Performance Evaluation of GraphCore IPU-M2000 Accelerator for Text Detection Application

Nupur Sumeet
nupur.sumeet@tcs.com
Tata Consultancy Services Research
Mumbai, Maharashtra, India

Karan Rawat
rawat.karan@tcs.com
Tata Consultancy Services Research
Mumbai, Maharashtra, India

Manoj Nambiar
m.nambiar@tcs.com
Tata Consultancy Services Research
Mumbai, Maharashtra, India

ABSTRACT

The large compute load and memory footprint of modern deep neural networks motivates the use of accelerators for high throughput deployments in application spanning multiple domains. In this paper, we evaluate throughput capabilities of a comparatively new hardware from Graphcore, IPU-M2000 that supports massive parallelism and in-memory compute. For a text detection model, we measured the throughput and power variations with batch size. We also evaluate compressed versions of this model and analyze performance variation with model precision. Additionally, we compare IPU (Intelligence Processing Unit) results with state-of-the-art GPU and FPGA deployments of a compute intensive text region detection application. Our experiments suggest, IPU supports superior throughput, 27×, 1.89×, and 1.56× as compared to CPU, FPGA DPU and A100 GPU, respectively for text detection application.

CCS CONCEPTS

• **Hardware → Emerging architectures**; • **Computer systems organization → Single instruction, multiple data**; • **Computing methodologies → Neural networks**; • **Applied computing → Optical character recognition**.

KEYWORDS

New Technologies, Performance Evaluation, High-throughput deployment, Text detection

ACM Reference Format:

Nupur Sumeet, Karan Rawat, and Manoj Nambiar. 2022. Performance Evaluation of GraphCore IPU-M2000 Accelerator for Text Detection Application. In *Companion of the 2022 ACM/SPEC International Conference on Performance Engineering (ICPE '22 Companion), April 9–13, 2022, Bejing, China.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3491204.3527469

1 INTRODUCTION

The increasing computational load and memory footprint of modern deep neural networks translates to large inference times. On the contrary, high throughput requirement has become a universal ask from applications of all domains, vision, recommendation, speech processing *etc.* The contrasting constraint of high compute and memory with fast-acting systems renders creates a need for model optimization techniques and efficient hardware alternatives for high throughput deployments. In a CPU-alone implementation, a document processing pipeline[21] with CRAFT-based[1, 2] text region detection amounts for 90% of processing time and is the bottleneck task. Central Processing Unit (CPU) systems equipped with accelerators such as Tensor Processing Unit (TPU), Graphics Processing Unit (GPU) and Field Programmable Gate Array (FPGA) are conventionally preferred hardware choices for such performance sensitive and computationally intensive workloads.

Due to recent advances in hardware architectures, new hardware platforms have entered the play field that perform better as compared to conventional accelerators. The Graphcore IPU processors[9, 26], is a custom hardware platforms targeted to accelerate machine learning workloads through micro-architecture changes like in-memory processing. Due to tightly coupled memory, IPU cores pay no penalty when their control flows diverge or when the addresses of their memory accesses diverge and realise uncorrelated memory accesses without memory access overhead. Cores access data from their respective local memory at a fixed cost that is independent of access patterns. This makes IPUs more efficient than GPUs at executing applications with irregular or random data access patterns and/or applications that are control-flow dominated, provided that working sets fit in IPU memory.

In this paper, we evaluate the performance of Graphcore IPU-M2000 machine for a vision application. We present a roofline model that provide insights on required compute and memory capabilities from a hardware platform to achieve a certain throughput for the target application. Our roofline model compares GPU, FPGA DPUs (Deep learning Processing Units)[24] with Graphcore IPUs and establish the suitability of later for text detection model. We present results on throughput variation with respect to batch size, compute precision as well as power trends for compute-intensive workload. Additionally, we compare and contrast the IPU results with state-of-the art conventional accelerators like GPU and FPGA.

The rest of the paper is organized as follows. Section 2 contains architecture and system-level details of IPU-M2000. The text detection application and its ML model (CRAFT) with its compressed versions are discussed in section 3. The roofline model for compressed CRAFT model is presented in section 4. Section 5 presents results on variation of throughput and power with # of IPUs, batch size and comparison with other hardware. This is followed by conclusion in section 6.

2 GRAPHCORE IPU-M2000 MACHINE

Graphcore IPU-M2000 helps in making the training and inference of large models more scalable and efficient. TheIPU-M2000 contains

ICPE '22 Companion, April 9–13, 2022, Bejing, China

ACM ISBN 978-1-4503-9159-7/22/04…$15.00
https://doi.org/10.1145/3491204.3527469

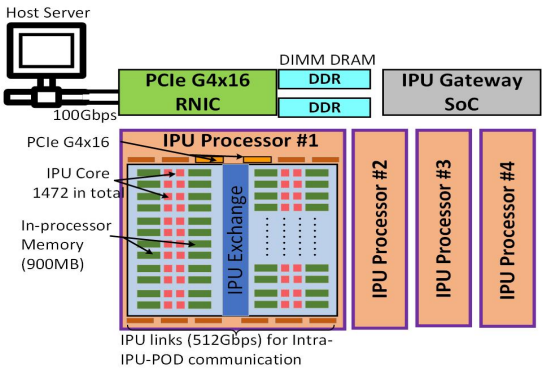

Figure 1: The system level architecture of IPU-M2000 machine.

4 GC200 IPU processors, together housing 5,888 processor cores with 35k independent parallel threads giving a total compute of 1PFLOP/s (peta Floating-Point OPerations per second) for FP16 precision and 250TFLOP/s for FP32. As cache memory, there exists 900MB ultra-high-speed SRAM in-processor memory for each IPU that is distributed alongside each processor core for lowest energy access per bit. The system level architecture of IPU-M2000 machine is shown in Fig. 1. A server host machine running Linux O/S is used as an interface to the Graphcore IPU-M2000 system. This host is connected to the Graphcore system by a 100Gbps Ethernet link. The 4 IPUs within the IPU-M2000 system are interconnected by a 2.8Tbps IPU-Fabric.

IPU systems are supported by Poplar SDK (Software Development Kit)[18], which provides a complete platform for AI deployment and evaluation. All standard ML libraries like Pytorch, Tensorflow, ONNX, Keras *etc.* are supported and integrated in the development environment. An analysis tool (PopVision) provides cycle by cycle accounting and analysis of the workload. The input model is accepted by Poplar framework and passed to Graph Compiler which simplifies the IPU programming by handling the scheduling and work partitioning of the target workload. Graph compile domain allows single application to be programmed across multiple IPUs and yields data and model parallel executions simultaneously.

3 TEST APPLICATION: TEXT DETECTION

DeepReader[21] is document processing application which facilitates information extraction from document images. To extract textual entities in scene text or colored images present in the document, DeepReader use the CRAFT[1] model for text segmentation.

3.1 Text Detection Model: CRAFT

The CRAFT (Character Region Awareness For Text detection) framework is based on convolutional neural network (CNN) producing the character region score and affinity score. The region score is used to localize individual characters and denotes the probability of a pixel lying in the center of a text character. The affinity score is used to group each character into a single instance and represents probability of the space between adjacent characters. Based on the two scores, bounding boxes are created over text regions of the image as part of post-processing step. Figure 2 shows the text detection pipeline. CRAFT network contains convolution (3×3, 1×1), batch normalization, ReLU, maxpool(2×2), upsampling and concatenation operations. The CRAFT model processes and transforms the features shown in Fig. 3. CRAFT is based on VGG-16-bn[19] backbone with skip connections in the decoding part. The pre-trained CRAFT model is available in open-source and has accuracy metrics- 96.04% (Recall) and 95.79% (Precision) on scanned receipt dataset IC19[10], as reported on the robust reading competition[4] leader-board. The model has high flexibility on detecting complicated cases, such as long, curved, and/or arbitrarily shaped texts.

Table 1: Compressed CRAFT model architecture.

Conv Stage	Down-Scaled		LTH Pruned		
	Cn	C8	C18	C12	C6
Conv1	n	8	18	12	6
Conv2	2n	16	24	24	20
Conv3	4n	32	40	36	34
Conv4	8n	64	50	48	44
Conv5	8n	128	50	48	44
Conv6	16n	256	70	60	56
UpConv1	8n	64	50	48	44
UpConv2	4n	32	40	36	24
UpConv3	2n	16	24	24	20
UpConv4	n	8	18	12	6
Conv7	n/2	4	9	6	3
Total Filters	93n/2	844	735	666	585

Table 2: Accuracy of native and compressed CRAFT models on IC15 dataset.

	CRAFT	C32	C18	C16	C12	C8	C6	C4
Precision	89.7%	85.7%	88.6%	84.2%	86.4%	83.5%	84.3%	80.7%
Recall	83.2%	80.4%	80.6%	79.1%	80.2%	78.9%	78.6%	75.5%

3.2 Compressed CRAFT Models

CRAFT has high computational and memory load which directly translates to large inference time and consequently limits its usability in real-life scenarios. For inference speed-up, we adopted knowledge distillation[3] based model compression technique. The compression specific details are beyond the scope of this paper. We have seven compressed models for CRAFT that are compressed by 14-180×. The accuracy for these models remain within ≈5% of the native model. The accuracy for compressed models is shown in Table 2. The method of model compression and detailed discussions on resultant accuracy trade-offs is beyond the scope of this paper.

The compressed models are categorized as down-scaled and LTH-pruned based on the method used to obtain these architectures. The down-scaled versions are obtained by proportionally reducing # of filters by factors 2, 4, 8 and so on, in the network. The generalized down-scaled student architectures is presented in column 2 Table 1 where n takes value 32, 16, 8, 4, 2 and 1. For instance, student architecture with n=8 is shown in column 3 in Table 1. For C8, Conv1 contains 2 layers with 8 channels followed by 2 layers of 16 channels. The model architecture for LTH pruned models is obtained by iteratively applying LTH[7] with different pruning % on native CRAFT model. The LTH pruned model architectures (C18, C12 and C6) are shown in Table 1.

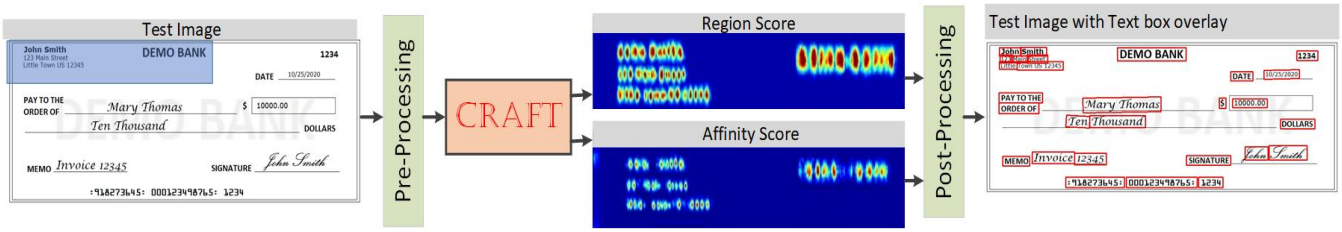

Figure 2: CRAFT-based Text Detection pipeline[1].

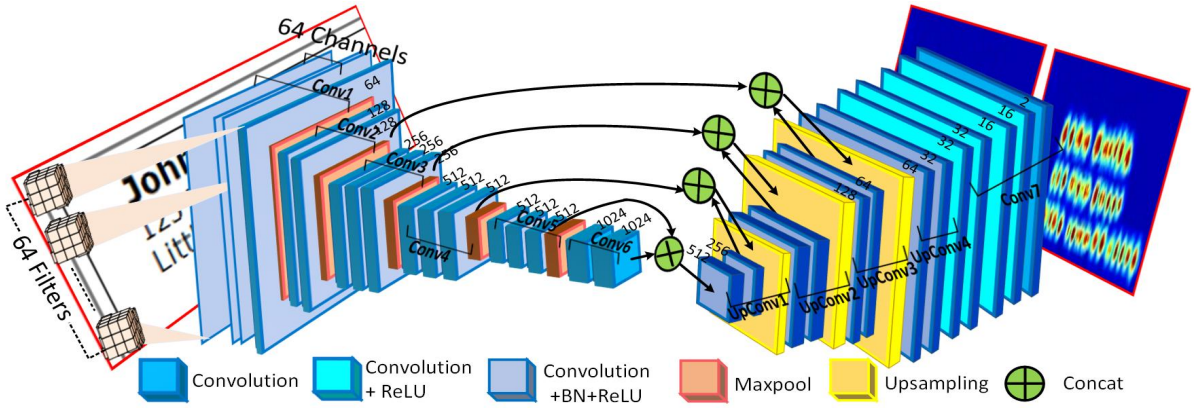

Figure 3: Representative diagram of Feature transformations due to CRAFT[1].

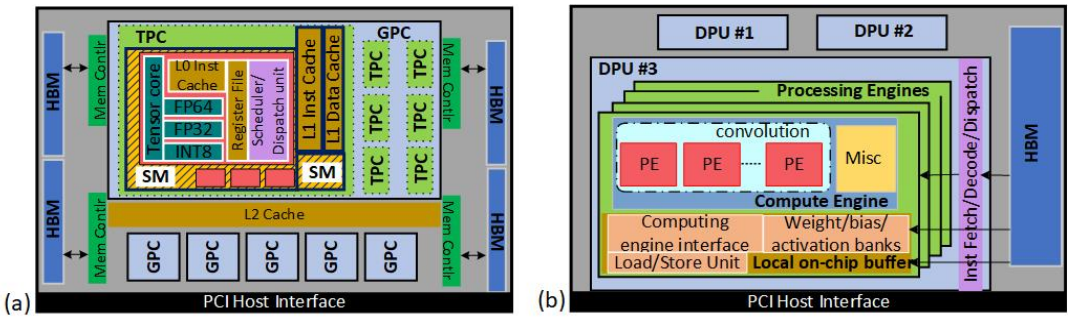

Figure 4: Processing element and HBM for (a) V100 GPU, and (b) FPGA DPU. GPC- GPU Processing Clusters, TPC- Texture Processing Clusters, SM- Streaming Microprocessors, PE- Processing Elements.

3.3 Training and testing

The compressed model training is done on NVIDIA A100 GPU with a batch size of 10 using ADAM Optimizer[13] and L2 regularization[5] for 150-500 epochs. The models are tested on IC15 dataset that consists of 500 testing images with texts in English and RGB format with a resolution of 1280×768. The text instances are labeled at the word level and ground truth consists of quadrilateral boxes in 8-point coordinate system for every word.

4 ROOFLINE MODEL FOR COMPRESSED CRAFT (C6)

For a DL model, the maximum compute and memory bandwidth capabilities required to meet a certain fps is regarded as the roofline modeling. We present the roofline model for C6 model in this section. For this, we estimate the computation and memory load of every layer in C6 model (see Table 3). The layer-wise GFLOps (Giga FLoating-point Operations) is estimated using flops counter[20]. The memory footprint is calculated by estimating the size of the output at every layer. For instance, the first layer (Layer #1) is convolution followed by batch normalization and ReLU activation. It has 1.42GFLOps computation load and has an output feature size of 1280×768×6 that amounts to a footprint of 22.5MB. We observe that initial and few final layers of the model have high computation load. The memory footprint reciprocate the same behaviour with upsampling and concatenation layers also generating large requirements on memory. This is because of the large feature size at these layers. The concatenation operation joins features on the depth dimension and considered as memory manipulation.

We choose a throughput of n fps, to calculate the layer-wise required compute and memory bandwidth capabilities. This requires a

Table 3: Layer-wise required compute and memory capabilities for 500fps throughput for Compressed CRAFT (C6 model).

Layer#	Layer Name	# of filters	Comp. Load (in GFLOps)	Req. Comp. Cap. (in GFLOP/s)	O/P Mem. footprint (in MB)	Req. Mem. Cap. (in GB/s)
1	Conv_BN_ReLU	6	1.42	708	22.5	10.99
2	Conv_BN_ReLU_MaxPool	6	2.67	1336	5.62	2.75
3	Conv_BN_ReLU	20	4.37	2184	18.75	9.16
4	Conv_BN_ReLU_MaxPool	20	6.82	3412	4.69	2.29
5	Conv_BN_ReLU	34	2.78	1388	7.97	3.89
6	Conv_BN_ReLU	34	4.41	2204	7.97	3.89
7	Conv_BN_ReLU_MaxPool	34	4.42	2212	1.99	0.97
8	Conv_BN_ReLU	44	1.51	756	2.58	1.26
9	Conv_BN_ReLU	44	2.07	1036	2.58	1.26
10	Conv_BN_ReLU_MaxPool	44	2.08	1040	0.64	0.31
11	Conv_BN_ReLU	44	0.52	260	0.64	0.31
12	Conv_BN_ReLU_MaxPool	44	0.52	260	0.64	0.31
13	Conv	56	0.66	328	0.82	0.40
14	Conv	56	0.10	48	0.82	0.40
-	Concat	100	-	-	1.46	0.71
15	Conv_BN_ReLU	44	0.14	68	0.64	0.31
16	Conv_BN_ReLU	34	0.38	192	0.49	0.24
-	Upsample	-	0.05	22.5	1.9	0.97
-	Concat	78	-	-	4.57	2.24
17	Conv_BN_ReLU	34	0.29	148	1.9	0.97
18	Conv_BN_ReLU	20	0.69	346	1.17	0.57
-	Upsample	-	0.133	66.5	4.69	2.29
-	Concat	54	-	-	12.66	6.18
19	Conv_BN_ReLU	20	0.52	260	4.69	2.29
20	Conv_BN_ReLU	6	0.52	262	1.41	0.69
-	Upsample	-	0.13	67.5	5.62	2.75
-	Concat	26	-	-	24.37	11.90
21	Conv_BN_ReLU	6	0.34	168	5.62	2.75
22	Conv_BN_ReLU	6	0.70	352	5.62	2.75
23	Conv_ReLU	32	0.66	328	30	14.65
24	Conv_ReLU	32	17.48	8744	30	14.65
25	Conv_ReLU	16	8.74	4372	15	7.32
26	Conv_ReLU	16	0.54	272	15	7.32
27	Conv	2	0.06	32	1.87	0.91

Comp. Load- Computation load, **Req. Comp. Cap.**- Required Compute Capability, **Req. Mem. Cap.**- Required Memory Capability.

n times the computation load and memory footprint to be completed in one second. We obtain the required compute and memory capabilities by multiplying layer-wise computation and memory footprint by required fps. Table 3 contains compute and memory capabilities for a 500fps throughput. Using the layer-wise computational and memory requirements we note that layer 24 has the maximum compute and memory bandwidth requirements of 32.8 TFLOP/s (Tera FLoating-point OPerations/Second) and 14.65 GB/sec, respectively. In case all layers function in a pipelined-manner to deliver an overall throughput of 500 fps, the total requirements are sum of individual layers requirements. Consequently, the total compute requirement is 32.8 TFLOP/s and the memory bandwidth requirement is 120.6 GB/s.

We now discuss some architectural options for C6 model. The model architecture suggests that the layers can be processed in a cascade manner and the data dependency between layers necessitates that the computation of previous layer completes before successive layer processing can start. This applies to all layers and operations of the C6 model. A fully-pipelined design can support cascade-type of data dependency and would yield a throughput of 1fps/clock cycle. FPGAs are known to support deep pipelines owing to its datapath-based compute paradigm. However, implementing such a design in fully-pipelined manner would impose impractical compute and memory capability requirement on the FPGA. Alveo U280 is a datacentre FPGA and contains 9k+ DSPs (Digital Signal Processors), 45MB on-chip SRAM, 8GB HBM (High-Bandwidth Memory) with capacity to house 3 DPUs designed to accelerate ML workloads (see Fig. 4. The combined compute capability of all DPUs is 17.2TOP/s[24] for INT8 compute. The on-chip SRAM (45MB) has the lowest access time but is insufficient to store intermediate outputs of layers (247.1MB). DPUs use HBM to store the intermediate data but the transfer latency between HBM and FPGA is quite high ($\approx$ 55 cycles)[22]. These requirements on compute and memory limit the overall throughput that can be extracted from a fully-pipelined design. Other viable architectural option is to perform the computation layer-wise *i.e.* while extracting full parallelism within a layer. For such an option, the compute and memory capability requirement is governed by their respective bottleneck layers. For instance, the bottleneck layer for C6 model is Layer #24 with 8.7TFLOP/s and 14.65GB/s as capability constraints. GPUs are a fair choice to support highly parallel designs. In case of a V100 GPU, streaming microprocessors(SM), contain double-precision, single-precision, int and tensor processing cores. The SMs contain dedicated and shared hierarchical cache system (L0, L1 and L2) for instruction and data. V100 GPU micro-architecture is shown in Fig. 4 and can support 15.7TFLOP/s (FP32) with register file and L1 cache of 256KB and 6.1MB, respectively. The memory requirement for storing intermediate outputs is not met and high performance penalty is incurred due to data round-trips through HBM. A IPU-M2000 has 1 PFLOP/s in FP16 (250 TFLOP/s in FP32) compute and 900MB in-processor memory with 47.5TB/s memory bandwidth. The IPU satisfies the compute and memory requirements estimated by our roofline model and should support a 500 fps throughput while implementing compressed CRAFT in layer-wise

manner. However, the achieved throughput might differ based on the based on the data dependency of model, IPU architecture and operation scheduling by IPU compiler.

5 RESULTS AND DISCUSSIONS

5.1 Experimental Setup

We evaluated IPU for variation in throughput, power and accuracy capabilities with compute precision, # of IPUs and batch size. We used PopART (Poplar Advanced Run Time)[8] version 2.4 to compile models for IPU. The replicationFactor flag is used to vary the # of IPUs. For FPGAs, the batch size can be mentioned in terms of number of threads. For CPU and GPU, the batch is passed 4-D tensor where the 4^{th} dimension represents the batch size and <channel, height, width> are the other three dimensions.

5.1.1 Throughput Readings. To calculate throughput, we use the time commands and recorded time taken to run a batch of images. The result obtained by taking ratio of size of batch to batch processing time is regarded as throughput. The processing time includes the image and result transfers between host and device.

5.1.2 Power Readings. Power is recorded using gcmonitor, an in-built Graphcore tool with GCDA_MONITOR set to 1. The power reading for GPUs is taken using nvida-smi API. The ILO (Integrated Lights Out)[6] interface is used to record power for FPGA and CPU. The power number indicates the board power and reports the max total power which includes variable dynamic power and fixed static power. In all these devices, the power recorded when no workload is running is regarded as static power. The dynamic power is calculated by subtracting the static power from the total power reading recorded when workload is running on device.

5.1.3 Accuracy Readings. The predicted text boxes, obtained by creating polygons over predicted textual regions, are characterized as correctly predicted by calculating its overlap (IoU threshold = 0.4) with boxes from ground truth. From the predicted and ground truth text boxes, we estimate accuracy (precision and recall) of model.

5.2 Hardware Platforms Specifications

For comparative studies with Graphcore IPU, we have used Intel Xeon Gold CPU (base frequency- 2.3G*Hz*, 16.5MB cache)[11], Intel Xeon CPU E5-2690 (base frequency- 2.9G*Hz*, 20MB cache)[12], GPUs - NVIDIA V100[14] and A100[15], and Alveo U280 FPGA[23] for performance testing of the compressed CRAFT models. The A100 and V100 are high end GPUs with 7936 and 5120 CUDA cores, respectively. Additionally, A100 GPU has special hardware to accelerate sparse computations. We use VITIS AI development flow[25] for our experiments to evaluate neural networks on FPGA DPUs[24]. The Alveo U280 DPU contains three cores and a total of 14 processing engines and can process as many images in parallel at 300 M*Hz*. We measure power and present throughput efficiency for Intel Xeon CPU E5-2690 system.

5.3 Throughput Speed-up for Compressed Models

The IPU-M2000 machine is not able to run native CRAFT model due to its memory footprint (81MB) and returns out-of-memory error. We did not have a concrete reason for this behaviour and investigation is still in progress. However, the compressed CRAFT models runs on IPU-M2000 without errors. We present throughput for compressed CRAFT models in Table 4. All models record a throughput higher than 200fps. The highest throughput is obtained for most compressed models, C6 and C4, since their compute and memory load is smaller than other models. The ratio of footprints of native and compressed CRAFT is indicated as compression ratio. Model C8 is only 60× compressed but realises a higher throughput as compared to 80× compressed C18 model. This is because the in and out channels for C8 model are multiple of 8 whereas such structure is not seen in C18 models. This observation indicates that models with # of in/out channels as powers of 2 map better on IPU. The FP16 implementation yields higher throughput (upto 390fps) as compared to FP32 compute. We also observe that for FP32 compute, the IPU offers marginal improvement over A100 DGX[16] but the gains over DGX is higher (upto 1.33×) in case of FP16 compute.

Table 4: Throughput and speed-up over A100 DGX for compressed CRAFT models on 4 IPU configuration.

Model	Comp	FP32		FP16	
		Thrpt (in fps)	Gain over A100 DGX	Thrpt (in fps)	Gain over A100 DGX
C32	14x	205.34	1.01×	290.28	1.33×
C16	52x	231.48	1.03×	308.64	1.30×
C8	60x	260.42	1.02×	333.56	1.18×
C18	80x	252.53	1.08×	313.48	1.18×
C12	110x	265.25	1.06×	327.87	1.20×
C6	142x	271.33	1.06×	375.80	1.27×
C4	180x	308.64	1.06×	390.17	1.12×

Comp- Compression ratio, **A100 DGX** contains 8 A100 GPUs, **Thrpt**- Throughput

5.3.1 Resolving Network Bottleneck. Graphcore has an option to run the performance test with synthetic images. The synthetic images are generated on the IPUs instead of the conventional way of transfer from host to device. By doing this, the network bandwidth does not get engaged. The throughput achieved with synthetic images was >10× higher than test of IC15 images (results reported in Table 4). The network bandwidth roofline can be estimated using the transfer size and supported bandwidth (100Gbps-full duplex) of the host-IPU-M2000 interface. The input image and output features amount to 11.25MB and 1.875MB, respectively. Between these transfers, the input transfer is the bottleneck. The network can support 1060fps with full utilization of 100Gbps bandwidth. For C6 model, the throughput measured on synthetic dataset is 3984fps. This indicated that network bandwidth is limiting the maximum achievable throughput on real test dataset. As shown in Fig. 1, the IPU connects to host system through 100Gbps Ethernet interface. The achievable throughput for C4 (390fps) translates to a realized network bandwidth of 34.3Gbps for an input size of 11.25MB. We can see that there is scope for higher throughput gain by improving realizable network bandwidth. Additionally, for this use case it will be impossible to achieve a throughput of 3984 fps achieved in the synthetic benchmark because of network bottleneck, rendering the Graphcore IPU M2000 system underutilized.

We addressed this, by reducing the memory transfers pertaining to test images. We typecast the test image from FP32 to INT8 when transferring image to device and typecast back to compute precision

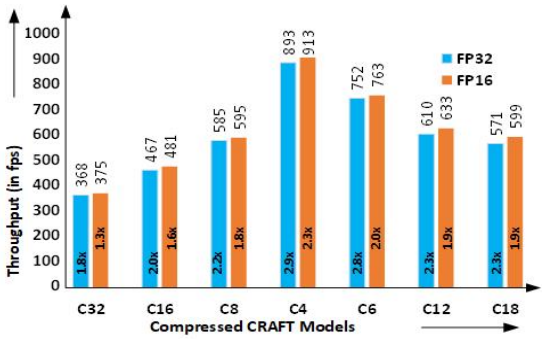

Figure 5: Throughput gain by typecasting test images for compressed CRAFT models.

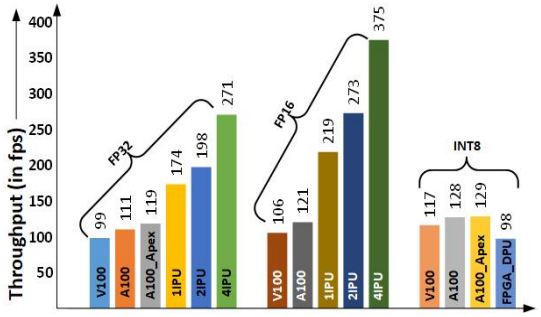

Figure 6: Throughput *vs.* compute precision for compressed CRAFT (C6 model) on IC15 dataset.

(FP32 or FP16) before the compute begins. The transfer size for input reduced from 11.25MB to 2.81MB because of typecasting. By doing this, we were able to siphon more throughput from IPUs as shown in Fig. 5. The maximum throughput is achieved for C4 model as 913fps after this optimization that is 2.3× higher as compared to image transfer with FP32 precision. Similar trend is observed in other models in both, FP32 and FP16, precision. This can be attributed to reduction in transfer time after typecast. This experiment helps us validate that the throguhput bottleneck in this case is the network transfer of images from the host. However, we observe that the accuracy drop is significant with recall falling from 78.6% to 23.7%.

5.4 Throughput *vs.* Precision Trade-off

The precision of computation denotes the number of bits and datatype used in calculations. Nowadays, CPU and commercial hardware accelerators support multiple precision. Lower precision compute offers higher computational power (in terms of GOP/S) which translates to performance improvements. For instance, V100 GPU supports FP32 and FP16 computations but the latter has 2× the computational power[14].

The Graphcore IPU supports two precisions for compute, FP32 and FP16. The default precision is FP32 and FP16 can be realised by adding *.half()* in model invocation. The throughput variation with compute precision in IPU for 1, 2 and 4 IPUs is shown in Fig. 6. We observe that for FP16 the IPUs support a higher (upto 1.3×) throughput as compared to FP32 precision. Additionally, the throughput obtained with 2 IPUs is 1.14× w.r.t throughput with 1

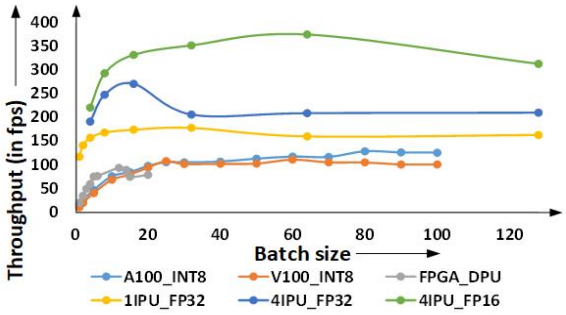

Figure 7: Throughput *vs.* Batchsize comparison of hardware platforms for compressed CRAFT (C6 model) on IC15 dataset.

IPU. The ratio between throughput between 4 IPUs and 1 IPU is 1.55. This indicates that the throughput scaling is not linear with # of IPUs *i.e.* increasing the amount of compute resources does not lead to proportional increase in throughput. This investigation for the underlying reason for this behaviour is in progress. However, a probable reason can be saturation of memory bandwidth owing to the nature of the test model which requires outputs from convolution network during upsampling.

5.4.1 Accuracy vs. Precision Trade-off. Reducing the precision affects the prediction accuracy since the number of bits used in computation reduces. The accuracy measured on IPU-M2000 for C6 model remains similar to actual accuracy for C6 model. This is desirable since a substantial drop in accuracy in exchange for throughput gain is not acceptable. An accuracy drop of ≈1-1.5% is seen for precision FP32 and FP16 as shown in Table 5.

Table 5: Test accuracy for C6 model on IC15 dataset.

Hardware	Compute Precision	Precision	Recall
GPU	FP32	84.3%	78.3%
Graphcore	FP32	83.11%	77.32%
	FP16	83.08%	77.19%

5.4.2 Throughput Comparison with other hardware. We compute throughput for C6 model on A100, V100 and FPGA in FP32, FP16 and INT8 compute precisions. The model is compiled in FP16 and INT8 using TensorRT[17] (maps computation to tensor cores) that quantizing models on GPU. In addition to TensorRT, Apex library is used to compile models for A100. The Apex library introduces 2:4 sparsity into models. 2:4 sparse models are supported on A100 GPU owing to its special hardware. The best batch size for A100, V100 and FPGA DPU are 65, 80 and 12, respectively.

Among other hardware, the maximum throughput achieved by C6 model is on A100 GPU with INT8 precision in Apex and TensorRT compile. We observe that IPU implementation outperforms other hardware in FP32 and FP16 precision. The 4 IPU implementation in FP16 gives the highest throughput of 375fps that is ≈3× higher that highest throughput supported by other hardware.

5.5 Throughput *vs.* Batch size Trade-off

We present the variation of throughput with batch size for C6 FP32 model on IC15 dataset in Fig. 7. The throughput variation for all IPUs indicates a sharp increase for small batch sizes that attain a

peak and levels off later for batch sizes ⟩50. The throughput attains a peak at batch size 16 for 1 and 4 IPU implementation, whereas 2 IPU peak is at batch size 32. The 4IPU configuration peaks at batch size 64 at FP16 precision.

5.5.1 Comparison with other hardware platforms. Among hardware other than IPUs, we observe that DPU have higher throughput as compared to GPUs for batch size upto 12 images (see Fig. 7) . For batch size upto 20 images, A100 and V100 have similar throughput beyond which A100 offers higher throughput than V100. However, the IPU implementations offer higher throughput gains as compared to all other hardware. The throughput gain over other hardware is prominent for small batch sizes as compared to large batches. Additionally, 4 IPU with FP16 precision has the highest throughput for all batch size with peak throughput at batch size of 64. IPU capability to support high performance on small batch sizes makes them useful for low latency and real-time applications.

5.6 Profiling impressions from PopVision tool

We analyse the performance profile of the C6 model using PopVision tool available for Graphcore hardwares. The C6 models contains 76 layers including convolution (3×3 and 1×1), maxpool (2×2), batch normalization and ReLU. Additionally, there are some mathematical operations such as concatenation and upsampling (2×2). The profiling information, in terms of clock cycles, is not available for mathematical operations.

The total clock cycles for C6 model is 5.4 million and its breakup among model layers is shown in Table 6. The operation taking 68.5% of cycles is streamcopy which includes test image transfer time over network. This indicates that only ≈30% of cycles are spent on compute. If we extrapolate this to 100% compute cyles of 4 IPUs, the estimated throughput is 3614fps which is very close to the throughput achieved with synthetic test image time minus network transfer (discussed in sec. 5.3.1). Thus, the host-IPU communication is acting as a bottleneck and optimizing for network bandwidth can help improve throughput further. After streamcopy, convolution and batch normalization are the compute bottleneck operations. This indicates that memory is limiting the throughput and not compute.

Table 6: PopVision profiling data for C6 model.

Layer Name	# of layers	% of cycles spent
Convolution	27	15.1%
Maxpool	5	3.3%
BN	20	11.2%
ReLU	24	1.9%
streamcopy amounts for 68.5% of the cycles		

5.7 Dynamic Power Consumption Observation

The dynamic power consumption observations are presented in Fig. 8. The 4IPU implementations consume more power as compared to 2 and 1 IPU deployments. The FP16 compute consumes less power as compared to FP32 designs for small batch sizes but the gap reduces for batch sizes beyond 128. This indicates that lower precision offers power as well as throughput advantages over higher compute precision.

5.7.1 Comparison with other hardware. The CPU (Intel Xeon E5-2690), A100 GPU and FPGA DPU consume higher power as compared to 1 and 2 IPU configurations for all batch sizes. For small batch sizes, the power consumption of A100 and CPU coincides with 4 IPU configuration but the former uses more power for higher batch sizes.

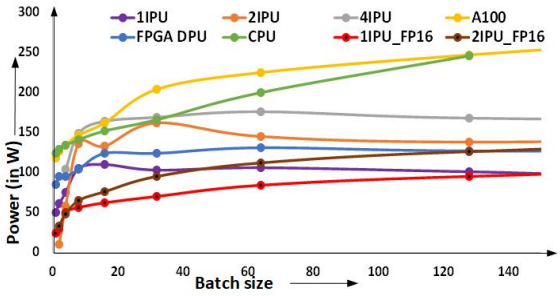

Figure 8: Dynamic power variation with batch size and # of IPUs.

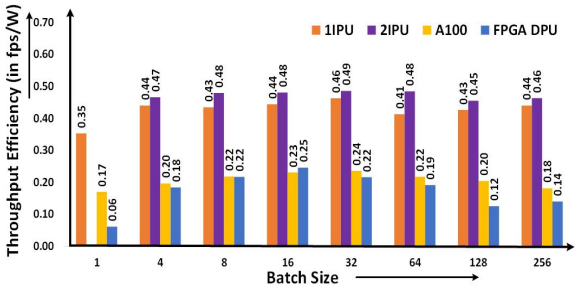

Figure 9: Throughput Efficiency comparison other devices.

5.8 Throughput Efficiency Variations

The throughput efficiency is measured as throughput per unit watt and a higher value denotes better deployment option. We present the throughput efficiency variation with # of IPUs in Fig. 9. We observe, 4 IPU configuration has maximum throughput efficiency of 0.61fps/W. This can be attributed to IPU micro-architecture where memory and compute units are closeby thereby saving power on data transfers between the two. For batch sizes >16, the 2 and 4 IPU configurations offer similar throughput efficiency.

5.8.1 Comparison with other hardware. We use devices A100 GPU and FPGA DPU for comparison and their throughput efficiency is presented in Fig. 9. The A100 achieves a maximum throughput efficiency of 0.24fps/W whereas the same for FPGA DPU is 0.25fps/W. IPU (1 and 2 IPUs) configurations have higher throughput efficiency as compared to GPU and FPGA devices. We also compare the IPU (4 IPU configuration) throughput efficiency with system, A100-DGX and Xeon CPU E5-2690 (see Fig. 10). The CPU system exhibits higher throughput efficiency at small batch sizes as compared to large batch sizes. The same trend is seen in A100-DGX. However, the throughput efficiency of A100-DGX is 11-27× than CPU. In case of A100-DGX, the throughput efficiency for large batch sizes

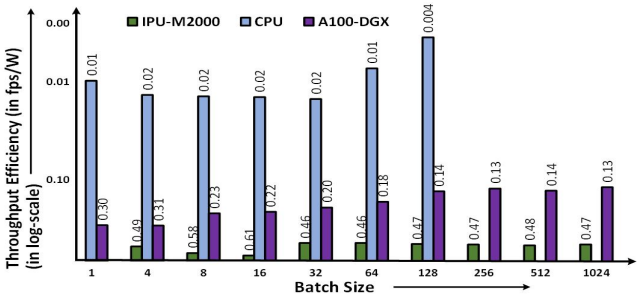

Figure 10: Throughput Efficiency comparison with other systems.

reduces because of increase in # of GPU units utilized and consequent increase in static power. For instance, 4 GPUs are used for batch size 8 and 8 GPUs for batch size ≥32. As compared to IPU configurations, the throughput efficiency for A100 and FPGA devices as well as CPU and DGX system is lower. This indicates energy-efficient processing capability of IPUs.

6 CONCLUSION

Graphcore IPU-M2000 is a new hardware available with in-memory processing capabilities to accelerate machine learning workloads. We use a compressed version of text region detection model (CRAFT) to evaluate its performance. Upon comparison with state-of-the-art hardware platforms, we realise IPU can support much higher throughput especially the FP16 implementations. Additionally, we analyse the profile of the target application and observe that the network transfer from interfacing host to Graphcore IPU-M2000 system as the bottleneck. We intend to look for ways to improve network utilization and use a pipelined architecture to maximize the inferencing throughput of the system. Our experiments on power comparison suggest IPU implementation are power-efficient as compared to GPU, FPGA and CPU. On comparison across systems (CPU and A100-DGX), we find that the Graphcore system delivers maximum throughput across all batch sizes. Additionally, it delivers maximum throughput efficiency which makes a good case for Graphcore to be adopted in data centers and by cloud service providers as ML accelerator.

ACKNOWLEDGMENTS

We would like to acknowledge Graphcore Team: Helen and Sudhakar for sharing access and technical details of IPU_POD_{16}. We also thank Rekha from TCS Research for facilitating the IPU access.

REFERENCES

[1] Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. 2019. Character Region Awareness for Text Detection. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), 9357–9366.

[2] Youngmin Baek, Seung Shin, Jeonghun Baek, Sungrae Park, Junyeop Lee, Daehyun Nam, and Hwalsuk Lee. 2020. Character Region Attention For Text Spotting. *ArXiv* abs/2007.09629 (2020).

[3] Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model Compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Philadelphia, PA, USA) *(KDD '06)*. Association for Computing Machinery, New York, NY, USA, 535–541. https://doi.org/10.1145/1150402.1150464

[4] Robust Reading Competition:. 2021. Scanned Receipts OCR and Information Extraction. Retrieved December 28, 2021 from https://rrc.cvc.uab.es/?ch=13&com=evaluation&task=1

[5] Corinna Cortes, M. Mohri, and Afshin Rostamizadeh. 2009. L2 Regularization for Learning Kernels. In *UAI*.

[6] HP Enterprise. 2016. HPE iLO 4 User Guide. Retrieved December 28, 2021 from https://www.ni.com/pdf/manuals/377263a.pdf

[7] Jonathan Frankle and Michael Carbin. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. *arXiv: Learning* (2019).

[8] Graphcore. 2021. PopART User Guide.

[9] Graphcore. 2021. V-IPU User Guide. Retrieved December 28, 2021 from https://docs.graphcore.ai/projects/vipu-user/en/latest/concepts_and_architecture.html

[10] Zheng Huang, Kai Chen, Jianhua He, Xiang Bai, Dimosthenis Karatzas, Shijian Lu, and C. V. Jawahar. 2019. ICDAR2019 Competition on Scanned Receipt OCR and Information Extraction. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*. 1516–1520. https://doi.org/10.1109/ICDAR.2019.00244

[11] Intel. 2019. Intel®Xeon®Gold 5118 Processor. Retrieved December 28, 2021 from https://ark.intel.com/content/www/us/en/ark/products/120473/intel-xeon-gold-5118-processor-16-5m-cache-2-30-ghz.html

[12] Intel. 2019. Intel®Xeon®Processor E5-2690. Retrieved December 28, 2021 from https://ark.intel.com/content/www/us/en/ark/products/64596/intel-xeon-processor-e52690-20m-cache-2-90-ghz-8-00-gts-intel-qpi.html

[13] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2015).

[14] nVIDIA. 2017. nVIDIA Tesla V100 GPU Architecture. Retrieved December 28, 2021 from https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf

[15] nVIDIA. 2020. NVIDIA A100 Tensor Core GPU Architecture. Retrieved December 28, 2021 from https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf

[16] nVIDIA. 2021. DGX A100 System User Guide. Retrieved December 28, 2021 from https://docs.nvidia.com/dgx/pdf/dgxa100-user-guide.pdf

[17] nVIDIA. 2021. nVIDIA TensorRT Developer Guide. Retrieved December 28, 2021 from https://docs.nvidia.com/deeplearning/tensorrt/developer-guide/index.html

[18] Cambrian AI Research. 2021. Graphcore'S AI Software Stach is now Customer-Driven. Retrieved 28-12-21 from https://www.graphcore.ai/hubfs/Cambrian%20AI%20white%20paper_Graphcores%20AI%20software%20stack%20is%20now%20customer%20driven.pdf

[19] K. Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR* abs/1409.1556 (2015).

[20] Vladislav Sovrasov. 2021. Flops Counter. Retrieved December 28, 2021 from https://github.com/sovrasov/flops-counter.pytorch

[21] D. Vishwanath, R. Rahul, Gunjan Sehgal, Swati, Arindam Chowdhury, Monika Sharma, L. Vig, Gautam M. Shroff, and A. Srinivasan. 2018. Deep Reader: Information extraction from Document images via relation extraction and Natural Language. *ArXiv* abs/1812.04377 (2018).

[22] Zeke Wang, Hongjing Huang, Jie Zhang, and Gustavo Alonso. 2020. Shuhai: Benchmarking High Bandwidth Memory On FPGAS. In *2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. 111–119. https://doi.org/10.1109/FCCM48280.2020.00024

[23] Xilinx. 2021. Alveo U280 Data Center Accelerator Card Data Sheet. Retrieved December 28, 2021 from https://www.xilinx.com/support/documentation/data_sheets/ds963-u280.pdf

[24] Xilinx. 2021. DPUCAHX8H for Convolutional Neural Networks. Retrieved December 28, 2021 from http://xilinx.eetrend.com/files/2021-08/wen_zhang_/100553088-217819-pg367-dpucahx8h.pdf

[25] Xilinx. 2021. Vitis AI User Guide. Retrieved December 28, 2021 from https://www.xilinx.com/products/design-tools/vitis/vitis-ai.html#:~:text=Optimal%20Artificial%20Intelligence%20Inference%20from,%2C%20models%2C%20and%20example%20designs

[26] Marco Maggioni Zhe Jia, Blake Tillman and Daniele P. Scarpazza. 2019. Dissecting the Graphcore IPU Architecture via Microbenchmarking. Retrieved December 28, 2021 from https://www.graphcore.ai/hubfs/assets/pdf/Citadel%20Securities%20Technical%20Report%20-%20Dissecting%20the%20Graphcore%20IPU%20Architecture%20via%20Microbenchmarking%20Dec%202019.pdf?hsLang=en