서비스형 엣지 머신러닝 기술 동향

Trend of Edge Machine Learning as-a-Service

나중찬 (J.C. Na, njc@etri.re.kr) 서울SW-SoC융합R&BD센터 책임연구원/센터장
전승협 (S.H. Jeon, shjeon00@etri.re.kr) 지역ICT융합연구실 책임연구원

ABSTRACT

The Internet of Things (IoT) is growing exponentially, with the number of IoT devices multiplying annually. Accordingly, the paradigm is changing from cloud computing to edge computing and even tiny edge computing because of the low latency and cost reduction. Machine learning is also shifting its role from the cloud to edge or tiny edge according to the paradigm shift. However, the fragmented and resource-constrained features of IoT devices have limited the development of artificial intelligence applications. Edge MLaaS (Machine Learning as-a-Service) has been studied to easily and quickly adopt machine learning to products and overcome the device limitations. This paper briefly summarizes what Edge MLaaS is and what element of research it requires.

KEYWORDS AutoML, Edge ML as-a-Service, Edge ML model compilation, MLOps, hardware-aware NAS, hardware-aware pruning, hardware-aware quantization

Ⅰ. 서론

머신러닝(ML: Machine Learning) 모델의 학습 및 추론 개발 프로세스는 대규모의 컴퓨팅 자원과 편리한 개발환경을 제공하는 클라우드 컴퓨팅 아키텍처의 도움으로 더욱 대중화되어 왔다. 하지만 클라우드 서비스의 지리적 요인으로 인한 지연, 개인정보 보호 또는 최종 노드의 처리 기능 제한으로 머신러닝 업무는 천천히 엣지로 이동하는 패러다임의 변화를 가져오고 있다. 사물 인터넷 영역이 계속 확장됨에 따라 엣지 컴퓨팅 또한 중요성이 커지고 있다.

이러한 패러다임 변화에 맞춰서 클라우드에서 머신러닝 개발을 효율화하는 도구를 서비스로 제공했던 서비스형 머신러닝은 엣지 디바이스 또는 초소형 엣지 디바이스의 잠재력을 극대화하기 위한 새로운 기술과 도구들을 반영함으로써 서비스형 엣지 머신러닝으로의 확장이 진행 중이다[1].

본고에서는 먼저 Ⅱ장에서 서비스형 엣지 머신러닝 개요를 살펴보고, Ⅲ과 Ⅳ장에서는 서비스형 엣지 머신러닝 플랫폼과 이를 구성하는 주요 기술 동향에 관해 살펴보며, 마지막으로 Ⅴ장에서 결론을 맺도록 한다.

* DOI: https://doi.org/10.22648/ETRI.2022.J.370505
* 이 논문은 2022년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임[No. 2022-0-00454, 스마트 엣지 디바이스 SW 개발 플랫폼 개발].

II. 서비스형 엣지 머신러닝 개요

1. 엣지 컴퓨팅 패러다임 전환

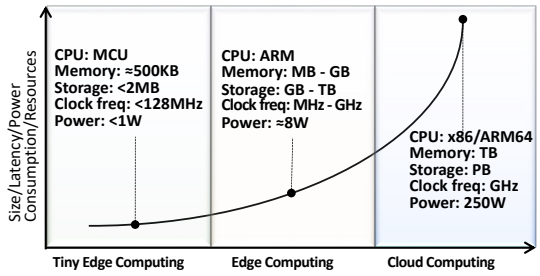
최근 컴퓨팅 요구 및 활용 사례에 대한 컴퓨팅 인프라 모델은 그림 1과 같이 구체화되고 있다.

클라우드 컴퓨팅(Cloud Computing)은 기가 바이트의 메모리와 초고속 처리가 필요한 수백만 개의 매개변수로 구성된 신경망 모델을 처리한다. 이러한 대규모 신경망 모델은 무제한 컴퓨팅 자원과 메모리에 액세스할 수 있으므로 주로 정확성과 속도에 중점을 두고 개발되고 있다.

그리고 엣지 컴퓨팅(Edge Computing)은 클라우드의 컴퓨팅 성능에 의존하지 않고 짧은 시간에 실시간 처리가 필요한 스마트폰, 분산된 서버, 웨어러블 디바이스, 자율주행 자동차와 같은 많은 분야에서 사용되며, 최근에는 일부 응용에서 원하지 않는 데이터 지연, 데이터 손실 및 개인정보보호 문제를 고려하기 위해 제한된 자원(처리 속도, 메모리, 소비전력 등)을 갖는 디바이스에서 머신러닝 알고리즘을 처리하는 초소형 엣지 컴퓨팅(Tiny Edge Computing) 연구가 촉발되고 있다[1-3].

클라우드 머신러닝(Cloud ML)은 인공지능 학습 및 추론에 클라우드 인프라를 활용하는 것으로, 엣지 머신러닝(Edge ML)은 이러한 업무 중 일부를 클라우드 인프라에서 개별 구축 서버, 게이트웨이 및 최종 디바이스에 배치하는 접근방식을 보여주고 있다. 클라우드 인프라에 남겨둘 업무와 엣지 컴퓨팅에 배치될 업무는 비용, 지연시간, 가용성 등의 요구사항에 의해 결정된다. 예를 들어, 클라우드에 상주하는 중앙 노드는 작업 할당, 자원 스케줄링 및 최적화를 담당하게 하고, 엣지 노드는 중앙 노드와 정보를 주고받을 필요 없이 즉각적으로 조처하도록 구성할 수 있다.

일반적으로 초소형 엣지 머신러닝(Tiny Edge ML)



출처 Reproduced with permission from [3].

그림 1 **컴퓨팅 패러다임 비교**

은 자원 부족으로 추론만 허용하고, 학습은 자원이 풍부한 서버 또는 클라우드 컴퓨터에 국한되는 것으로 가정하고 있다. 그러나 초소형 엣지 디바이스를 위한 효과적인 머신러닝 솔루션 생성을 위해서는 실제 데이터의 변동성을 고려해야 하며, 자원이 매우 제한된 디바이스가 최대한의 잠재력을 발휘하려면 인간 개입을 최소화할 수 있는 수준의 솔루션이 필요할 것으로 예상된다[3].

최근에는 머신러닝의 등장과 함께 머신러닝을 엣지에 적용하기 위한 노력이 활발히 진행되고 있으며, TinyML 커뮤니티는 센서 또는 액추에이터와 같은 MCU(Micro Controller Unit) 기반 초소형 엣지 디바이스 내에서 머신러닝의 통합을 촉진하고 있다[4,5].

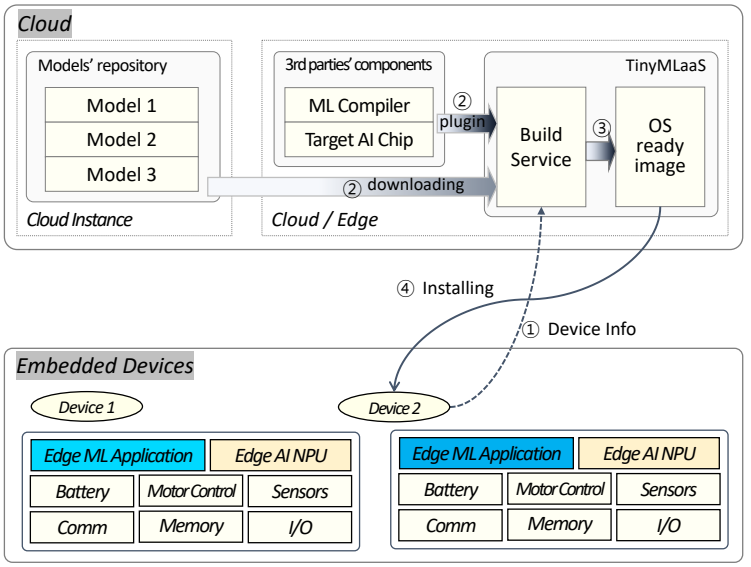
2. 서비스형 엣지 머신러닝

머신러닝 모델을 개발하기 위해서는 데이터 가공 및 학습을 위한 대용량의 컴퓨팅 자원, 복잡한 학습 모델 등이 필요하여 온프레미스 환경에서 구축하기에 어려움이 있다. 또한, 데이터 과학자, 머신러닝 모델 개발자로 구성된 팀을 설계하는 데 막대한 투자를 해야 하나, 대부분 조직은 그런 능력을 갖추고 있지 못하고, 소수의 조직만이 자체 머신러닝 기반 응용솔루션 개발환경을 구축·운영할 수 있다. 이러

한 문제점을 해결하기 위해 등장한 서비스가 서비스형 머신러닝(MLaaS: Machine Learning as a Service)으로 클라우드 컴퓨팅 환경에서 간편하고 효과적으로 머신러닝 모델을 개발할 수 있도록 한다[6].

최근 초소형 엣지 머신러닝의 이점이 더욱 분명해짐에 따라 머신러닝 모델을 매우 제한된 디바이스에 배포하고 원하는 상호 운용성을 보장하는 클라우드 또는 엣지 기반 서비스인 서비스형 초소형 엣지 머신러닝(TinyMLaaS: TinyML as-a-Service)이 제안되었다. 그림 2는 서비스형 엣지 머신러닝 구성요소 간의 상호작용을 보여주고 있다[7]. 즉, 초소형 임베디드 디바이스에 탑재될 머신러닝 모델을 개발 및 배포하기 위해 1) TinyMLaaS는 CPU 유형, RAM 및 ROM 크기, 사용 가능한 주변 디바이스, 기본 소프트웨어, 처리할 올바른 추론 모델과 같은 장치 자체에 대한 일부 정보를 수집하며, 2) TinyMLaaS 백엔드는 가장 적합한 머신러닝 컴파일러를 선택하고, 3) 위의 매개변수를 기반으로 컴파일된 머신러닝 추론 모듈을 생성하며, 4) 생성된 ML 추론 모듈이 다운로드되어 지정된 디바이스에 설치된다.

엣지 머신러닝 모델을 단순 개발하는 것과 개발을 실용적으로 하는 것과의 차이는 개발팀이 제품 출시 주기에 대해 예상되는 시간과 비용 목표 내에서 엣지 머신러닝 모델을 구현할 수 있도록 하거나 데이터 과학 및 코딩 배경 지식이 없는 경우에 자체 인프라 구축과 모델을 자체 개발하는 것보다 광범위한 애플리케이션을 위한 엣지 머신러닝 모델을 빠르고 더 쉽게 통합하고 모델의 배포 및 유지 관리를 가속할 수 있는 서비스형 엣지 머신러닝 플랫폼의 가용성에 달려 있다[8]. 따라서 대부분의 엣지 디바이스 개발업체 또는 개발자는 실용적인 측면에서 데이터 엔지니어링, 모델 생성 및 최적화, 펌웨어 코드 생성, 모니터링 및 재학습 등의 프로세스가 포함된 모델 개발·배포 서비스를 제공하는 서비스형 엣



출처 Reproduced with permission from [7].

그림 2 TinyML as-a-Service 구성도

지 머신러닝 플랫폼에 가입하거나 제3자로부터 외주를 선호할 가능성이 크다.

또한 서비스형 초소형 엣지 머신러닝 플랫폼은 로우 코드 또는 제로 코드 방법을 통해 엣지 머신러닝 모델의 지속적인 통합, 배포, 비정상적인 추론 결과 또는 이상상태 모니터링 서비스로 진화되고 있다.

Ⅲ. 서비스형 엣지 머신러닝 개발 동향

포브스지에 따르면 서비스형 머신러닝 시장 규모는 2020년 10억 달러에서 2026년 84.8억 달러에 이를 예정이며, Vertiv Group Corp.은 2019년부터 2025년까지 엣지 사이트 수가 226% 증가할 것으로 전망하고 있다[9,10]. 이와 같이, 새로운 엣지 디바이스들의 증가는 머신 러닝을 활용할 기회의 확대를 의미하며, 동시에 엣지 머신러닝 기술은 그 기회를 잡는데 중추적 역할을 할 것으로 예상된다. 표 1에서 알 수 있듯이 이러한 사업성을 일찍이 파악한 글로벌 클라우드 서비스 공급업체와 엣지 머신러닝 서비스 공급업체들은 앞다퉈 서비스형 머신러닝 사업에 뛰어든 상태다.

Amazon, Google, Microsoft 등의 주요 클라우드 서비스 제공업체들은 엣지 머신러닝 하드웨어 가속기 칩셋(CPU, GPU, FPGA, ASIC 등) 기업과 긴밀한 파트너십을 형성하여 다양한 엣지 머신러닝 모델을 배포하기 위한 생태계를 형성하고 있다. 또한 점점 더 많은 공급업체가 오픈소스 소프트웨어를 포함하거나 오픈소스로 솔루션을 만드는 방향으로 기울고 있으며, 서비스형 머신러닝 서비스 솔루션을 활용하는 기존 개발자들을 엣지 머신러닝 시장으로 끌어들이고 해당 솔루션을 확장함에 따라 엣지와 클라우드 포트폴리오를 지속해서 강화하고 있다[11-13].

또한, 종단 간 엣지 머신러닝 서비스 공급업체들인 OctoML, Edge Impulse, SensiML 등이 최적의 엣

표 1 서비스형 엣지 머신러닝 솔루션 개발 현황

기업	서비스형 엣지 머신러닝 솔루션 특징
Amazon	• 맞춤형 머신러닝 파이프라인을 자동화하여 서비스를 제공하는 'SageMaker'를 출시 • 엣지 머신러닝 모델 성능개선, 배포, 수명 주기 동안 상태를 모니터링하는 'SageMaker Edge Manager' 기능을 제공
Google	• 쉽게 통합, 모델 배포 및 유지 관리할 수 있는 'Vertex AI'를 발표, 또한 자동화된 프로세스를 사용하여 엣지에서 모델을 배포하고 모니터링이 가능한 'Vertex ML Edge Manager'를 소개 • 엣지 머신러닝 성능을 가속하기 위한 엣지 머신러닝 가속 칩셋인 Edge TPU를 출시
Microsoft	• 클라우드 서비스의 확장으로 설계된 엣지 인공지능 모델을 관리하는 'Azure Percept'를 발표 • 또한, 클라우드에서 Edge ML 모델을 학습시키고, 모델을 컨테이너화하여 배포하고, 클라우드에서 모든 컨테이너를 모니터링할 수 있는 'Azure IoT Edge'를 개발
OctoML	• 개발자가 모델을 입력하며, 클라우드 또는 엣지 디바이스를 대상으로 해당 모델의 성능을 자동으로 최적화하는 'Octomizer' SaaS 플랫폼을 출시 • 오픈소스를 기반으로 구축된 기계 학습 가속화 플랫폼을 제공
Edge Impulse	• 서비스형 Tiny ML을 제공하여 오픈소스 디바이스 SDK를 사용하여 딥러닝 모델을 제공 • 센서 데이터 수집, 원시 데이터에서 인공신경망까지의 실시간 신호 처리, 초저전력 MCU를 사용하여 모든 대상 장치에 대한 테스트 및 배포
SensiML	• 데이터 수집, 라벨링, 알고리즘 및 펌웨어 자동 생성, 테스트를 포괄하는 개발환경을 제공 • Arm Cortex-M 클래스 이상 마이크로컨트롤러 코어, Intel x86 명령어 세트 프로세서, FPGA 최적화를 통해 이기종 코어 QuickLogic SoC 및 QuickAI 플랫폼을 지원

출처 Reprinted from [11-16].

지 머신러닝을 개발 및 배포하기 위한 서비스형 엣지 머신러닝 솔루션 생태계에서의 주요 플레이어로 등장하고 있다[14-16].

Ⅳ. 서비스형 엣지 머신러닝 주요 기술

서비스형 엣지 머신러닝 아키텍처는 신경 아키텍처 검색, 모델 압축, 모델 컴파일, 지속적 통합 및 배포 등 다양한 기술을 보유해야 한다.

1. 엣지 머신러닝 프레임워크

대부분의 머신러닝 모델 개발자는 의도한 작업 및 대상 HW 자산(GPU, NPU, MCU 등)에 적합한 머신러닝 모델을 선택하고, TensorFlow, ONNX 및 PyTorch와 같은 주요 머신러닝 프레임워크를 기반으로 하는 모델에 대한 추론 엔진을 개발한다. 이러한 머신러닝 프레임워크는 사전에 만들어진 최적화된 머신러닝 알고리즘과 라이브러리를 제공함으로써 개발자들이 이를 이용하여 쉽고 빠르게 문제를 해결할 수 있도록 한다.

그러나 Tensorflow, Pytorch 및 SKLearn과 같은 머신러닝 프레임워크에서 생성된 모델은 메모리 요구사항 등으로 인해 초소형 엣지 머신러닝 모델 개발에 적합하지 않다. 대신에 엣지 또는 초소형 엣지 머신러닝 모델 개발에 적합한 프레임워크는 오픈소스 형태로 제공되고 있으며, 엣지 디바이스에서 실행할 수 있는 머신러닝 모델 개발을 지원하는 Google의 TensorFlow Lite[17,18], Arduino, Raspberry Pi 및 Microbit와 같은 ARM Cortex-A 및 Cortex-M 아키텍처를 기반으로 하는 모델 개발을 지원하는 임베디드 프레임워크인 Microsoft의 Embedded Learning Library[19], 초소형 임베디드 디바이스의 추론 기능을 개발하기 위한 리눅스 소프트웨어인 ARM-NN[20] 등이 있다.

2. 머신러닝 모델 개발 자동화

기존의 머신러닝 모델 개발방식은 전문가가 많은 시간을 투자하여 개발하는 형태로 이루어져 왔다. 그러나 머신러닝 모델 개발을 비전문가가 수행하기에는 어려운 측면이 있다. 또한, 응용에 맞는 다양한 머신러닝 모델들을 빠르게 개발하기 어려운 문제가 있다. 이러한 문제를 해결하기 위해 다양한 머신러닝 모델 개발을 쉽고 빠르게 할 수 있는 머신러닝 모델 개발 자동화(AutoML: Automated Machine Learning) 기술에 관한 관심이 매우 높고, 지금까지 많은 AutoML 관련 솔루션이 출시되어 활용되고 있다.

AutoML은 데이터 준비, 특징 엔지니어링, 모델 생성, 모델 평가 등을 포함한 머신러닝 개발 프로세스의 주요 단계를 자동화한다. 이러한 AutoML 도구는 2013년 AutoWeka를 시작으로 다양한 AutoML 도구들이 지속적으로 개발되어 공개되었다. 특히 2017년 이후부터 AutoML 관련 오픈소스 도구와 머신러닝 개발 플랫폼 관련 스타트업 기업 및 클라우드 서비스 제공업체(Google, Microsoft, Amazon, Huawei 등)에 의해 주로 개발된 상업용 도구 출시가 증가하고 있다.

드래그 앤 드롭 기능을 통해 모델 개발 및 배포를 모니터링하고 제어할 수 있다는 것은 모델 사용자를 매우 편리하게 한다. 머신러닝 모델 사용자 대부분이 데이터 과학자 또는 머신러닝 개발자가 아닌 점을 고려하여 모델 개발 프로세스를 최대한 쉽게 만들 필요가 있다. 현재 AutoML 솔루션 중에서는 개발자에게 편의성을 중시한 AutoKeras[21], 초소형 엣지 머신러닝 개발자를 위한 Edge Impulse[22], 그리고 Microsoft에서 개발한 오픈소스 도구인

NNI(Neural Network Intelligence)[23] 등의 사용자 수가 많이 증가하고 있다.

3. 하드웨어 인지형 모델 압축

컴퓨팅 자원이 제한된 디바이스에서 적합한 엣지 머신러닝 모델은 원하는 정확도를 유지하면서 모델 크기와 연산량 측면에서 모델을 더욱 압축해야 한다. 특히 머신러닝 모델의 연산 비용, 메모리 공간 및 에너지 소비 등을 줄이는 모델 압축 기술 중 일부는 배포된 자원 제약을 갖는 하드웨어 플랫폼의 지원과 해당 하드웨어 특징을 고려한 모델 최적화인 경우에만 최대 잠재력에 도달할 수 있다[24].

가. 하드웨어 인지형 모델 양자화

양자화(Quantization)는 모델 파라미터 크기나 연산 수를 최소화하기 위해 가중치 정밀도를 줄이는 데 그 목적이 있으며, 이진화(Binarization)는 신경망의 모든 가중치와 중간 계산을 -1 또는 +1의 이진값으로 변환함으로써 기존 부동 소수점 연산과 비교하여 연산량과 메모리양을 크게 압축시키는 데 사용되는 기술이다.

가중치 정밀도를 변환하는 양자화 기술은 초소형 엣지 노드에서 훨씬 더 중요한 역할을 하며, 그동안 정밀도 값을 훨씬 더 작은 고정 소수점 정수, 즉 int4 및 int2로 변환하기 위해 상당한 노력을 기울여 왔다. 단일 또는 2비트 양자화를 통해 수행되며, 이를 통해 딥러닝 모델은 활성화 및 가중치에 대해서 이진 및 삼항 값을 사용하여서 이진 및 삼항 신경망을 형성할 수 있다. 이렇게 하면 실행 시간, 모델 크기 및 메모리 접근이 줄어들어 전력 효율성이 크게 향상된다.

그러나 양자화는 머신러닝 모델을 압축하기 위해 널리 사용되는 기술이나, 기존의 양자화 방법은 모든(또는 대부분의) 계층에 대해 같은 비트 폭을 사용하는 균일 정밀도를 적용하므로 초저정밀도 영역에서 종종 상당한 정확도 저하를 겪고 새로운 하드웨어 가속기가 혼합 정밀도 연산을 지원하기 시작한다는 사실을 무시하고 있다.

최근에는 엣지 디바이스의 하드웨어에서 지원하는 연산 정밀도로 정확도 손실을 최소화하는 방향으로 혼합 정밀도 기반 연산 방식과 실행될 하드웨어 대상으로 한 성능 측정결과를 양자화에 대한 피드백으로 사용하는 방식을 고려한 하드웨어 인지형 양자화(Hardware-Aware Quantization) 연구가 활발히 진행되고 있다[25,26]. 이러한 혼합 정밀도 지원 양자화 기법은 CPU, GPU, NPU와 같은 다양한 연산 유닛에 맞춰서 정밀도를 최적으로 조절하면 정확도 손실을 발생시키지 않으면서도 저정밀도 연산을 통해 메모리 사용 감소, 에너지 소모 감소, 추론 지연 시간 감소의 효과를 얻을 수 있다. 이는 서로 다른 자원 제약 조건(즉, 대기 시간, 에너지 및 모델 크기)에서 서로 다른 하드웨어 아키텍처에 따라 크게 다르다는 것을 보여주었다.

나. 하드웨어 인지형 가지치기

모델 가지치기 기술은 중요하지 않고 중복되는 뉴런 제거를 통해 압축 모델을 생성하는 것이며, 가중치의 중복성을 판단하는 다양한 알고리즘에 따라 필터 또는 채널을 제거하는 구조적 가지치기(Structured Pruning) 기법과 가지치기할 개별 가중치를 선택하는 비구조적 가지치기(Unstructured Pruning) 기법으로 분류된다.

기존의 모델 가지치기 방법은 자원이 제한된 엣지 디바이스에서 머신러닝 모델 실행을 위해 모델 자체에 집중하여 압축된 모델을 생성하거나 컴파일러 최적화를 사용하여 효율적인 코드를 생성하는 데 집중하였다[27-29]. 그러나 컴파일러 최적화 고

려 없이 모델 자체에만 집중한 가지치기 방식은 어떤 모델이 정확도를 보장하면서 실행 속도가 보장되는지는 알 수 없으며, 생성된 경량 머신러닝 모델은 정확도를 보장하는 반면에 실행 속도는 보장되지 않는다는 결과를 발표하였다[30].

최근에는 정확도 요구사항을 충족하면서 머신러닝 모델의 실행 속도를 크게 향상시킬 수 있는 하드웨어 인지형 가지치기(Hardware-aware Pruning)에 관한 새로운 연구가 진행되고 있다[31,32].

다. 하드웨어 인지형 NAS

AutoML의 하위 집합인 NAS(Neural Architecture Search) 기술은 주어진 데이터 세트에 대한 신경 아키텍처 설계 프로세스를 자동화하는 데 사용된다. 그러나 기존 NAS 방식은 실행될 하드웨어 환경을 고려하지 않고 단순히 최고의 성능만을 지표로 하여 ML 모델을 설계하였으며, 이러한 설계된 모델은 크기가 매우 크고, 많은 컴퓨팅 자원을 필요로 할 수 있으므로 엣지 컴퓨팅 환경에서 활용되기에는 어려움이 있을 수 있다. 또한, 다양한 시나리오와 상황에서 사용할 수 있는 하드웨어 장치가 너무 많으므로 모든 경우에 대해 하나의 시스템을 선택하기가 매우 어렵다.

이를 해결하기 위해 네트워크 검색 공간, 대상 하드웨어 특성을 입력으로 사용하여 최적의 성능을 지닌 하드웨어 적응형 신경망 설계를 자동화하는 하드웨어 인지형 NAS(HW-NAS: Hardware-aware NAS) 연구가 활발하게 이루어지고 있다[33]. HW-NAS 관련 연구는 주로 DNN(Deep Neural Network) 관련 연구가 80% 이상을 차지하고 있으며, 지연시간, 전력 소비, 메모리 사용량과 같은 성능 지표와 정확도 사이에 좋은 균형을 이루는 모델 설계를 위해 심층 신경망의 설계 프로세스를 자동화하는 것을 목표로 연구가 수행되었다.

4. 엣지 머신러닝 모델 컴파일

엣지 머신러닝 모델을 배포하기 전에 엣지 디바이스 하드웨어가 이해할 수 있는 언어로 변환해야 하며, 이를 위해 주로 컴파일러(Compiler) 방식과 인터프리터(Interpreter) 방식이 사용되고 있다.

컴파일러 방식은 편의성 면에서 선호되는 방식이나, 공급업체별로 파편화된 머신러닝 컴파일러는 상호 운용성과 이식성 부족으로 독점적인 특징을 갖는다. NEST-C[34], TensorFlow XLA[35], ONNC[36], nGraph[37], Glow[38], TVM[39], PlaidML[40] 등과 같은 머신러닝 컴파일러는 더 나은 성능을 위해 코드를 최적화하고 다양한 하드웨어 명령어 세트에 대응시킨다. 또한, Micro TVM은 마이크로컨트롤러를 대상으로 하는 베어메탈 디바이스에서 TVM을 실행하는 것으로 현재 개발하고 있다(표 2 참고).

이에 반해 머신러닝 인터프리터 방식은 모델 구조가 머신러닝 프레임워크에 연결되어 있지 않기 때문에 쉽게 이식할 수 있다. 또한 모델을 개별적으로 최적화하고 요구사항과 디바이스에 맞게 편리하게 조정할 수 있지만, 성능과 메모리 사용량에 약간의 오버헤드가 있을 수 있다. 초소형 엣지 머신러닝 인터프리터 오픈소스 주도권은 Google의 TFLM(TensorFlow Lite Micro)[41], Microsoft의 ELL(Embedded Learning Library)[42] 등이 주도하고 있다.

5. 엣지 MLOps

엣지 디바이스에서 머신러닝 모델을 지속해서 모니터링, 유지 관리 및 개선할 수 있게 해주는 엣지 MLOps(Machine Learning Operation) 도구는 머신러닝을 확장하는 데 도움이 되는 중요한 기술로 부상하고 있다.

표 2 머신러닝 모델 컴파일러 개발 현황

기술(기관명)	머신러닝 모델 컴파일러 기술/제품
NEST-C (ETRI)	• C Code, Relay-IR, CPU code, eVTA code 등 다양한 백엔드 코드 생성 • Layer Fusion, Operator Scheduling, Latency Hiding, Auto-tuning, Post-Training Quantization 등 고성능 하드웨어 독립적 및 종속적 최적화 지원 • ZCU102 & Ultra96 configuration 하드웨어 지원
XLA (Google)	• High(HLO)/Low(HLO) Level IR 최적화를 수행, 양자화(int8/int16) 지원 • TensorFlow, ONNX 포맷 지원 • CPU/GPU/TPU/Customized 하드웨어 지원
ONNC (OpenSource)	• ONNX IR을 통해 그래프 최적화를 수행하고, ONNC IR을 통해 HW에 대한 최적화 • CPU인 경우에는 ONNC IR 동작 후에 LLVM IR을 이용한 LLVM 백엔드 동작이 필요 • Caffe2, Chainer, Cognitive Toolkit MXNet, PyTorch, PaddlePaddle, ONNX 포맷 지원 • NVDLA 기반 하드웨어 지원
nGraph (Intel)	• High(nGraph IR)/Low(None) Level IR 최적화를 수행, 양자화(int8) 지원 • TensorFlow, PaddlePaddle, ONNX 포맷 지원 • CPU/Intel GPU/NNP/GPU/Customized(Use OpenCL support in PlaidML) 지원
Glow (Meta)	• High(own)/Low(own) Level IR 최적화를 수행, 양자화(int8) 지원 • PyTorch, Caffe2, TensorFlow Lite, ONNX 포맷 지원 • CPU/GPU/Customized 하드웨어 지원
TVM (Apache)	• High(Relay)/Low(Halide) Level IR 최적화를 수행, 양자화(int8/fp16) 지원 • TensorFlow, TensorFlow Lite, Keras, PyTorch, Caffe2, MXNet, CoreML, DarkNet, ONNX 포맷 지원 • CPU/GPU/ARM/FPGA/Customized(Use VTA) 지원
PlaidML (OpenSource)	• MLIR 컴파일러 구조 채택하였으며, nGraph 컴파일러 내부 컴포넌트로 활용 가능 • Keras. ONNX 포맷 지원 • AMD, Intel, NVIDIA 지원

출처 Reprinted from [34-40].

엣지 MLOps의 초점은 무중단 업데이트, 지속적인 전달 및 피드백 루프를 통해 엣지 머신러닝의 전체 배포 주기를 조정하고 모니터링할 수 있는 프로세스를 만드는 것이다. 최종 사용자가 머신러닝 모델에서 비정상적인 추론 결과 또는 이상을 감지하면 최종 사용자는 피드백을 기반으로 머신러닝 모델을 업데이트하고 모델을 다시 배포한다. 이를 달성하는 한 가지 방법은 엣지 머신러닝용 CI/CD (Continuous Integration/Continuous Deployment) 파이프라인을 유지하는 것이다. 이러한 CI/CD 외에도 데이터가 지속해서 변경됨에 따라 새로 등장하는 데이터에서 모델을 자동으로 재학습하는 Continuous Training 단계가 추가되고 있다.

그러나 엣지 환경은 각기 다른 소프트웨어 지원 및 하드웨어 기능을 가진 다양한 공급업체의 다양한 디바이스로 구성되어 있다. 또한, 보다 에너지 효율적인 방식으로 머신러닝 애플리케이션을 지원하는 맞춤형 하드웨어 가속기[43]에 대한 추세를 보이고 있다. 따라서 엣지 디바이스에 응용 프로그램을 배포하려면 먼저 기본 플랫폼에서 필요한 모든 작업을 지원하는지 확인해야 하며, 그런 다음 모델을 양자화하거나 추가 공급업체별 최적화를 수행하는 경우를 고려해야 한다.

V. 결론

머신러닝은 네트워크 코어에서 엣지로 꾸준히 이동하고 있다. 이와 동시에 초소형 엣지 디바이스의 수가 해마다 급격히 증가하면서 사물인터넷이 기하급수적으로 성장하고 있다. 이러한 엣지 디바이스

나 초소형 디바이스를 지능화하기 위해 머신러닝이 사용되고 있으나 디바이스의 자원이 제한적이고 파편화되어 있어서 기존의 머신러닝을 사용하기 힘들게 하며, 머신러닝 개발을 어렵게 하고 있다. 이러한 문제점을 해결하기 위해서 기존에 클라우드에서 머신러닝 개발을 효율화하는 도구를 서비스로 제공했던 서비스형 머신러닝을 쉽고 빠르게 엣지나 초소형 엣지 디바이스로 확장할 수 있도록 활발하게 연구되고 있다. 또한, 확장된 서비스형 엣지 머신러닝은 머신러닝을 지원하는 디바이스의 개발 시간을 단축시키고, 그 완성도를 높이는 데 도움을 줄 것으로 예상된다.

용어해설

Cloud ML 엔터프라이즈 내부 또는 클라우드의 데이터 센터에서 머신러닝의 응용프로그램

Edge ML 여러 지역에 있는 소규모 데이터 센터의 엔터프라이즈급 컴퓨터/디바이스에서 머신러닝의 응용 프로그램

Tiny ML 엣지 가장자리에 있는 디바이스의 다양한 마이크로컨트롤러에서 초저전력 머신러닝의 응용 프로그램(스마트 카메라, 원격 모니터링 장치, 웨어러블 장치, 오디오 캡처 하드웨어 및 다양한 센서 등이 해당)

서비스형 머신러닝 다양한 머신러닝 기반 기능을 포함하여 타사 공급업체가 고객사에 서비스 형태로 제공하는 머신러닝 소프트웨어 배포 및 모니터링

약어 정리

ASIC	Application-Specific Integrated Circuit
AutoML	Automated Machine Learning
CI/CD	Continuous Integration/Continuous Delivery
CPU	Central Processing Unit
DNN	Deep Neural Network
FPGA	Field Programmable Gate Array
GPU	Graphics Processing Unit
MCU	Micro Controller unit
MLaaS	Machine Learning as-a-Service
MLOps	Machine Learning Operations
NAS	Neural Architecture Search
NNI	Neural Network Intelligence
NPU	Neural Processing Unit
ONNX	Open Neural Network Exchange
TinyML	Tiny Edge Machine Learning
TPU	Tensor Processing Unit
TVM	Tensor Virtual Machine

참고문헌

[1] P.P. Ray, "A review on TinyML: State-of-the-art and prospects," J. King Saud Univ.: Comput. Inf. Sci., vol. 34, no. 4, 2021.

[2] S. Soro, "TinyML for ubiquitous edge AI," arXiv preprint, CoRR, 2021, arXiv: 2102.01255.

[3] V. Rajapakse, I. Karunanayake, and N. Ahmed, "Intelligence at the extreme edge: A survey on reformable TinyML," arXiv preprint, CoRR, 2022, arXiv: 2204.00827.

[4] R. Sanchez Iborra and A.F. Skarmeta, "TinyML-Enabled frugal smart objects challenges and opportunities," IEEE Circuits Syst. Mag., vol. 20, no. 3, 2020, pp. 4-8.

[5] https://www.tinyml.org/

[6] 신성필, "서비스형 기계학습(MLaaS, Machine Learning as a Service) 시장 동향 및 기능 요구사항 표준," TTA저널, 제198호 제11/12월호, 2021.

[7] H. Doyu et al., "A tinymlaas ecosystem for machine learning in iot: Overview and research challenges," in Proc. Int. Symp. VLSI Des., Autom. Test (VLSI-DAT), (Hsinchu, Taiwan), Apr. 2021.

[8] C. Rogers, "The opportunity for AI at the edge and beyond," SensiML, Apr. 2020.

[9] 김대우, "2021년 머신러닝과 인공지능(AI) 트렌드-MLaaS," Jan. 2021, https://sqlermail.medium.com

[10] Vertiv, Data Center 2025-Closer To The Edge, 2019, https://www.vertiv.com/en-emea/about/news-and-insights/articles/pr-campaigns-reports/data-center-2025-closer-to-the-edge/

[11] https://aws.amazon.com/ko/sagemaker/edge/

[12] https://cloud.google.com/vertex-ai

[13] https://docs.microsoft.com/ko-kr/azure/iot-edge/

[14] https://octoml.ai/

[15] https://www.edgeimpulse.com/

[16] https://sensiml.com/

[17] Google Inc., TensorFlow Lite, https://www.tensorflow.org/lite/

[18] Google Inc., TensorFlow Lite for Microcontrollers, https://www.tensorflow.org/lite/microcontrollers

[19] Microsoft Corp., Embedded Learning Library, https://microsoft.github.io/ELL/

[20] ARM In., ARM-NN, https://www.arm.com/products/silicon-ip-cpu/ethos/arm-nn

[21] https://autokeras.com

[22] https://www.tiriasresearch.com/wp-content/uploads/2021/09/Edge-Impulse-Accelerates-MLOps-with-EON.pdf

[23] https://github.com/microsoft/nni

[24] S. Leroux et al., "TinyMLOps: Operational challenges for widespread edge AI Adoption," arXiv preprint, CoRR, 2022, arXiv: 2203.10923v2.

[25] M. Rastegari et al., "XnorNet: ImageNet classification using binary convolutional neural networks," arXiv preprint, CoRR, 2016, arXiv: 1603.05279.

[26] W. Chen, P. Wang, and J. Cheng, "Towards mixed-precision quantization of neural networks via constrained cptimization," arXiv preprint, CoRR, 2021, arXiv: 2110.06554.

[27] K. Wang et al., "HanHAQ: Hardware-aware automated quantization," arXiv preprint, CoRR, 2019, arXiv: 1811.08886.

[28] Y. He et al., "Filter pruning via geometric median for deep convolutional neural networks acceleration," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., (Long Beach, CA, USA), June 2019, pp. 4340-4349.

[29] Y. He et al., "Amc: Automl for model compression and acceleration on mobile devices," in Proc. Eur. Conf. Comput. Vis. (ECCV), (Munich, Germany), Sept. 2018, pp. 784-800.

[30] J. Martinez et al., "Permute, quantize, and fine-tune: Efficient compression of neural networks," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., (Virtual), June 2021, pp. 15699-15708.

[31] T. Kim et al., "CPrune: Compiler-informed model pruning for efficient target-aware DNN execution," arXiv preprint, CoRR, 2022, arXiv: 2207.01260.

[32] T.J. Yang et al., "Netadapt: Platform-Aware neural network adaptation for mobile applications," in Proc. Eur. Conf. Comput. Vis. (ECCV), (Munich, Germany), Sept. 2018, pp. 285-300.

[33] K.T. Chitty-Venkata and A.K. Somani, "Neural architecture search survey: A hardware perspective," ACM Comput. Surv., 2022, doi: 10.1145/3524500.

[34] https://github.com/etri/nest-compiler

[35] X. Team, Xla: Domain-Specific Compiler for Linear Algebra that Optimizes Tensorflow Computations, 2019.

[36] W.F. Lin et al., "Onnc: A compilation framework connecting onnx to proprietary deep learning accelerators," in Proc. IEEE Int. Conf. Artif. Intell. Circuits Sys. (AICAS), (Hsinchu, Taiwan), Mar. 2019, pp. 214-218.

[37] S. Cyphers et al., "Intel nGraph: An intermediate representation, compiler, and executor for deep learning," arXiv preprint, CoRR, 2018, arXiv: 1801.08058.

[38] N. Rotem et al., "Glow: Graph lowering compiler techniques for neural networks," arXiv preprint, CoRR, 2018, arXiv: 1805.00907.

[39] T. Chen et al., "Tvm: An automated end-to-end optimizing compiler for deep learning," in Proc. USENIX Symp. Oper. Syst. Des. Implement., (Carlsbad, CA, USA), Oct. 2018, pp. 578-594.

[40] M. Li et al., "The deep learning compiler: A comprehensive survey," arXiv preprint, CoRR, 2020, arXiv: 2002.03794v4.

[41] R. David et al., "TensorFlow lite micro: Embedded machine learning for TinyML systems," in Proc. Mach. Lear. Syst., (San Jose, CA, USA), 2021.

[42] Microsoft Corp., The Embedded Learning Library-Embedded Learning Library(ELL), 2022, Retrieved from https://microsoft.github.io/ELL/

[43] W. Li and M. Liewig, "A survey of AI accelerators for edge environment," in World Conference on Information Systems and Technologies, Springer, Cham, Switzerland, 2020, pp. 35-44.