CE4052/ES6152 Project Report – 2015

Evaluation of TMS320C6678 DSP's Performance for Deeplearning Algorithms

Team Members: *Nachiappan Ramasamy, Gourav Modi,Ragunathan Mahesh Kumar*

Nanyang Technological University
Singapore

Chapter 1

Abstract

Deep learning-based object recognition using Convolutional Neural Network (CNN) has recently emerged as one of the leading approaches for machine learning. Currently, most of the deep learning algorithm implementations run only on high-performance server machines at the back-end. The implementation of this algorithm in embedded device is highly attractive since it would reduce network dependency and latency and thereby improve the overall response time for soft-real time applications such as Voice Assistant Systems, Advanced Driver Assistant Systems in automobiles, Image Recognition Systems to name a few. With passing years, the detection accuracy of networks are improving and more divergent object classes are being added to existing networks through learning transfer. Therefore, the easy availability of trained models and widely available bench marking tools will reduce the training time of the models to almost zero.This would enable us to exhaustively use such highly desirable features in mobile and embedded devices. In this work, we have implemented the fore-mentioned network on TI DSP TMS320C6678 platform using ImageLib library and did the Performance vs Power analysis. Our implementation has outperformed all the comparable in the platform while consuming slightly more power. We achieved a performance speed up factor of 5.24x when compared to Parallela, 19.4x with MXP and 46.6x increase with Jetson TK1. The cost borne is in the area of power where the same platform consumes 1.8x more power than Jetson TK1, 4.25x more than MXP and 8.45x more than Parallela.

Chapter 2

Introduction

Embedded systems are getting cheaper and faster at the same time due to technological advancement. This enables more and more computationally intensive algorithms to run on embedded platforms. The main challenge in this domain comes due to constraints involved in terms of cost, power and performance of the systems. A simple deep learning algorithm with image size of 28x28 requires around 2.5 million multiplications, 5 million additions with high memory bandwidth requirement such as 1900 MB/s. CNN requires repetitive multiplication and addition operations hence platforms which supports SIMD processor is more suitable. The platform which we will use for implementing this computational extensive algorithm is TI DSP board which exploits the underlying multi-core CPU organization with 8 cores and up to 8 MB of on-chip memory. The on-chip memory is basically divided in to L2 SRAM of 512Kb per core and a global MSMC RAM of 4MB which is shared among all the cores. And it has multicore navigator pheripheral of bandwidth 1Tbps which might be utilised in case we want ot move the date in between the cores. Since our application doesn't really requires a RTOS serives, we ran our applicaiton over bare metal to reduce the code space and latency. In our application We have manually created different memory sections in MSMCRAM and L2 SRAM for our memory requirement.

Our main work includes:

- Implementation of Deep learning algorithm in TI DSP in all 8 cores
- Optimizing memory bandwidth
- optimizing the performance with IMGLIB and DSPLIB
- Performance and power usage of TI DSP under different configurations

Chapter 3

Motivation

Artificial vision systems have diversified applications in the fields of autonomous robots, security systems, micro-UAV's and more recently, mobile phones and automobiles. The algorithm running behind these applications should have the capability to recognize objects with a high degree of accuracy while being able to execute in soft real-time. Many recent algorithms which have shown promise for use in visual understanding is CNN which can perform robust feature extractions for recognition, and scene detection and understanding. However, CNN's are computationally very expensive and require high performance computers or graphics processing units (GPU's) to run in real time, hence for low-power platforms like smart-phones, computations are usually sent to off-site servers systems for processing but this requires constant and reliable connectivity with the off-site server which is not a guarantee. This constraint the limits of the use of CNN's in mobile environments where very predictable performance is allowed between the input and the result. Although, mobile processors have increasingly become smart and powerful but their real time performance is application dependent and thereby may result into under-utilization of hardware resources due to cache misses, frequent, unpredictable branches and a large amount of memory accesses. In this work, our software implementation allows for maximum utilization of the underlying hardware thereby giving a peak performance.

Chapter 4

Methodology

We have implemented Convolutional Neural Network on TI DSP platform as mentioned in the prior sections. The network takes images from MNIST database which is subjected to different operations of feature extraction and re-construction of images. The array of operations performed are convolution for feature extraction, dilation for reducing the effect of noise, sub-sampling for image compression, weighted-add for feature pooling.
The structure of our network has one input layer which is fed with 28X28 MNIST Gray scale image, 2 hidden layers for deep learning and fully connected output layer for classification. The images and all the intermediate outputs are taken to be 16 bits and signed. The first layer (L1) has 50 maps with convolution performed for a matrix size of 7x7, dilation and sub-sampling whereas the second layer(L2) has 128 maps with convolution performed for a matrix size of 5x5, dilation, weighted-add and sub-sampling. The interconnection between these layers are done using 10 random connections per map. The algorithm uses Rectified Linear Unit (ReLU) activation function for introducing non-linearity into the system for better feature extraction and mapping. This is achieved by making the negative values of pixels to zero. All the kernels required for convolution and feature extraction are randomized.
We have explored several strategies to achieve the optimal performance in terms of a trade-off between peak performance and power consumption which is described in details in the result section. The input image (28x28) occupies 12Kb of space and is kept in L2 SRAM of size 512Kb for each core. All the cores are given designated space under critical section in MSMC with only write access to their respective areas in-order to prevent memory corruption. The intermediate maps from the two hidden layers are stored in MSMC which are used during the communication from L1 and L2 layers. Therefore, an inter-communication between cores are achieved using MSMC as a shared memory by giving read access to other cores. We have implemented the pointer reference methodology for referencing these MSMC locations which resides in DDR3 RAM. These pointer references are fetched only once during the start of the algorithm.
The operations CNN involves is multiple additions and multiplications which is accelerated using ImageLib and DSP optimized library. The library exploits Single Instruction Multiple Data (SIMD) units by parallelizing instructions and doing vectorization.
To achieve the peak performance we have divided the entire CNN operations among all 8 cores in a balanced way to achieve maximum performance. As mentioned earlier, the number of maps in L1 is 50, therefore, we have divided it into 6 per cores with extra map taken by core 0 and 1 each. Moreover, all the operations corresponding to each map in L1 are operated in all the cores. Since, core 0 and 1 performs more operations, hence it consumes more time units which is observed in later sections. Furthermore, in L2, each core performs operation corresponding to 16 maps. Thus,

this defines the logic bifurcation among the cores to achieve the best performance. The intention of our experiment is to compare metrics such as power and performance with other platforms. The total time taken by the CNN network in TI DSP platform is 685 microseconds whereas the power consumed is 16.9 watts. The power is measured under different configuration by power gating modules such as DDR3 RAM, Unused peripherals such as Network Co-processor, SRIO, PCIe, Hyperlink and Packet accelerators. It was observed that even after switching off certain modules the overall power gain was not significant since the base line power of the board is 11.5 Watts. The trade-off between power and timing was also noted with varying frequency of operations to find the optimal point where both the metrics meet. Similar analysis for power and timing was also performed by running the algorithm in 1,2,4 and 8 cores configuration and measuring corresponding power consumed which is shown in detail in the following section.

Chapter 5

Results

Our implementation of CNN with the above mentioned design has approximately two and half million multiplications and 315168 additions operations in the convolution stages of layer1 and layer2. TMS320C6678 has 2 multipliers unit along with 6 arithmetic units running a 4 way SIMD design providing up to 32 fixed point 16x16 MACs/CPU cycle.

5.0.1 Performance Achievements

Thus, for an input image of resolution 28x28 and for the parameters stated in the previous section, we achieved the computation time of 685 microseconds. However, as seen from the figure 5.1 – we could observe that the computation time would increase to a maximum of 1506 microseconds when none of the optimized libraries like Imglib, DSPlib are used. In addition to the optimized libraries being used, parallelization of the algorithm across 8 cores of DSP has resulted in 8x performance when compared with unicore processor.

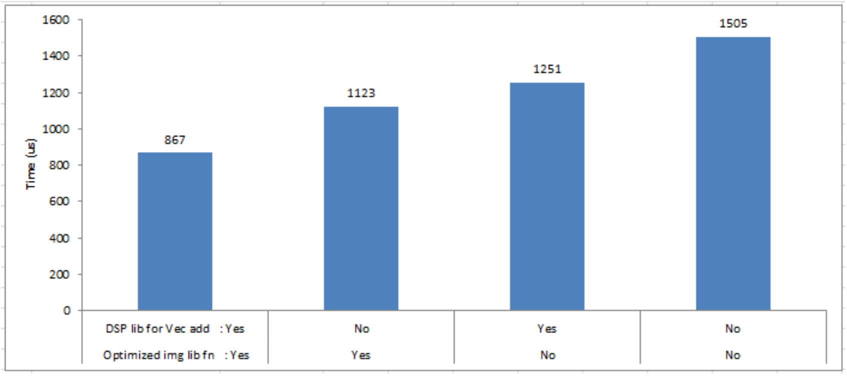

Figure 5.1: Performance of TI DSP using optimized libraries

The overall timing for performing deep learning can be broadly accounted for Operation Layer 1 and 2 which in turn can be subdivided and mapped to the basic image processing functions which are shown in figure 5.2:

The total time taken for operate layer 1 is 253 microseconds and for operate layer 2 is 468 microseconds. From the figure 2, it is evident that in both the layers, dilate function accounts for majority of the timing. This is because dilate is the only function that has been constructed using 'c' normal code whereas all the other functions are SIMD optimized in Imglib library. Power and the performance are the two important driving criteria for an efficient hardware platform. Having

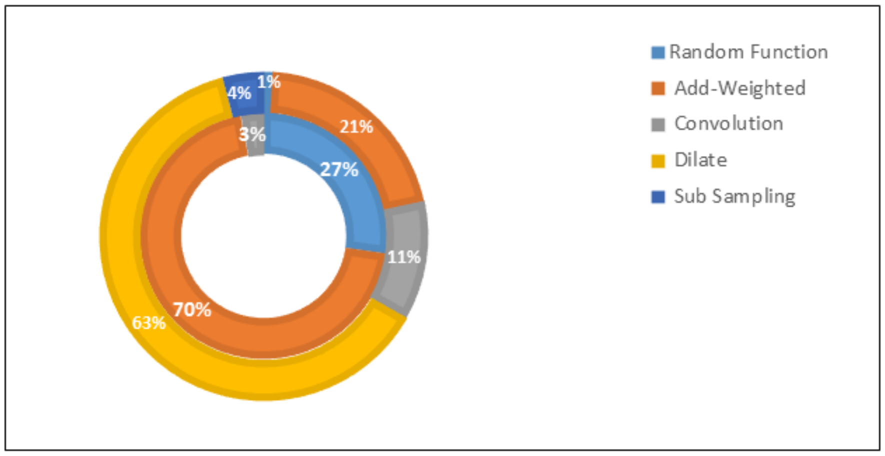

Figure 5.2: L1 and L2 computation breakup

achieved the performance, the below plots 5.3 and 5.4 1 and 2 explains the 'power vs frequency' and 'power vs no. of cores' in use of the TI DSP for deeplearning computation.

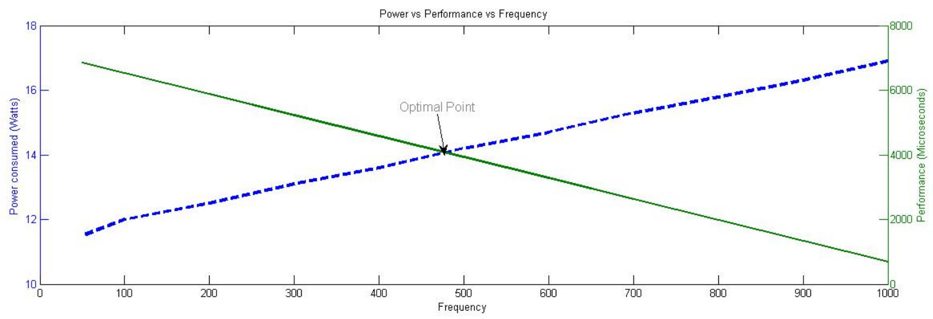

Figure 5.3: Power vs Frequency vs Performance

The above plot exhibits linear relation of power and performance for varying frequencies. This is justified because our algorithm is CPU intensive and does not depend upon any IO or shared resources.

Since, the power consumed by the DSP is very high, we exploited the option of reducing the power by utilizing less number of cores. Though this technique has an overhead in terms of performance, we tried and found out the optimum point that would be beneficial under power and performance constraints.

Comparisons of Energy and performance of TI DSP with other hardware platforms

From the above figure, it can be seen that for deep learning computation, parallela consumes only 7.2 Wms of energy and TI DSP consumes 11Wms. This proves that TI DSP is second best energy efficient board (among the listed) for deeplearning computation.

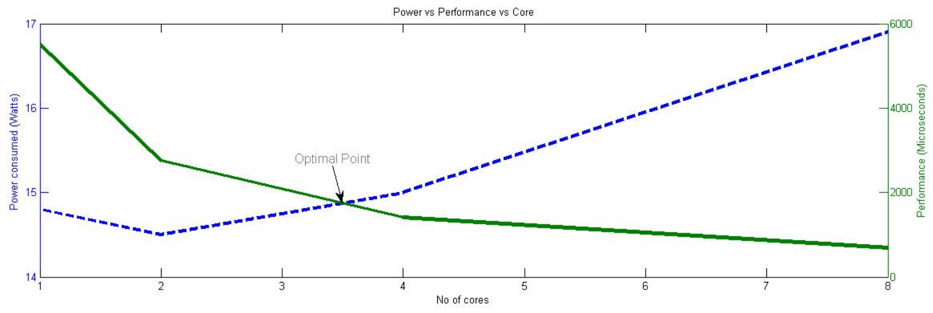

Figure 5.4: Power vs No. of cores vs Performance

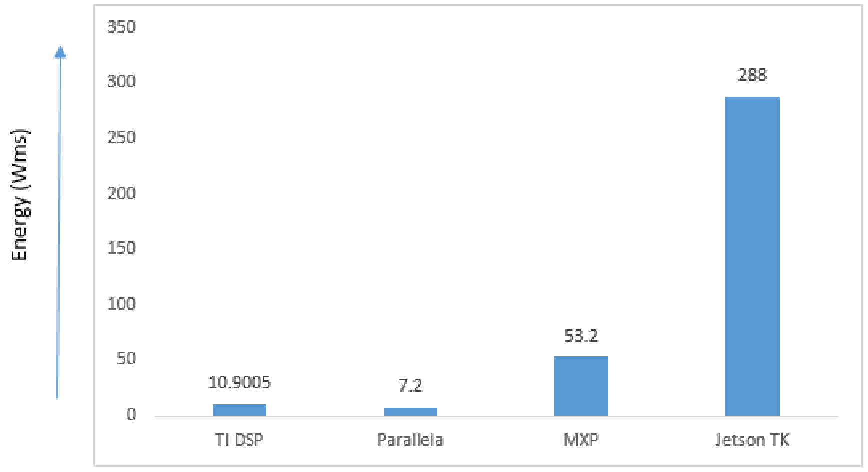

Figure 5.5: Comparisons with different architectures

Chapter 6

Conclusion

The problem statement of performing compute intensive task pertaining to CNN on embedded platforms is achievable by using embedded board such as TI DSP which gives a lot of processing power and large on-chip memory. We found out that by exploiting the underlying hardware layer completely a drastic improvement in overall performance is with processing time ranging in microseconds whereas the boards mentioned earlier looses with the operation taking in milliseconds. . The assumption that we have taken is that the image is 28x28 size while comparing it withachieved other embedded platforms. Since, the image size was low, the entire image was accomodated inside the L@ SRAM itself. In the future work, will try to process Full HD images/videos by extending our design by processing images/frames patch by patch each of size 28x28. Another approach which can be exploited is to try a serial processing of the images with each core dedicated for a one of set of operations and propagating the output to the next core.

Chapter 7

References

[1] *http : //www.embedded − vision.com/summit/resources/may −* 2015
[2] *http : //www.embedded − vision.com/platinum − members/synopsys/embedded − vision − training/videos/pages/may −* 2015 *− embedded − vision − summit*
[3] *http : //www.computervisionblog.com/*2015/03/*mobileyes−quest−to−put−deep−learning.html*
[5] *http : //www.technologyreview.com/news/*534736/*deep − learning − squeezed − onto − a − phone/*
[6] *http : //www.techdesignforums.com/practice/technique/embedded−vision−object−detection−convolutional − neural − network/*
[7] *http : //www.teradeep.com/papers/mwscas −* 2014*.pdf*
[8] *http : //deeplearning.net/tutorial/lenet.html*
[9] *http : //cdn.intechopen.com/pdfs − wm/*9274*.pdf*