counts, with T1 corresponding to the policy with the lowest FLOP count and T10 the highest.

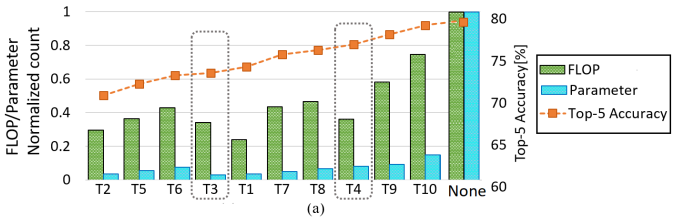
Fig. 5 shows that even with a mild thresholding policy as in T10, the FLOP and parameter counts are reduced by up to 25% and 80%, respectively, with negligible accuracy loss. Note that in MS-CNN not all of the parameters are required for all the images. Since early stages of MS-CNN deploy smaller FC layers, the number of parameters for images classified in the early stages is significantly lower than the latter stages. Hence, early termination results in significant drop in the parameter count by removing the need to execute the larger FC layers of next stages.

B. Context-aware Pruning Policy

1) CAP_{FC}: Fig. 6-a shows the overall accuracy and average parameter count for various FC layer pruning policies, while the table in Fig. 6-b captures the setting of the pruning policy by listing the number of remaining classes after pruning the classes with low probability at each stage of investigated MS-CNN. The last bar shows the MS-CNN results when none of the classes are pruned in any of the stages. P1 to P8 are sorted based on their parameter count, with P1 having the lowest parameter count and P8 the highest.

Fig. 6 shows that by increasing the number of pruned classes (i.e., lower P_{th}) the accuracy drops; however, intelligent selection of the pruning policy yields reductions in the parameter count, with negligible accuracy loss. For instance P8 yields a 17% reduction in the parameter count with negligible accuracy loss. With CAP_{FC} all the images go through all the stages in this scheme. Thus, the only reduction in the FLOP count is due to the pruning of the last FC layer in each stage. Newer and deeper CNNs (e.g., VGGNet and GoogleNet) deploy fewer FC layers as opposed to 3 FC layers in AlexNet. Hence, applying pruning policy to deeper CNNs increases the rate of FLOP count reduction as well as parameter count reduction in their multi-stag version. Note that moving the large set of parameters required for the FC from the memory accounts for a significant energy and execution delay.

2) CAP_{CONV}: Fig. 7 captures the result of filter pruning for the last CONV layer (i.e., CONV5) in stages 2 to 7. CONV5 of each stage consists of 32 filters, and in each

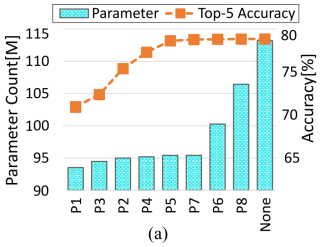


stg	Thresholding Policy									
	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
s1	0.8	0.5	0.85	0.8	0.5	0.5	0.9	0.8	0.95	0.95
s2	0.7	0.5	0.7	0.8	0.55	0.6	0.8	0.8	0.9	0.95
s3	0.6	0.5	0.55	0.75	0.6	0.7	0.7	0.7	0.85	0.95
s4	0.5	0.5	0.4	0.75	0.65	0.8	0.6	0.7	0.8	0.95
s5	0.4	0.5	0.25	0.7	0.7	0.9	0.5	0.6	0.75	0.95
s6	0.3	0.5	0.1	0.7	0.75	0.9	0.4	0.6	0.7	0.95

e.g. T9 policy: for stage 3 D_{ct}=0.85

(b)

Fig. 5: Variable thresholding policies, (a) The accuracy and average FLOP and parameter count (b) The table including the $\bar{D_{ct}}$ values in each stage for each thresholding policy. The optimized policies are the circled ones, in which higher accuracy is maintained with reduced FLOP count and parameter count.



stg	Pruning Policy							
	P1	P2	P3	P4	P5	P6	P7	P8
s1	1k	1k	1k	1k	1k	1k	1k	1k
s2	10	20	150	100	100	200	400	400
s3	10	20	60	100	100	200	320	400
s4	10	20	30	100	100	200	256	400
s5	10	20	18	100	100	200	205	400
s6	10	20	13	50	100	200	164	400
s7	10	20	10	25	100	200	131	400

(b)

Fig. 6: CAP_{FC} policies, (a) The accuracy and average parameter count (b) Number of labels *not* pruned in each stage of each policy.

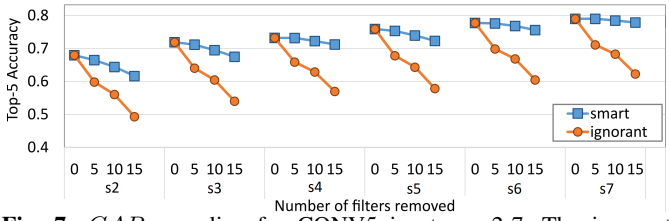


Fig. 7: CAP_{FC} policy for CONV5 in stages 2-7. The ignorant trend-line shows the accuracy for pruning filters that least effect the accuracy of all classes, while the smart trend-line shows the results for when feedback from previous stages are used to prune the filters.

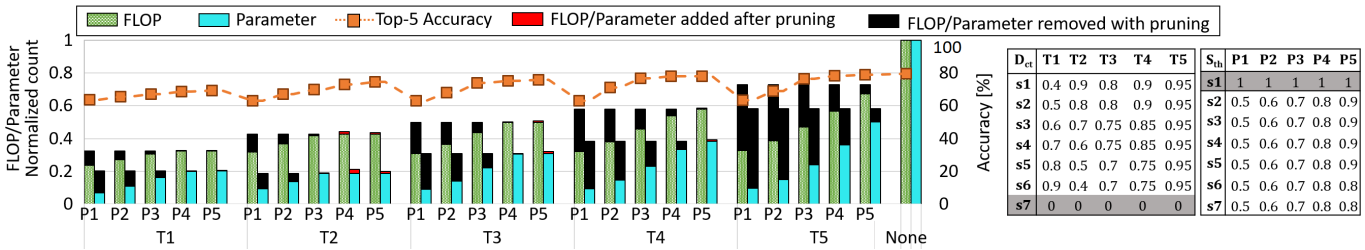
stage, 5, 10 and 15 filters least affecting the top-5 classes from the previous stage are pruned (i.e., $rm = 5, 10, 15$ and $P_{th} = 5$). Our proposed smart approach takes advantage of the feedback from previous stages and depending on the remaining classes removes the least contributing filters, while the ignorant approach prunes filters based on how strongly each filter contributes to the overall accuracy of the network across all classes. As illustrated in Fig. 7, the contextual knowledge generated after each classification, and selective pruning of filters based on remaining classes significantly reduces the loss in classification accuracy when pruning filters.

C. Pruning and Thresholding Hybrid Policy

Fig. 8.left shows the accuracy, and normalized FLOP and parameter counts for the $PTHP$ policy, while Fig. 8.right captures the setting used for experimented thresholding and pruning policies in $PTHP$. The figure is divided into 5 sections, one for each thresholding policy. Each segment assess the impact of 5 different pruning policies on total parameter and flop count when combined with thresholding in $PTHP$.

Fig. 8 shows that with aggressive thresholding (e.g., T1), pruning results in significant drop in accuracy. However, with moderate thresholding some (but not all) of the pruning policies allow the parameter and FLOP counts to significantly drop with little impact on the accuracy. This is due to the fact that with $PTHP$ only the high probability classes (with accumulated probability of S_{th}) proceed to the next stages.

In the $PTHP$ the classification is terminated if 1) a top-5 confidence of D_{ct} is reached, or 2) the number of classes selected for the next stage is no bigger than 5. In the second case, proceeding to the next stage does not increase the top-5 accuracy and MS-CNN is terminated while the confidence threshold is not reached. The effect of this early termination is twofold: On one hand, it identifies images that would never reach a high classification confidence, and for which processing of more stages is a waste of resources, hence reducing computation with no accuracy loss. On the other hand, it could prematurely terminate the processing of images that could have reached top-5 in the proceeding stage, negatively affecting the accuracy and the top-5 confidence of



D_{ct}	T1	T2	T3	T4	T5
s1	0.4	0.9	0.8	0.9	0.95
s2	0.5	0.8	0.8	0.9	0.95
s3	0.6	0.7	0.75	0.85	0.95
s4	0.7	0.6	0.75	0.85	0.95
s5	0.8	0.5	0.7	0.75	0.95
s6	0.9	0.4	0.7	0.75	0.95
s7	0	0	0	0	0

S_{th}	P1	P2	P3	P4	P5
s1	1	1	1	1	1
s2	0.5	0.6	0.7	0.8	0.9
s3	0.5	0.6	0.7	0.8	0.9
s4	0.5	0.6	0.7	0.8	0.9
s5	0.5	0.6	0.7	0.8	0.9
s6	0.5	0.6	0.7	0.8	0.8
s7	0.5	0.6	0.7	0.8	0.8

Fig. 8: The accuracy and average FLOP and parameter counts (normalized to MS-CNN with no pruning or threshold policy) for $PTPH$. For each thresholding policy (i.e., D_{ct} vector), multiple pruning policies (i.e., S_{th} vector) are investigated. The red and black bars show the increase and reduction respectively, in the number of FLOP and/or parameter counts of a thresholding policy due to pruning.

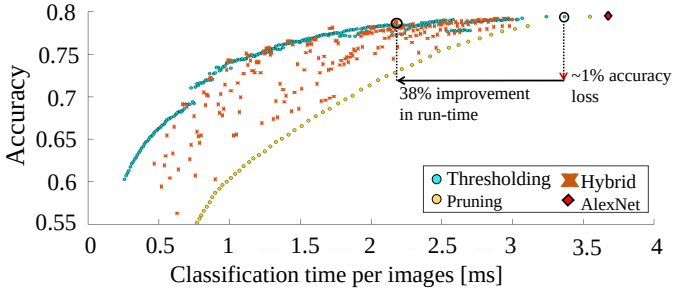


Fig. 9: The average per image run-time of multi-stage AlexNet for 50k validation images on K80 GPU. The figure marks how in a hybrid policy pruning further reduces the run-time with no performance loss.

the subsequent stages. This forces MS-CNN to proceed to the next stage, and increases the parameter and FLOP counts (see red bars in Fig. 8). Fig. 8 shows that when the value of S_{th} at each stage is selected slightly higher than D_{ct} of the next stage, reductions in FLOP and parameter count has the least impact on the accuracy (See T4-P5 hybrid policy).

D. Run-time and overall accuracy

Adopting the pruning or thresholding policies creates a trade-off between average delay and the accuracy of classification. The pruning and thresholding policies reduce the parameter and FLOP counts, respectively. However, as shown in Fig. 6 and 5, increased resources does not always translate into higher accuracy. The hybrid policies which combine both pruning and thresholding, exhibit the same behavior. Consequently, finding the optimal strategy in which the target accuracy is reached with minimal resources (in this case execution time), is realized by exploration and tuning of thresholding and pruning parameters as shown in Fig. 9.

Fig. 9 captures the design space of the MS-CNN when trading the accuracy for reducing run-time through thresholding and pruning policies. In this figure, each point denotes a unique combination of thresholding and pruning policies. As illustrated, the stand-alone thresholding policy yields better run-time/accuracy trade-off compared to the stand-alone pruning policy. However their combination in hybrid policy could lead to optimal solutions. For instance Fig. 9 highlights a stand-alone thresholding policy and a hybrid policy derived by combining the same thresholding policy with pruning. Fig. 9 shows that the hybrid policy reduces the classification time by 38% with less than 1% accuracy loss.

VII. Conclusions

In this paper we studied how the contextual knowledge gained from execution of each stage in a multi-stage CNN architecture could be used to reduce computational complexity and parameter count in the subsequent stages. We proposed various policies to improve the efficiency of MS-CNNs and to construct a dynamic architecture suitable for a wide range of applications with varied accuracy requirements, resources, and time-budget, without further need for network re-training. We combined confidence-based early-termination and context-aware pruning to significantly reduce the computational complexity, parameter count and run-time of a MS-CNN with negligible accuracy loss. Moreover, we proposed variable and dynamic bit-length selection to solve the problem of increased parameter size associated with multi-stage CNN.

VIII. Acknowledgement

This research leading to this publication is sponsored by National Science Foundation (award number 1718538).

References

[1] H. Tann, S. Hashemi, I. Bahar, and S. Reda, "Runtime configurable deep neural networks for energy-accuracy tradeoff," in *(CODES+ ISSS)*, 2016.
[2] K. Neshatpour and et al., "ICNN: An iterative implementation of convolutional neural networks to enable energy and computational complexity aware dynamic approximation," in *DATE*, March 2018, pp. 551–556.
[3] P. Panda, A. Sengupta, and K. Roy, "Conditional deep learning for energy-efficient and enhanced pattern recognition," in *DATE*, 2016.
[4] S. Teerapittayanon and et al., "Branchynet: Fast inference via early exiting from deep neural networks," in *ICPR*. IEEE, 2016.
[5] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.
[6] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Process. Mag*, vol. 29, no. 6, 2012.
[7] M. Jaderberg *et al.*, "Speeding up convolutional neural networks with low rank expansions," *arXiv preprint arXiv:1405.3866*, 2014.
[8] V. L. et al., "Speeding-up convolutional neural networks using fine-tuned cp-decomposition," *arXiv preprint arXiv:1412.6553*, 2014.
[9] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Penksy, "Sparse convolutional neural networks," in *ICVPR*, 2015, pp. 806–814.
[10] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," in *ICVPR*, 2016, pp. 4820–4828.
[11] A. Krizhevsky and et al., "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.
[12] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *ECCV*. Springer, 2014, pp. 818–833.
[13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[14] C. Szegedy *et al.*, "Going deeper with convolutions," in *CVPR*, 2015.
[15] K. He and et al., "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.
[16] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *ACM int. conf. on Multimedia*, 2014, pp. 675–678.
[17] S. Anwar, K. Hwang, and W. Sung, "Structured pruning of deep convolutional neural networks," *JETC*, vol. 13, no. 3, Feb. 2017.
[18] M. Courbariaux, Y. Bengio, and J.-P. David, "Low precision arithmetic for deep learning," *CoRR, abs/1412.7024*, vol. 4, 2014.
[19] S. Gupta, A. Agrawal, , and P. Narayanan, "Deep learning with limited numerical precision," in *ICML*, 2015, pp. 1737–1746.
[20] M. Rastegari and et al., "Xnor-net: Imagenet classification using binary convolutional neural networks," in *ECCV*. Springer, 2016, pp. 525–542.
[21] D. D. Lin, S. S. Talathi, and V. S. Annapureddy, "Fixed point quantization of deep convolutional networks," in *ICML*, 2016, pp. 2849–2858.