영상인식 및 분류용 인공지능 가속기의 최신 성능평가: MLPerf를 중심으로

□ 서영호, 박성호*, 박장호* / 광운대학교, *미국 Cosignon

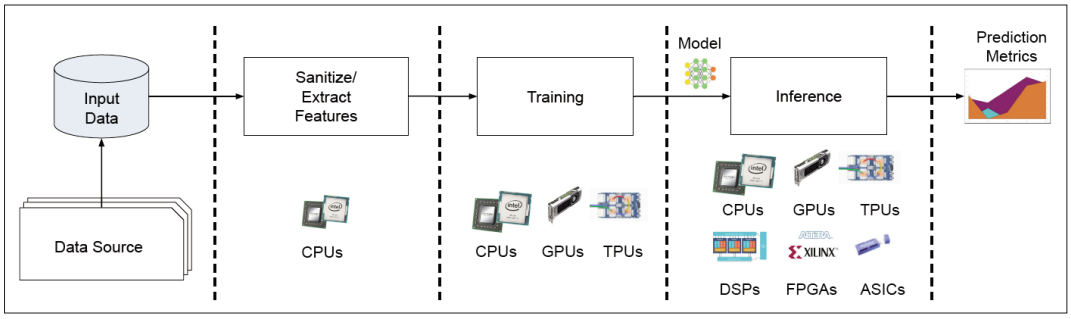
요약

인공지능의 고속화를 위한 인공지능용 혹은 딥러닝용 하드웨어 및 소프트웨어 시스템에 대한 수요가 폭발적으로 증가하고 있다. 또한 딥러닝 모델에 따라 다양한 추론 시스템이 끊임없이 연구되고 소개되고 있다. 최근에는 전세계에서 100개가 넘는 회사들에서 인공지능용 추론 칩을 개발하고 있고, 임베디드 시스템에서 데이터센터 솔루션에 이르기까지 다양한 분야를 위한 것들이 존재한다. 이러한 하드웨어의 개발을 위해서 12개 이상의 소프트웨어 프레임 워크 및 라이브러리가 활용되고 있다. 하드웨어와 소프트웨어가 다양한 만큼 이들을 중립적으로 평가하기가 매우 어려운 실정이다. 따라서 업계 표준의 인공지능을 위한 벤치마킹 및 평가기준이 필요한데, 이러한 요구로 인해 MLPerf 추론이 만들어졌다. MLPerf는 30개 이상의 기업과 200개 이상의 머신러닝 연구자 및 실무자들에 의해 운영되고, 전혀 다른 구조를 갖는 시스템을 비교할 수 있는 일관성 있는 규칙과 방법을 제시한다. MLPerf에 의해 제시된 규칙에 의해 2019년도에 처음으로 다양한 인공지능용 추론 하드웨어가 벤치마킹을 수행했다. 여기에는 14개의 회사에서 600개 이상의 추론 결과를 측정하였으며, 30개가 넘는 시스템이 이러한 추론에 사용되었다. 본 원고에서는 MLPerf의 학습과 추론을 중심으로 하여 최근에 개발된 다양한 회사들의 인공지능용 하드웨어, 즉 가속기들의 성능을 살펴보고자 한다.

Ⅰ. 머신러닝 파이프라인

기계 학습에는 일반적으로 〈그림 1〉과 같은 일련의 복잡한 작업으로 구성된다. 거의 모든 머신러닝용 파이프라인은 모델을 훈련하고 테스트하기 위해 데이터를 수집하는 것으로 시작한다. 실제 데이터에는 종종 머신러닝 모델의 품질과 정확성을 저하시키는 요인들이 포함되어 있기 때문에 원래의 데

※ 본 원고는 MLPerf의 Training Benchmark whitepaper(https://arxiv.org/abs/1910.01500) 및 Inference Benchmark whitepaper(https://arxiv.org/abs/1910.01500)를 참고하여 작성하였다. 본 원고의 그림은 위의 두 문서의 그림을 참고하였고, 훈련 및 추론 결과는 MLPerf에서 공개한 결과를 이용하였다.



〈그림 1〉 일반적인 머신러닝 파이프 라인의 단계

이터는 일반적으로 사용하기 전에 전처리 및 정규화 과정을 거친다.

머신러닝의 벤치마킹은 학습과 추론의 두 단계로 구성된다. 훈련하는 동안 모델은 입력 값을 이용하여 결과를 예측하는 방법을 학습한다. 예를 들어, 모델은 사진의 내용이 무엇인지 또는 영어에서 한글로 문장을 번역하는 등의 예측하는 방법을 학습한다. 반면 추론하는 동안 모델은 입력에 대해 예측은 하지만 더 이상 학습하지는 않는다. 머신러닝이 연구 단계에서 다양한 상용화 단계로 이동하고 보편화되면서 이러한 두 가지 단계는 매우 중요해지고 있다.

II. 학습을 위한 벤치마킹

기계 학습은 컴퓨터 비전, 자연어 처리, 자율 주행 및 자율 로봇 등의 분야에서 널리 이용되고 있는 기술이지만, 실제로 이들을 상용화 수준에서 사용하기 위한 소프트웨어는 매우 거대하고 좋은 성능을 내기 위해서는 학습 및 추론에 많은 비용을 투자해야 한다[1-6]. 이러한 비용 증가를 해결하기 위해서 하드웨어, 소프트웨어 및 시스템 개발자들은 최적화된 하드웨어, 소프트웨어 및 시스템을 설계하고 다양한 최적화 방법을 적용함으로써 추론 성능을 높이고자 노력하고 있다. 예측치에 따르면 100개가 넘는 회사가 최적화된 추론용 칩을 생산하거나 상용화를 할 계획에 있다. 그에 비해서 학습을 위한 칩은 약 20여개의 회사만이 개발하고 있는 것으로 보고되고 있다.

딥러닝의 훈련을 위한 하드웨어 및 소프트웨어 시스템 옵션의 수가 증가함에 따라 보편성을 갖는 표준화된 벤치마킹 방법이 필요하게 되었다[7-13]. 지금까지 이러한 벤치마킹을 위한 노력이 꾸준히 진행되었고, 최근에 그러한 움직임이 매우 활발해져왔다[14]. 예를 들어, 1980년대의 마이크로프로세서 및 데이터베이스 시스템의 혁신은 Unix 서버용 표준 성능 평가 공사(SPEC)(1991년)와 트랜잭션 처리 및 데이터베이스 용 트랜잭션 처리 성능위원회(TPC)를 탄생시켜서 널리 활용되었다[15]. SPEC 및 TPC의 성공에서 영감을 얻은 산업계와 교육기관의 컨소시엄인 MLPerf는 딥러닝의 표준화된 벤치마킹 문제를 해결하기 위해 설립되었다. 다른 분야들과 달리 딥러닝은 본질적으로 확률론적이며 다양한 통계기법, 하드웨어 및 소프트웨어 최적화를 적용할 수 있다. 이러한 모델의 최적화는 성능과 학습 속도 등을 향상시킬 수 있지만 일부 최적화는 그

렇지 않은 결과를 가져오기도 한다. 모델의 최종적인 품질 혹은 정확도를 변경하기 위해서는 모델 자체의 변경을 수반하는 경우도 있어서 성능을 측정하는 방법을 결정하는데 매우 어려움이 있다. 딥러닝은 근사적이고 확률론적이기 때문에 정답이라는 것이 존재할 수도 있지만 확인하기 어려운 경우도 있고, 결과의 이분법적인 평가가 어려운 경우도 있다. 또한 동일한 결과에 대해서도 다양한 학습과정과 모델이 사용될 수도 있기 때문에 벤치마킹을 위한 포괄적인 방법과 과정을 정의하기는 매우 어려운 일이다.

Ⅲ. 추론을 위한 벤치마킹

각 시스템과 칩들은 추론을 위한 고유의 개발방법을 적용하여 대시기간, 처리량, 전력, 모델의 품질 간의 균형을 유지하고 있다. 예를 들어, 양자화 및 정밀도의 감소는 정확도를 낮추었지만 추론에 대한 대기시간, 처리량 및 전력의 효율을 향상시키는 강력한 방법이다[16,17]. 부동소수점 숫자로 학습한 후 모델 가중치를 압축하면 메모리 대역폭을 감소시킬 수 있고, 계산당 처리량을 증가시켜 시간적인 성능을 향상시킬 수 있다. 또한 가중치를 제거함으로써 메모리에 대한 비용을 줄일 수도 있다[16][18][19]. 이러한 다양한 기술을 적용하는 것은 각 칩들에 따라 전혀 다르지만 결국 최종 모델의 품질을 감소시킬 수 있는 위험이 있다. 그러나 현재로써는 표준화된 혹은 시장에서 공인된 평가 방법이 없기 때문에 이들의 성능에 대해 보장을 할 수 없다. 따라서 구조적으로 중립적이고, 대표성을 가지면서 재현할 수 있는 머신러닝용 추론을 위한 벤치마킹 방법과 체계가 필요하다.

표준화된 벤치마킹 방법이 어려운 이유는 인공지능 분야의 생태계가 기계 학습 작업, 모델, 데이터 세트, 프레임 워크, 도구 세트, 라이브러리, 아키텍처 및 추론 엔진을 조합하여 여러 가지 가능한 조합으로 추론 벤치마킹을 거의 다루기 어렵게 만드는 것에 있다. 머신러닝의 분야는 이미지 분류 및 지역화, 객체 감지 및 분할, 기계 번역, 자동 음성 인식, 텍스트 음성 변환 등이 포함되는데 최근에는 적용되는 분야가 기하급수적으로 늘어나고 있다. 이렇게 다양한 머신러닝 분야에서 요구되는 정밀도와 시간적인 요구사항이 매우 다르고, 사용하고 있는 프레임 워크에 따라서 구현 및 동작하는 방법도 달라서 일관성있는 벤치마킹을 하는 것은 매우 복잡한 문제로 여겨지고 있다. 학계와 산업계에서는 각각 머신러닝을 위한 벤치마크를 개발해 왔었다. 학계를 중심으로 Fathom 및 DAWNBench 등이 제안되었었고, 산업계에서는 AIMatrix, EEMBC 머신러닝Mark, AIX-PRT, AI Benchmark 및 TBD 등이 개발되었었다[20-24]. 이러한 벤치마크들은 머신러닝의 벤치마크에 많은 기여를 했었지만, 학계 및 산업계의 의견을 수렴하는데 부족했었기 때문에 널리 사용되는 데에는 큰 성과를 거두지 못했다. 벤치마킹 방식을 만드는 데에는 추론에 대한 올바른 성능 지표를 고안하고 평가에 대한 모델의 실제 작동 방법 등을 정확하게 반영하는 것이 중요하다. 예를 들어 대기지연의 제한에 대한 것은 클라우딩 방식의 시스템과 큰 연관이 있는 것과 같은 것이다.

Ⅳ. MLPerf 학습 벤치마킹

최근의 머신러닝 및 딥러닝 워크로드에 대해 공

정하고 유용한 벤치마크 방식을 만들기 위해 여러 주요 머신러닝 영역에서 대표적인 작업 세트를 정리하는 것을 시작으로 MLPerf의 학습을 위한 작업이 시작된다. 벤치마크는 주로 산업 및 연구 관련성을 기반으로 선택되었으며 다양한 연산에 대한 주제를 나타낸다. 일반성과 공공성을 확립하기 위해 MLPerf를 지원하는 수십 개의 산업 및 학술 기관과 연계하면서 작업이 진행되었다. 벤치마크를 저비용으로 운영하기 위해 〈표 1〉에 요약된 것처럼 대표적인 7가지 벤치마크 세트를 지정하였다. 이들 벤치마크는 이미 광범위한 작업을 포함하고 있지만 추가 벤치마크를 지속적으로 확장하기 위한 노력을 진행 중이다.

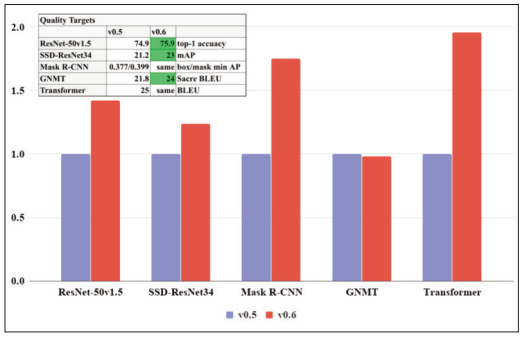
MLPerf를 이용한 성능평가를 위해 성능결과를 제출하기 위해서는 시스템 설명, 학습 세션 로그 파일 및 해당 학습 세션을 재현하는데 필요한 모든 코드와 라이브러리로 구성된다. 이 모든 것은 MLPerf 결과 게시와 동시에 MLPerf GitHub에서 공개적으로 제공되므로 재현이 가능하고 이후의 제출 시 다양한 지원자에 의해 결과를 개선할 수 있다. 시스템 설명에는 하드웨어 설명(노드 수, 프로세서 및 가속기 수 및 유형, 노드 당 스토리지, 네트워크 상호 연결) 및 소프트웨어 설명(운영 체제, 라이브러리 및 해당 버전)이 모두 포함된다. 학습 세션 로그 파일에는 중요한 작업 단계에 대한 타임 스탬프, 미리 규정된 간격으로 평가된 품질 지표, 그리고 하이퍼 파라미터의 선택 등을 포함한 다양한 정보가 포함된다. 이 로그는 후속 결과 분석을 위한 기초가 된다.

MLPerf 벤치마크에 의한 결과 보고서는 각 벤치마크에 대해 학습하는 시간을 제공한다. 전체 제품군에 걸친 제출에 대한 단일 요약 점수가 시스템 비교에 바람직 할 수 있지만, 요약 점수는 다음의 두 가지 주요 이유로 MLPerf에 적합하지 않다. 첫째, 요약 점수는 개별 벤치마크 점수의 가중치를 의미한다. MLPerf 제품군이 다루는 다양한 시스템 사용자 및 광범위한 응용 분야를 고려할 때 보편적으로 대표되는 가중치는 없을 것이다. 둘째, 제출된 단일 시스템이 제품군의 모든 벤치마크에 대한 결과를 보고하지 않으면 요약 점수의 의미가 낮아진다. 제출 과정에서 일부 벤치마크를 생략할 수 있는 여러 가지 이유가 존재할 수 있다. 그것은 모든 벤치마크가 모든 규모의 시스템들에서 실용적이거나 유효한 것은 아니다. 예를 들어, 일부 네트워크는

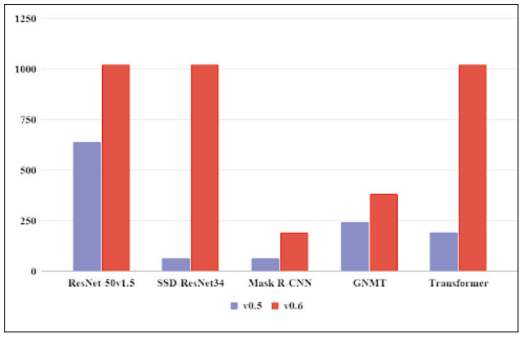
〈표 1〉 MLPerf 학습 v0.5 벤치마크

Benchmark	Dataset	Model	Quality Threshold
Image Classification	ImageNet [25]	ResNet-50v1.5 [26]	74.9% Top-1 Accuracy
Object detection (light weight)	COCO 2017 [27]	SSD-ResNet-34 [3]	21.2 mAP
Instance segmentation and object detection (heavy weight)	COCO 2017 [27]	Mask R-CNN [1]	0.377 Box min AP, 0.339 Mask min AP
Translation (recurrent)	WMT16 EN-DE [28]	GNMT [29]	21.8 Scare BLEU
Translation (non-recurrent)	WMT17 EN-DE [30]	Transformer [31]	25.0 BLEU
Recommendation	MovieLens-20M [32]	NCF [33]	0.635 HR@10
Reinforcement Learning	Go(9x9 Board)	MiniGo [34]	40.0% pro move prediction

현재 가장 큰 시스템에서 데이터 병렬 훈련에 필요한 미니 배치 크기로 훈련 할 수 없다는 사실은 모든 개발자가 이미 알고 있는 사실일 것이다. 또한 일부 프로세서는 특정 응용 분야만 대상으로 할 수



〈그림 2〉 향상된 품질 목표와 함께 MLPerf 버전 v0.5에서 v0.6으로 가장 빠른 16 칩 엔트리의 속도 향상



〈그림 3〉 MLPerf 버전 v0.5에서 v0.6으로 전체 점수가 가장 빠른 시스템에서 사용되는 칩 수가 증가한다.

〈표 2〉 MLPerf 벤치마킹을 이용한 추론 결과(https://MLPerf.org/inference-results)

#	Submitter	System	Processor	#	Accelerator	#	Benchmark results (minutes)							Details
								Image classification	Object detection, light-weight	Object detection, heavy-wt.	Translation, recurrent	Translation, non-recur.	Recommendation	
								ImageNet	COCO	COCO	WMT E-G	WMT E-G	MovieLens-20M	
							Software	ResNet-50 v1.5	SSD w/ResNet-34	Mask-R-CNN	NMT	Transformer	NCF	
Available in cloud														
0.6-1	Google	TPUv3.32			TPUv3	16	TensorFlow, TPU 1.14.1.dev	42.19	12.61	107.03	12.25	10.2	[1]	
0.6-2	Google	TPUv3.128			TPUv3	64	TensorFlow, TPU 1.14.1.dev	11.22	3.89	57.46	4.62	3.85	[1]	
0.6-3	Google	TPUv3.256			TPUv3	128	TensorFlow, TPU 1.14.1.dev	6.86	2.76	35.6	3.53	2.81	[1]	
0.6-4	Google	TPUv3.512			TPUv3	256	TensorFlow, TPU 1.14.1.dev	3.85	1.79		2.51	1.58	[1]	
0.6-5	Google	TPUv3.1024			TPUv3	512	TensorFlow, TPU 1.14.1.dev	2.27	1.34		2.11	1.05	[1]	
0.6-6	Google	TPUv3.2048			TPUv3	1024	TensorFlow, TPU 1.14.1.dev	1.28	1.21			0.85	[1]	
Available on-premise														
0.6-7	Intel	32x 2S CLX 8260L	CLX 8260L	64			TensorFlow						[1]	14.43
0.6-8	NVIDIA	DGX-1			Tesla V100	8	MXNet, NGC19.05	115.22					[1]	
0.6-9	NVIDIA	DGX-1			Tesla V100	8	PyTorch, NGC19.05		22.36	207.48	20.55	20.34	[1]	
0.6-10	NVIDIA	DGX-1			Tesla V100	8	TensorFlow, NGC19.05						[1]	27.39
0.6-11	NVIDIA	3x DGX-1			Tesla V100	24	TensorFlow, NGC19.05						[1]	13.57
0.6-12	NVIDIA	24x DGX-1			Tesla V100	192	PyTorch, NGC19.05			22.03			[1]	
0.6-13	NVIDIA	30x DGX-1			Tesla V100	240	PyTorch, NGC19.05		2.67				[1]	
0.6-14	NVIDIA	48x DGX-1			Tesla V100	384	PyTorch, NGC19.05				1.99		[1]	
0.6-15	NVIDIA	60x DGX-1			Tesla V100	480	PyTorch, NGC19.05					2.05	[1]	
0.6-16	NVIDIA	130x DGX-1			Tesla V100	1040	MXNet, NGC19.05	1.69					[1]	
0.6-17	NVIDIA	DGX-2			Tesla V100	16	MXNet, NGC19.05	57.87					[1]	
0.6-18	NVIDIA	DGX-2			Tesla V100	16	PyTorch, NGC19.05		12.21	101	10.94	11.04	[1]	
0.6-19	NVIDIA	DGX-2H			Tesla V100	16	MXNet, NGC19.05	52.74					[1]	
0.6-20	NVIDIA	DGX-2H			Tesla V100	16	PyTorch, NGC19.05		11.41	95.2	9.87	9.8	[1]	
0.6-21	NVIDIA	4x DGX-2H			Tesla V100	64	PyTorch, NGC19.05		4.78	32.72			[1]	
0.6-22	NVIDIA	10x DGX-2H			Tesla V100	160	PyTorch, NGC19.05					2.41	[1]	
0.6-23	NVIDIA	12x DGX-2H			Tesla V100	192	PyTorch, NGC19.05			18.47			[1]	
0.6-24	NVIDIA	15x DGX-2H			Tesla V100	240	PyTorch, NGC19.05		2.56				[1]	
0.6-25	NVIDIA	16x DGX-2H			Tesla V100	256	PyTorch, NGC19.05				2.12		[1]	
0.6-26	NVIDIA	24x DGX-2H			Tesla V100	384	PyTorch, NGC19.05				1.8		[1]	
0.6-27	NVIDIA	30x DGX-2H, 8 chips each			Tesla V100	240	PyTorch, NGC19.05		2.23				[1]	
0.6-28	NVIDIA	30x DGX-2H			Tesla V100	480	PyTorch, NGC19.05					1.59	[1]	
0.6-29	NVIDIA	32x DGX-2H			Tesla V100	512	MXNet, NGC19.05	2.59					[1]	
0.6-30	NVIDIA	96x DGX-2H			Tesla V100	1536	MXNet, NGC19.05	1.33					[1]	
Preview														
0.6-31	Intel	1x 2S CLX 9282	CLX 9282	2			TensorFlow						[1]	77.95
Research														
0.6-32	Alibaba	64x Tesla V100 PCIE			Tesla V100	64	Sinian	24.37					[1]	

도 있다.

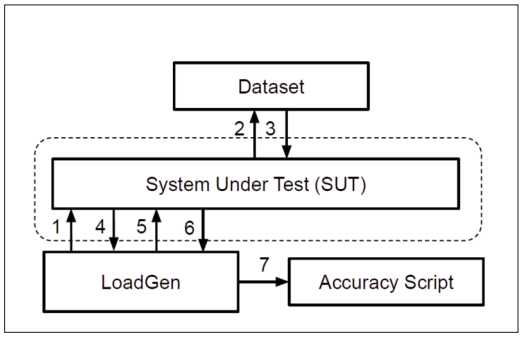
MLPerf는 모든 벤치마크와 마찬가지로 해당 분야의 발전을 장려하기 위해 경쟁을 유도하기 위한 것을 하나의 목표로 한다. 제출 라운드 간의 결과를 비교하여 이러한 공공의 목표를 향한 진행 상황을 분석하여 보고한다. 현재까지 MLPerf 학습 벤치마크에는 v0.5와 v0.6의 두 가지 제출 라운드가 있었다. 두 라운드는 6개월 간격이었고 기본 하드웨어 시스템은 바뀌지 않았다. 제출 라운드 사이에 수정되지 않았거나 제한적으로 수정된 5가지 벤치마크에 대한 결과는 MLPerf가 성능 및 구현 및 소프트웨어 스택의 스케일링에서 빠른 개선을 주도하고 있음을 보여주는 좋은 예시이다. 〈그림 2〉는 두 개의 제출 라운드 사이에서 16 칩 시스템에 제출된 최상의 성능 결과가 더 높은 품질 목표에도 불구하고 평균 1.3배 증가한 것을 보여준다. 〈그림 3〉은 두 개의 제출 라운드 사이에서 시스템의 칩이 가장 우수한 전체 성능 결과를 생성하는데 평균 5.5배 증가한 것을 보여준다. 이러한 개선 중 일부는 벤치마크 자체가 개선되었기 때문에 발생한 것이고, 일부는 대규모 ResNet 배치 크기에 대해 LAR 최적화 프로그램 허용과 같은 규칙 변경으로 가능하였다. 전세계의 여러 회사에 의해 제출된 벤치마킹 결과를 〈표 2〉에 모두 요약하여 나타내었다.

V. MLPerf 추론 벤치마킹

초기 버전 0.5에서 부족했던 부분을 반영하여 개발 프로세스를 촉진하기 위한 최소한의 벤치마크를 만드는 접근 방식을 선택함으로써 MLPerf Training v0.6 제품군이 출시되었다[26][34][35]. 산업계와 학계의 의견을 통해서 가장 적당한 범위를 갖는 현재의 버전이 만들어졌다. MLPerf Inference v0.5는 이미지 분류(ResNet-50 [1] 및 MobileNet-v1 [36]), 객체 감지(SSD-ResNet34, 즉 SSD)를 위한 3가지 작업과 5가지 모델로 구성된다[3]. ResNet34 백본 및 SSD-MobileNet-v1 및 머신 변환(GNMT [29])이 포함된다.

여러 참여 기업 및 학계와의 협력을 통해 벤치마크 제작 작업과 모델을 선택하고 일반성을 보장하기 위해 여러 피드백을 심도있게 고려하였다. 벤치마크 모델은 광범위한 기관들의 지원을 받았다. 업계에서는 이를 연구하고 효율적인 시스템을 구축할 수 있으므로 벤치마킹이 가능하며 머신러닝 시스템의 상태를 보여주는 지표가 될 수 있다. 또한 벤치마크를 모듈식으로 설계함으로써 새로운 모델을 추가하거나 작업을 하는데 있어서 많은 비용을 저감하였다. MLPerf Inference 사용자는 새로운 디자인을 쉽게 추가할 수 있는데, 앞으로 더 많은 영역, 작업, 모델 등을 포함하도록 범위를 확장할 계획이다. 또한 훈련 및 추론 벤치마크 간의 일관성과 정렬을 유지하는 것도 필요할 것이다.

완전한 MLPerf 추론 시스템에는 데이터 세트, 테



〈그림 4〉 테스트 대상 MLPerf 유추 시스템 (SUT) 및 구성 요소

〈표 3〉 MLPerf 추론 v0.5의 머신러닝 작업

AREA	TASK	REFERENCE MODEL	DATA SET	QUALITY TARGET
VISION	IMAGE CLASSIFICATION (HEAVY)	RESNET-50 V1.5 25.6M PARAMETERS 7.8 GOPS / INPUT	IMAGENET (224X224)	99% OF FP32 (76.456%) TOP-1 ACCURACY
VISION	IMAGE CLASSIFICATION (LIGHT)	MOBILENET-V1 224 4.2M PARAMETERS 1.138 GOPS / INPUT	IMAGENET (224X224)	98% OF FP32 (71.676%) TOP-1 ACCURACY
VISION	OBJECT DETECTION (HEAVY)	SSD-RESNET34 36.3M PARAMETERS 433 GOPS / INPUT	COCO (1,200X1,200)	99% OF FP32 (0.20 MAP)
VISION	OBJECT DETECTION (LIGHT)	SSD-MOBILENET-V1 6.91M PARAMETERS 2.47 GOPS / INPUT	COCO (300X300)	99% OF FP32 (0.22 MAP)
LANGUAGE	MACHINE TRANSLATION	GNMT 210M PARAMETERS	WMT16 EN-DE	99% OF FP32 (23.9 SACREBLEU)

스트 대상 시스템(SUT), 로드 생성기(LoadGen) 및 정밀도 스크립트와 같은 여러 구성 요소가 포함된다. 〈그림 4〉는 MLPerf 추론 시스템의 개요를 보여준다. 데이터 세트, LoadGen 및 정확도 스크립트는 모든 벤치마킹 과정에 대해 고정되어 있으며 MLPerf에서 제공하는 것을 사용해야 한다. 제출자는 아키텍처 요구 사항과 엔지니어링 판단에 따라 SUT를 구현할 수 있는 재량을 가질 수 있으나 MLPerf에 의한 구성요서는 반드시 명시를 함으로써 구별될 수 있도록 해야 한다.

머신러닝 벤치마크의 설계는 머신러닝 이외의 벤치마크의 설계와 근본적으로 다르다. MLPerf는 머신러닝 시스템이 수행할 수 있는 이미지 분류와 같은 고급 작업들을 정의한다. 각각에 대해 널리 사용되는 몇 가지 프레임 워크에서 표준 참조 모델을 제공하는데, 참조모델과 가중치는 머신러닝 작업의 구체적인 인스턴스를 제공하지만 공식적으로는 수학적으로 같을 필요는 없다. 참조 모델의 개념과 동등한 수준으로 구현된 유효한 클래스들은 대부분의 머신러닝 시스템에 자유도를 제공하면서도 추론 시스템들 간에 비교할 수 있게 한다. MLPerf는 32비트 부동 소수점 가중치를 사용하는 참조 모델을 제공하고 편의상 TensorFlow 및 PyTorch와 같이 가장 널리 사용되는 몇 가지 모델을 제공한다[37].

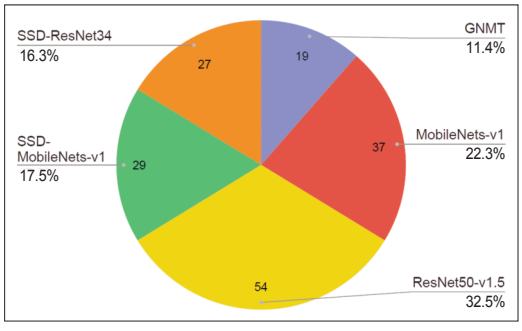
〈표 3〉에서 알 수 있듯이 관련 참조 모델과 함께 일련의 비전 및 언어 작업을 선택하였는데, 비전과 번역은 에지 장치에서 클라우드 데이터 센터에 이르기까지 모든 컴퓨팅 시스템에서 널리 사용되기 때문에 선택되었고, CNN과 RNN과 같이 서로 다른 아키텍처를 가진 완성도 있고 정상적으로 작동하는 참조 모델을 사용할 수 있다.

다양한 추론을 위한 응용 프로그램에는 다양한 사용 모델과 성능 지수가 있으며 여러 성능 메트릭이 필요하다. 이러한 모델을 다루기 위해 〈표 4〉와 같이 추론 응용 프로그램을 나타내는 네 가지 시나리오를 지정하였다. LoadGen은 시나리오를 시뮬레이션하고 성능을 측정하는데 사용된다.

〈표 4〉 시나리오 설명 및 메트릭 각 시나리오는 고객 및 공급 업체 입력을 기반으로 실제 사용 사례를 대상으로 한다.

SCENARIO	QUERY GENERATION	METRIC	SAMPLES/QUERY	EXAMPLES
SINGLE-STREAM (SS)	SEQUENTIAL	90TH-PERCENTILE LATENCY	1	TYPING AUTOCOMPLETE, REAL-TIME AR
MULTISTREAM (MS)	ARRIVAL INTERVAL WITH DROPPING	NUMBER OF STREAMS SUBJECT TO LATENCY BOUND	N	MULTICAMERA DRIVER ASSISTANCE, LARGE-SCALE AUTOMATION
SERVER (S)	POISSON DISTRIBUTION	QUERIES PER SECOND SUBJECT TO LATENCY BOUND	1	TRANSLATION WEBSITE
OFFLINE (O)	BATCH	THROUGHPUT	AT LEAST 24,576	PHOTO CATEGORIZATION

MLPerf 추론 제출에는 SUT 성능 점수, 벤치마크 코드, SUT의 주요 구성 특성(가속기 수, CPU 수, 소프트웨어 릴리스 및 메모리 시스템 등)을 강조하는 시스템 설명 파일 및 LoadGen 로그 파일에 대한 정보가 포함된다. 성능 및 정밀도는 일련의 작업 및 시나리오 조합에 의해 실행된다. 이 모든 데이터는 공개 GitHub 리포지토리에 업로드되어 릴리스 전에 피어 검토 및 유효성 검사를 수행한다. MLPerf Inference는 광범위한 적용 범위를 보장하는 작업 및 시나리오이지만 제출에는 하위 작업 및 시나리오가 포함될 수 있다. SPEC CPU와 같은 많은 기존 벤치마크에서는 모든 구성 요소에 대한 제출이 필요한데 이 방법은 임의의 코드를 실행하는 범용 프로세서에 논리적이지만 머신러닝 시스템은 이보다 더욱 복잡하고 다양한 형태를 갖기 때문에 더 많은 것을 고려해야 한다. 다른 하나는 단일 스트림 응용 프로그램과 같은 특정 시나리오를 대상으로 하며 서버 스타일 응용 프로그램을 위한 것이 아니라는 것이다. 따라서 제출자에게 작업 및 시나리오를 선택할 때 유연성을 제공 할 수 있다. MLPerf Inference v0.5 제출자는 시스템 성능을 평가할 작업을 선택할 수 있다. 따라서 여러 작업에 결과를 배포하면 해당 작업이 머신러닝 시스템 공급 업체에 관심이 있는지 여부를 알 수 있다. 전체 작업 범위를 결정하기 위해 제출 내용을 분석하였는데, 〈그림 5〉는 모델에 대한 분석을 보여준다. 가장 인기 있는 모델은 의심할 여지없이 ResNet-50 v1.5이지만 가장 인기가 적은 모델인 GNMT은 그에 비해 3배나 낮다.



〈그림 5〉 MLPerf Inference가 초기 v0.5 벤치마크에 대한 모델 분포

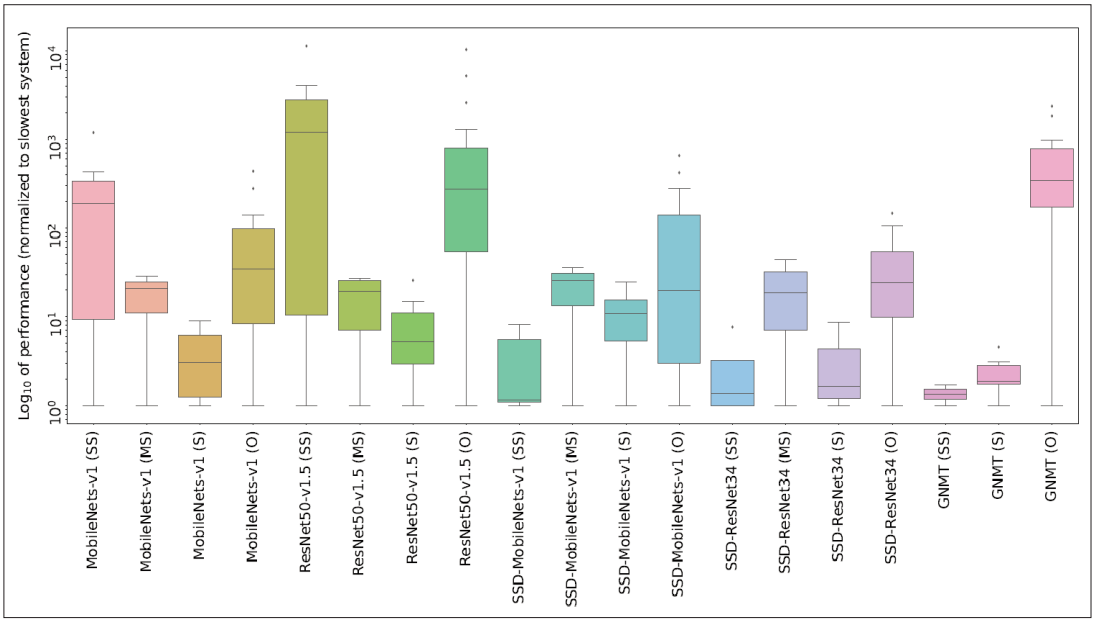
LoadGen과 시나리오는 실제로 시스템을 어떻게 사용할 것인가를 평가하는 것을 목표로 한다. 이를 위해 〈표 5〉는 다양한 작업 및 시나리오 조합에 대한 결과 분포를 보여준다.

〈표 5〉 태스크 및 LoadGen 시나리오

	SINGLE-STREAM	MULTISTREAM	SERVER	OFFLINE
GNMT	2	0	6	11
MOBILENET-V1	18	3	5	11
RESNET-50 V1.5	19	5	10	20
SSD-MOBILENET-V1	8	3	5	13
SSD-RESNET34	4	4	7	12
TOTAL	51	15	33	67

〈표 6〉 소프트웨어 프레임 워크 및 하드웨어 아키텍처 요약

	ASIC	CPU	DSP	FPGA	GPU
ARM NN		X			X
FURIOSA-AI				X	
HAILO SDK	X				
HANGUANG-AI	X				
ONNX		X			
OPENVINO		X			
PYTORCH		X			
SNPE			X		
SYNAPSE	X				
TENSORFLOW	X	X			X
TF-LITE		X			
TENSORRT					X



〈그림 6〉 단일 스트림(SS), 멀티 스트림(MS), 서버(S) 및 오프라인(O) 시나리오에 대한 모델에서 로그 스케일(log10)의 정규화된 성능

〈표 7〉 MLPerf 벤치마킹을 이용한 추론 결과(https://MLPerf.org/training-results-0-6)

Closed Division Times																												
ID	Submitter	System	Benchmark results (Single Stream in milliseconds, MultiStream in no. streams, Server in QPS, Offline in inputs/second)																				Processor	#	Accelerator	#	Software	Notes
			Image classification								Object detection								Translation									
			ImageNet				ImageNet				COCO				COCO 1200x1200				WMT E-G									
			MobileNet-v1				ResNet-50 v1.5				SSD w/ MobileNet-v1				SSD w/ ResNet-34				NMT									
			Stream	MultiS	Server	Offline	Stream	MultiS	Server	Offline	Stream	MultiS	Server	Offline	Stream	MultiS	Server	Offline	Stream	MultiS	Server	Offline						
CATEGORY: Available																												
Inf-0.5-1	Alibaba Cloud	Alibaba Cloud T4				17,473.60				5,540.10				7,431.20				128.08				342.35	Intel Xeon Platinum 8163	1	Nvidia Tesla T4	1	TensorRT 6.0, CUDA 10.1, cuDNN 7.6.3	ECC off
Inf-0.5-2	Dell EMC	Dell EMC R740			67,124.18	71,214.50			20,742.83	22,438.00			28,293.31	30,407.90			485.23	546.69			828.57	1,417.62	Intel(R) Xeon(R) Gold 6154	2	Nvidia T4	4	NGC19.09 TensorRT	ECC off
Inf-0.5-3	Dell EMC	Dell EMC R740xd with 2nd generation Intel® Xeon® Scalable Processor									1.54			3,744.24									Intel(R) Xeon(R) Gold 6248 CPU @ 2.50GHz	2			OpenVINO	
Inf-0.5-4	Dell EMC	Dell EMC R740xd with 2nd generation Intel® Xeon® Scalable Processor									1.69			4,266.46									Intel(R) Xeon(R) Platinum 8276 CPU @ 2.20GHz	2			OpenVINO	
Inf-0.5-5	dividiti	Raspberry Pi 4 (rpi4)	394.34				1,916.65																Broadcom BCM2711B0	1			TFLite v1.15.0-rc2	
Inf-0.5-6	dividiti	Raspberry Pi 4 (rpi4)	103.6				448.31																Broadcom BCM2711B0	1			ArmNN v19.08 (Neon)	
Inf-0.5-7	dividiti	Linaro HiKey960 (hikey960)	121.11				518.07																HiSilicon Kirin960	1			TFLite v1.15.0-rc2	Mobile chip in embedded form factor (development
Inf-0.5-8	dividiti	Linaro HiKey960 (hikey960)	50.77				203.99																HiSilicon Kirin960	1	Arm Mali-G71 MP8	1	ArmNN v19.08 (OpenCL)	Mobile chip in embedded form factor (development
Inf-0.5-9	dividiti	Linaro HiKey960 (hikey960)	143.07				494.9																HiSilicon Kirin960	1			ArmNN v19.08 (Neon)	Mobile chip in embedded form factor (development
Inf-0.5-10	dividiti	Huawei Mate 10 Pro (mate10pro)	74.2				354.13																HiSilicon Kirin970	1	Arm Mali-G72 MP12	1	ArmNN v19.08 (OpenCL)	
Inf-0.5-11	dividiti	Huawei Mate 10 Pro (mate10pro)	111.6				494.92																HiSilicon Kirin970	1			ArmNN v19.08 (Neon)	
Inf-0.5-12	dividiti	Firefly-RK3399 (firefly)	120.56				695.11																Rockchip RK3399	1			TFLite v1.15.0-rc2	
Inf-0.5-13	dividiti	Firefly-RK3399 (firefly)	106.49				447.9																Rockchip RK3399	1	Arm Mali-T860 MP4	1	ArmNN v19.08 (OpenCL)	
Inf-0.5-14	dividiti	Firefly-RK3399 (firefly)	80.12				391.02																Rockchip RK3399	1			ArmNN v19.08 (Neon)	
Inf-0.5-15	Google	Cloud TPU v3							16,014.29	32,716.00							320.2	656.4			1,401.40	3,082.10	Intel Skylake	2	TPU v3	4	TensorFlow, TPU 1.15.dev	
Inf-0.5-16	Google	2x Cloud TPU v3								65,431.40								1,317.40				6,127.70	Intel Skylake	2	TPU v3	8	TensorFlow, TPU 1.15.dev	
Inf-0.5-17	Google	4x Cloud TPU v3								130,833.00								2,623.40				11,981.90	Intel Skylake	2	TPU v3	16	TensorFlow, TPU 1.15.dev	
Inf-0.5-18	Google	8x Cloud TPU v3								261,587.00								5,260.60				22,509.10	Intel Skylake	2	TPU v3	32	TensorFlow, TPU 1.15.dev	
Inf-0.5-19	Google	16x Cloud TPU v3								524,978.00								7,274.20				29,061.10	Intel Skylake	2	TPU v3	64	TensorFlow, TPU 1.15.dev	
Inf-0.5-20	Google	32x Cloud TPU v3								1,038,510.00													Intel Skylake	2	TPU v3	128	TensorFlow, TPU 1.15.dev	
Inf-0.5-21	Habana Labs	HL-102-Goya PCI-board					0.24	700		14,451.00					3.85	18	296.7	326.3					Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz	1	Goya	1	Synapse-V0.2.0	
Inf-0.5-22	Intel	Intel® Xeon® Platinum 9200 processors									1.4		5,262.00	9,468.00									Intel® Xeon® Platinum 9282 Processor	2			OpenVINO	
Inf-0.5-23	Intel	Intel® Xeon® Platinum 9200 processors	0.49		27,244.81	29,203.30	1.37		4,850.62	5,965.62													Intel® Xeon® Platinum 9282 Processor	2			PyTorch Caffe2	
Inf-0.5-24	Intel	DELL ICL i3 1005G1	3.55			507.71	13.58			100.93	6.67			217.93									Intel® Core™ i3-1005G1 Processor	1	Intel® UHD Graphics	1	OpenVINO	
Inf-0.5-25	NVIDIA	Supermicro 4029GP-TRT-OTO-28 8xT4 (T4x8)		6,320.00	135,073.00	141,807.00		1,920.00	41,546.64	44,977.80		2,624.00	56,620.00	60,871.60	6.7	56	1,010.49	1,095.11	41.32		1,581.20	2,834.75	Intel(R) Xeon(R) Platinum 8280 CPU @ 2.70GHz	2	NVIDIA Tesla T4	8	TensorRT 6.0, CUDA 10.1, cuDNN 7.6.3	ECC off
Inf-0.5-26	NVIDIA	Supermicro 6049GP-TRT-OTO-29 20xT4 (T4x20)							103,532.10	113,592.00			129,056.80	143,084.00			2,581.24	2,765.69			3,776.07	7,154.88	Intel(R) Xeon(R) Platinum 8280 CPU @ 2.70GHz	2	NVIDIA Tesla T4	20	TensorRT 6.0, CUDA 10.1, cuDNN 7.6.3	ECC off
Inf-0.5-27	NVIDIA	SCAN 3XS DBP T496X2 Fluid (TitanRTXx4)		8,704.00	199,098.30	222,388.00		2,560.00	60,030.57	66,250.40		3,640.00	82,007.78	91,779.70	3.84	88	1,550.75	1,661.29	24.43		2,580.99	4,242.33	Intel(R) Xeon(R) 8268	2	NVIDIA TITAN RTX	4	TensorRT 6.0, CUDA 10.1, cuDNN 7.6.3	
Inf-0.5-28	NVIDIA	NVIDIA Jetson AGX Xavier (Xavier)	0.58	302		6,520.75	2.04	100		2,158.93	1.5	102		2,485.77	29.43	2		49.5					NVIDIA Carmel (ARMv8.2)	1	NVIDIA Xavier	1	TensorRT 6.0, Jetpack 4.3-DP, CUDA 10.0, cuDNN 7.6.3	GPU and both DLAs are used in Offline and MultiStream scenarios
Inf-0.5-29	Qualcomm	SDM855 QRD	3.02				8.95																Qualcomm Kryo485	1	Qualcomm Hexagon 690 Processor: Hexagon Vector Extensions (HVX), Hexagon Tensor Accelerator (HTA)	1	Snapdragon Neural Processing Engine (SNPE) V1.30	Hexagon Vector Extensions (HVX) being used
CATEGORY: Preview																												
Inf-0.5-31	Alibaba T-Head	Alibaba HanGuang					0.17	2,692.00	45,169.48	69,306.60													Intel Xeon Platinum 8163	2	HanGuang 800	1	HanGuangAI 1.0.3	
Inf-0.5-32	Centaur Technology	Centaur Technology Reference Design v1.0	0.33			6,042.34	1.05			1,218.48	1.54			651.89								12.28	Centaur Integrated x86 CPUs	1	Centaur Integrated AI Coprocessor	1	TF + Centaur ML Library	
Inf-0.5-33	Intel	2x NNP-I 1000							10,262.63	10,567.20													Intel(R) Xeon(R) Silver 4116 Processor	1	Intel® Nervana™ Neural Network Processor	2	ONNX	
Inf-0.5-34	Hailo	Hailo-8	7.98			645.61	12.48			545.4	13.3			373.38									Intel(R) Core(TM) i7-7820HQ	1	Hailo8	1	Hailo SDK	
Inf-0.5-30	Tencent	Tencent Cloud with 2nd generation Intel® Xeon® Scalable Processor	0.52		21,929.00	24,866.00	1.49		4,021.00	5,169.00													Intel Xeon 8255C CPU	4			PyTorch Caffe2	
CATEGORY: Research, Development, and Other																												
Inf-0.5-35	Google	Cloud TPU v3							21,032.00								390.19				1,506.80		Intel Skylake	2	TPUv3	4	TensorFlow, Internal stack	
Inf-0.5-36	FuriosaAI	furiosa-single-fpga	3.29								11.55												Intel(R) Core(TM) i3-7100	1	Renegade	1	FuriosaRuntime/TensorflowLite	
Inf-0.5-37	FuriosaAI	furiosa-multi-fpga				2,367.65								491.65									Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz	2	Renegade	7	FuriosaRuntime/TensorflowLite	

다양한 하드웨어 유형 외에도 많은 머신러닝 소프트웨어 프레임 워크가 있다. 〈표 6〉은 하드웨어 플랫폼을 벤치마킹하는데 사용되는 다양한 소프트웨어 프레임 워크를 보여준다. 머신러닝 소프트웨어는 하드웨어 성능을 향상시키는데 중요한 역할을 한다. 일부 런타임은 특정 유형의 하드웨어와 함께 작동하여 기능을 완전히 활용하도록 특별히 설계되었다. 해당 프레임 워크없이 하드웨어를 사용하면 동작은 할 수 있지만 성능은 하드웨어의 잠재력에 미치지 못할 수 있다. 이 표는 CPU의 프레임 워크 다양성이 가장 높고 TensorFlow의 아키텍처 다양성이 가장 높다는 것을 보여준다.

〈그림 6〉은 모든 작업과 시나리오에 대한 결과를 보여준다. MobileNet-v1 단일 스트림 시나리오(SS), ResNet-50 v1.5 SS 및 SSD-MobileNet-v1 SS와 같은 경우 시스템의 성능 차이는 100배이다. 이러한 모델에는 많은 응용 프로그램이 있기 때문에 대상 시스템은 저전력 임베디드 장치에서 고성능 서버에 이르는 모든 것을 포함하고 있다. GNMT 서버(S)는 시스템 간의 성능 변화가 훨씬 적은 특성을 나타낸다. 〈표 7〉에는 여러 기업들에 의한 추론 결과를 하나의 표에 정리한 것이다.

Ⅵ. 결 론

본 원고에서는 MLPerf 벤치마크를 바탕으로 하여 현재 머신러닝 및 딥러닝 분야에서 쏟아져 나오고 있는 다양한 제품과 개발물들에 대한 성능평가가 어떻게 이루어지고 있고, 현재 상황은 어떠한지를 살펴보고자 하였다. 현재 전세계적으로 학습 및 추론을 위한 새로운 시스템으 개발하기 위한 활발한 활동이 펼쳐지고 있고, 새로운 시스템의 우수성을 검증받고자 노력하고 있다. 이러한 과정의 일환으로 벤치마크가 존재하는 것이다. 이러한 벤치마크가 해외의 대기업들을 위한 광고의 장이 되지 않도록 국내의 기술력을 결집시키고 발전시킬 필요가 있음을 상기해야할 것이다.

참 고 문 헌

[1] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learn-ing for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778, 2016.
[2] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In Advances in neural information processing systems, pp. 2672-2680, 2014.
[3] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. Ssd: Single shot multibox detector. In European conference on computer vision, pp. 21-37. Springer, 2016.
[4] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pp. 1097-1105, 2012.
[5] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.
[6] Badrinarayanan, V., Kendall, A., and Cipolla, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE transactions on pattern analysis and machine intelligence, 39(12):2481-2495, 2017.

[7] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. TensorFlow: A System for Large-Scale Machine Learning. In OSDI, volume 16, pp. 265-283, 2016.

[8] Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., and Zhang, Z. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. arXiv preprint arXiv:1512.01274, 2015.

[9] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. Caffe: Convolutional Architecture for Fast Feature Embedding. In ACM International Conference on Multimedia, pp. 675-678. ACM, 2014.

[10] Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., et al. In-datacenter performance analysis of a tensor processing unit. In 2017 ACM/IEEE 44th Annual Inter-national Symposium on Computer Architecture (ISCA), pp. 1-12. IEEE, 2017.

[11] Chen, T., Moreau, T., Jiang, Z., Zheng, L., Yan, E., Shen, H., Cowan, M., Wang, L., Hu, Y., Ceze, L., et al. fTVMg: An automated end-to-end optimizing compiler for deep learning. In 13th fUSENIXg Symposium on Operating Systems Design and Implementation (fOSDIg 18), pp. 578-594, 2018.

[12] Markidis, S., Der Chien, S. W., Laure, E., Peng, I. B., and Vetter, J. S. Nvidia tensor core programmability, performance & precision. arXiv preprint arXiv:1803.04014, 2018.

[13] Intel. Bigdl: Distributed deep learning library for apache spark, 2019. URL https://github.com/ intel-analytics/BigDL.

[14] Hennessy, J. L. and Patterson, D. A. Computer architecture: a quantitative approach. Elsevier, 2011.

[15] Council, T. P. P. Transaction processing performance council. Web Site, http://www. tpc. org, 2005.

[16] Han, S., Mao, H., and Dally, W. J. Deep compres-sion: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149, 2015.

[17] Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., and Dally, W. J. Eie: efficient inference engine on compressed deep neural network. In 2016 ACM/IEEE 43rd Annual International Symposium on Computer Ar-chitecture (ISCA), pp. 243-254. IEEE, 2016.

[18] Molchanov, P., Tyree, S., Karras, T., Aila, T., and Kautz, J. Pruning convolutional neural networks for resource efficient inference. arXiv preprint arXiv:1611.06440, 2016.

[19] Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710, 2016.

[20] Adolf, R., Rama, S., Reagen, B., Wei, G.-Y., and Brooks, D. Fathom: Reference Workloads for Modern Deep Learning Methods. In Workload Characterization (IISWC), 2016 IEEE International Symposium on, pp. 1-10. IEEE, 2016.

[21] Coleman, C., Narayanan, D., Kang, D., Zhao, T., Zhang, J., Nardi, L., Bailis, P., Olukotun, K., Re,´ C., and Zaharia, M. DAWNBench: An End-to-End Deep Learning Benchmark and Competition. NIPS 머신러닝 Systems Workshop, 2017.

[22] EEMBC. Introducing the eembc 머신러닝mark benchmark.

[23] Zhu, H., Akrout, M., Zheng, B., Pelegris, A., Jayarajan, A., Phanishayee, A., Schroeder, B., and Pekhimenko, G. Benchmarking and analyzing deep neural network training. In 2018 IEEE International Symposium on Workload Characterization (IISWC), pp. 88-100. IEEE, 2018.

[24] Alibaba. Ai matrix. https://aimatrix.ai/ en-us/, 2018.

[25] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, pp. 248-255. Ieee, 2009.

[26] MLPerf. MLPerf Reference: ResNet in TensorFlow. https://github.com/MLPerf/training/tree/master/image_classification/tensorflow/official, 2019

[27] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ra-manan, D., Dollar,´ P., and Zitnick, C. L. Microsoft COCO: Common Objects in Context. In European Conference on Computer Vision, pp. 740-755. Springer, 2014.

[28] WMT. First conference on machine translation, 2016. URL http://www.statmt.org/wmt16/.

[29] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144, 2016.

[30] WMT. Second conference on machine translation, 2017.
[31] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In Advances in neural information processing systems, pp. 5998-6008, 2017.
[32] GroupLens. Movielens 20m dataset, Oct 2016. URL https://grouplens.org/datasets/ movielens/20m/.
[33] He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.-S. Neural collaborative filtering. In Proceedings of the 26th international conference on world wide web, pp. 173-182. International World Wide Web Conferences Steering Committee, 2017b.
[34] MLPerf. MLPerf Reference: MiniGo. https: //github.com/MLPerf/training/tree/ master/reinforcement, 2019a.
[35] Mattson, P., Cheng, C., Coleman, C., Diamos, G., Micikevicius, P., Patterson, D., Tang, H., Wei, G.-Y., Bailis, P., Bittorf, V., Brooks, D., Chen, D., Dutta, D., Gupta, U., Hazelwood, K., Hock, A., Huang, X., Jia, B., Kang, D., Kanter, D., Kumar, N., Liao, J., Narayanan, D., Oguntebi, T., Pekhimenko, G., Pentecost, L., Reddi, V. J., Robie, T., John, T. S., Wu, C.-J., Xu, L., Young, C., and Zaharia, M. MLPerf training benchmark, 2019.
[36] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861, 2017.
[37] Bai, J., Lu, F., Zhang, K., et al. Onnx: Open neural network exchange. https://github.com/onnx/ onnx, 2019.
[38] "MLPerf Training Benchmark", https://arxiv.org/abs/1910.01500
[39] "MLPerf Inference Benchmark", https://arxiv.org/abs/1911.02549

필 자 소 개

서 영 호

– 1999년 2월 : 광운대학교 전자재료공학과 졸업(공학사)
– 2001년 2월 : 광운대학교 일반대학원 졸업(공학석사)
– 2004년 8월 : 광운대학교 일반대학원 졸업(공학박사)
– 2005년 9월 ~ 2008년 2월 : 한성대학교 조교수
– 2008년 3월 ~ 현재 : 광운대학교 전자재료공학과 교수
– ORCID : http://orcid.org/0000–0003–1046–395X
– 주관심분야 : 실감미디어, 디지털 홀로그램, 딥러닝, 뉴로모픽 시스템, SoC 설계

박 성 호

– 2000년 : 광운대학교 전자재료공학 전공, 전자공학 부전공 학사
– 2004년 : 광운대학교 전자재료공학 전공, 석사
– 2004년 ~ 2013년 : LG전자 선임연구원
– 2013년 ~ 2015년 : 미국 Uniquify Sr. ASIC Manager
– 2015년 ~ 2017년 : 미국 Northwest Logic, Sr. Designer
– 2017년 ~ 2018년 : 미국 Qualcomm, Sr. Staff Engineer
– 2018년 ~ 2019년 : 퓨리오사 AI, 수석연구원
– 2019년 ~ 현재 : 미국 CoSignOn, 대표이사
– 주관심분야 : 시스템 반도체/IP, 메모리 시스템, 뉴럴 프로세서

필 자 소 개

박 장 호

– 2008년 : 청주대학교 전자공학 전공, 통신공학 부전공 학사
– 2010년 : 광운대학교 전자재료공학 전공, 석사
– 2010년 ~ 2012년 : 어보브반도체 연구원
– 2012년 ~ 2015년 : Uniquify 한국지사 Jr. DDR IP Engineer
– 2015년 ~ 2018년 : 텔레칩스 Sr. SoC Design Engineer
– 2018년 ~ 2019년 : 퓨리오사 AI, Sr. Engineer
– 2019년 ~ 현재 : 미국 CoSignOn, 수석연구원
– 주관심분야 : 시스템 반도체/IP, 메모리 서브 시스템