특집논문 (Special Paper)
방송공학회논문지 제26권 제4호, 2021년 7월 (JBE Vol. 26, No. 4, July 2021)
https://doi.org/10.5909/JBE.2021.26.4.368
ISSN 2287-9137 (Online) ISSN 1226-7953 (Print)

컴퓨팅 부하 예측 DNN 모델 기반 디지털 트윈 소프트웨어 개발 프레임워크

김 동 연[a], 윤 성 진[a)b)], 김 원 태[a)b)‡]

A Digital Twin Software Development Framework based on Computing Load Estimation DNN Model

Dongyeon Kim[a], Seongjin Yun[a)b)], and Won-Tae Kim[a)b)‡]

요 약

인공지능 클라우드는 학습된 모델 공유 및 실행 환경을 제공하여 인공지능 기술과 제어 기술을 융합하는 자율 사물 개발을 지원한다. 기존 자율 사물 개발 기술은 인공지능 모델의 정확도만을 고려하여 은닉 계층 수 및 커널 수 증가 등 모델의 복잡성을 증가시켜 결과적으로 많은 연산량을 요구하게 한다. 자원 제약적 컴퓨팅 환경은 해당 모델이 필요로 하는 충분한 자원을 제공할 수 없어 자율 사물의 실시간성 장애를 발생시킬 수 있다. 본 논문은 컴퓨팅 환경에 최적화된 인공지능 모델을 선택하는 디지털 트윈 소프트웨어 개발 프레임워크를 제안한다. 제안 프레임워크는 DNN 기반 부하 예측 모델을 활용하여 제어 소프트웨어를 개발한다. 부하 예측 모델은 디지털 트윈을 활용하여 인공지능 모델의 부하를 예측하여 특정 컴퓨팅 환경에 최적의 모델 선택을 지원한다. 대표적인 CNN 모델을 활용한 부하 예측 실험으로 제안 부하 예측 DNN 모델이 수식 기반 부하 예측 대비 최대 20%의 오류를 보임을 확인했다.

Abstract

Artificial intelligence clouds help to efficiently develop the autonomous things integrating artificial intelligence technologies and control technologies by sharing the learned models and providing the execution environments. The existing autonomous things development technologies only take into account for the accuracy of artificial intelligence models at the cost of the increment of the complexity of the models including the raise up of the number of the hidden layers and the kernels, and they consequently require a large amount of computation. Since resource-constrained computing environments, could not provide sufficient computing resources for the complex models, they make the autonomous things violate time criticality. In this paper, we propose a digital twin software development framework that selects artificial intelligence models optimized for the computing environments. The proposed framework uses a load estimation DNN model to select the optimal model for the specific computing environments by predicting the load of the artificial intelligence models with digital twin data so that the proposed framework develops the control software. The proposed load estimation DNN model shows up to 20% of error rate compared to the formula-based load estimation scheme by means of the representative CNN models based experiments.

Keyword : Artificial intelligence cloud, data-driven model, load estimation, digital twin, autonomous things

Ⅰ. 서 론

인공지능 기술과 제어 시스템이 결합된 자율 사물 개발 기술에 대한 연구가 활발하게 이루어지고 있다[1]. 자율 비행 드론으로 대표되는 자율 사물은 카메라, LiDAR 등의 센서 데이터들을 인공지능 기술로 처리하여 객체 인식, 공간 인지 등 상황인지를 수행하고 이를 활용하여 미션을 수행한다[2]. 인공지능 클라우드는 데이터의 유형과 사용 목적에 맞는 사전 학습된 모델과 인공지능 모델들을 위한 컴퓨팅 자원을 포함한 실행 환경을 제공해 주어 임베디드 소프트웨어 개발자가 인공지능 기술과 제어 기술이 융합된 자율 사물을 손쉽게 개발할 수 있도록 지원한다. 클라우드 기반의 학습 및 실행 환경의 제공은 인공지능 모델이 자원의 제약 없이 실행할 수 있게 만들어 시스템의 제약 없이 다양한 산업 도메인에서 활용될 수 있도록 해주었다.

자율 사물은 사람의 개입 없이 동작하는 신뢰성을 갖춘 시스템이므로 실시간 상황인지 및 의사결정이 이루어져야 한다[3]. 하지만 클라우드를 통한 인공지능 모델 실행은 통신 지연, 메시지 처리 지연 등의 요인으로 인해 발생하는 지연 시간으로 인해 자율 사물의 실시간성 요구사항을 만족시키지 못하게 만들 수 있다[4]. 예를 들어 자율 비행 드론의 경우 초당 수십 프레임 이상의 고해상도의 이미지 처리를 바탕으로 의사결정을 수행하는데 클라우드와의 통신으로 인한 통신 지연으로 인해 장애물을 늦게 인식하게 되어 충돌을 회피하지 못하게 하는 등의 사고의 원인이 될 수 있다[5].

따라서 실시간 의사결정을 위해 필요한 인공지능 모델들은 자율 사물에 직접 탑재되어야 한다. 이때 정확도만을 고려하여 탑재할 모델을 결정하면 은닉 계층 수 및 커널의 수가 큰 모델을 선택하게 되어 모델 실행에 있어 많은 양의 컴퓨팅 자원을 필요로 하게 될 수 있다. 자원 제약적인 컴퓨팅 환경에서 정확도만을 고려한 모델 탑재는 컴퓨팅 자원의 부족으로 인해 실시간성을 만족시키지 못하게 되어 물리 시스템의 오작동을 초래할 수 있다[6][7].

본 논문에서는 이러한 문제를 해결하기 위해 컴퓨팅 부하 예측 DNN (Deep neural network) 모델 기반 디지털 트윈 소프트웨어 개발 프레임워크를 제안한다. 제안한 프레임워크는 하드웨어의 성능, 가용한 자원 상태 그리고 센싱 데이터 등을 포함하고 있는 디지털 트윈을 통하여 지능형 제어 소프트웨어를 설계하고 검증을 수행한다. 타겟 물리 시스템에 적합한 인공지능 모델을 선택하기 위하여 사용자 요구사항 정의를 바탕으로 한 인공지능 모델 탐색 기법 및 컴퓨팅 부하 예측 DNN 모델을 활용한 부하 예측 메커니즘을 적용하여 디지털 트윈을 통한 신뢰할 수 있는 지능형 제어 소프트웨어 개발을 수행한다.

본 논문의 나머지 부분은 아래의 내용들을 다룬다. Ⅱ장에서는 손쉬운 인공지능 기반 제어 소프트웨어 개발을 위해 제시되었던 기존 인공지능 프레임워크와 인공지능 클라우드에 대해 다루고, 인공지능 모델의 부하 예측 기법들에 대하여 다룬다. Ⅲ장에서는 제시하는 프레임워크의 구조와 핵심 기법에 관하여 서술하고, Ⅳ장에서는 제시한 부하 예측 기법의 성능 검증을 수행하였다. Ⅴ장을 통해 논문을 결론 짓는다.

Ⅱ. 관련 연구

1. 모델 성능 분석 및 모델 선택

Marco Vicent Sanz[8]는 임베디드 보드에서 CNN (Convolutional neural network) 모델을 효율적으로 실행할 수 있도록 정확도와 추론 시간을 고려하여 주어진 입력에 따라 사전 훈련된 CNN 모델 세트에서 모델을 동적으로 선택하는 저비용 예측 모델을 제안하였다. 입력 값에 따라 CNN

a) 한국기술교육대학교 컴퓨터공학과(The Department of Computer Science and Engineering, Korea University of Technology and Education)
b) 한국기술교육대학교 미래융합공학전공(Future Convergence Engineering, Korea University of Technology and Education)
‡ Corresponding Author : 김원태(Won-Tae Kim)
E-mail: wtkin@koreatech.ac.kr
Tel: +82-41-560-1485
ORCID:https://orcid.org/0000-0003-3426-3792
※본 논문은 2020년도 한국기술교육대학교 교수 교육연구진흥과제의 지원과 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원(No. 2018-0-01456) 지원을 통하여 연구되었음.
※This work was partly supported by the Institute for Information & communications Technology Promotion (IITP) (No. 2018-0-01456) from the Korea government (MSIT), and Education and Research promotion program of KOREATECH in 2020.
· Manuscript received May 18, 2021; Revised July 19, 2021; Accepted July 19, 2021.

모델의 성능이 달라지기 때문에 KNN (K-Nearest Neighbour) 알고리즘을 기반으로 한 최적의 모델을 선택을 통해 단일 모델을 사용하는 것보다 정확도 향상과 추론 시간 감소를 달성하였다. 해당 연구는 사전 학습된 CNN 세트에서만 모델을 선택할 수 있는 반면, 제안한 프레임워크의 부하 예측 DNN 모델은 CNN 모델의 연산량, 연산강도(Operational Intensity) 등 구조적 요인들을 고려하여 모델이 지속적으로 갱신되는 인공지능 클라우드에서 최적의 모델을 선택할 수 있다.

Aleksandar Ilic[9]은 특정 컴퓨터 아키텍처의 성능 한계를 정확하게 표현하기 위하여 기존 Roofline 모델에 캐시 인식 개념을 도입한 Cache-aware roofline 모델을 제안했다. 기존의 Roofline 모델은 DRAM 메모리에서 중앙처리장치로의 대역폭만을 고려하기 때문에 캐시 인식 개념을 도입한 Cache-aware roolifne 모델은 DRAM 메모리에서 중앙처리장치로의 대역폭뿐만 아니라, 캐시 대역폭을 고려하여 인공지능 모델의 성능을 정확하게 표현할 수 있었다. 해당 연구는 인공지능 모델이 이상적인 환경에서 갖는 성능을 표현하기 때문에 실제 인공지능 모델이 동작하는 환경에서의 성능을 정확하게 예측하기 어려울 수 있다. 제안한 프레임워크에서는 Roofline 모델의 달성 가능한 최대 FLOPS (Floating point operation per second)를 도출하는 수식을 확장하여 실제 환경에서 인공지능 모델의 예상 추론 시간을 구하는 컴퓨팅 부하 예측 DNN 모델을 설계했다.

Kaiming He[10]은 CNN 모델의 구조에 따른 시간복잡도 수식을 통해 모델의 성능을 분석하여 제한된 시간 비용 하에서 최적의 모델을 선택하는 방법을 제시하였다. 은닉 계층의 깊이, 필터 수, 필터 크기로 계산한 이론적 시간복잡도와 정확도의 비교를 통해 최적의 모델을 선택했다. 하지만 해당 연구의 제한적 요소만을 고려한 시간복잡도 수식은 하드웨어 성능을 고려하지 않고 CNN 모델의 연산량 계산에 사용되는 요소만을 고려하기 때문에 실제 실행시간과 오차가 발생할 수 있다. 따라서, 제안한 프레임워크는 디지털 트윈 하드웨어 성능과 컴퓨팅 자원을 고려한 컴퓨팅 부하 예측 DNN 모델을 통해서 높은 정확도의 추론 시간 예측을 수행한다.

Ningning Ma[11]는 4가지 기준을 바탕으로 한 CNN 모델의 추론 성능 평가 및 최적화 방법을 제시하였다. 그들은 하드웨어 플랫폼의 특성을 고려하여 입출력 채널 수, 그룹 Convolution 수, 신경망 단편화 수준 그리고 element-wise 연산 수의 4가지 기준을 활용한 성능 평가를 통해 대표적인 CNN 모델인 ShuffleNet을 분석하여 최적의 구조를 갖는 ShuffleNet V2를 개발하였다. 제안한 기준을 바탕으로 성능 평가 및 최적의 모델 선택을 수행하기 위해서는, 인공지능 모델이 타겟 하드웨어에서 직접 실행되어야 하기 때문에 다수의 인공지능 모델에 대한 평가를 위해서는 많은 시간과 비용이 소요될 수 있다. 제안하는 프레임워크의 부하 예측 모델은 실제 실행 없이 모델의 구조, 하드웨어 정보를 입력으로 하여 모델의 부하를 측정할 수 있다.

Ⅲ. 본 론

1. 디지털 트윈을 활용한 예측 가능한 지능형 제어 소프트웨어 개발 프레임워크

그림 1은 제안한 디지털 트윈 기반 신뢰할 수 있는 지능형 제어 소프트웨어 개발 프레임워크 구성도이다. 제안 프레임워크는 물리 시스템과 연결되어 작동 상태를 반영하고 있는 디지털 트윈을 활용하여 지능형 제어 소프트웨어를 생성한다. Requirement description interface는 사용자로부터 요구사항을 입력받을 수 있는 인터페이스 역할을 수행한다. 사용자는 이를 통해 개발하고자 하는 지능형 제어 소프트웨어의 요구사항을 명세한다. AI model explorer는 사용자의 물리 시스템에 대응하는 디지털 트윈으로부터 시스템 정보를 전달받고 사용자가 입력한 요구사항과 대조하여 인공지능 클라우드에서 인공지능 모델을 탐색한다. 탐색된 인공지능 모델은 Control SW code generator로 전달되어 실행 가능한 제어 소프트웨어로 변환되고 디지털 트윈을 통하여 물리 시스템에 탑재될 수 있도록 한다.

2. 프레임워크 상세 설계

본 논문에서 제안한 디지털 트윈을 활용한 예측 가능한 지능형 제어 소프트웨어 개발 프레임워크의 핵심 기능은

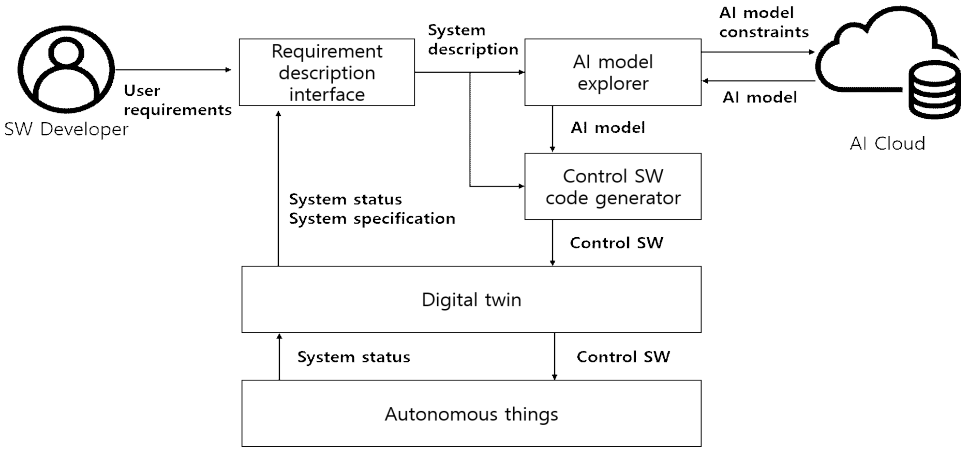

그림 1. 컴퓨팅 부하 예측 DNN 모델 기반 디지털 트윈 소프트웨어 개발 프레임워크
Fig. 1. Overview of Digital Twin Software Development Framework based on Computing Load Estimation DNN Model

자율 사물과 디지털 트윈의 상위 계층에 속하는 애플리케이션 계층의 모듈인 Requirement description, AI model explorer, Control SW code generator이다. 각각의 모듈들이 인공지능 모델의 조건 설정, 조건에 맞는 인공지능 모델 탐색, 인공지능 모델을 위한 코드 생성을 통해 지능형 제어 소프트웨어를 개발한다.

2.1 Requirement description interface

Requirement description interface는 물리 시스템에서 사용 가능한 모델 선택을 위해 사용자로부터 사용자 요구사항을 입력받는 인터페이스를 제공한다. 사용자로부터 자율 제어 소프트웨어를 위한 인공지능 모델의 사용자 요구사항을 전달받고, 디지털 트윈으로부터는 물리 시스템의 실행 상태와 연산 성능을 전달받아 명세화한다. 본 프레임워크에서 입력받는 사용자 요구사항은 모델의 특성을 나타내는 정적 명세, 하드웨어의 성능, 제어 소프트웨어의 유형에 따라 달라지는 동적 명세로 나누어진다. 정적 명세로는 분류, 객체 인식과 같은 인공지능 모델의 동작 분류(Category), 모델의 인식 대상을 나타내는 Class, 정확도를 나타내는 Accuracy를 입력받는다. 동적 명세로는 모델이 실행되어야 하는 제약 시간인 추론 시간을 입력받는다. 물리 시스템 명세로는 초당 처리할 수 있는 최대 성능을 가리키는 Peak FLOPS, 최대 메모리 대역폭(Peak memory bandwidth)을 입력받는다. Requirement description interface를 통해 명세되는 사용자 요구사항 명세는 표 1과 같다.

표 1. 사용자 요구사항 명세 양식
Table 1. User requirements specification form

User requirements		System specification
Static specification	Dynamic specification	
Category	Inference time	Peak FLOPS
Class		
Accuracy		Peak memory bandwidth

2.2 AI model explorer module

AI model explorer는 사용자 요구사항 명세를 만족하기 위해서 그림 2의 모델 선택 알고리즘을 수행한다. 처음으로, 사용자 요구사항 중 시스템 성능에 영향을 받지 않는 정적 명세를 기준으로 인공지능 모델 목록을 인공지능 클라우드로부터 전달 받는다. 다음으로, 연산 수행 능력을 의미하는 연산 강도와 물리 시스템의 최대 메모리 대역폭, Peak FLOPS를 수식[9]에 대입하여 모델이 하드웨어에서 달성 가능한 최대 FLOPS와 계산된 실행시간(Calculated execution time)을 구한다. 수식을 통해 계산된 실행시간은 이론상 최대 시간이며 모델의 실제 실행시간과 다르다. 따라서 수식이 반영하지 못하는 특징들을 고려한 정확한 부하 예측을 위해 그림 3과 같이 CNN 모델의 연산량(FLOPs), 메모리 접근량, 달성 가능한 최대 FLOPS를 입력

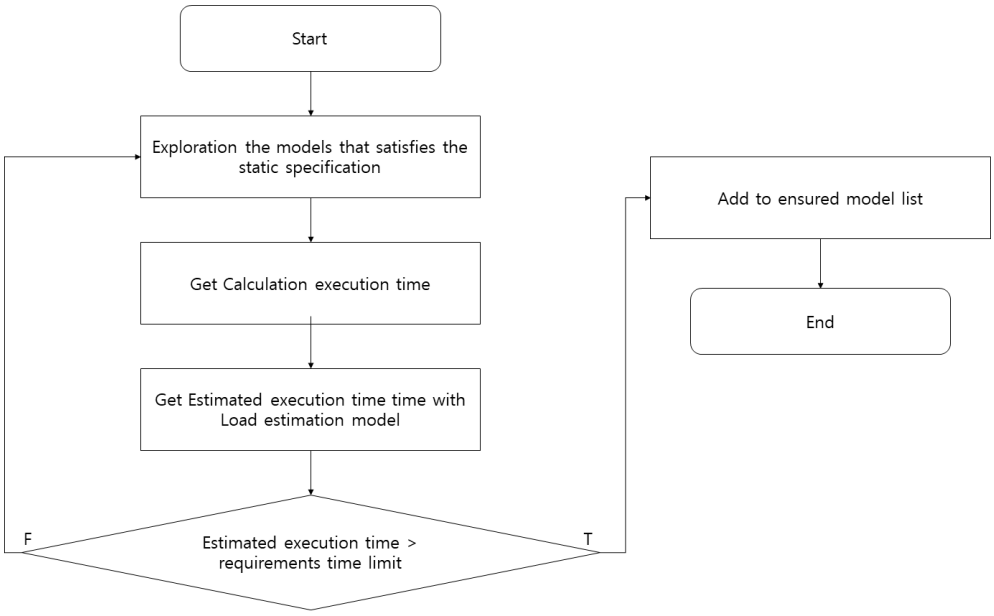

그림 2. 모델 선택 알고리즘
Fig. 2. Model selection algorithm

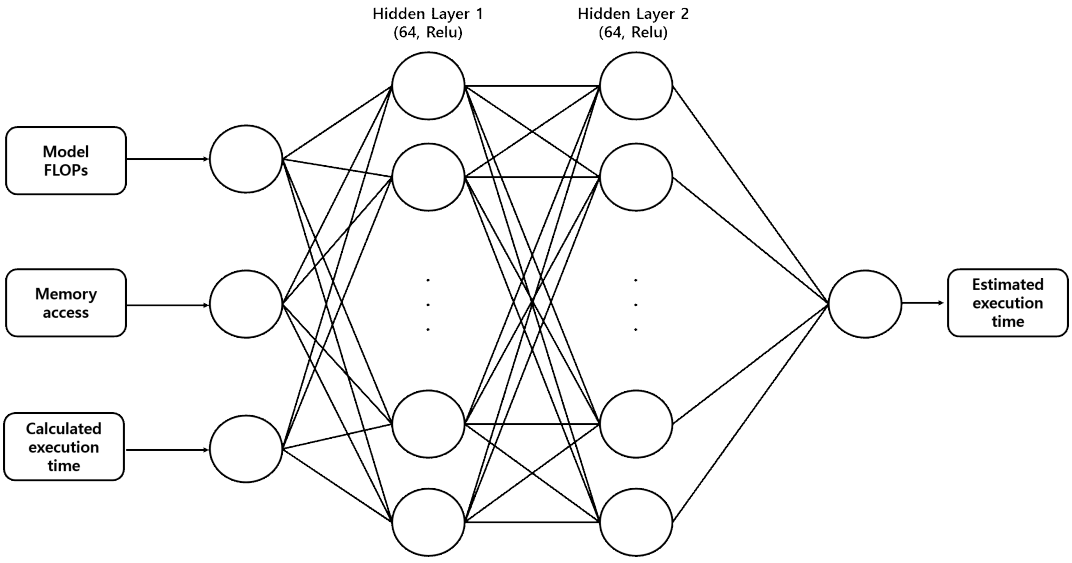

그림 3. 컴퓨팅 부하 예측 DNN 모델
Fig. 3. Computing load estimation DNN model

으로 사용하는 컴퓨팅 부하 예측 DNN 모델을 통해서 모델의 예측 실행시간(Estimated execution time)을 구한다. 사용자 요구사항의 추론 시간 조건을 충족하는 모델 중 가장 짧은 연산 시간을 갖는 모델을 선택하여 인공지능 클라우드로부터 내려 받는다.

2.3 Control SW code generator

Control SW code generator는 AI model explorer를 통해

선택된 인공지능 모델을 사용한 제어 실행 코드를 생성해 낸다. 본 논문에서는 텐서플로우 라이브러리를 활용한 코드를 생성하여 제공한다. 생성된 코드는 디지털 트윈으로 전달되어 작동 검증을 수행하고, 선택된 인공지능과 함께 물리 시스템으로 전달되어 탑재된다.

3. 동작 순서 및 데이터 흐름

제안 프레임워크의 동작 순서는 그림 4에서 확인할 수 있다. 프레임워크를 이용하여 제어 소프트웨어를 개발하고자 하는 사용자는 사용자 요구사항을 Requirements description interface에 입력한다. AI model explorer는 작성된 요구사항 명세를 전달받아 인공지능 모델 탐색을 수행한다. 먼저 사용자 요구사항의 정적 명세를 만족하는 인공지능 모델 목록을 인공지능 클라우드로 제공 받은 뒤, 제시된 모델들의 명세정보와 사용자의 하드웨어 성능을 고려하여 모델의 예상 추론 시간을 구하여 발생 가능한 부하를 예측한다. 이후 사용자 요구사항의 추론 시간 조건을 충족하는 모델이 결정되면 인공지능 클라우드로부터 내려 받아 Control SW code generator로 전달하여 실행 가능한 코드를 생성한다. 생성된 코드는 디지털 트윈에 탑재되어 테스트를 거친 뒤 제어 소프트웨어 형태로 물리 시스템으로 전달된다.

Ⅳ. 실험

1. 실험 환경

본 실험의 부하 예측 DNN 모델은 Imagenet 데이터를 통해 학습된 CNN 모델의 연산량, 메모리 접근량, 수식 기반으로 계산된 실행시간을 입력받아서 CNN 모델의 실행시간을 예측한다. 부하 예측 DNN 모델 학습을 위한 손실함수로는 MSE(Mean Squared Error)를 사용하였고, 평가지표로 MAE(Mean Absolute Error)를 사용하였다. 또한 동일한 구조의 부하 예측 모델을 K개 생성한 뒤 K개로 나눈 학습데이터를 훈련데이터와 학습데이터로 교차하여 학습시키는 기법인 K-fold cross validation을 사용하여 모델의 성능을

표 2. 실험에 사용된 하드웨어 사양
Table 2. Hardware spec for experiments

Hardware name	Peak FLOPS (TFLOPs/s)	Peak memory bandwidth (GB/s)
RTX 3090	35.38	936.2

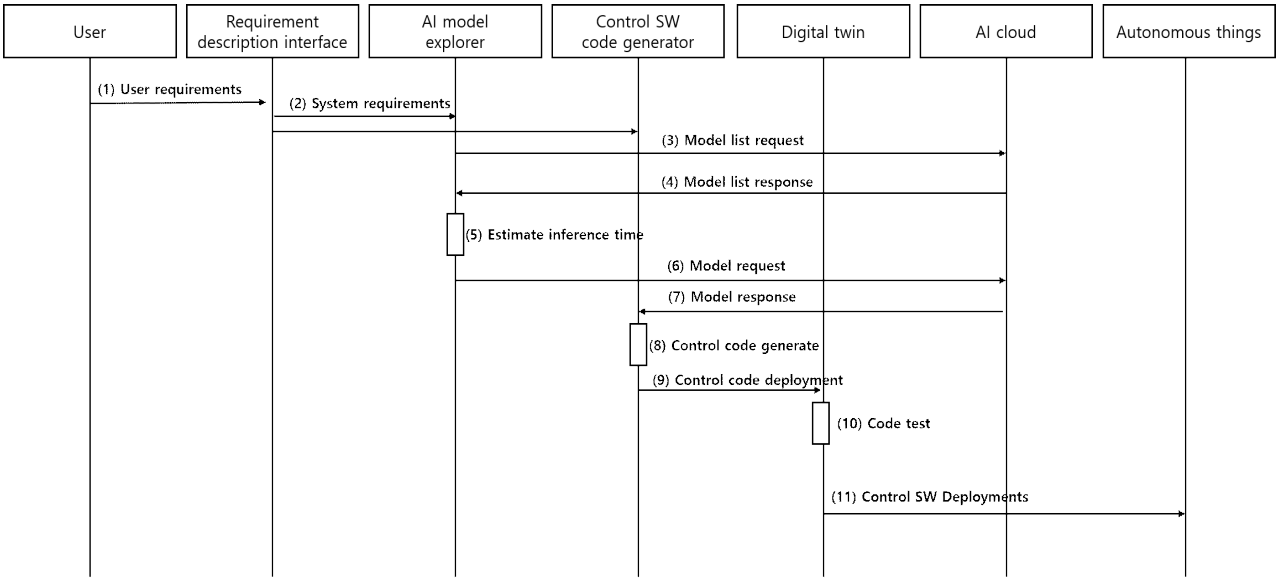

그림 4. 제안한 프레임워크의 동작 순서도
Fig. 4. Sequence diagram of the proposed framework

표 3. 실험에 사용된 모델 사양
Table 3. Model spec for experiments

Category	Model	Input	FLOPS(GFLOPs)	Memory access(MB)
Classification	VGG16	(224, 224, 3)	15.5	746.89
	VGG19	(224, 224, 3)	19.67	787.05
	ResNet50	(224, 224, 3)	4.12	317,43
	ResNet101	(224, 224, 3)	7.84	494
	ResNet152	(224, 224, 3)	11.57	682.3
	MobileNetV2	(224, 224, 3)	0.32024	162.2
	DenseNet121	(224, 224, 3)	2.88	359.7
	DenseNet169	(224, 224, 3)	3.42	448.55
	DenseNet201	(224, 224, 3)	4.37	581.5

향상시켰다. 본 실험에서 사용된 하드웨어는 표 2와 같고, 본 실험에서 사용된 CNN 모델은 표 3과 같다.

본 실험에서 사용한 수식 기반의 예측 실행시간은 아래의 수식 (1)-(4)을 통해 계산된다. 수식 (1)은 모델의 메모리 접근량 대비 연산량을 의미하는 연산 강도를 계산하고, 수식 (2)는 연산 강도와 최대 메모리 대역폭의 곱을 통해 타겟 하드웨어에서 모델의 성능을 의미하는 F^{OP}을 계산한다. 수식 (3)는 하드웨어의 최대 FLOPS와 모델의 성능을 비교하여 타겟 하드웨어에서 CNN 모델이 달성 가능한 최대 FLOPS를 구한다. 수식 (4)은 CNN 모델의 연산량을 달성 가능한 최고 성능으로 나눠서 CNN 모델의 계산된 실행시간을 구한다. 계산된 실행시간은 CNN 모델의 연산량, 메모리 접근량과 함께 CNN 모델의 실행시간 예측을 위한 부하 예측 모델의 입력으로 사용된다. 수식 (1)-(4)는 다음과 같다.

$$Operational\ Intensity = \frac{FLOPs}{Memory access} \quad (1)$$

$$F^{OP} = Operational Intensity * Peak Memory bandwidth \quad (2)$$

$$Maximum\ FLOPS = MIN(Peak\ FLOPS, F^{OP}) \quad (3)$$

$$Calculated\ execution\ time = \frac{FLOPs}{Maximum\ FLOPS} \quad (4)$$

2. 실험 결과 및 분석

그림 5는 기존 수식을 통해 계산된 실행시간(Calculated execution time)과 부하 예측 DNN 모델을 통해 예측 실행시간(Estimated execution time)의 비교를 위해 예측시간($T_{estimated}$)과 실제 모델을 실행해서 구한 실행시간(T_{real})을 수식 (5)의 평균 절대 오차(MAE)를 통해 비교한 결과이다.

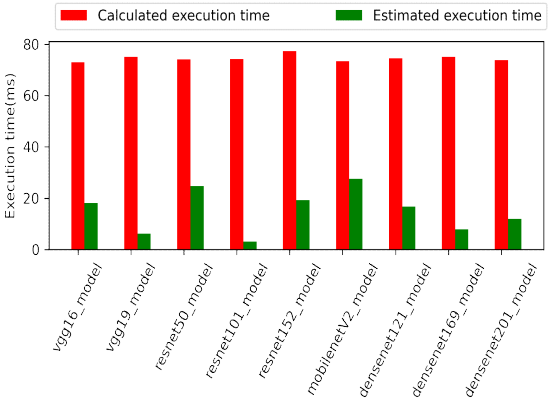

그림 5. 기존 수식 기반과 부하 예측 모델의 평균 절대 오차 비교
Fig. 5. Comparison of the average absolute error between the existing equation and the load estimation model

$$MAE = \frac{1}{n}\sum_{i=1}^{n} | T_{estimated} - T_{real} | \quad (5)$$

기존 수식 기반 예측 기법 대비 부하 예측 DNN 모델이 기존 대비 20% 수준의 적은 오차가 발생함을 알 수 있었다. 단순한 수식만으로 계산되어 모델의 이론상 최대 속도만을 계산했던 기존 수식 기반 예측 기법의 예측 값에 비해서 CNN 모델의 연산량, CNN 모델의 메모리 접근량, 기존 수식 기반 예측 기법 예측 값을 사용하여 학습된 회귀 모델인 부하 예측 모델이 실행시간 예측을 더 정확히 해낼 수 있음을 확인하였다.

V. 결 론

본 연구에서는 컴퓨팅 부하 예측 DNN 모델을 활용한 디지털 트윈 기반 지능형 제어 소프트웨어 개발 프레임워크를 제안하였다. 인공지능 모델의 정확도, 연산 복잡도 그리고 가용한 컴퓨팅 자원을 고려하여 인공지능 모델을 선택하는 알고리즘을 통해 제약된 컴퓨팅 자원을 가지는 임베디드 환경에서 최적의 인공지능 모델의 탑재가 가능하게 하였다. 대표적인 CNN 모델을 활용한 부하 예측 실험을 통해 인공지능 모델 실행시간 예측에서 DNN 기반 부하 예측 모델이 수식 기반 예측 대비 20% 수준의 오차를 보임을 확인했다. 제안한 프레임워크를 활용하면 다양한 산업 도메인에서 자율 사물의 도입을 촉진할 수 있게 될 것으로 기대된다. 향후 부하 예측을 활용한 오프로딩 기술 등 분산 컴퓨팅 기술 연구를 통해 에너지 소모, 저장 공간 제약 등 다양한 제약을 갖는 엣지 디바이스를 지원하여 스마트 홈, 스마트 시티 등 다양한 사물인터넷 서비스에서 손쉽게 인공지능 기술을 활용할 수 있도록 제안 프레임워크를 확장할 것이다.

참 고 문 헌 (References)

[1] Software Policy & Research Institute, Report on the AI Talents in New Southern and Northern Countries, 2020.

[2] R. Kanan, O. Elhassan, R. Bensalem, "An IoT-based autonomous system for workers' safety in construction sites with real-time alarming, monitoring, and positioning strategies," Automation in Construction, Vol.88, pp.73-86, Apr 2018.

[3] C.J. Marshalla, B. Roberts, M. Grenn, "Adaptive and automated reasoning for autonomous system resilience in uncertain worlds," Disciplinary Convergence in Systems Engineering Research, pp.799-812, 2018, doi:10.1007/978-3-319-62217-0_56.

[4] N. Tijtgat, W. Van Ranst, T. Goedeme, B. Volckaert, F. De Turck, "Embedded real-time object detection for a UAV warning system," Proceedings of the IEEE International Conference on Computer Vision Workshops, pp.2110-2118, 2017.

[5] F. Luo, C. Jiang, S. Yu, J. Wang, Y. Li, Y. Ren, "Stability of cloud-based UAV systems supporting big data acquisition and processing," IEEE Transactions on Cloud Computing, Vol.7, No.3, pp.866-877, Apr 2017.

[6] D. Kim, S. Yun, W.T. Kim, “Intelligent solution software development framework for easy Autonomous thing development,” Proceedings of Symposium of the Korean Institute of communications and Information Sciences, Jeju, Korea, pp 770-771, 2021 June.

[7] R. Huang, J. Pedoeem, C. Chen, "YOLO-LITE: a real-time object detection algorithm optimized for non-GPU computers," In 2018 IEEE International Conference on Big Data (Big Data), pp.2503-2510, Dec 2018.

[8] V.S. Marco, B. Talyor, Z. Wang, Y. Elkhatib, "Optimizing deep learning inference on embedded systems through adaptive model selection," ACM Transactions on Embedded Computing Systems (TECS), Vol.19, No.1, pp.1-28, Feb 2020.

[9] A. Ilic, F. Pratas, L. Sousa, "Cache-aware roofline model: Upgrading the loft," IEEE Computer Architecture Letters, Vol.13, No.1, pp.21-24, Apr 2013.

[10] K. He, J. Sun, "Convolutional neural networks at constrained time cost", Proceedings of the IEEE conference on computer vision and pattern recognition, pp.5353-5360, 2015.

[11] N. Ma, X. Zhang, H.T. Zheng, J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," Proceedings of the European conference on computer vision (ECCV), pp.116-131, 2018.

저 자 소 개

김 동 연

- 2021년 : 한국기술교육대학교 컴퓨터공학과 학사
- 2021년 ~ 현재 : 한국기술교육대학교 컴퓨터공학과 석사과정
- ORCID : https://orcid.org/0000-0002-2249-4094
- 주관심분야 : AI, CPS, Digital Twin, Cloud computing, Autonomous things

저 자 소 개

윤 성 진

- 2016년 : 한국기술교육대학교 컴퓨터공학 학사
- 2019년 : 한국기술교육대학교 컴퓨터공학 석사
- 2019년 ~ 현재 : 한국기술교육대학교 컴퓨터공학 박사과정
- ORCID : https://orcid.org/0000-0003-4082-2052
- 주관심분야 : Digital Twin, XAI, CPS, SDN, Cloud computing

김 원 태

- 1994년 : 한양대학교 공학사
- 1996년 : 한양대학교 공학석사
- 2000년 : 한양대학교 공학박사
- 2001년 ~ 2005월 : ㈜로스틱테크놀로지 기술이사
- 2005년 ~ 2010년 : 한국전자통신연구원 임베디드SW연구부 선임연구원
- 2010년 ~ 2015년 : 한국전자통신연구원 CPS연구실 실장/책임연구원
- 2015년 ~ 2019년 : 한국기술교육대학교 컴퓨터공학부 조교수
- 2019년 ~ 현재 : 한국기술교육대학교 컴퓨터공학부 부교수
- ORCID : https://orcid.org/0000-0003-3426-3792
- 주관심분야 : Digital Twin, CPS, XAI, WTSN, Autonomous Computing