Performance Analysis of Deep Learning Based Image Compression Algorithms on Xilinx FPGAs

SRI JANANI RENGARAJAN DEBRAJ DAS
sjre | debrajd @kth.se

October 25, 2020

Abstract: With the advancements of imaging technologies, there is a variety of image types available in the market like VR and stereoscopic images. To efficiently compress these, advanced methodologies are needed. Autoencoder models have shown good compression sizes and quality comparable to JPEG, a traditional and extensively used compression algorithm. In this study, the latency of compression for Autoencoder model is analysed to determine expected frame rates in a Field Programmable Gate Array (FPGA). An end-to-end Deep Learning (DL) based image compression algorithm using Convolutional Autoencoder (CAE) is implemented and deployed on FPGA to understand the performance of custom image compression algorithms on Edge devices. The proposed algorithm's compression quality is analysed using the metrics - Peak Signal-to-Noise Ratio (PSNR) and Multi-Scale Structural Similarity Index (MS-SSIM). The hardware performance profiles are extracted showing the achievable quality and performance for deploying the compression algorithm on Xilinx® Deep Learning Processor Units (DPUs).

Keywords: Autoencoder, FPGA, Image compression.

Contents

1 Introduction

Storing or transferring raw images incurs a lot of data transmission expenses in terms of storage resources, energy requirements, and network bandwidth usage. Traditional Encoder and/or Decoders (CODECs) like JPEG [1] and JPEG2000 [2] can compress the data, but with a cost in image quality. Whereas, DL based image compression algorithms are established to have at-par or above par image compression capabilities with higher quality [3]. The rationale behind this work is to reduce network and storage resource utilization on mobile platforms by reducing the size of commonly shared data, images, at a better quality than traditional CODECs, and provide real-time (30 Frames per second (FPS)) compressed streaming capability. This can prolong the battery operation time of mobile devices while streaming videos and reduce network congestion and energy consumption to transfer images.

Along with improvements in DL model complexity and accuracy, there have been significant improvements in their hardware implementation. Graphics Processing Units (GPUs) are the preferred choice for DL applications over Central Processing Units (CPUs), however, ”with software-hardware co-design, FPGA can achieve more than 10x better speed and energy efficiency than state-of-the-art GPU” [4]. As FPGAs are a promising alternative for GPUs and CPUs, we use them for the hardware deployment of DL based image compression and understand the degree of speedup (latency) achievable in this use-case. Implementation using ASICs is ignored in this work due to budget constraints. The primary focus is on Xilinx’s product DPU, offering substantial DL accelerations [5], for rapid prototyping of image compression algorithms on an FPGA. A potential benefit of such an implementation would be to embed the Intellectual Property (IP) along with the Camera Controllers to compress the image at the source and open up avenues for compression of a variety of image types, image processing like denoising, and greater control over copyrighted data by controlling the distribution of the decoder.

2 Theoretical Framework

This project was inspired by Deep Learning for lossy image compression in real-world applications [6] and using CAEs for image compression [7]. These two papers explored compression techniques with critical real-time constraints on desktop-grade CPU and GPU. The authors assessed the visual quality of compression using metrics like Mean Squared Error (MSE), PSNR [8], Structural Similarity Index (SSIM) [9] and for the hardware performance - time, CPU load and throughput were used as performance metrics. The following sections discuss the current state of research related to learning based compression and hardware implementation of neural networks.

2.1 Deep Learning-based image compression

In [7] the authors demonstrated a novel CAE architecture using an approximate rate-distortion loss function that outperformed JPEG2000 by 13.7% decreased Bjontegaard-Delta (BD) rate, on the Kodak database [10]. They have used Parametric Rectified Linear Unit (PReLU) activation function with *Adam* optimizer, and also have shown that PReLU improves the quality of the compression when compared to Rectified Linear Unit (ReLU).

The future work of Z. Cheng, et al. [7] led to [11] using MS-SSIM metric for checking the image quality post-compression and the utilization of Generative Adversarial Network (GAN) for image compression. Z. Cheng, et al. [7] [11] compared the compression performance of CAE, GAN, and Super Resolution (SR) methods. Their results indicate that GAN and CAE architectures are better than JPEG. In terms of metrics, GAN and CAE have similar PSNR but GAN has better MS-SSIM value compared to CAE. In [6], the authors propose a network architecture called Convolutional Residual Auto Encoder (CRAE), which is similar to [7] with additional residual layers. The residual layers are used to speed-up the convergence. Also, they didn’t add a quantizer or an entropy block between the encoder and decoder which are generally present in a CAE model used for compression. From the available literature, it can be understood that CAE architecture is the way forward for DL based image compression.

2.2 DNN Accelerator

Deploying trained neural networks on an embedded platform can be costly due to hardware constraints. Complicated models can have parameters worth of 500+ MBs like that of VGG-16 which is both memory and computationally extensive [12]. Multiple techniques have been implemented to optimize computation and memory DL models for inference on FPGA deployments. Methods like Deep Compression consisting of pruning and quantization with weight sharing are developed to reduce the size of the models by 40x [12]. By reducing the number of weights used to process a layer in the model the memory data traffic is also reduced. DL's rapid algorithmic advancements need the hardware to be in line with the computation required, it is observed custom hardware like FPGA and ASIP [13] [14] supports DL inference better than GPU with throughput, latency and energy requirements. Hence, DPU is chosen to deploy the image compression algorithm.

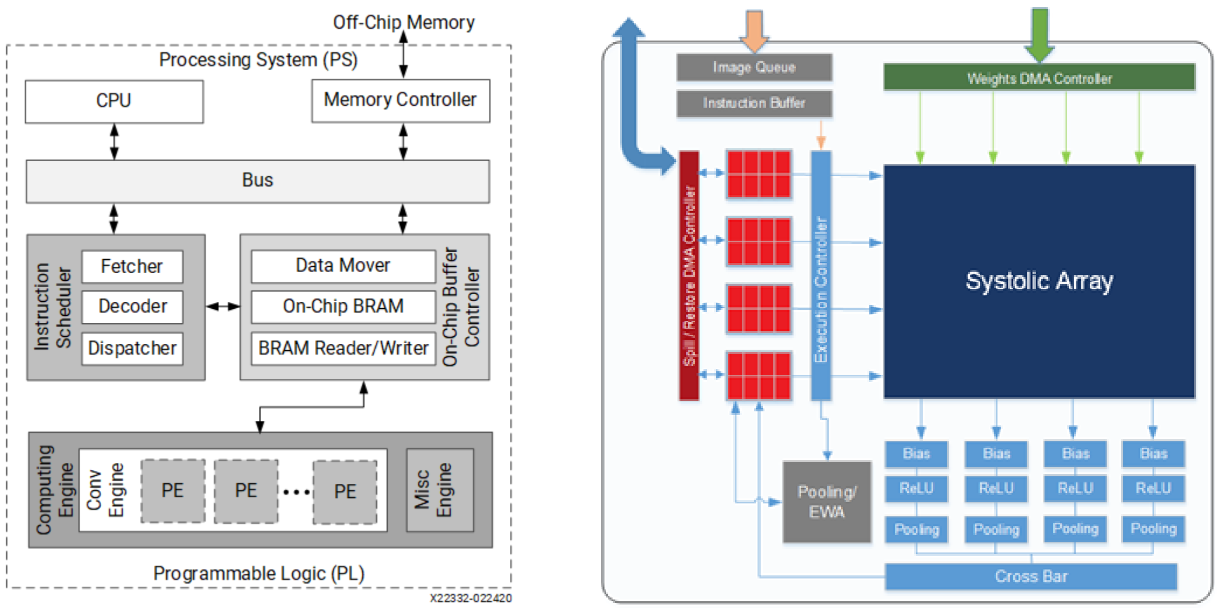
Xilinx has demonstrated almost double throughput, latency, and energy optimizations compared to CPU and GPU implementations by using their xDNN IP on their Alveo accelerator cards[15]. Given the performance benefits demonstrated by Xilinx's IP, a scaled-down version of Xilinx's Deep Neural Network Development Kit (DNNDK^{TM}), leveraging DPU customized for edge devices on Ultra96v2 board [16] will be used to deploy the autoencoder model. The models will be designed to run at 150 MHz which is the maximum safe frequency to ensure timing closure in Xilinx Zynq UltraScale+ MPSoC ZU3EG A484 (MPSoC). Table 1 shows the available DPU architecture provided by Xilinx, and their configuration and peak operations per second. B512, B1024, and B2304 are chosen for analysis, since they are spread out uniformly in terms of operations per second and the IPs greater than B2304 require FPGAs with greater programmable logic area than MPSoC.

Convolution Architecture	Pixel Parallelism	Input Channel Parallelism	Output Channel Parallelism	Peak operations per clock
B512	4	8	8	512
B800	4	10	10	800
B1024	8	8	8	1024
B1152	4	12	12	1150
B1600	8	10	10	1600
B2304	8	12	12	2304
B3136	8	14	14	3136
B4096	8	16	16	4096

Table 1: DPU architectures and channel configurations [17]

2.3 DPU Hardware Functioning

The DPU follows a custom tensor-level instruction set processed by the Instruction Scheduler [18] in Figure 1a and consists of Processing Engines and Miscellaneous Engines. These processing engines incorporate multiple sequences of Digital Signal Processing (DSP) blocks, multipliers, adders, and accumulators combined to form a Systolic array [19] as shown in Figure 1b. The IP can accelerate Convolutions, Maxpooling, and Batch Normalization layers [20]. The weights and input data are coordinated by the On-Chip Buffer controller with guidance from the instructions. The architecture is deeply pipelined and through the combined action of the Instruction Scheduler and the On-Chip Buffer controller the Computing can reuse the weights and have the data processed at the fastest rate possible. Moreover, the IP works parallel to an on-chip CPU which can either coordinate the inference activities for more complex designs or it is free to process other tasks.



(a) DPU Hardware Architecture (b) DPU Functional Components

Figure 1: DPU Hardware design, both images reproduced from [17]

2.4 Compilation for DPU

Xilinx has developed a set of tools to enable hardware deployment as shown in Figure 2a. The tools help to take a trained model from Tensorflow down to a special set of instructions and here the flow of tools and steps are described. The model as trained needs to prepared to be ready for deployment as follows:

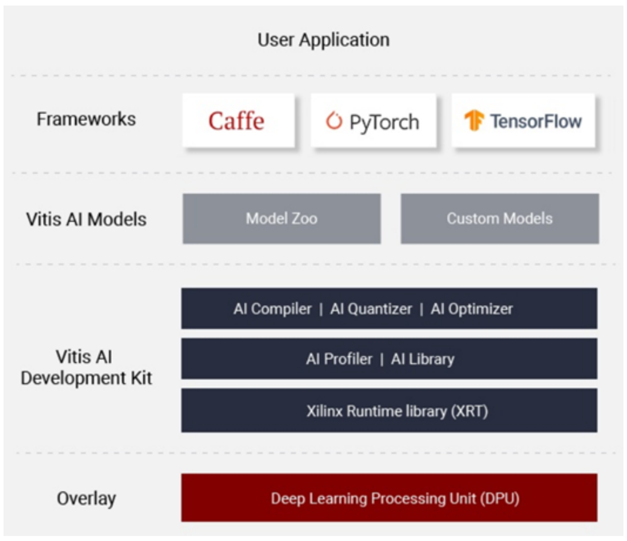
1. Freezing the layers - To remove any training specific layers like Dropout layers.
2. Pruning - Not performed due to license requirements of Vitis AI Optimizer tool.
3. Quantization - To reduce the number of weights and enable weight sharing.
4. Compilation - To generate assembly code for the DPU to utilize during runtime.

Compilation of the layer generates a DPU specific assembly code which is used in the runtime to fetch the weights, activations and coordinate the processing in the DPU. The compilation is done as shown in Figure 2b. The quantized model is parsed and fed to the optimizer which merges layers and organizes the model layers such that the data movement is reduced. The optimized data is used by the code generator to be compiled into the assembly code.

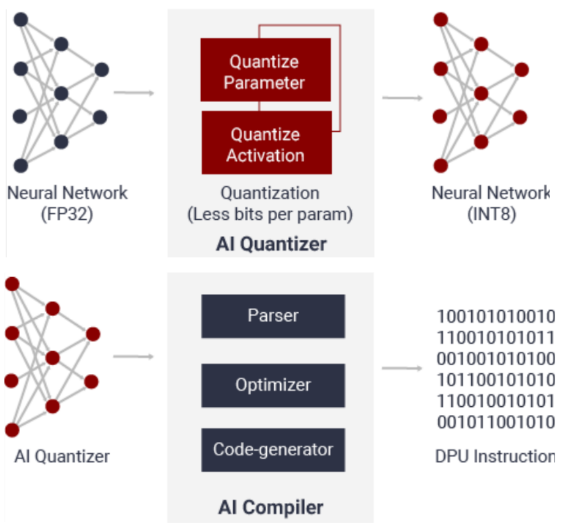
3 Research Questions

The proposed research is carried out to answer the following questions:

- How to compress images using Deep Neural Networks (DNNs)?
- How to deploy image compression algorithms developed using neural networks on FPGA?
- What latency to expect on FPGA for Image Compression?
- What would be the quality of the image compression and latency when the Deep Learning (DL) based image compression is run on FPGAs for Edge devices?



(a) Vitis AI toolchain



(b) Vitis AI model preparation

Figure 2: Vitis AI compiler toolset, both images reproduced from [17]

4 Hypothesis

DL based image compression algorithms optimized for hardware implementation would potentially degrade the quality of the image post-compression. There would be an improvement to the latency of images processed when deployed on DPU when compared to deployment on CPU or GPU.

5 Methodology

In order to extract performance and quality metrics of DL based image compression, a CAE model is implemented based on the literature study. Xilinx provides a set of DPU architectures with different capabilities. After identifying the best latency vs FPGA utilization, a set of hardware configuration is generated and prepared to utilize the CAE model and extract latency data. The methodology followed in this project can be divided into two major components: software and hardware implementations. Each implementation is explained in detail in the following subsections.

5.1 Software implementation

Software implementation workflow begins with data pre-processing, defining the CAE model, training and validation of the model. Once the model is trained, the quality of compression offered by the model is evaluated on standard test images based on quality metrics such as PSNR and MS-SSIM. *Tensorflow version 1.15* is used as the backend for all the operations performed as it is compatible with the hardware environment being tested in this work. Other major dependencies include *OpenCV version 4.1* and *Python 3.6.9*

5.1.1 Data pre-processing

The input images are obtained from Challenge on Learned Image Compression (CLIC) hosted by Conference on Computer Vision and Pattern Recognition (CVPR) [21], [22]. The total number of training and validation images used are 1834 and 161 respectively. All the images are of High Resolution (HR) with a varying number of pixels. Images used for testing are from the Kodak database [10]- uncompressed images of dimensions 512 (height) x 768 (width), they are used as standard images to test compression algorithms across the literature. The images have three colour channels- Red, Green, Blue (R, G, B) and have pixel intensity values between 0 and 255. Through the literature study, it was understood that while using images as input data for DL applications, a common approach followed is to have images to be of the same dimensions (height and width) and pixel intensity values to be in the range of 0 to 1. In line with these approaches from the literature, the images are cropped to the dimensions- 512 x 768 and pixel intensity values are normalized to the range of 0 to 1. The dimensions 512 x 768 were chosen in accordance with the dimensions of the test images. Pixel normalization is a standard method to prevent the numerical training weights of the model going out of bounds and also to have a faster convergence of weights while training the network.

5.1.2 CAE architecture and training

The CAE model consists of two sub-models- *encoder* and *decoder* that are concatenated one after the other to form the *autoencoder*. The proposed CAE architecture can be visualized as presented in Figure 4. Each layer in the encoder model convolves the input data with a 3 x 3 filter either 32 or 64 number of times, thereby creating 32 or 64 feature maps of the input data. Figure 3 shows two-dimensional convolution and feature map creation of an image with three colour channels (R, G, B). Each filter has weights associated with it and they are the trainable parameters of the model. The decoder performs inverse convolution with 32 or 64 number of filters, each of dimension 3 x 3. The encoder takes in an image matrix of dimensions 512 x 768 x 3 (height x weight x number of channels) and encodes it into a multi-dimensional feature matrix of dimensions 62 x 94 x 32. This feature matrix is reconstructed back into an image (512 x 768 x 3) using the decoder. This compression technique has information loss.

The proposed CAE model is trained for several epochs and validated on the validation dataset at the end of each epoch. Epochs refer to the number of times the learning algorithm is trained on the whole training dataset and it is a hyper-parameter. Epoch is an integer value, in literature it is usually set to 10, 50, 100 or 1000 and the training dataset is shuffled before the start of each epoch. In this work, the model is trained for 50 epochs and the validation accuracy is computed at the end of each epoch. During training, the validation accuracy is monitored. The model, as defined by its parameters, that offers the highest validation accuracy is saved for hardware deployment.

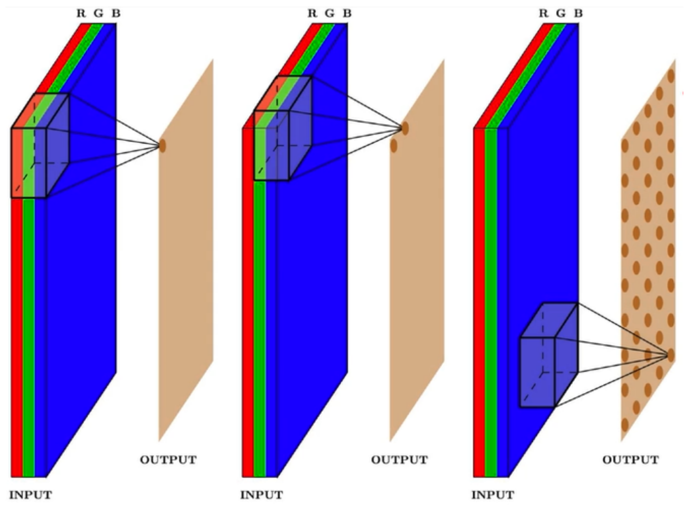
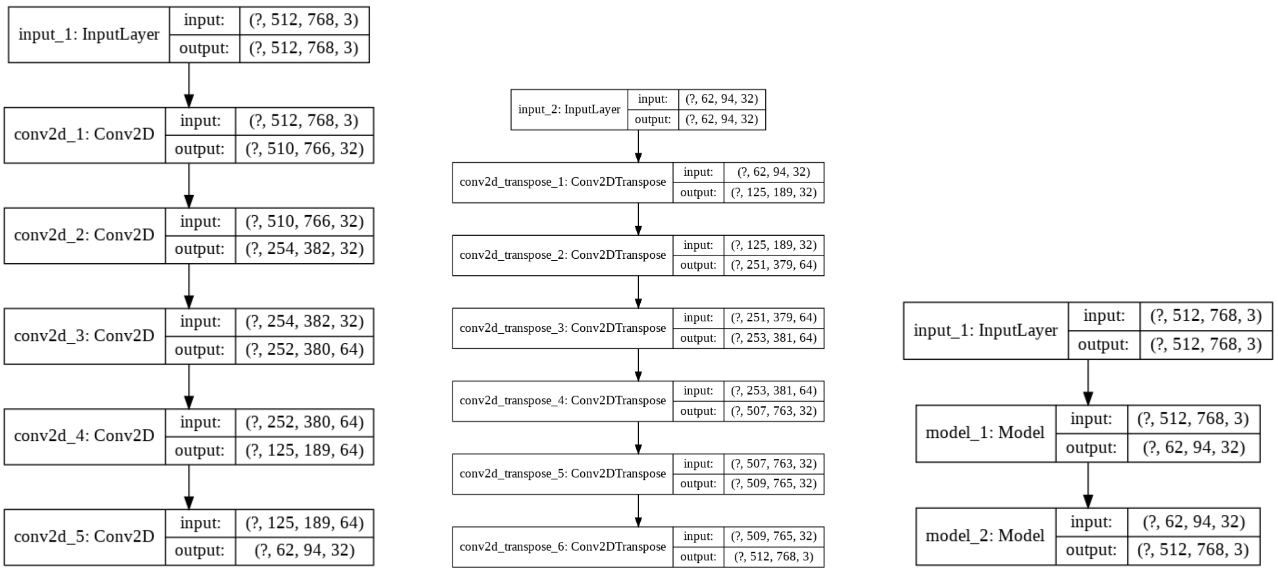


Figure 3: 2D Image convolution with a single filter reproduced from [23]

5.1.3 Analysis of compression quality

Commonly used metrics to check the compression quality are PSNR and MS-SSIM. *Scikit-Image* library offers APIs to compute the said metrics with ease. The PSNR metric gives the peak signal-to-noise ratio between two images in decibels. It makes use of the differences in the pixel values between the images. MS-SSIM computes the structural differences between the images and it is based on the structural information of the image. Structural information is a concept that there are strong interdependencies between pixels that


input_1: InputLayer | input: (?, 512, 768, 3) | output: (?, 512, 768, 3)
conv2d_1: Conv2D | input: (?, 512, 768, 3) | output: (?, 510, 766, 32)
conv2d_2: Conv2D | input: (?, 510, 766, 32) | output: (?, 254, 382, 32)
conv2d_3: Conv2D | input: (?, 254, 382, 32) | output: (?, 252, 380, 64)
conv2d_4: Conv2D | input: (?, 252, 380, 64) | output: (?, 125, 189, 64)
conv2d_5: Conv2D | input: (?, 125, 189, 64) | output: (?, 62, 94, 32)
input_2: InputLayer | input: (?, 62, 94, 32) | output: (?, 62, 94, 32)
conv2d_transpose_1: Conv2DTranspose | input: (?, 62, 94, 32) | output: (?, 125, 189, 32)
conv2d_transpose_2: Conv2DTranspose | input: (?, 125, 189, 32) | output: (?, 251, 379, 64)
conv2d_transpose_3: Conv2DTranspose | input: (?, 251, 379, 64) | output: (?, 253, 381, 64)
conv2d_transpose_4: Conv2DTranspose | input: (?, 253, 381, 64) | output: (?, 507, 763, 32)
conv2d_transpose_5: Conv2DTranspose | input: (?, 507, 763, 32) | output: (?, 509, 765, 32)
conv2d_transpose_6: Conv2DTranspose | input: (?, 509, 765, 32) | output: (?, 512, 768, 3)
input_1: InputLayer | input: (?, 512, 768, 3) | output: (?, 512, 768, 3)
model_1: Model | input: (?, 512, 768, 3) | output: (?, 62, 94, 32)
model_2: Model | input: (?, 62, 94, 32) | output: (?, 512, 768, 3)

(a) Encoder (b) Decoder (c) Autoencoder

Figure 4: Convolutional Autoencoder architecture

are spatially close. MS-SSIM exploits this concept when comparing two images for similarity. MS-SSIM values are typically between 0 to 1. 0 indicating that there is no structural similarity between images whereas 1 indicates identical images. The trained CAE model is tested on the Kodak test dataset, PSNR and MS-SSIM metrics are computed between the input and output images in order to understand the compression quality of the proposed methodology.

5.2 Hardware Methodology

The model as designed and trained using Tensorflow will be passed through a series of processes, as discussed in section 2.4, to quantize and compile it into Xilinx's runtime assembly instructions. This assembly code will be deployed to the FPGA platform which incorporates the Xilinx DPU. The hardware will be running a PetaLinux Operating System which can be accessed over the WiFi connection available. The process is initiated via Secure Shell (SSH) and data is collected. This process is repeated by varying the hardware configuration and the program is run and profiled to extract the latency metrics.

5.2.1 Hardware Customization

Xilinx provides a set of DPU architectures as a customizable IP. The module was incorporated into the FPGA design utilizing the Xilinx Vitis model compiler which prepares the hardware architecture. PetaLinux [24] is used to provide a Linux Operating System to enable the programs to run and have interaction with the hardware DPU through the runtime libraries.

5.2.2 Model Deployment

The trained model was exported from Tensorflow in Hierarchical Data Format ver. 5 (HDF5) format. The HDF5 file was processed to freeze the layers and fed to VAI_Q_Tensorflow quantizer available in Xilinx's DNNDK^{TM} toolset. The quantized model was calibrated using images to ensure the weights are scaled correctly and the quantization has minimal effect on the model's weights. The calibrated model was then fed to VAI_C_Tensorflow available in Xilinx's DNNDK^{TM} toolset to generate the assembly instructions for the DPU to use during the processing of the image on Ultra96-V2.

5.2.3 Performance Metrics

The total runtime of the compression algorithm when deployed on the hardware and the processing time of the model in the DPU is measured using the profiling tools available in DNNDK^{TM} libraries. The runtime to process images i.e. image preparation (comprising of normalization) and image processing on DPU is averaged over multiple samples per hardware configurations. The workload on the layers is measured as Multiply and Accumulate (MAC) Operations (MOP) indicating two operations. The memory size for the node is recorded in Megabits (MB). The runtime of the layer is recorded per node in ms. The complete program execution time is ignored since it was implemented using Python which is known to be slower than C/C++ as it is an interpreted language, also our objective is to primarily analyse the IP. The operations per layer are recorded as Giga operations per second (GOPS). The processing capability utilization is also recorded as a percentage. And the throughput of the nodes is recorded as MB/s. The workload, memory utilization and runtime are totalled, whereas the operations, utilization and throughput are averaged. It should also be noted that some workload, memory and runtime is shared between the layers due to pipelining. The FPS is calculated as per equation 1.

$$FPS = \frac{1000}{runtime(ms)} \tag{1}$$

6 Results

The accuracy and loss incurred during 50 epochs of training are plotted in Figures 5(a) and 5(b) respectively. The maximum and minimum values of accuracy and loss are presented in Table 2. The trained CAE is tested with 17 images, which are of dimensions 512 x 768, from the Kodak dataset. The images are encoded and reconstructed using the decoder. The decoded images are compared with original images using PSNR and MS-SSIM values, these values convey the similarity between the images and are presented in Figure 6. The compression ratio regarding representation is constant.

$$compression\_ratio = \frac{compressed\_image}{uncompressed\_image} = \frac{62*94*32*(4byte)}{764*512*3*(1byte)} = 0.6356$$

The array output from the autoencoder model was compressed using 7zip utility and the resulting file had a size of 108.4 kb. When the same images were compressed using JPG using GNU Image manipulation tool the file had a size of 172.1 kb retaining 90% of quality and compressing further using 7zip utility there was no reduction in size. Following is the result in the case of least compression improvement.

$$compression\_improvement = \frac{compressed\_filesize}{JPG\_filesize} = \frac{108.4kb}{172.1kb} = 0.6298$$

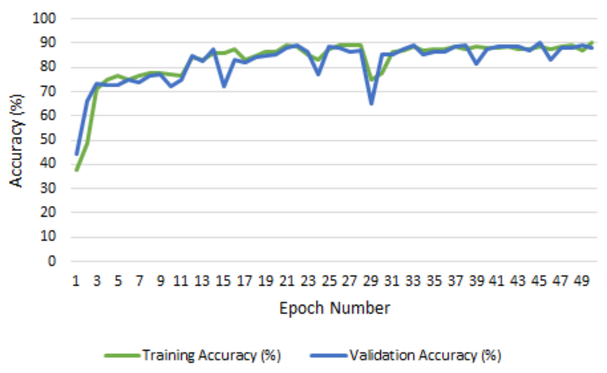
Quantity	Maximum	Minimum
Training accuracy	0.9018	0.3753
Training loss	0.0311	0.0012
Validation accuracy	0.9005	0.4451
Validation loss	0.0128	0.0013

Table 2: Accuracy and loss values

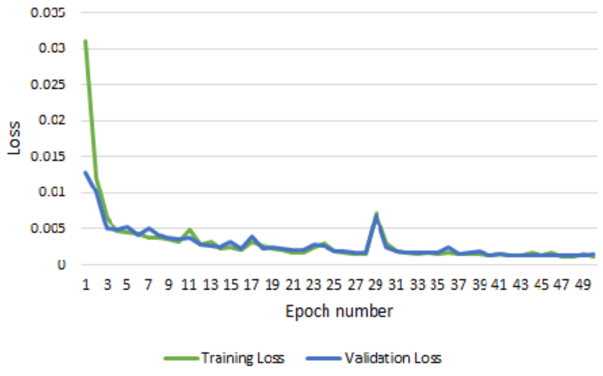
6.1 On-Device results

Tables 3, 4, and 5 show the detailed layer by layer processing information and profiling as extracted using Vitis-AI provided profiling tools. The data was collected while running the model in debug mode.

From the Tables 3, 4, and 5 the corresponding average latency of the hardware architectures are calculated in Table 6.

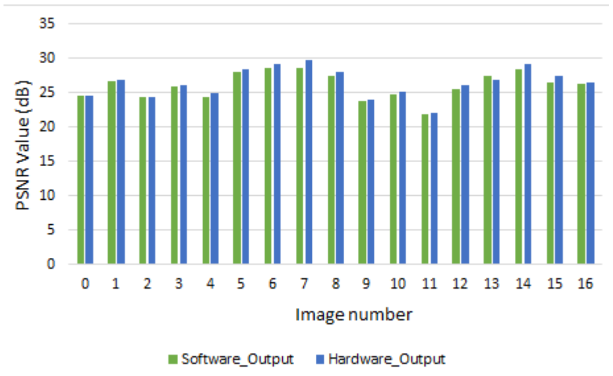


(a) Training and Validation Accuracy

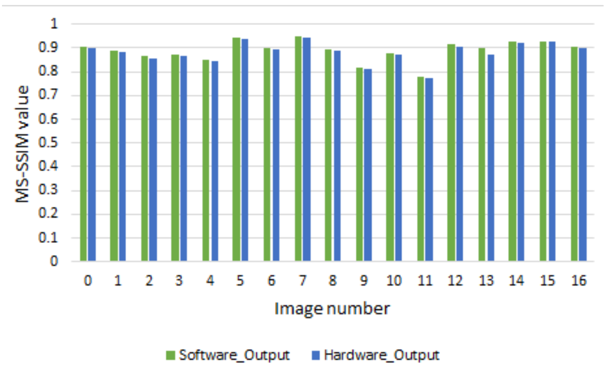


(b) Training and Validation Loss

Figure 5: Evolution of accuracy and loss



(a) PSNR values



(b) MS-SSIM values

Figure 6: Compression quality metrics for Kodak test images

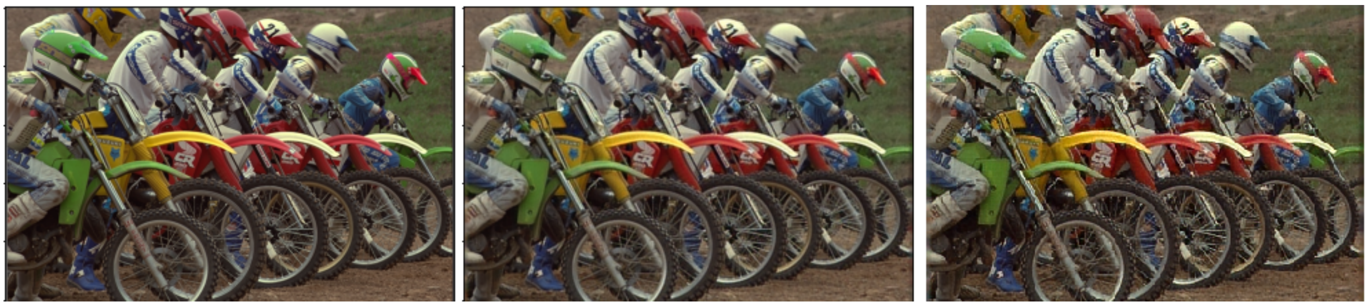


Figure 7: Image 5- Original(left), CAE software output (middle) and output from FPGA(right)

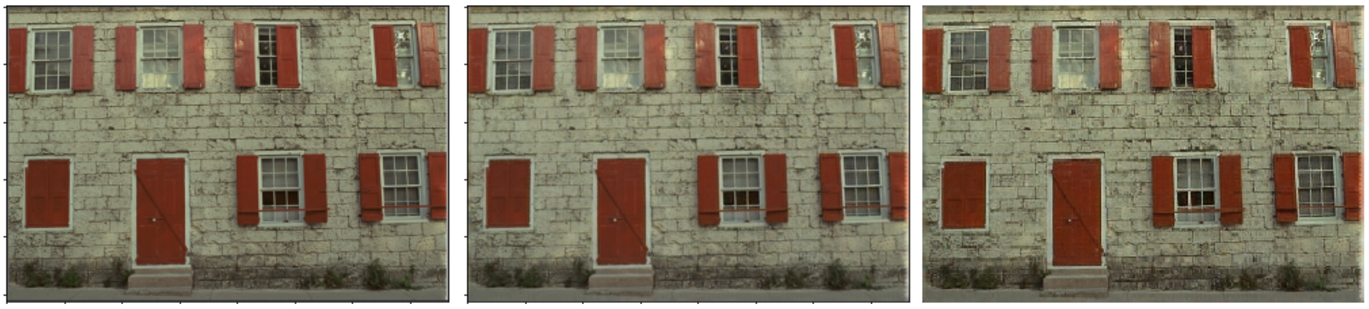


Figure 8: Image 1- Original(left), CAE software output (middle) and output from FPGA(right)

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.12	15.851	42.6	27.70%	828
2	conv2d_2_Conv2D	1788.42	15.06	23.708	75.4	49.10%	635.1
3	conv2d_3_Conv2D	3530.097	8.88	46.178	76.4	49.80%	192.2
4	conv2d_4_Conv2D	1741.824	7.43	23.463	74.2	48.30%	316.6
5	conv2d_5_Conv2D	214.843	1.66	2.944	73	47.50%	563.3
	All	7950.245	48.39	112.144	Avg=70.9	Avg=46.20%	Avg=431.5

Table 3: Profiling results for architecture B512

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	10.6	63.7	20.70%	1236.8
2	conv2d_2_Conv2D	1788.42	15.03	11.885	150.5	49.00%	1264.9
3	conv2d_3_Conv2D	3530.097	8.86	23.457	150.5	49.00%	377.8
4	conv2d_4_Conv2D	1741.824	7.4	11.759	148.1	48.20%	629.7
5	conv2d_5_Conv2D	214.843	1.66	1.499	143.3	46.70%	1104.3
	All	7950.245	48.3	59.2	Avg=134.3	Avg=43.70%	Avg=815.9

Table 4: Profiling results for architecture B1024

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	7.927	85.2	12.30%	1653.4
2	conv2d_2_Conv2D	1788.42	14.99	7.061	253.3	36.60%	2122.5
3	conv2d_3_Conv2D	3530.097	8.86	13.232	266.8	38.60%	669.5
4	conv2d_4_Conv2D	1741.824	7.38	6.638	262.4	38.00%	1111.2
5	conv2d_5_Conv2D	214.843	1.65	0.862	249.2	36.10%	1915.7
	All	7950.245	48.21	35.72	Avg=222.6	Avg=32.20%	Avg=1349.7

Table 5: Profiling results for architecture B2034

Architecture	Latency	FPS
B512	112.144	8.91710
B1024	59.2	16.89189
B2034	35.72	27.99552

Table 6: Architecture image compression latency

7 Discussion and Analysis

High values of PSNR and MS-SSIM values indicate that the input and output images are quite similar to each other with subtle variations. Visual inspection of the CAE outputs presented in Figures 7 and 8 indicates that the proposed compression algorithm results in a subtle reduction in sharpness of the images. This can be attributed to the fact that convolutions are essentially smoothening operations on images. The autoencoder based image compression algorithm was deployed on the Ultra96-V2 board and the compression quality was analysed. The decoded images are grainy images when deployed on FPGA and it was expected due to the quantization of the model weight parameters. This can be improved through more rigorous calibration to correctly spread out the weight distribution of the encoder.

The execution of the model on different hardware designs shows very low FPS values, typical real-time image processing involving human perception requires an average minimum of 30 FPS. The architecture B2034 comes fairly close to the value but would still encounter issues if program overheads are considered which will add to the latency of processing. The hardware is more suitable for non-human eye inferences like CCTV camera-based theft recognition [25]. It should also be noted that none of the models execution had the architectures loaded beyond 50% of capacity. This signifies these architectures can handle larger images with ease.

Regarding resolution, a 16:9 aspect ratio was chosen and designed to compare accuracy with the software models developed by the authors discussed in Section 2. Higher resolution requirements may reduce the attainable FPS.

8 Conclusion

This research project resulted in an image compression algorithm using Deep Neural Networks (DNNs). The proposed algorithm transformed the information content of an image from a higher dimensional space to a lower dimensional space. Image compression algorithm can be deployed on FPGA following the methodology described in Section 5 for DPU with custom architecture targeting Xilinx FPGAs.

The trained model is capable of compressing images of 512x768 with a FPS of approximately 28. The hypothesis of improvement in latency stands correct. There is an improvement in running time/latency where our method has 0.03572 s, compared to Z. Cheng, et al's publication [11] where they noted CPU consumed 2.29 s, and GPU consumed 0.67 s which are an order of magnitude slower than our model.

The design analyzed here is near real time, however it does not leave enough room for any processor overheads. It is worth noting that the Accelerator IP was functioning at the frequency of 150 MHz compared to CPU and GPU designs running at a speed of 1 GHz or more, which shows DPU is a very capable technology for DNN acceleration.

The CAE based image compression algorithm attained a 63.56 % compression ratio and retained an average of 90% of quality. Therefore, CAE based image compression is established as a viable alternative to JPEG encoders for real-time image processing.

9 Future Work

The compression based on autoencoder is thus established as a viable alternative for traditional JPEG encoders. This can be used to compress a wide variety of images. It would be interesting to develop models that can similarly compress stereoscopic images or VR images. There are techniques to improve the quality of images through super-resolution methods [26] which can overcome the loss of quality observed in the Autoencoder compression. The methodology could also be used to embed copyrighted data into the compressed image through steganography techniques. From the hardware improvements perspective, there could be HLS techniques developed to improve the utilization of the layers and produce higher FPS.

Acronyms

BD	Bjontegaard-Delta
CAE	Convolutional Autoencoder
CLIC	Challenge on Learned Image Compression
CODEC	Encoder and/or Decoder
CPU	Central Processing Unit
CRAE	Convolutional Residual Auto Encoder
CVPR	Conference on Computer Vision and Pattern Recognition
DL	Deep Learning
DNN	Deep Neural Network
DNNDK™	Deep Neural Network Development Kit
DPU	Xilinx® Deep Learning Processor Unit
DSP	Digital Signal Processing
FPGA	Field Programmable Gate Array
FPS	Frames per second
GAN	Generative Adversarial Network
GOPS	Giga operations per second
GPU	Graphics Processing Unit
HDF5	Hierarchical Data Format ver. 5
IP	Intellectual Property
MAC	Multiply and Accumulate
MB	Megabits
MOP	MAC Operations
MPSoC	Xilinx Zynq UltraScale+ MPSoC ZU3EG A484
MS-SSIM	Multi-Scale Structural Similarity Index
MSE	Mean Squared Error
PReLU	Parametric Rectified Linear Unit
PSNR	Peak Signal-to-Noise Ratio
ReLU	Rectified Linear Unit
SSH	Secure Shell
SSIM	Structural Similarity Index

References

[1] G. K. Wallace, “The jpeg still picture compression standard,” *Commun. ACM*, vol. 34, no. 4, p. 30–44, Apr. 1991. doi: 10.1145/103085.103089. [Online]. Available: https://doi.org/10.1145/103085.103089

[2] M. Rabbani and R. Joshi, “An overview of the jpeg 2000 still image compression standard,” *Signal Processing: Image Communication*, vol. 17, no. 1, pp. 3 – 48, 2002. doi: https://doi.org/10.1016/S0923-5965(01)00024-8 JPEG 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0923596501000248

[3] G. Valenzise, A. Purica, V. Hulusic, and M. Cagnazzo, “Quality assessment of deep-learning-based image compression,” in *2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP)*, Aug 2018. doi: 10.1109/MMSP.2018.8547064. ISSN 2473-3628 pp. 1–6.

[4] K. Guo, S. Zeng, J. Yu, Y. Wang, and H. Yang, “[dl] a survey of fpga-based neural network inference accelerators,” *ACM Trans. Reconfigurable Technol. Syst.*, vol. 12, no. 1, Mar. 2019. doi: 10.1145/3289185. [Online]. Available: https://doi.org/10.1145/3289185

[5] J. Zhu, L. Wang, H. Liu, S. Tian, Q. Deng, and J. Li, “An efficient task assignment framework to accelerate dpu-based convolutional neural network inference on fpgas,” *IEEE Access*, vol. 8, pp. 83 224–83 237, 2020. doi: 10.1109/ACCESS.2020.2988311

[6] V. W. Anelli, Y. Deldjoo, T. Di Noia, and D. Malitesta, “Deep learning-based adaptive image compression system for a real-world scenario,” in *2020 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, 2020. doi: 10.1109/EAIS48028.2020.9122753 pp. 1–8.

[7] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, “Deep convolutional autoencoder-based lossy image compression,” in *2018 Picture Coding Symposium (PCS)*, 2018. doi: 10.1109/PCS.2018.8456308 pp. 253–257.

[8] Z. Wang and A. C. Bovik, *Modern Image Quality Assessment*, ser. Synthesis Lectures on Image, Video, and Multimedia Processing. San Rafael, Calif. (1537 Fourth Street, San Rafael, CA 94901 USA): Morgan & Claypool Publishers LLC, 2006, vol. 2, no. 1. ISBN 1598290223

[9] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, April 2004. doi: 10.1109/TIP.2003.819861

[10] R. Franzén, “Kodak lossless true color image suite: Photocd pcd0992,” 2002, accessed: 2020-10-10. [Online]. Available: http://r0k.us/graphics/kodak/

[11] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, “Performance comparison of convolutional autoencoders, generative adversarial networks and super-resolution for image compression,” in *2018 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2018, Salt Lake City, UT, USA, June 18-22, 2018.* IEEE Computer Society, 2018, pp. 2613–2616. [Online]. Available: http://openaccess.thecvf.com/content_cvpr_2018_workshops/w50/html/Cheng_Performance_Comparison_of_CVPR_2018_paper.html

[12] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding,” 2015.

[13] E. Wang, J. J. Davis, R. Zhao, H.-C. Ng, X. Niu, W. Luk, P. Y. K. Cheung, and G. A. Constantinides, “Deep neural network approximation for custom hardware: Where we’ve been, where we’re going,” *ACM Comput. Surv.*, vol. 52, no. 2, May 2019. doi: 10.1145/3309551. [Online]. Available: https://doi-org.focus.lib.kth.se/10.1145/3309551

[14] E. Nurvitadhi, G. Venkatesh, J. Sim, D. Marr, R. Huang, J. Ong Gee Hock, Y. T. Liew, K. Srivatsan, D. Moss, S. Subhaschandra, and G. Boudoukh, "Can FPGAs Beat GPUs in Accelerating Next-Generation Deep Neural Networks?" in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '17. New York, NY, USA: Association for Computing Machinery, 2017. doi: 10.1145/3020078.3021740. ISBN 978-1-4503-4354-1 pp. 5–14, event-place: Monterey, California, USA. [Online]. Available: https://doi.org/10.1145/3020078.3021740

[15] Xilinx, "WP504 Accelerating DNNs with Xilinx Alveo Accelerator Cards," Oct. 2018. [Online]. Available: https://www.xilinx.com/support/documentation/white_papers/wp504-accel-dnns.pdf

[16] Avnet. Ultra96-v2 board. [Online]. Available: https://www.avnet.com/wps/portal/us/products/new-product-introductions/npi/aes-ultra96-v2

[17] Xilinx, "Vitis AI user guide (UG1414)," p. 198. [Online]. Available: https://www.xilinx.com/support/documentation/sw_manuals/vitis_ai/1_2/ug1414-vitis-ai.pdf

[18] ——, "UG1327 DNNDK User Guide," Jun. 2019. [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug1327-dnndk-user-guide.pdf

[19] J. H. Moreno and T. Lang, *Systolic-Type Arrays for Matrix Algorithms*. Boston, MA: Springer US, 1992, pp. 15–43. ISBN 978-1-4615-3610-9. [Online]. Available: https://doi.org/10.1007/978-1-4615-3610-9_2

[20] H.-I. Suk, "Chapter 1 - an introduction to neural networks and deep learning," in *Deep Learning for Medical Image Analysis*, S. K. Zhou, H. Greenspan, and D. Shen, Eds. Academic Press, 2017, pp. 3 – 24. ISBN 978-0-12-810408-8. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B978012810408800002X

[21] G. Toderici, W. Shi, R. Timofte, J. B. L Theis, E. Agustsson, N. Johnston, and F. Mentzer, "Workshop and challenge on learned image compression (clic2020)," CVPR, 2020. [Online]. Available: http://www.compression.cc

[22] E. Agustsson and R. Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[23] "Understanding convolution neural networks," accessed: 2020-10-10. [Online]. Available: https://towardsdatascience.com/understanding-convolution-neural-networks-the-eli5-way-785330cd1fb7

[24] Xilinx, "PetaLinux tools documentation : UG1144." [Online]. Available: https://www.xilinx.com/support/documentation/sw_manuals_j/xilinx2020_1/ug1144-petalinux-tools-reference-guide.pdf

[25] H. Keval and A. Sasse, "To catch a thief - you need at least 8 frames per second: The impact of frame rates on user performance in a cctv detection task," 01 2008. doi: 10.1145/1459359.1459527 pp. 941–944.

[26] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy image compression with compressive autoencoders," *ArXiv*, vol. abs/1703.00395, 2017.

A Appendix A : Hardware Profiling details

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.060	13.12	15.851	42.6	27.7%	828.0
2	conv2d_2_Conv2D	1788.420	15.06	23.708	75.4	49.1%	635.1
3	conv2d_3_Conv2D	3530.097	8.88	46.178	76.4	49.8%	192.2
4	conv2d_4_Conv2D	1741.824	7.43	23.463	74.2	48.3%	316.6
5	conv2d_5_Conv2D	214.843	1.66	2.944	73.0	47.5%	563.3
	All	7950.245	48.39	112.144	70.9	46.2%	431.5

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.060	13.12	15.854	42.6	27.7%	827.8
2	conv2d_2_Conv2D	1788.420	15.06	23.708	75.4	49.1%	635.1
3	conv2d_3_Conv2D	3530.097	8.88	46.164	76.5	49.8%	192.3
4	conv2d_4_Conv2D	1741.824	7.43	23.463	74.2	48.3%	316.6
5	conv2d_5_Conv2D	214.843	1.66	2.943	73.0	47.5%	563.5
	All	7950.245	48.39	112.132	70.9	46.2%	431.5

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.060	13.12	15.861	42.6	27.7%	827.5
2	conv2d_2_Conv2D	1788.420	15.06	23.708	75.4	49.1%	635.1
3	conv2d_3_Conv2D	3530.097	8.88	46.164	76.5	49.8%	192.3
4	conv2d_4_Conv2D	1741.824	7.43	23.462	74.2	48.3%	316.7
5	conv2d_5_Conv2D	214.843	1.66	2.943	73.0	47.5%	563.5
	All	7950.245	48.39	112.138	70.9	46.2%	431.5

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.060	13.12	15.862	42.6	27.7%	827.4
2	conv2d_2_Conv2D	1788.420	15.06	23.708	75.4	49.1%	635.1
3	conv2d_3_Conv2D	3530.097	8.88	46.176	76.4	49.8%	192.2
4	conv2d_4_Conv2D	1741.824	7.43	23.463	74.2	48.3%	316.6
5	conv2d_5_Conv2D	214.843	1.66	2.943	73.0	47.5%	563.5
	All	7950.245	48.39	112.152	70.9	46.2%	431.5

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.060	13.12	15.854	42.6	27.7%	827.8
2	conv2d_2_Conv2D	1788.420	15.06	23.711	75.4	49.1%	635.1
3	conv2d_3_Conv2D	3530.097	8.88	46.165	76.5	49.8%	192.3
4	conv2d_4_Conv2D	1741.824	7.43	23.464	74.2	48.3%	316.6
5	conv2d_5_Conv2D	214.843	1.66	2.943	73.0	47.5%	563.5
	All	7950.245	48.39	112.137	70.9	46.2%	431.5

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.060	13.12	15.860	42.6	27.7%	827.5
2	conv2d_2_Conv2D	1788.420	15.06	23.709	75.4	49.1%	635.1
3	conv2d_3_Conv2D	3530.097	8.88	46.165	76.5	49.8%	192.3
4	conv2d_4_Conv2D	1741.824	7.43	23.462	74.2	48.3%	316.7
5	conv2d_5_Conv2D	214.843	1.66	2.943	73.0	47.5%	563.5
	All	7950.245	48.39	112.139	70.9	46.2%	431.5

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.060	13.12	15.853	42.6	27.7%	827.9
2	conv2d_2_Conv2D	1788.420	15.06	23.708	75.4	49.1%	635.1
3	conv2d_3_Conv2D	3530.097	8.88	46.176	76.4	49.8%	192.2
4	conv2d_4_Conv2D	1741.824	7.43	23.463	74.2	48.3%	316.6
5	conv2d_5_Conv2D	214.843	1.66	2.941	73.1	47.6%	563.9
	All	7950.245	48.39	112.141	70.9	46.2%	431.5

Table 7: Network Layer profiling data for B512

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.060	13.12	15.860	42.6	27.7%	827.5
2	conv2d_2_Conv2D	1788.420	15.06	23.708	75.4	49.1%	635.1
3	conv2d_3_Conv2D	3530.097	8.88	46.168	76.5	49.8%	192.3
4	conv2d_4_Conv2D	1741.824	7.43	23.463	74.2	48.3%	316.6
5	conv2d_5_Conv2D	214.843	1.66	2.943	73.0	47.5%	563.5
	All	7950.245	48.39	112.142	70.9	46.2%	431.5

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.060	13.12	15.861	42.6	27.7%	827.5
2	conv2d_2_Conv2D	1788.420	15.06	23.708	75.4	49.1%	635.1
3	conv2d_3_Conv2D	3530.097	8.88	46.165	76.5	49.8%	192.3
4	conv2d_4_Conv2D	1741.824	7.43	23.463	74.2	48.3%	316.6
5	conv2d_5_Conv2D	214.843	1.66	2.944	73.0	47.5%	563.3
	All	7950.245	48.39	112.141	70.9	46.2%	431.5

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.060	13.12	15.855	42.6	27.7%	827.8
2	conv2d_2_Conv2D	1788.420	15.06	23.706	75.4	49.1%	635.2
3	conv2d_3_Conv2D	3530.097	8.88	46.162	76.5	49.8%	192.3
4	conv2d_4_Conv2D	1741.824	7.43	23.462	74.2	48.3%	316.7
5	conv2d_5_Conv2D	214.843	1.66	2.944	73.0	47.5%	563.3
	All	7950.245	48.39	112.129	70.9	46.2%	431.5

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.060	13.12	15.858	42.6	27.7%	827.6
2	conv2d_2_Conv2D	1788.420	15.06	23.708	75.4	49.1%	635.1
3	conv2d_3_Conv2D	3530.097	8.88	46.169	76.5	49.8%	192.3
4	conv2d_4_Conv2D	1741.824	7.43	23.462	74.2	48.3%	316.7
5	conv2d_5_Conv2D	214.843	1.66	2.940	73.1	47.6%	564.1
	All	7950.245	48.39	112.137	70.9	46.2%	431.5

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.060	13.12	15.851	42.6	27.7%	828.0
2	conv2d_2_Conv2D	1788.420	15.06	23.708	75.4	49.1%	635.1
3	conv2d_3_Conv2D	3530.097	8.88	46.172	76.5	49.8%	192.2
4	conv2d_4_Conv2D	1741.824	7.43	23.464	74.2	48.3%	316.6
5	conv2d_5_Conv2D	214.843	1.66	2.943	73.0	47.5%	563.5
	All	7950.245	48.39	112.138	70.9	46.2%	431.5

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.060	13.12	15.853	42.6	27.7%	827.9
2	conv2d_2_Conv2D	1788.420	15.06	23.707	75.4	49.1%	635.2
3	conv2d_3_Conv2D	3530.097	8.88	46.177	76.4	49.8%	192.2
4	conv2d_4_Conv2D	1741.824	7.43	23.463	74.2	48.3%	316.6
5	conv2d_5_Conv2D	214.843	1.66	2.943	73.0	47.5%	563.5
	All	7950.245	48.39	112.143	70.9	46.2%	431.5

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.060	13.12	15.855	42.6	27.7%	827.8
2	conv2d_2_Conv2D	1788.420	15.06	23.708	75.4	49.1%	635.1
3	conv2d_3_Conv2D	3530.097	8.88	46.163	76.5	49.8%	192.3
4	conv2d_4_Conv2D	1741.824	7.43	23.462	74.2	48.3%	316.7
5	conv2d_5_Conv2D	214.843	1.66	2.942	73.0	47.5%	563.7
	All	7950.245	48.39	112.130	70.9	46.2%	431.5

Table 8: Network Layer profiling data for B512

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.060	13.12	15.862	42.6	27.7%	827.4
2	conv2d_2_Conv2D	1788.420	15.06	23.706	75.4	49.1%	635.2
3	conv2d_3_Conv2D	3530.097	8.88	46.170	76.5	49.8%	192.3
4	conv2d_4_Conv2D	1741.824	7.43	23.463	74.2	48.3%	316.6
5	conv2d_5_Conv2D	214.843	1.66	2.944	73.0	47.5%	563.3
	All	7950.245	48.39	112.145	70.9	46.2%	431.5
ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.060	13.12	15.853	42.6	27.7%	827.9
2	conv2d_2_Conv2D	1788.420	15.06	23.706	75.4	49.1%	635.2
3	conv2d_3_Conv2D	3530.097	8.88	46.166	76.5	49.8%	192.3
4	conv2d_4_Conv2D	1741.824	7.43	23.464	74.2	48.3%	316.6
5	conv2d_5_Conv2D	214.843	1.66	2.943	73.0	47.5%	563.5
	All	795	48.39	112.132	68.34	44.48	507.1
ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.060	13.12	15.854	42.6	27.7%	827.8
2	conv2d_2_Conv2D	1788.420	15.06	23.708	75.4	49.1%	635.1
3	conv2d_3_Conv2D	3530.097	8.88	46.167	76.5	49.8%	192.3
4	conv2d_4_Conv2D	1741.824	7.43	23.463	74.2	48.3%	316.6
5	conv2d_5_Conv2D	214.843	1.66	2.944	73.0	47.5%	563.3
	All	7950.245	48.39	112.136	70.9	46.2%	431.5
ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.060	13.12	15.850	42.6	27.7%	828.0
2	conv2d_2_Conv2D	1788.420	15.06	23.706	75.4	49.1%	635.2
3	conv2d_3_Conv2D	3530.097	8.88	46.174	76.5	49.8%	192.2
4	conv2d_4_Conv2D	1741.824	7.43	23.463	74.2	48.3%	316.6
5	conv2d_5_Conv2D	214.843	1.66	2.944	73.0	47.5%	563.3
	All	7950.245	48.39	112.137	70.9	46.2%	431.5

Table 9: Network Layer profiling data for B512

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	10.6	63.7	20.70%	1236.8
2	conv2d_2_Conv2D	1788.42	15.03	11.885	150.5	49.00%	1264.9
3	conv2d_3_Conv2D	3530.097	8.86	23.457	150.5	49.00%	377.8
4	conv2d_4_Conv2D	1741.824	7.4	11.759	148.1	48.20%	629.7
5	conv2d_5_Conv2D	214.843	1.66	1.499	143.3	46.70%	1104.3
	All	7950.245	48.3	59.2	134.3	43.70%	815.9

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	10.594	63.7	20.70%	1237.5
2	conv2d_2_Conv2D	1788.42	15.03	11.883	150.5	49.00%	1265.1
3	conv2d_3_Conv2D	3530.097	8.86	23.457	150.5	49.00%	377.8
4	conv2d_4_Conv2D	1741.824	7.4	11.758	148.1	48.20%	629.8
5	conv2d_5_Conv2D	214.843	1.66	1.496	143.6	46.70%	1106.6
	All	7950.245	48.3	59.188	134.3	43.70%	816.1

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	10.619	63.6	20.70%	1234.6
2	conv2d_2_Conv2D	1788.42	15.03	11.885	150.5	49.00%	1264.9
3	conv2d_3_Conv2D	3530.097	8.86	23.457	150.5	49.00%	377.8
4	conv2d_4_Conv2D	1741.824	7.4	11.76	148.1	48.20%	629.7
5	conv2d_5_Conv2D	214.843	1.66	1.496	143.6	46.70%	1106.6
	All	7950.245	48.3	59.217	134.3	43.70%	815.7

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	10.571	63.9	20.80%	1240.2
2	conv2d_2_Conv2D	1788.42	15.03	11.883	150.5	49.00%	1265.1
3	conv2d_3_Conv2D	3530.097	8.86	23.458	150.5	49.00%	377.8
4	conv2d_4_Conv2D	1741.824	7.4	11.758	148.1	48.20%	629.8
5	conv2d_5_Conv2D	214.843	1.66	1.499	143.3	46.70%	1104.3
	All	7950.245	48.3	59.169	134.4	43.70%	816.4

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	10.602	63.7	20.70%	1236.5
2	conv2d_2_Conv2D	1788.42	15.03	11.882	150.5	49.00%	1265.2
3	conv2d_3_Conv2D	3530.097	8.86	23.457	150.5	49.00%	377.8
4	conv2d_4_Conv2D	1741.824	7.4	11.759	148.1	48.20%	629.7
5	conv2d_5_Conv2D	214.843	1.66	1.497	143.5	46.70%	1105.8
	All	7950.245	48.3	59.197	134.3	43.70%	816

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	10.568	63.9	20.80%	1240.5
2	conv2d_2_Conv2D	1788.42	15.03	11.883	150.5	49.00%	1265.1
3	conv2d_3_Conv2D	3530.097	8.86	23.458	150.5	49.00%	377.8
4	conv2d_4_Conv2D	1741.824	7.4	11.76	148.1	48.20%	629.7
5	conv2d_5_Conv2D	214.843	1.66	1.498	143.4	46.70%	1105.1
	All	7950.245	48.3	59.167	134.4	43.70%	816.4

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	10.572	63.9	20.80%	1240.1
2	conv2d_2_Conv2D	1788.42	15.03	11.884	150.5	49.00%	1265
3	conv2d_3_Conv2D	3530.097	8.86	23.457	150.5	49.00%	377.8
4	conv2d_4_Conv2D	1741.824	7.4	11.761	148.1	48.20%	629.6
5	conv2d_5_Conv2D	214.843	1.66	1.497	143.5	46.70%	1105.8
	All	7950.245	48.3	59.171	134.4	43.70%	816.3

Table 10: Network Layer profiling data for B1024

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	10.603	63.7	20.70%	1236.4
2	conv2d_2_Conv2D	1788.42	15.03	11.883	150.5	49.00%	1265.1
3	conv2d_3_Conv2D	3530.097	8.86	23.457	150.5	49.00%	377.8
4	conv2d_4_Conv2D	1741.824	7.4	11.761	148.1	48.20%	629.6
5	conv2d_5_Conv2D	214.843	1.66	1.497	143.5	46.70%	1105.8
	All	7950.245	48.3	59.201	134.3	43.70%	815.9
ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	10.594	63.7	20.70%	1237.5
2	conv2d_2_Conv2D	1788.42	15.03	11.885	150.5	49.00%	1264.9
3	conv2d_3_Conv2D	3530.097	8.86	23.457	150.5	49.00%	377.8
4	conv2d_4_Conv2D	1741.824	7.4	11.761	148.1	48.20%	629.6
5	conv2d_5_Conv2D	214.843	1.66	1.497	143.5	46.70%	1105.8
	All	7950.245	48.3	59.194	134.3	43.70%	816
ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	10.571	63.9	20.80%	1240.2
2	conv2d_2_Conv2D	1788.42	15.03	11.884	150.5	49.00%	1265
3	conv2d_3_Conv2D	3530.097	8.86	23.457	150.5	49.00%	377.8
4	conv2d_4_Conv2D	1741.824	7.4	11.76	148.1	48.20%	629.7
5	conv2d_5_Conv2D	214.843	1.66	1.496	143.6	46.70%	1106.6
	All	7950.245	48.3	59.168	134.4	43.70%	816.4
ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	10.577	63.8	20.80%	1239.5
2	conv2d_2_Conv2D	1788.42	15.03	11.883	150.5	49.00%	1265.1
3	conv2d_3_Conv2D	3530.097	8.86	23.457	150.5	49.00%	377.8
4	conv2d_4_Conv2D	1741.824	7.4	11.761	148.1	48.20%	629.6
5	conv2d_5_Conv2D	214.843	1.66	1.499	143.3	46.70%	1104.3
	All	7950.245	48.3	59.177	134.3	43.70%	816.3
ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	10.568	63.9	20.80%	1240.5
2	conv2d_2_Conv2D	1788.42	15.03	11.884	150.5	49.00%	1265
3	conv2d_3_Conv2D	3530.097	8.86	23.457	150.5	49.00%	377.8
4	conv2d_4_Conv2D	1741.824	7.4	11.76	148.1	48.20%	629.7
5	conv2d_5_Conv2D	214.843	1.66	1.497	143.5	46.70%	1105.8
	All	7950.245	48.3	59.166	134.4	43.70%	816.4
ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	10.576	63.8	20.80%	1239.6
2	conv2d_2_Conv2D	1788.42	15.03	11.885	150.5	49.00%	1264.9
3	conv2d_3_Conv2D	3530.097	8.86	23.458	150.5	49.00%	377.8
4	conv2d_4_Conv2D	1741.824	7.4	11.759	148.1	48.20%	629.7
5	conv2d_5_Conv2D	214.843	1.66	1.497	143.5	46.70%	1105.8
	All	7950.245	48.3	59.175	134.4	43.70%	816.3
ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	10.57	63.9	20.80%	1240.3
2	conv2d_2_Conv2D	1788.42	15.03	11.885	150.5	49.00%	1264.9
3	conv2d_3_Conv2D	3530.097	8.86	23.46	150.5	49.00%	377.8
4	conv2d_4_Conv2D	1741.824	7.4	11.761	148.1	48.20%	629.6
5	conv2d_5_Conv2D	214.843	1.66	1.499	143.3	46.70%	1104.3
	All	7950.245	48.3	59.175	134.4	43.70%	816.3

Table 11: Network Layer profiling data for B1024

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	10.603	63.7	20.70%	1236.4
2	conv2d_2_Conv2D	1788.42	15.03	11.884	150.5	49.00%	1265
3	conv2d_3_Conv2D	3530.097	8.86	23.457	150.5	49.00%	377.8
4	conv2d_4_Conv2D	1741.824	7.4	11.763	148.1	48.20%	629.5
5	conv2d_5_Conv2D	214.843	1.66	1.498	143.4	46.70%	1105.1
	All	7950.245	48.3	59.205	134.3	43.70%	815.9
ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	10.572	63.9	20.80%	1240.1
2	conv2d_2_Conv2D	1788.42	15.03	11.885	150.5	49.00%	1264.9
3	conv2d_3_Conv2D	3530.097	8.86	23.459	150.5	49.00%	377.8
4	conv2d_4_Conv2D	1741.824	7.4	11.758	148.1	48.20%	629.8
5	conv2d_5_Conv2D	214.843	1.66	1.498	143.4	46.70%	1105.1
	All	795	48.3	59.172	131.28	42.74%	816.3
ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	10.579	63.8	20.80%	1239.2
2	conv2d_2_Conv2D	1788.42	15.03	11.885	150.5	49.00%	1264.9
3	conv2d_3_Conv2D	3530.097	8.86	23.458	150.5	49.00%	377.8
4	conv2d_4_Conv2D	1741.824	7.4	11.759	148.1	48.20%	629.7
5	conv2d_5_Conv2D	214.843	1.66	1.498	143.4	46.70%	1105.1
	All	7950.245	48.3	59.179	134.3	43.70%	816.2
ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	10.572	63.9	20.80%	1240.1
2	conv2d_2_Conv2D	1788.42	15.03	11.886	150.5	49.00%	1264.8
3	conv2d_3_Conv2D	3530.097	8.86	23.458	150.5	49.00%	377.8
4	conv2d_4_Conv2D	1741.824	7.4	11.76	148.1	48.20%	629.7
5	conv2d_5_Conv2D	214.843	1.66	1.499	143.3	46.70%	1104.3
	All	7950.245	48.3	59.175	134.4	43.70%	816.3

Table 12: Network Layer profiling data for B1024

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	7.927	85.2	12.30%	1653.4
2	conv2d_2_Conv2D	1788.42	14.99	7.061	253.3	36.60%	2122.5
3	conv2d_3_Conv2D	3530.097	8.86	13.232	266.8	38.60%	669.5
4	conv2d_4_Conv2D	1741.824	7.38	6.638	262.4	38.00%	1111.2
5	conv2d_5_Conv2D	214.843	1.65	0.862	249.2	36.10%	1915.7
	All	7950.245	48.21	35.72	222.6	32.20%	1349.7

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	7.925	85.2	12.30%	1653.8
2	conv2d_2_Conv2D	1788.42	14.99	7.057	253.4	36.70%	2123.7
3	conv2d_3_Conv2D	3530.097	8.86	13.223	267	38.60%	669.9
4	conv2d_4_Conv2D	1741.824	7.38	6.633	262.6	38.00%	1112
5	conv2d_5_Conv2D	214.843	1.65	0.861	249.5	36.10%	1917.9
	All	7950.245	48.21	35.699	222.7	32.20%	1350.5

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	7.924	85.2	12.30%	1654
2	conv2d_2_Conv2D	1788.42	14.99	7.019	254.8	36.90%	2135.2
3	conv2d_3_Conv2D	3530.097	8.86	13.223	267	38.60%	669.9
4	conv2d_4_Conv2D	1741.824	7.38	6.635	262.5	38.00%	1111.7
5	conv2d_5_Conv2D	214.843	1.65	0.861	249.5	36.10%	1917.9
	All	7950.245	48.21	35.662	222.9	32.30%	1351.9

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	7.929	85.1	12.30%	1652.9
2	conv2d_2_Conv2D	1788.42	14.99	7.011	255.1	36.90%	2137.6
3	conv2d_3_Conv2D	3530.097	8.86	13.224	266.9	38.60%	669.9
4	conv2d_4_Conv2D	1741.824	7.38	6.634	262.6	38.00%	1111.9
5	conv2d_5_Conv2D	214.843	1.65	0.861	249.5	36.10%	1917.9
	All	7950.245	48.21	35.659	223	32.30%	1352

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	7.927	85.2	12.30%	1653.4
2	conv2d_2_Conv2D	1788.42	14.99	7.132	250.8	36.30%	2101.4
3	conv2d_3_Conv2D	3530.097	8.86	13.224	266.9	38.60%	669.9
4	conv2d_4_Conv2D	1741.824	7.38	6.634	262.6	38.00%	1111.9
5	conv2d_5_Conv2D	214.843	1.65	0.862	249.2	36.10%	1915.7
	All	7950.245	48.21	35.779	222.2	32.10%	1347.5

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	7.926	85.2	12.30%	1653.6
2	conv2d_2_Conv2D	1788.42	14.99	7.023	254.7	36.80%	2134
3	conv2d_3_Conv2D	3530.097	8.86	13.223	267	38.60%	669.9
4	conv2d_4_Conv2D	1741.824	7.38	6.633	262.6	38.00%	1112
5	conv2d_5_Conv2D	214.843	1.65	0.866	248.1	35.90%	1906.8
	All	7950.245	48.21	35.671	222.9	32.20%	1351.6

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	7.926	85.2	12.30%	1653.6
2	conv2d_2_Conv2D	1788.42	14.99	7.079	252.6	36.60%	2117.1
3	conv2d_3_Conv2D	3530.097	8.86	13.224	266.9	38.60%	669.9
4	conv2d_4_Conv2D	1741.824	7.38	6.634	262.6	38.00%	1111.9
5	conv2d_5_Conv2D	214.843	1.65	0.862	249.2	36.10%	1915.7
	All	7950.245	48.21	35.725	222.5	32.20%	1349.5

Table 13: Network Layer profiling data for B2034

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	7.93	85.1	12.30%	1652.7
2	conv2d_2_Conv2D	1788.42	14.99	7.031	254.4	36.80%	2131.5
3	conv2d_3_Conv2D	3530.097	8.86	13.223	267	38.60%	669.9
4	conv2d_4_Conv2D	1741.824	7.38	6.635	262.5	38.00%	1111.7
5	conv2d_5_Conv2D	214.843	1.65	0.861	249.5	36.10%	1917.9
	All	7950.245	48.21	35.68	222.8	32.20%	1351.3

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	7.927	85.2	12.30%	1653.4
2	conv2d_2_Conv2D	1788.42	14.99	7.042	254	36.70%	2128.2
3	conv2d_3_Conv2D	3530.097	8.86	13.222	267	38.60%	670
4	conv2d_4_Conv2D	1741.824	7.38	6.635	262.5	38.00%	1111.7
5	conv2d_5_Conv2D	214.843	1.65	0.861	249.5	36.10%	1917.9
	All	7950.245	48.21	35.687	222.8	32.20%	1351

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	7.925	85.2	12.30%	1653.8
2	conv2d_2_Conv2D	1788.42	14.99	7.119	251.2	36.30%	2105.2
3	conv2d_3_Conv2D	3530.097	8.86	13.225	266.9	38.60%	669.8
4	conv2d_4_Conv2D	1741.824	7.38	6.633	262.6	38.00%	1112
5	conv2d_5_Conv2D	214.843	1.65	0.864	248.7	36.00%	1911.2
	All	7950.245	48.21	35.766	222.3	32.20%	1348

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	7.928	85.1	12.30%	1653.2
2	conv2d_2_Conv2D	1788.42	14.99	7.075	252.8	36.60%	2118.3
3	conv2d_3_Conv2D	3530.097	8.86	13.225	266.9	38.60%	669.8
4	conv2d_4_Conv2D	1741.824	7.38	6.635	262.5	38.00%	1111.7
5	conv2d_5_Conv2D	214.843	1.65	0.863	248.9	36.00%	1913.4
	All	7950.245	48.21	35.726	222.5	32.20%	1349.5

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	7.924	85.2	12.30%	1654
2	conv2d_2_Conv2D	1788.42	14.99	7.046	253.8	36.70%	2127
3	conv2d_3_Conv2D	3530.097	8.86	13.224	266.9	38.60%	669.9
4	conv2d_4_Conv2D	1741.824	7.38	6.634	262.6	38.00%	1111.9
5	conv2d_5_Conv2D	214.843	1.65	0.859	250.1	36.20%	1922.3
	All	7950.245	48.21	35.687	222.8	32.20%	1351

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	7.933	85.1	12.30%	1652.1
2	conv2d_2_Conv2D	1788.42	14.99	7.058	253.4	36.70%	2123.4
3	conv2d_3_Conv2D	3530.097	8.86	13.222	267	38.60%	670
4	conv2d_4_Conv2D	1741.824	7.38	6.635	262.5	38.00%	1111.7
5	conv2d_5_Conv2D	214.843	1.65	0.861	249.5	36.10%	1917.9
	All	7950.245	48.21	35.709	222.6	32.20%	1350.2

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	7.925	85.2	12.30%	1653.8
2	conv2d_2_Conv2D	1788.42	14.99	7.052	253.6	36.70%	2125.2
3	conv2d_3_Conv2D	3530.097	8.86	13.223	267	38.60%	669.9
4	conv2d_4_Conv2D	1741.824	7.38	6.635	262.5	38.00%	1111.7
5	conv2d_5_Conv2D	214.843	1.65	0.86	249.8	36.10%	1920.1
	All	7950.245	48.21	35.695	222.7	32.20%	1350.7

Table 14: Network Layer profiling data for B2034

ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	7.924	85.2	12.30%	1654
2	conv2d_2_Conv2D	1788.42	14.99	7.049	253.7	36.70%	2126.1
3	conv2d_3_Conv2D	3530.097	8.86	13.224	266.9	38.60%	669.9
4	conv2d_4_Conv2D	1741.824	7.38	6.635	262.5	38.00%	1111.7
5	conv2d_5_Conv2D	214.843	1.65	0.861	249.5	36.10%	1917.9
	All	7950.245	48.21	35.693	222.7	32.20%	1350.8
ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	7.932	85.1	12.30%	1652.3
2	conv2d_2_Conv2D	1788.42	14.99	7.056	253.5	36.70%	2124
3	conv2d_3_Conv2D	3530.097	8.86	13.223	267	38.60%	669.9
4	conv2d_4_Conv2D	1741.824	7.38	6.636	262.5	38.00%	1111.5
5	conv2d_5_Conv2D	214.843	1.65	0.86	249.8	36.10%	1920.1
	All	795	48.21				
ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	7.923	85.2	12.30%	1654.2
2	conv2d_2_Conv2D	1788.42	14.99	7.073	252.9	36.60%	2118.9
3	conv2d_3_Conv2D	3530.097	8.86	13.222	267	38.60%	670
4	conv2d_4_Conv2D	1741.824	7.38	6.634	262.6	38.00%	1111.9
5	conv2d_5_Conv2D	214.843	1.65	0.86	249.8	36.10%	1920.1
	All	7950.245	48.21	35.712	222.6	32.20%	1350
ID	NodeName	Workload(MOP)	Mem(MB)	RunTime(ms)	Perf(GOPS)	Utilization	MB/S
1	conv2d_1_Conv2D	675.06	13.11	7.927	85.2	12.30%	1653.4
2	conv2d_2_Conv2D	1788.42	14.99	7.052	253.6	36.70%	2125.2
3	conv2d_3_Conv2D	3530.097	8.86	13.223	267	38.60%	669.9
4	conv2d_4_Conv2D	1741.824	7.38	6.637	262.4	38.00%	1111.4
5	conv2d_5_Conv2D	214.843	1.65	0.861	249.5	36.10%	1917.9
	All	7950.245	48.21	35.7	222.7	32.20%	1350.5

Table 15: Network Layer profiling data for B2034