https://doi.org/10.2352/ISSN.2470-1173.2017.19.AVM-020

Efficient Pre-Processor for CNN

Mihir Mody, Manu Mathew, Shyam Jagannathan;
Email: {mihir, mathew.manu, shyam.jagannathan}@ti.com
Automotive Processor, Texas Instruments Incorporated, Bangalore, INDIA

Abstract

Convolution Neural Networks (CNN) are rapidly deployed in ADAS and Autonomous driving for object detection, recognition, and semantic segmentation. The prior art of supporting CNN (HW IP or multi-core SW) doesn't address efficient implementation for the first layer, YUV color space, and output stride support. The given paper proposes a new pre-processing technique to enhance CNN based HW IP or multi-core SW solution. The pre-processor enables new features namely (1) Higher parallelism for the first layer with boosting of first layer (2) Efficient YUV color space (3) Efficient output stride support. The pre-processor uses novel phase-split method to enable supporting above features. The proposed solution splits input to multiple phases based on spatial location e.g. 2 phases for YUV 4:2:0 format, 4 phases for output strides 2 etc. The proposed solution is a unified solution that enables utilization (>90%) for the first layer and reduction of bandwidth of 2-4x for output stride of 2. For YUV color space, this reduces the computation by factor 2 along saving of ~0.1 mm2 of silicon area with negligible loss in accuracy.

Introduction

Traditional camera based systems with computer vision processing are used in various markets e.g. Automotive, Robotics, Industrial, Wearable computing, and Mobile [1][2]. In the past, these systems used hand-crafted features (e.g. SIFT, HOG, Haar) [3] followed by a pre-trained classifier (e.g. SVM, AdaBoost) [3] to classify objects of interests in the images. Modern day systems implement Deep learning techniques such as Convolution Neural Network (CNN) for image classification, thanks to rapid advances in computation power along with wide-spread usage of the low-cost camera providing big data.

Typical Convolution Neural Network (CNN) structure is illustrated as shown in Fig. 1. Input feature vectors are convolved with a set of pre-trained receptive field weights followed by a non-linear activation function. Max-pooling enables translation invariance and reduces the output feature vector size. The learned output feature vector is fed to the fully connected neural network for classification, with Softmax layer normalizes the results. There are multiple network topologies e.g. LeNet5[4], AlexNet[5] etc. These networks have multiple convolution layers and fully connected layers, which results in huge compute complexity going in hundreds of Giga or Tera Multiply and Add operations (GOPS or TOPS).

Typical front camera systems will comprise of a CMOS sensor, which captures real world image in raw Bayer format. The raw format is converted to either RGB or YUV formats on using image signal processor (ISP) [6][7][8]. YUV 4:2:0 and/or 4:2:2 formats are most preferred in embedded systems as it reduces the memory footprint and data bandwidth between external and on-chip memory.

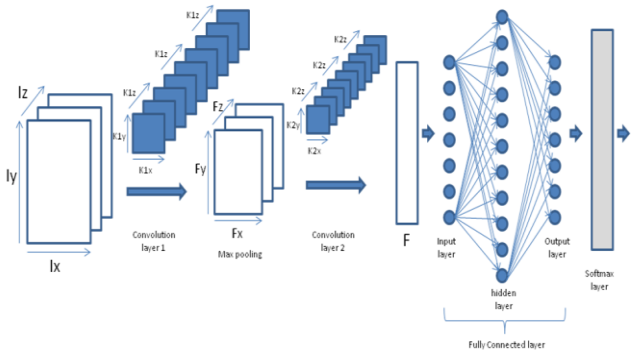

Figure 1. A typical Convolution Neural Network with two convolution layers and fully connected layer followed by softmax

Pretty much every known state of art CNN accepts input in RGB 8-bit format. This format is also in line with the basic network architecture where each layer accepts a set of N input channels of same width and height and produces M output channels of same or different dimensions. Although some ISPs can produce both RGB and YUV outputs, a resource constraint embedded system would prefer to work with just one format for obvious reasons due to the huge task of optimizations [9][10]. A YUV 4:2:0 or 4:2:2 inputs requires some preprocessing especially with chroma channels such as de-interleaving, up-sampling and bit-depth conversions from 12-bit to 8-bit and finally a YUV to RGB conversion which can then be fed to the network as shown in Figure 2.

Deeper networks with multiple layers work with many input channels (e.g. 16, 32, 64, 128, 512 …) in the later stages but the first layer will have just a 3 channel RGB input. Modern day embedded solutions will be designed to optimally handle input channels in a quantum of 8 or 16 channels both from data feed and compute aspects. This will result in a typical underflow for the first layer with just 3 channels. Although the compute contribution of the initial layer is less compared to the later stages, a hardware designer would find it difficult to handle this imbalance and end up spending extra gates to handle these corner cases.

Typical classification networks begin by accepting a fairly sized image, (e.g. 64x64, 128x128, 224x224…) and end up downsizing the image as the network progress. The initial convolution layers will have an output stride of 2 or 4. This means to produce an X by Y output, the hardware should fetch 2X by 2Y or 4X by 4Y input with sufficient padding for the convolution. This requires the hardware designer to implement wider data bus, add more local data registers etc. just to handle the first layer efficiently.

In this paper, we present novel techniques to handle these challenges. We propose the "phase-split" technique which

encourages the use of embedded friendly YUV 4:2:0 formats. The paper also presents the impact of training the network directly in YUV 4:2:0 and 4:4:4 color formats with acceptable classification accuracy for the CIFAR-10 network.

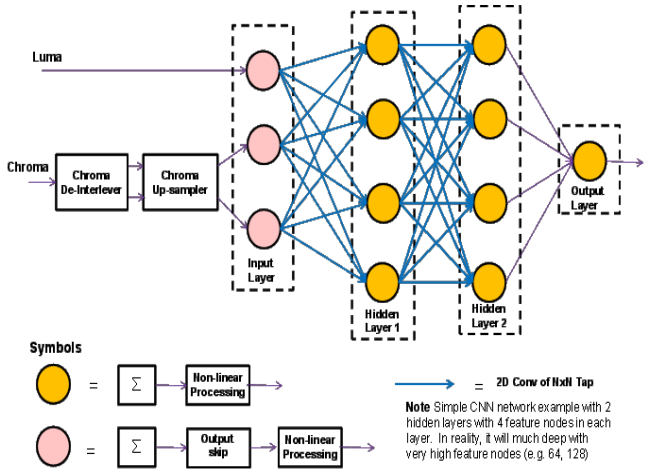

Figure 2. Prior Art

Proposed Solution

In most state-of-art CNN the first layer will accept RGB channels with a fixed width X and height Y. The number of input channels for the first layer is 3 and number of output channels is N (e.g. 16, 32, 64, 128 …). Hence the number of convolution kernels will be 3 * k * k * N, where k is the kernel size (e.g. 3, 5, 7 …). Networks accepting larger input such as 224x224 as in AlexNet[5] will have a larger kernel size in the first layer such as 5x5 along with a stride of 2 or 4 in both horizontal and vertical direction. This is mainly to reduce the input size as the network progress. Each input channel is convolved with its respective kernel and the result is summed up. This summed up the result is passed through a nonlinear activation function such as 'ReLU', 'tanh', 'sigmoid' to produce output. It is important to note that after convolution of input channels the result is summed up point wise. This requires the width and height of all the input channels to be same.

In YUV color space, the 4:4:4 format maintains the dimensions of the input in both luma (Y) and chroma (UV) spaces. Embedded friendly formats such as 4:2:0 and 4:2:2 will have the chroma channels in reduced dimensions compared to luma. YUV 4:2:0 format will have the UV dimensions as half of Y dimensions in both horizontal and vertical direction. YUV 4:2:2 format will have the UV dimensions as half of luma in the horizontal direction and same as luma in the vertical direction. This poses a problem when supplying YUV 4:2:0 or 4:2:2 directly as an input to the network. One way of solving this problem is to convert the chroma 4:2:0 or 4:2:2 space to 4:4:4. There are several techniques to perform this conversion ranging from simple pixel replicate to complex interpolating techniques using poly-phase filters. In this paper, we propose a "phase-split" technique which enables us to feed YUV 4:2:0 input with minor modification to the network structure as shown in Figure 3.

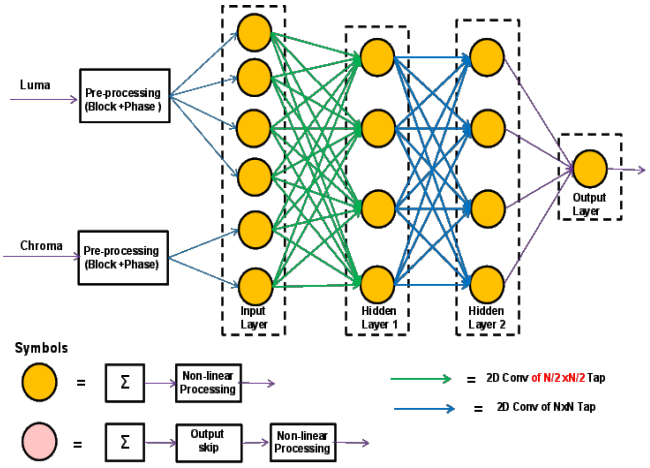

Figure 3. Proposed Solution

For a YUV 4:2:0 input, the luma channel is split into 4 phases. Each luma phase has the dimensions as W/2, H/2 with an input stride of 2 as shown in Figure 4. This reshapes the luma channel and brings the dimension same as the chroma channel. Hence we now have 4 channels of luma and 2 channels of chroma at a quarter of the original resolution. For a YUV 4:2:2 input, along with phase split of luma channel, we can split the 2 chroma channels to 4 as shown in figure 5. Hence for a 4:2:2 input we have 4 luma channels and 4 chroma channels again at a quarter of the original resolution.

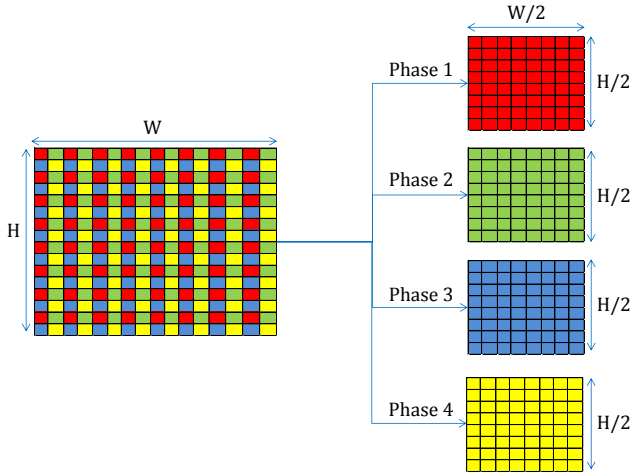

Figure 4. Luma phase split produces 4 channels

This proposal of input skip is complementary to the approach of output skip that happens in typical CNN layers. Convolution layers with large kernel size (e.g. k = 5, 7, 9 …) will typically have an output stride of 2 or 4 in the horizontal and vertical direction. Also, most of the state-of-art networks will have a pooling layer with output stride of 2 to reduce the feature size every layer.

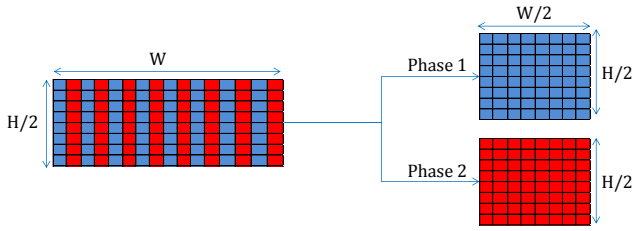

Figure 4a. Chroma phase split of 4:2:0 format produces 2 channels

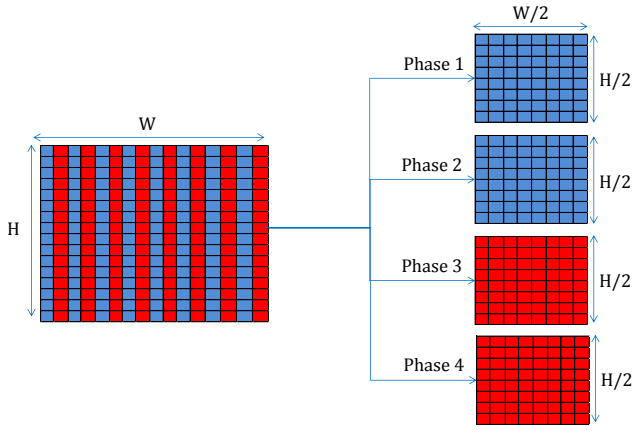

Figure 4b. Chroma phase split of 4:2:2 format produces 4 channels

Standard networks can be reconfigured to accept this multi-phase YUV input by reducing the output stride in the convolution layers by a factor of 2 in both horizontal and vertical direction or by removing the pooling layer right after the first convolution layer. By using transfer learning techniques the network can be made to learn the newly connected input which will help train the network faster.

CNN Network Details and Simulations

The CIFAR-10 dataset comprises of 60,000 images and 10 classes. These 60,000 images are divided into 5 batches of 10,000 images for training and the remaining 10,000 images are set aside for testing. Images are 8-bit RGB of size 32x32. The inputs are distorted and randomly cropped at the center with an active dimension of 24x24. Additionally, inputs are randomly flipped horizontally, image contrast and brightness is varied randomly and image mean is subtracted and divided by the variance of the pixels, The data set is shuffled to have good mixing properties. This creates 100,000 images for training. For a batch size of 256 images, it takes two epochs to completely cover 50,000 unique images twice. The initial learning rate is set at 0.1 which was exponentially decayed after every 30,000 iterations. Hence for a total of 100,000 iterations, the learning rate is changed 3 times. For testing only image resizing and mean subtraction is done on the input.

TABLE I
NETWORK STRUCTURE FOR CIFAR-10 RGB AND YUV 4:4:4 INPUT

Layer Type	Input size	Input Ch	Kernel size	Output Ch	Output size
Convolution	24x24	3	5x5	64	24x24
Max pool	24x24	64	3x3	64	12x12
Norm	12x12	64	-	64	12x12
Convolution	12x12	64	5x5	64	12x12
Norm	12x12	64	-	64	12x12
Max pool	12x12	64	3x3	64	6x6
Reshape	6x6	64	-	1	384
FC	384	1	384x192	1	192
Softmax	192	1	192x10	1	10

FC = fully connected layer, Ch - channels

Google's Tensorflow deep learning framework was used for simulations. The network used for training is described in Table I. The network comprises of two convolution layers, two max-pooling layers, two normalization layers, two fully connected layers and a softmax layer at the end. Dropout was implemented in the last fully connected layer before the softmax layer with dropout probability set at 50%. Notice that the first convolution layer has 5x5 kernels with a stride of 1 and the second max pool layer has 3x3 kernels at a stride of 2. The input reduces from a spatial resolution of 24x24 to 12x12 after the first two layers. We take advantage of this in the phase-split method. The input is reshaped from 24x24x3 to 12x12x6 before feeding to the network. Now to keep the rest of the parameters the same, we remove the max pooling layer with a stride of 2. The modified network structure is shown in Table II.

TABLE II
NETWORK STRUCTURE FOR CIFAR-10 YUV 4:2:0 INPUT

Layer Type	Input size	Input Ch	Kernel size	Output Ch	Output size
Convolution	12x12	6	5x5	64	12x12
Norm	12x12	64	-	64	12x12
Convolution	12x12	64	5x5	64	12x12
Norm	12x12	64	-	64	12x12
Max pool	12x12	64	3x3	64	6x6
Reshape	6x6	64	-	1	384
FC	384	1	384x192	1	192
Softmax	192	1	192x10	1	10

FC = fully connected layer, Ch - channels

With these two network structures we ran 4 simulations, direct RGB input, RGB to YUV 4:4:4 which used the same network structure. RGB to YUV 4:2:0 which uses the phase-split method and modified network structure and YUV 4:2:0 to YUV 4:4:4 by replicating chroma pixels. The test accuracy results are as shown in Table III. It is observed that given a network structure for RGB input, the accuracy does not vary much when we provide YUV 4:4:4 input or even with the YUV 4:2:0 replicated to YUV 4:4:4. The phase-split method observes a loss of 5% largely due to reduced input spatial resolution. The results are expected to improve with the much smaller loss for larger input resolution.

The proposed solution is designed to run up-to 600 MHz in a low power 28nm CMOS process. For accurate estimation, an RTL design for the core data-path (i.e MAC) coded and synthesized. The top level control HW area & cycles are estimated based on previous similar designs. The proposed solution is the unified

TABLE III
TRAINING AND TESTING ACCURACY FOR VARIOUS COLOR FORMATS USING CIFAR-10 DATASET

Input	Training Accuracy %	Testing Accuracy %
RGB input	99.8	86.1
RGB to YUV 4:4:4	99.7	85.6
RGB to YUV 4:2:0	99.4	80.5
YUV 4:2:0 to YUV 4:4:4	99.7	85.3

solution for both color and output skip processing. The proposed solution enables utilization (>90%) for the first layer and reduction of bandwidth of 2-4x for output stride of 2. For YUV color space, this reduces the computation by factor 2 along saving of ~0.1 mm2 of silicon area with negligible loss in accuracy. The proposed solution is generic and can be realized in HW IP for silicon or SW algorithm for a multi-core solution.

Conclusion

The paper introduces challenges of working directly with YUV domain image from ISP. The paper proposed the novel solution of "phase-split" technique which encourages the use of embedded friendly YUV 4:2:0 formats with future research to work directly Bayer domain. The paper provides the quantitative impact of training the network directly in YUV 4:2:0 and 4:4:4 color formats with acceptable classification accuracy for the CIFAR-10 network.

References

[1] P. Vishwanath, K. Chitnis, P. Swami et. al, “A Diverse Low Cost High Performance Platform for Advanced Driver Assistance System (ADAS) Application”, Computer Vision and Pattern Recognition (CVPR), 2016

[2] S. Dabral, M. Mody, S. Kamath, B. Zhang, V. Appia and U. Batur, "Trends in camera based Automotive Driver Assistance Systems (ADAS)," Circuits and Systems (MWSCAS), 2014 IEEE 57th International Midwest Symposium on, pp.1110-1115, 3-6 Aug. 2014.

[3] Richard Szeliski, " Computer Vision: Algorithms and Applications", Springer Book, 2010

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition”, Proceeding of IEEE, 1988

[5] Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton, “ImageNet Classification with Deep Convolutional Neural Network”, NIPS (2012)

[6] M. Mody, H. Sanghvi, N. Nandan, , et. al., "High performance and flexible imaging Sub-system," Advances in Computing, Communications and Informatics (ICACCI), IEEE Conference, Sept. 2014

[7] M. Mody, N. Nandan, S, Dabral, et. al. “Image Signal Processing for Front Camera based Automated Driver Assistance System”, Consumer Electronics (ICCE) Berlin, 2015, IEEE Conference, Sep. 2015.

[8] M. Mody, P. Swami, K. Chitnis et.al, “High Performance Front Camera ADAS Applications on TI’s TDA3X Platform”, High Performance Computing (HiPc), 2015, IEEE Conference, Dec. 2015.

[9] Shyam J., Mihir Mody and Manu Mathew, " Optimizing convolutional neural network on DSP", Consumer Electronics, ICCE, 2016

[10] Mihir Mody, Shyam J., and Manu Mathew et. al. " Efficient Mapping of 2D Convolution on DSP for Convolution Neural Network", Consumer Electronics, Asia ICCE-Asia, 2016

Author Biography

Mihir Mody is Senior Principal Architect for Automotive business at Texas Instruments (TI) Incorporated. His domain of interest are video coding, image processing, computer vision & machine/deep learning algorithms with focus on embedded HW & SW solution. He holds Master of Engineering (ME) degree from Indian Institute of Science (IISc) in 2000 and Bachelor of Engineering (BE) from GCOEP in 1998.

Manu Mathew is a Principal Engineer for Automotive business at Texas Instruments (TI) Incorporated. His areas of interest include computer vision, machine learning and video compression algorithms for real-time embedded systems. He holds a master’s degree (ME) from Indian Institute of Science (IISc), Bangalore in 2000 and a Bachelor of Technology (B.Tech) from Regional Engineering College (currently National Institute of Technology), Calicut in 1997.

Shyam Jagannathan is Senior Technical Lead for Automotive business at Texas Instruments (TI) Incorporated. His areas of interest are video compression, image processing, computer vision and machine learning with expertise is real time embedded systems. He holds a master’s degree (MS) from Illinois Institute of Technology, Chicago in 2014 and Bachelor of Engineering (BE) from R.V. Engineering College, Bangalore in 2005