논문 2023-18-12

임베디드 엣지 플랫폼에서의 경량 비전 트랜스포머 성능 평가

(Performance Evaluation of Efficient Vision Transformers on Embedded Edge Platforms)

이 민 하, 이 성 재, 김 태 현*
(Minha Lee, Seongjae Lee, Taehyoun Kim)

Abstract : Recently, on-device artificial intelligence (AI) solutions using mobile devices and embedded edge devices have emerged in various fields, such as computer vision, to address network traffic burdens, low-energy operations, and security problems. Although vision transformer deep learning models have outperformed conventional convolutional neural network (CNN) models in computer vision, they require more computations and parameters than CNN models. Thus, they are not directly applicable to embedded edge devices with limited hardware resources. Many researchers have proposed various model compression methods or lightweight architectures for vision transformers; however, there are only a few studies evaluating the effects of model compression techniques of vision transformers on performance. Regarding this problem, this paper presents a performance evaluation of vision transformers on embedded platforms. We investigated the behaviors of three vision transformers: DeiT, LeViT, and MobileViT. Each model performance was evaluated by accuracy and inference time on edge devices using the ImageNet dataset. We assessed the effects of the quantization method applied to the models on latency enhancement and accuracy degradation by profiling the proportion of response time occupied by major operations. In addition, we evaluated the performance of each model on GPU and EdgeTPU-based edge devices. In our experimental results, LeViT showed the best performance in CPU-based edge devices, and DeiT-small showed the highest performance improvement in GPU-based edge devices. In addition, only MobileViT models showed performance improvement on EdgeTPU. Summarizing the analysis results through profiling, the degree of performance improvement of each vision transformer model was highly dependent on the proportion of parts that could be optimized in the target edge device. In summary, to apply vision transformers to on-device AI solutions, either proper operation composition and optimizations specific to target edge devices must be considered.

Keywords : Vision Transformer, On-device AI, Image Classification, Quantization, Hardware Accelerators, Performance Evaluation

Ⅰ. 서 론

급격한 하드웨어 성능 발전에 힘입어 딥러닝은 컴퓨터 비전, 자연어 처리와 같은 다양한 분야에서 주목할 만한 성과를 내고 있다. 일반적인 딥러닝 모델은 많은 연산량이 필요하므로 고성능 Central Processing Unit (CPU)과 Graphics Processing Unit (GPU)이 장착된 클라우드 서버에서 연산이 수행된다. 그러나 클라우드 서버 기반 딥러닝 응용 서비스는 서비스 수행 시마다 클라이언트와 서버가 입출력 데이터를 교환하기 때문에 네트워크 사용에 따른 비용 부담이 높다. 또한, 네트워크 상태에 따라 클라이언트 요청에 대한 응답성이 저하될 수 있고, 데이터 전송 중 해킹 등으로 인한 보안 문제가 생길 우려가 있다 [1, 2].

클라우드 서버뿐만 아니라 모바일 디바이스나 임베디드 보드와 같은 엣지 디바이스의 하드웨어 성능도 마찬가지로 향상됨에 따라 최근에는 클라우드 서버 기반 artificial intelligence (AI) 시스템의 단점을 완화하기 위한 온-디바이스 (On-device) AI 시스템이 주목받고 있다. 온-디바이스 AI 시스템은 엣지 컴퓨팅 개념에 AI를 접목한 것으로, 데이터 취득을 담당하는 엣지 디바이스가 데이터 취득 후 곧바로 AI 모델 연산을 수행하는 시스템이다 [3, 4]. 클라우드 기반 AI 응용 시스템과 달리, 온-디바이스 AI 시스템은 데이터에 대한 연산을 위해 클라우드 서버까지 데이터를 전송할 필요가 없으므로 네트워크 비용이 적게 들고 보안 위협에 노출될 가능성도 적다. 또한, 네트워크 전송에 의한 지연시간이 없어 응답성이 향상될 수 있으며 네트워크 연결 없이도 데이터를 처리할 수 있으므로 저전력 동작이 가능하다. 그러나, 엣지 디바이스는 클라우드 서버보다 상대적으로 프로세서 성능과 메모리 용량 등 하드웨어 리소스가 제한되

*Corresponding Author (thkim@uos.ac.kr)
Received: Feb. 15, 2023, Revised: Mar. 10, 2023, Accepted: Mar. 21, 2023.
M. H. Lee: University of Seoul (M.S. Student)
S. J. Lee: University of Seoul (Ph.D. Candidate)
T. H. Kim: University of Seoul (Prof.)
※ 이 논문은 정부 (과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. 2022R1F1A1060231).

ⓒ IEMEK J. Embed. Sys. Appl. 2023 Jun. 18(3) 89-100
ISSN : 1975-5066
http://dx.doi.org/10.14372/IEMEK.2023.18.3.89

어 있어 모델 연산에 더 많은 시간이 걸리거나 모델 파라미터가 많은 모델은 동작시키기 힘들다는 한계를 가지고 있다. 이러한 한계를 극복하기 위해 임베디드 GPU 플랫폼 [5]이나 신경망 처리 유닛을 Application-Specific Integrated Circuit (ASIC) 형태로 구현한 Tensor Processing Unit (TPU) [6]을 활용하여 엣지 디바이스의 모델 추론을 가속화하려는 시도가 활발하게 이루어지고 있다 [7-9].

온-디바이스 AI 시스템은 차량이나 건물에 장착되어 실시간으로 이미지를 처리해야 하는 자율주행, 스마트시티와 같은 컴퓨터 비전 응용에 활발하게 적용되고 있다 [10-12]. 전통적으로 컴퓨터 비전에 사용되는 딥러닝 모델은 Convolutional Neural Network (CNN) 구조에 근거한 모델 [13-15]이 대부분이었으나, 2020년 이후부터 Natural Language Processing (NLP) 분야에서 큰 성과를 거둔 트랜스포머 모델에 근거한 비전 트랜스포머 (Vision Transformer) [16] 모델들이 큰 관심을 얻고 있다. 비전 트랜스포머 모델들은 초기에는 Fully Connected (FC) 레이어만으로 CNN 모델에 버금가는 추론 정확도를 얻었다는 점으로 주목받았으나, 그 이후 연구가 활발히 진행되며 최근에는 CNN의 정확도를 능가하는 성능을 보일 정도로 발전하였다 [17-19].

그러나 초기 비전 트랜스포머 모델들은 파라미터 수와 연산량이 많아 리소스 제약이 큰 온-디바이스 AI 환경에는 적합하지 않다. 따라서 비전 트랜스포머 모델을 온-디바이스 AI 시스템에 적용하기 위해 양자화 (Quantization)나 프루닝 (Pruning)과 같은 모델 압축 기법을 적용하거나 [20-23], 비전 트랜스포머 모델의 아키텍처를 효율적으로 설계하여 경량화하는 연구 [24, 25]가 활발히 수행되고 있다. 그러나 온-디바이스 AI 시스템에서 CNN 모델을 벤치마킹한 연구는 많이 수행되었던 데 반해 비전 트랜스포머의 성능을 온-디바이스 AI 환경에서 벤치마킹한 연구는 여전히 부족한 상황이다.

따라서 본 연구는 다양한 경량 비전 트랜스포머 모델의 이미지 분류 성능을 엣지 디바이스에서 평가하고 그 결과를 제시한다. 먼저, 세 가지 비전 트랜스포머 아키텍처를 선정하여 CPU 기반의 엣지 디바이스에서 ImageNet 데이터셋 [26]에 대한 각 모델의 이미지 분류 성능을 평가하여 비교하였다. 그 다음, 각 모델에 양자화 기법을 적용하여 양자화에 의한 성능 변화를 확인했으며 각 모델의 연산 구성을 프로파일링하여 모델별 추론 속도 및 양자화 전후 성능 변화의 원인을 분석하였다. 또한, GPU 및 TPU 기반의 엣지 디바이스로 각 모델을 이식하여 하드웨어 가속기 상에서의 성능을 평가하였다. 마지막으로, 성능 평가 결과를 바탕으로 비전 트랜스포머의 온-디바이스 AI 솔루션 적용 가능성을 제시하였다.

본 논문의 구성은 다음과 같다. II장에서는 연산 효율성을 고려한 비전 트랜스포머 아키텍처와 컴퓨터 비전 분야 온-디바이스 벤치마킹 관련 연구를 정리한다. III장은 평가 대상이 되는 비전 트랜스포머 아키텍처와 사용되는 엣지 디바이스, 성능 평가 척도 및 전반적인 성능 평가 방법에 대해 설명한다. IV장에서는 성능 평가 결과를 제시하며 그 고찰을 정리하고 V장에서 논문을 마무리한다.

II. 관련 연구

1. 효율적인 비전 트랜스포머

ViT [16], DeiT [27]과 같은 초기 비전 트랜스포머는 모든 층에서 이미지 해상도가 동일하게 유지되는 특성을 가지고 있어서 CNN 아키텍처보다 상대적으로 크기가 크고 연산이 비효율적이라는 단점을 가지고 있다. 이를 개선하기 위해 비전 트랜스포머의 효율성을 향상시키는 연구가 많이 진행되고 있다 [21-25, 28]. LeViT [24]과 MobileViT [25]은 비전 트랜스포머와 CNN의 장점을 조화시켜 모델의 연산 효율성을 높인 모델이다. 두 모델은 일반적인 CNN 구조와 같이 층을 거침에 따라 이미지 해상도가 감소하고 채널 수가 많아지는 구조를 가지므로 초기 비전 트랜스포머보다 상대적으로 파라미터 수와 연산량이 적은 경량 모델이다.

2. 온-디바이스 AI를 위한 비전 태스크 벤치마킹

한편, 모델 아키텍처와 엣지 디바이스 특성에 따른 온-디바이스 AI 솔루션의 성능을 파악하기 위한 벤치마킹 연구가 많이 수행되고 있다. AI-Benchmark [29, 30]는 안드로이드 모바일 디바이스에서 비전 태스크 벤치마킹을 수행하는 애플리케이션으로 다양한 모바일 디바이스에서 수행된 벤치마킹 점수를 제공한다. 또한, 임베디드 GPU 플랫폼과 CPU 플랫폼에서 CNN 기반의 작은 모델을 데이터셋의 크기를 바꿔가며 학습시켜 이미지 분류 성능을 비교한 연구도 있다 [31]. 또 다른 연구 [32]에서는 임베디드 GPU 및 TPU 플랫폼에서 여러 CNN 모델들을 벤치마킹하여 추론 속도, 메모리 사용량, 전력 소모량, 그리고 정확도를 플랫폼별로 비교하였고 추가로 병렬성 활용에 따른 성능 향상 효과를 검증하기 위해 임베디드 GPU 플랫폼과 범용 GPU 시스템을 비교한 결과를 제시하고 있다.

그러나, 온-디바이스 AI를 위한 대부분의 비전 태스크 벤치마킹 대상은 CNN 기반 아키텍처로 한정되어 있다. 최신 비전 트랜스포머 모델의 높은 성능과 경량 비전 트랜스포머 아키텍처의 등장을 고려했을 때, 비전 트랜스포머의 온-디바이스 AI 적용을 위한 벤치마킹 연구는 여전히 부족한 상황이다. 여러 모바일 디바이스 상에서 응답 시간을 최소화하고자 양자화 및 프루닝을 적용한 후, 프로파일링을 통해 비전 트랜스포머의 추론 성능을 파악한 연구 [33]가 있으나, 연구에 사용된 비전 트랜스포머 아키텍처들은 효율성이 고려되지 않은 초기 아키텍처로 모델 압축 기법을 적용하여도 기존 CNN 아키텍처에 비해 느린 추론 속도를 보였다. 또한, 벤치마킹 과정에서 GPU 혹은 TPU 등의 하드웨어 가속기 사용에 제약이 있어 그 효과를 확인하지 못하였다는 한계가 있다.

III. 비전 트랜스포머 성능 평가

딥러닝을 활용한 컴퓨터 비전 분야에서, 이미지 분류 태스크는 다른 비전 태스크들의 기본이 되기 때문에 많은 벤

치마킹 연구들이 이미지 분류 태스크를 주로 다루고 있다 [31-33]. 따라서 본 연구도 성능 평가의 범위를 이미지 분류 태스크로 한정하여, 엣지 디바이스 상에서 경량 비전 트랜스포머의 성능을 평가하는 것을 목표로 한다. 이를 위해, 세 가지 비전 트랜스포머 아키텍처를 선정하였으며 두 가지 CNN 아키텍처를 대조군으로 선정하였다. 먼저 각 모델을 CPU 기반의 엣지 디바이스로 이식하여 성능을 측정하였다. 이후 각 모델에 양자화를 적용하여 양자화에 의한 각 모델의 성능 변화를 측정하였다. 또한, 측정된 각 모델의 성능을 분석하기 위해 연산 구성을 프로파일링하였다. 마지막으로 각 모델을 GPU와 TPU 기반의 엣지 디바이스로 이식하여 하드웨어 가속기에 의한 추론 속도 변화를 측정하였다.

1. 성능 평가 대상 비전 트랜스포머 아키텍처

본 연구는 성능 평가 대상 비전 트랜스포머 아키텍처로 DeiT, LeViT, MobileViT을 사용하였다. DeiT의 경우 비교적 초기 비전 트랜스포머 아키텍처로, 최초의 비전 트랜스포머 아키텍처인 ViT과 구조적으로 동일하다. ViT의 가장 핵심 아이디어는 입력 이미지를 여러 개의 이미지 패치로 구분하여 시퀀스를 만든 후 트랜스포머의 입력으로 사용하는 것이다. ViT은 먼저 입력 이미지를 컨볼루션 레이어를 사용하여 이미지 패치로 임베딩한다. 이후 각 이미지 패치가 위치에 대한 정보를 가질 수 있도록 positional encoding을 이미지 패치에 더해주며 마지막으로 이미지 분류를 위한 classification 토큰을 시퀀스에 이어 붙여 최종 시퀀스를 형성한다. ViT은 이렇게 만들어진 시퀀스를 여러 개의 transformer encoder block에 순차적으로 통과시켜 이미지 내의 특징들을 추출하고 최종적으로 FC 레이어로 이루어진 classification head에 classification 토큰을 통과시켜 이미지 분류 결과를 도출한다.

ViT에서 사용된 transformer encoder block은 NLP에서 사용된 것과 마찬가지로 어텐션 레이어, 그 뒤를 따르는 Multi-layer Perceptron (MLP) 레이어, 각각의 레이어 앞에 존재하는 정규화 레이어로 구성되어 있다. 어텐션 레이어와 MLP 레이어 중에서도 트랜스포머의 핵심은 어텐션 레이어인데, 어텐션 레이어는 각 패치 간 전역적인 정보를 학습하는 역할을 수행하며 ViT은 이러한 정보를 통해 추론 결과를 도출한다.

DeiT은 ViT이 공개되지 않은 데이터셋인 JFT-300M 데이터셋을 썼다는 것을 지적하며, 기본 ViT 모델과 동일한 아키텍처를 사용하면서도 지식 증류 (Knowledge Distillation)와 같은 학습 기법을 적용하여 ImageNet 훈련 데이터셋에 대한 학습만으로 모델의 정확도를 높였다. DeiT은 연산 효율성을 고려하여 설계된 경량 비전 트랜스포머는 아니지만 ViT에 비해 각 이미지 패치의 임베딩 차원이 더 작은 모델들을 제시하고 있으므로 초기 비전 트랜스포머 구조의 온-디바이스 AI 성능을 비교, 평가하기 위해 선택하였다.

LeViT의 기본 구조는 그림 1과 같다. LeViT은 저층부의 컨볼루션 레이어가 모델 성능에 긍정적 영향을 미친다는 사실 [24, 34]에 근거하여 이미지 패치화 과정에서 여러 층의 컨볼루션 레이어를 사용한다. LeViT은 모델을 크게 3개의 스테이지로 나누는데, 각 스테이지를 지날수록 이미지의 해상도는 낮추고 각 이미지 패치의 임베딩 차원은 증가시켜 아키텍처의 효율성을 향상시켰다. 또한, LeViT은 MLP 레이어의 은닉층의 차원 증가폭을 기존 4배에서 2배로 줄여 MLP 레이어의 파라미터 수와 연산량을 절감했다.

LeViT의 핵심 요소는 shrink 어텐션 레이어와 어텐션 바이어스이다. 각 스테이지 사이에서 임베딩 차원을 증가시키

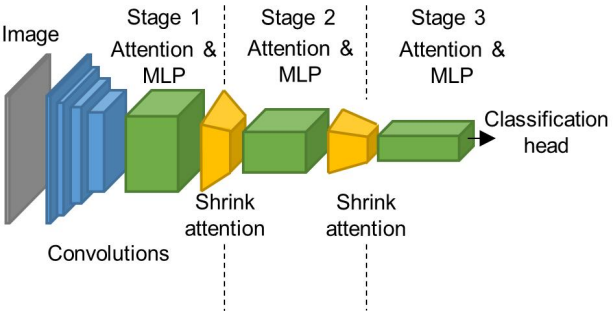
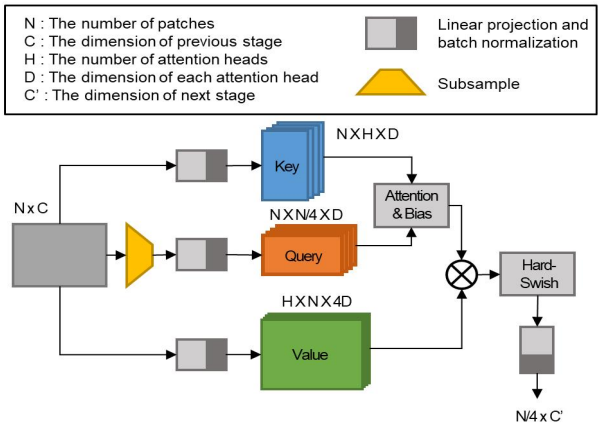
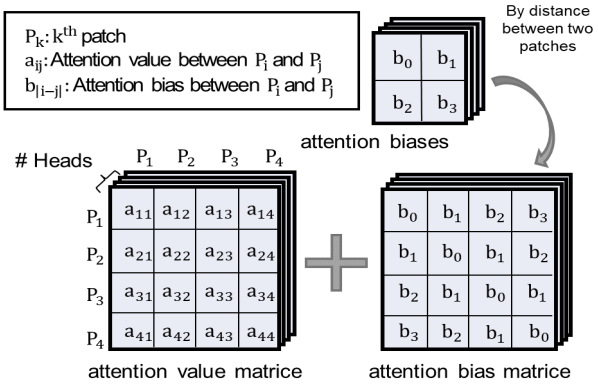


그림 1. LeViT 아키텍처
Fig. 1. LeViT architecture

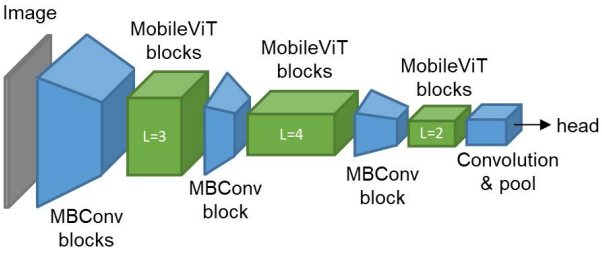


(a) Shrink attention

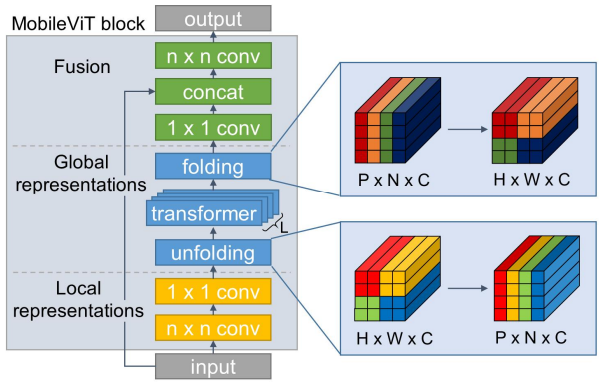


(b) Attention bias

그림 2. LeViT의 주요 방법론
Fig. 2. Main methodology of LeViT



(a) MobileViT architecture



(b) MobileViT Block

그림 3. MobileViT 아키텍처 및 MobileViT Block
Fig. 3. MobileViT architecture and MobileViT Block

고 이미지의 해상도를 낮추는 과정은 shrink 어텐션 레이어를 통해 수행된다. 그림 2 (a)는 shrink 어텐션 레이어의 구조를 나타낸다. 각 shrink 어텐션 레이어에서는 이미지 패치를 일정한 간격으로 샘플링하여 query 행렬을 계산하므로 정보의 손실이 발생할 수 있다. 이를 보완하기 위해 value 행렬의 차원을 기존 어텐션 레이어보다 큰 값으로 사용한다. LeViT은 모델에 이미지 패치들의 공간적 정보를 주입하기 위해 기존 비전 트랜스포머 아키텍처의 positional encoding 대신에, 그림 2 (b)에 나타난 것과 같이 각 어텐션 레이어마다 이미지 패치 간 거리에 따른 어텐션 바이어스를 어텐션 스코어에 더하는 방식을 사용한다.

MobileViT은 MobileNet [35]에서 사용되는 Inverted Residual Block (MBConv)과 트랜스포머 블록을 포함하는 MobileViT Block으로 구성된다. MBConv는 그림 3 (a)에 나타난 것과 같이 모델의 저층부와 각 MobileViT Block 사이에서 입력을 다운샘플링하며 주로 이미지의 지역적 특성을 추출하는 역할을 한다. 'Transformers as Convolutions' 방식이라고도 불리는 MobileViT Block은 이미지의 지역적 특성과 전역적 특성을 잘 혼합하기 위해 고안된 구조로 그림 3 (b)와 같이 세 부분으로 이루어져 있다. 첫 번째 부분에서 일반적인 컨볼루션 연산과 point-wise 컨볼루션 연산을 통해 입력 이미지의 지역적 특성을 추출한 후 두 번째 부분에서 트랜스포머 블록을 통해 입력 이미지의 전역적 특성을 추출한다. 마지막으로 최초로 입력된 이미지와 지역적 특성과 전역적 특성이 순차적으로 추출된 이미지를 이어 붙이고 컨볼루션 레이어를 사용하여 추출된 특성들을 융합함으로써 최종 활성화 맵을 얻는다. MobileViT Block에서 이미지의 전역적 특성은 각 이미지 패치의 동일 위치에 있는 픽셀들끼리 각각의 시퀀스를 이루며 트랜스포머 블록을 통과함으로써 얻어진다. MobileViT Block에서는 컨볼루션 레이어와 트랜스포머 블록을 혼용하기 위해 컨볼루션 레이어를 통과한 활성화 맵을 트랜스포머 블록의 입력으로 쓸 수 있도록 형태를 조작하는 unfolding 과정과 트랜스포머 블록을 통과한 활성화 맵을 컨볼루션 레이어의 입력으로 쓸 수 있도록 변환하는 folding 과정이 수행된다.

표 1. 성능 평가 대상 모델
Table 1. Models for performance evaluation

Category		Model	Parameters	ImageNet Top-1 Accuracy (%)
Transformer	Pure	DeiT-tiny	6M	74.5
		DeiT-small	22M	81.2
	Mixed (Conv + Transformer)	LeViT-128s	7.8M	76.6
		LeViT-128	9.2M	78.6
		LeViT-192	11M	80.0
		LeViT-256	19M	81.6
		LeViT-384	39M	82.6
		MobileViT-xxs	1.3M	69.0
		MobileViT-xs	2.3M	74.7
		MobileViT-s	5.6M	78.3
CNN		EfficientNet-Lite0	4.7M	75.1
		EfficientNet-Lite1	5.4M	76.7
		EfficientNet-Lite2	6.1M	77.6
		EfficientNet-Lite3	8.2M	79.8
		EfficientNet-Lite4	13M	81.5
		MobileNetV3-Large-100	5.4M	75.2
		MobileNetV3-Small-100	2.5M	67.4

본 논문에서 진행한 성능 평가는 모든 스케일의 비전 트랜스포머를 대상으로 하되, 약 8,700만 개의 파라미터를 가지는 DeiT의 Base 스케일은 그 크기로 인해 엣지 플랫폼에 적용하는 것이 적합하지 않아 평가 대상에서 제외하였다. 또한, 비전 트랜스포머와 기존 CNN 아키텍처 간 성능 비교를 위해 대표적인 경량 CNN 아키텍처인 EfficientNet-Lite [36]와 MobileNetV3 [15]에 대해서도 성능 평가를 수행하였다. 표 1은 평가 대상으로 사용한 각 비전 트랜스포머 아키텍처와 CNN 아키텍처의 스케일별 파라미터 수와 ImageNet 검증 데이터셋에 대한 추론 정확도를 나타낸다.

2. 사용 데이터셋

본 연구에서는 비전 트랜스포머의 추론 성능 파악을 위해 ImageNet 검증 데이터셋을 사용했다. ImageNet 데이터셋은 총 1,000개의 클래스를 가진다. 훈련 데이터셋은 약 128만 장의 이미지로 이루어져 있고, 성능 평가에 사용된 검증 데이터셋의 경우 클래스별 50장씩, 총 50,000장의 이미지로 이루어져 있다. 검증 데이터셋에 대한 정확도를 측정하기 위해 각 모델은 공식 Github 레포지토리 [24, 27] 및 timm 라

표 2. 엣지 디바이스별 사양 및 성능 평가 환경
Table 2. Platform specifications and performance evaluation environments by edge devices

	Raspberry Pi 4B	Jetson Nano dev. kit	Coral EdgeTPU USB type
Processor	Quad core ARM Cortex-A72 1.5GHz with VideoCore VI GPU	Quad core ARM Cortex-A57 1.43GHz (128 core Maxwell)	-
Memory (RAM)	2 GB	4 GB	-
OS	Raspberry Pi OS	L4T 32.7.1 (Linux kernel 4.9)	-
Python version	3.8.9	3.6.9	3.8.9
Framework	TFLite 2.9.2 (runtime)	Tensorflow 2.7.0 TensorRT 8.2.1.8	TFLite 2.9.2 (runtime) EdgeTPU runtime 14
Price	USD 45	USD 156	USD 60
Note	GPU not applicable	Docker container used	INT8 operations required

이브러리 [37]에서 제공하는 ImageNet 훈련 데이터셋으로 사전 학습된 가중치를 가져와 사용했다.

3. 사용 플랫폼 및 플랫폼별 모델 이식 방법

각 모델에 대한 성능 평가는 CPU, GPU, TPU 기반의 세 가지 엣지 디바이스 상에서 수행되었다. CPU 기반 엣지 디바이스로는 Raspberry Pi 4B [38], GPU 기반의 엣지 디바이스로는 NVIDIA CUDA GPU [39]를 탑재한 Jetson Nano dev. kit [40]을 각각 사용하였다. TPU 기반의 엣지 디바이스로는 Coral Edge TPU USB type (EdgeTPU) [41]을 Raspberry Pi 4B에 연결하여 사용했다. 표 2는 각 엣지 디바이스별 사양 및 성능 평가 환경을 나타낸다.

성능 평가에 사용된 각 모델은 우선 Tensorflow [42]로 구현된 후 각 엣지 디바이스에 이식되었다. 그림 4는 엣지 디바이스별 모델 이식 과정을 보여 준다. ImageNet 훈련 데이터셋으로 사전 학습된 각 모델은 기본적으로 32-bit 부동소수점 (FP32)의 정밀도를 지닌다. CPU 기반 엣지 디바이스에서는 FP32 모델을 TFLite [43] 플랫폼을 통해 ARM 아키텍처 CPU에 최적화된 모델로 변환한 후 이식하였다. NVIDIA CUDA GPU 기반 엣지 디바이스에서는 TF-TensorRT (TF-TRT) 모듈 [44]을 활용하여 모델을 GPU 활용에 최적화된 형태로 변환한 후 이식했다. CPU와 GPU 플랫폼에서는 FP32 모델을 그대로 이식하여 사용할 수 있는 반면, TPU 플랫폼에서는 모델을 8-bit 정수로 양자화하여 사용해야 한다. 따라서 TPU 플랫폼에서는 TFLite를 통해 변환된 FP32 모델을 8-bit로 양자화한 후 TPU에서 사용할 수 있는 형태로 컴파일하여 [45] 사용하였다.

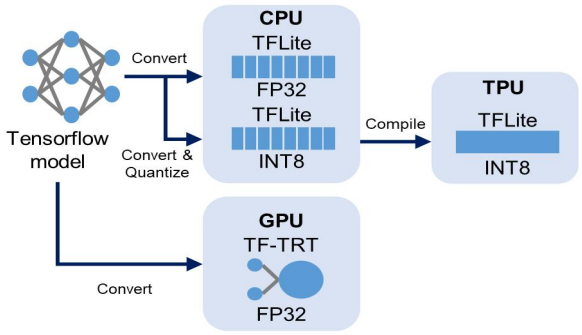


그림 4. 엣지 디바이스에 따른 모델 이식 워크플로우
Fig. 4. Model migration workflows by edge devices

4. 양자화 방법

양자화를 사용하면 모델의 크기를 감소시키고 응답 속도를 향상시킬 수 있지만, 모델의 정확도가 손실될 수 있다. 본 연구는 양자화된 비전 트랜스포머 모델에 대한 성능 평가를 위해 가장 보편적으로 사용되는 8-bit 정수 양자화를 적용하여 비전 트랜스포머의 양자화 전후 추론 속도와 정확도의 변화를 확인하였다.

모델 전체를 8-bit 정수로 양자화하기 위해서는 캘리브레이션 과정이 필요하다. 캘리브레이션 과정에서는 캘리브레이션 데이터셋을 사용하여 각 레이어의 양자화 스케일과 영점 (zero point)을 조정한다. 본 연구에서는 ImageNet 검증 데이터셋에서 각 클래스별로 이미지 1장씩을 임의로 선정하여 총 1,000장의 캘리브레이션 데이터셋을 구성한 후 모든 양자화 대상 모델에서 동일하게 사용하였다.

5. 성능 지표 선정

각 모델의 성능 평가는 ImageNet 검증 데이터셋에 대한 정확도 및 추론 시간을 측정하여 이뤄졌다. 이때, 각 모델에 따라 학습에 사용된 전처리 방식을 검증 이미지에 그대로 적용하여 정확도를 측정하였으며 추론 시간 측정 시에는 이미지 전처리를 위해 소요된 시간은 제외하고 모델의 연산이 수행된 시간만을 측정하였다. 또한, 각 비전 트랜스포머의 연산 구성을 프로파일링하기 위해 CPU 기반 엣지 디바이스에서 TFLite의 벤치마킹 툴을 사용하였다. Ⅳ장에서 사용되는 모든 성능 지표 및 프로파일링 결과는 평균값을 사용하였다.

Ⅳ. 실험 및 고찰

1. CPU 기반 플랫폼에서의 기본 성능 평가

본 연구에서는 먼저 CPU 기반의 대표적인 엣지 디바이스로 선정한 Raspberry Pi 4B 플랫폼 상에서 ImageNet 검증 데이터셋에 대한 추론 성능을 평가해 보았다. 각 모델의 추론 시간과 정확도를 기준으로 한 성능 분포는 그림 5와 같다.

실험 결과로 볼 때, 비전 트랜스포머 아키텍처 중에서는 LeViT이 정확도 대비 추론 속도가 가장 빠른 것을 알 수 있다. 그 이유는 그림 1과 같은 LeViT의 구조적 특성상 각

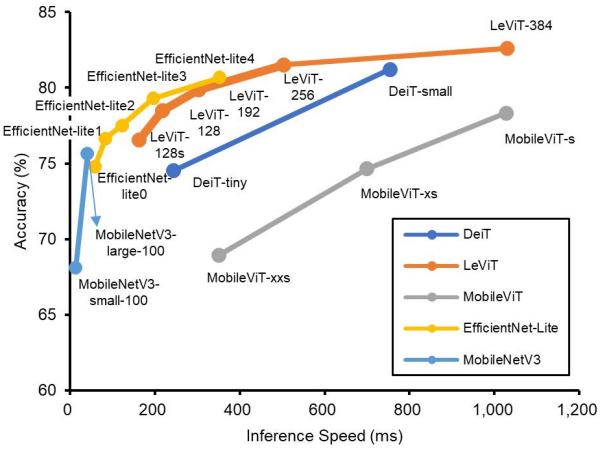


그림 5. Raspberry Pi 4B 상에서 모델 별 추론 성능 비교
Fig. 5. Performance comparison of models on Raspberry Pi 4B

표 3. LeViT-256의 주요 레이어의 스테이지별 실행시간 비율
Table 3. Response time occupancy of LeViT-256's key layers by stages

Stage \ Layer		Attention	MLP	Shrink Attention
Stage 1	Total	18.32 %	11.82 %	**8.7 %**
	Avg.	**4.58 %**	**2.95 %**	
Stage 2	Total	8.5 %	12.52 %	**5.81 %**
	Avg.	**2.13 %**	**2.50 %**	
Stage 3	Total	7.71 %	11.83 %	-
	Avg.	**1.93 %**	**2.37 %**	

스테이지를 지날수록 활성화 맵의 해상도는 감소하고 임베딩 차원은 증가함으로써 효율성이 향상된 결과로 볼 수 있으며, LeViT 아키텍처 중에서도 특히 LeViT-256 모델에서 이러한 구조의 효과가 잘 드러난다. 표 3은 LeViT-256 모델의 주요 레이어들의 스테이지별 실행시간 분포를 나타낸다. 각 실행시간 비율은 모델의 전체 실행시간에 대한 비율을 의미하며 스테이지별로 어텐션 레이어와 MLP 레이어의 수가 다르므로 스테이지 내에서 각 레이어의 총 실행시간 비율과 평균 실행시간 비율을 각각 제시하고 있다. 어텐션 레이어, MLP 레이어, shrink 어텐션 레이어 각각의 평균 실행시간은 스테이지를 지날수록 감소하는 것을 알 수 있다. 구체적으로는 어텐션 레이어와 MLP 레이어는 세 번째 스테이지에서 첫 번째 스테이지 대비 평균 실행시간이 각각 2.38배, 1.25배 감소하였고 shrink 어텐션 레이어의 경우 두 번째 스테이지에서 첫 번째 스테이지 대비 1.5배 감소하였다. 또한, LeViT의 구조적 효율성은 초기 비전 트랜스포머와 동일한 구조를 가진 DeiT 아키텍처와의 성능 차이로도 잘 드러나는데, LeViT-256 모델은 크기가 비슷한 DeiT-small 모델과 비슷한 정확도를 보이지만 훨씬 빠른 추론 속도를 나타냈으며, LeViT-128s 모델 역시 비슷한 크기의 DeiT-tiny 모델보다 높은 정확도를 보이면서 더 빠른 추론 속도를 나타내었다.

한편, MobileViT은 엣지 디바이스 성능 테스트 결과 초

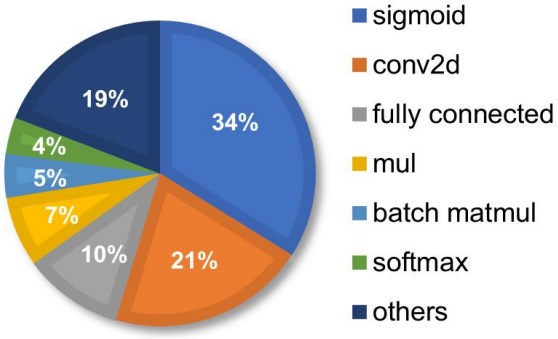


그림 6. MobileViT-s 모델의 주요 연산별 실행시간 비율
Fig. 6. Response time occupancy of main operations in MobileViT-s

기 비전 트랜스포머 아키텍처인 DeiT뿐만 아니라 모든 평가 대상 모델 중 가장 성능이 낮았다. MobileViT의 성능 감소 요인을 분석한 결과 그 원인은 크게 두 가지로 볼 수 있다. 첫 번째는 LeViT과 DeiT 아키텍처는 입력 이미지 해상도가 224 x 224인 반면 MobileViT의 입력 해상도는 256 x 256으로 LeViT과 DeiT보다 더 큰 해상도의 이미지를 처리해야 하기 때문이다. 그리고 두 번째는 MobileViT 아키텍처에서 사용되는 활성화 함수가 성능에 큰 악영향을 미치기 때문이다. MobileViT은 식 (1)과 같이 입력 x와 시그모이드 함수의 곱으로 계산되는 스위시 (Swish) 활성화 함수를 사용한다. MobileViT 아키텍처의 연산을 프로파일링한 결과, 그림 6과 같이 스위시 활성화 함수 계산에 사용되는 시그모이드 연산이 전체 추론 시간의 34%가 소요되었다. 반면, 식 (2)와 같이 시그모이드 연산을 사용하지 않은 하드 스위시 (Hard-swish) 활성화 함수가 사용된 LeViT 아키텍처에서는 활성화 함수 계산이 전체 추론 시간의 1.63%만을 차지하였다. 이 결과로, CPU 기반 엣지 디바이스에서는 MobileViT의 스위시 활성화 함수 계산이 추론 속도의 큰 병목 지점임을 알 수 있다.

$$\mathrm{Swish}(x) = \frac{x}{1-e^{-x}}, \tag{1}$$

$$\mathrm{Hardswish}(x) = \begin{cases} 0 & x \le -3 \\ x & x \ge 3 \\ \dfrac{x(x+3)}{6} & \text{otherwise} \end{cases}. \tag{2}$$

LeViT은 비전 트랜스포머 아키텍처 중에서는 가장 좋은 성능 지표를 나타내었으나, CNN 아키텍처인 EfficientNet-Lite와 유사한 정확도 수준의 스케일에서 추론 시간이 1.43-1.94배만큼 더 소요되었다. 또한, MobileViT은 아키텍처의 효율성을 고려하여 설계되었으나, 비교 대상 모델 중 가장 낮은 추론 정확도를 보였으며 MobileNetV3 아키텍처와 유사한 정확도 수준의 스케일에서 17-25배만큼 추론 시간이 더 소요되었다. 따라서, 기존 비전 트랜스포머를 구조

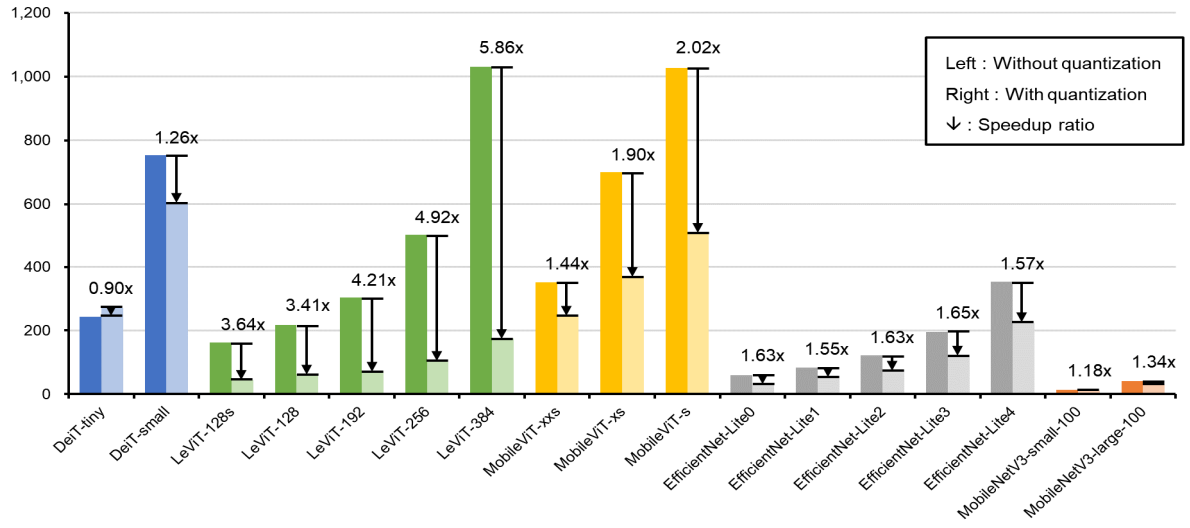


그림 7. 양자화 전후 각 모델의 추론 시간 및 양자화에 의한 추론 속도 향상율
Fig. 7. Model inference times and speedup ratios by quantization

표 4. 양자화 전과 후의 DeiT과 MobileViT 계열 모델들의 레이어 정규화의 모델 추론 시간 대비 실행시간 비율
Table 4. Response time occupancy of layer normalization in DeiT and MobileViT architectures with and without quantization

Model	Without quantization	With quantization
DeiT-tiny	12.86 %	51.79 %
DeiT-small	7.56 %	42.76 %
MobileViT-xxs	6.40 %	32.63 %
MobileViT-xs	4.97 %	30.59 %
MobileViT-s	5.97 %	34.14 %

적으로 효율화한 경량 아키텍처들도 타겟 엣지 디바이스 환경을 고려한 추가적인 최적화 없이는 온-디바이스 AI 솔루션으로 바로 적용하기 어려움을 알 수 있다.

2. CPU 기반 플랫폼에서의 양자화 적용 효과 실험

양자화를 통한 추가적 최적화가 비전 트랜스포머 아키텍처의 성능을 온-디바이스 AI 솔루션에 적용 가능한 수준으로 개선할 수 있는지 확인하기 위해 각 모델을 양자화하여 추론 속도와 정확도를 측정해 성능 변화를 확인하였다. 그림 7은 DeiT-tiny 모델을 제외한 모든 아키텍처 스케일에서 추론 속도가 향상되었음을 보여준다.

그러나, DeiT 아키텍처와 MobileViT 아키텍처는 LeViT 아키텍처에 비해 추론 속도 향상율이 비교적 낮았으며, 특히 DeiT-tiny 모델의 추론 시간은 양자화 이후 오히려 증가하였다. 아키텍처별로 양자화에 따른 추론 속도 향상율이 크게 달라지는 것은 정규화 기법의 연산 특성 때문으로 파악되었다. 배치 정규화 (Batch Normalization)를 사용하는 LeViT 아키텍처와 달리 DeiT과 MobileViT 아키텍처는 레이어 정규화를 사용한다. 표 4는 양자화 전후 DeiT 및 MobileViT 아키텍처의 각 모델 추론 시간 대비 레이어 정규화의 실행시간 비율을 나타낸다. 레이어 정규화는 모든 모델에서 양자화 후 실행시간이 증가하여 모델 추론 시간 중 차지하는 비율이 증가하였는데, 그림 8과 같이 레이어 정규화를 구성하는 연산 중에서도 특히 두 텐서 간 제곱 오차를 계산하는 squared difference와 sub 연산의 단일 실행시간이 크게 증가하였다. 레이어 정규화와 달리 배치 정규화는 추론 시에는 계산 효율성을 위해 평균과 표준 편차를 근삿값으로 고정하여 배치 정규화 계산을 선형 변환으로 대체하기 때문에 squared difference와 sub 연산이 사용되지 않는다. 본 실험 결과에 따르면, 레이어 정규화를 구성하는 주요 연산들이 양자화에 대해 최적화되지 않아 결과적으로 레이어 정규화가 두 아키텍처의 양자화에 의한 성능 향상에 병목이 되었다.

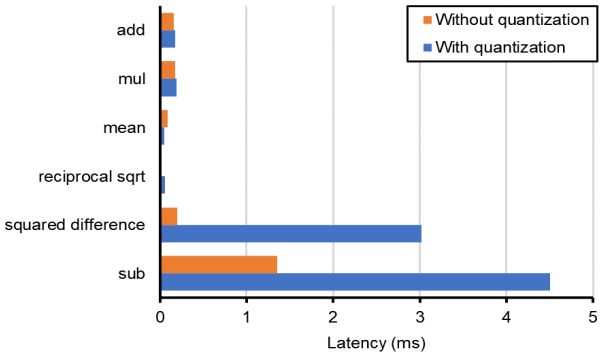


그림 8. 레이어 정규화를 구성하는 연산의 DeiT-small 모델 내 양자화 전후 단일 실행시간 변화
Fig. 8. Latency changes of operations composing the layer normalization in DeiT-small by quantization

또한, DeiT-tiny 모델이 양자화 후 추론 시간이 오히려 소폭 증가한 원인은 표 5에 제시한 DeiT 아키텍처의 각 레이어별 양자화 후 실행속도 향상율에서 찾을 수 있다. 양자

표 5. 양자화 후 DeiT의 주요 레이어별 실행속도 향상율
Table 5. Latency enhancement of main layers in DeiT architecture by quantization

Layers / Models	Attention	MLP	Layer normalization
DeiT-tiny	1.22x	2.41x	0.22x
DeiT-small	1.57x	2.65x	0.22x

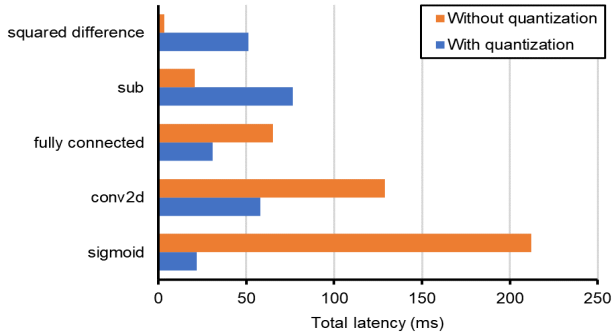


그림 9. MobileViT-s 모델의 양자화 후 주요 연산별 총 실행시간 변화
Fig. 9. Total latency changes of main operations in MobileViT-s by quantization

화에 의한 실행속도 향상율이 가장 큰 것은 MLP 레이어다. MLP 레이어는 임베딩 차원이 클수록 실행속도 향상 효과가 크므로 임베딩 차원이 비교적 큰 DeiT-small 모델은 MLP 레이어에 의한 실행속도 향상이 레이어 정규화에 의한 실행속도 저하보다 컸으나, 임베딩 차원이 비교적 작은 DeiT-tiny 모델은 MLP 레이어에 의한 실행속도 향상보다 레이어 정규화에 의한 실행속도 저하가 더 컸기 때문에 오히려 추론 시간이 더 증가한 것으로 판단된다.

반면에, 레이어 정규화를 사용하는 두 아키텍처 중에서도 MobileViT 아키텍처의 추론 속도 향상율은 DeiT 아키텍처보다 높게 나타났고, 양자화를 적용한 후의 추론 속도는 DeiT 아키텍처와 큰 차이가 없었다. 그 이유는 양자화 전 성능의 가장 큰 병목이었던 시그모이드 연산의 실행시간이 양자화 이후 크게 감소했기 때문이다. 그림 9는 MobileViT-s를 구성하는 연산들의 양자화 전후 총 실행시간의 비교 결과를 나타낸다. 그림 9와 같이 MobileViT-s 모델의 시그모이드 연산의 총 실행시간은 큰 폭으로 감소하여, 모델 추론 시간 대비 기존 36%에서 양자화 이후 약 10%로 감소하였다.

LeViT 아키텍처는 양자화에 의한 추론 속도 향상 측면에서도 가장 좋은 결과를 보였다. 이는 양자화 후 성능 향상에 병목이 되는 레이어 정규화 대신 배치 정규화를 사용하여 비교적 양자화 친화적인 연산 구성을 가지고 있기 때문이다. LeViT 아키텍처는 스케일이 증가함에 따라 최소 3.41배에서 최대 5.86배까지 추론 속도 향상율도 함께 증가하는 경향을 보였으나 LeViT-128 모델은 LeViT-128s 모델보다 추론 속도 증가율이 더 낮았다. 그 원인을 찾기 위해 두 모델에서의 주요 레이어별 속도 변화를 추가로 분석해 본 결과는 표 6과 같다. 기본적으로 LeViT-128 모델과 LeViT-128s 모델의 각 스테이지별 임베딩 차원은 동일하지만 LeViT-128 모델의 어텐션 레이어의 수가 더 많다. LeViT-128 모델과 LeViT-128s 모델에서는 각 레이어 중 어텐션 레이어의 양자화 후 추론 속도 증가율이 가장 낮은데, LeViT-128 모델의 경우 어텐션 레이어의 수가 많아 전체 실행시간 중 어텐션 레이어가 차지하는 비율이 LeViT-128s 모델보다 크기 때문에 양자화에 의한 추론 속도 향상 효과가 상대적으로 감소했음을 알 수 있다.

표 6. LeViT 아키텍처의 주요 레이어별 양자화 후 추론 속도 증가율
Table 6. Inference speed improvement of main layers in LeViT after quantization

Layers	LeViT-128s	LeViT-128
Attention	2.5x	2.5x
Shrink attention	3.4x	3.5x
MLP	5.6x	5.4x
Convolutions	3.9x	4.0x

표 7. 비전 트랜스포머의 양자화 후 정확도 변화
Table 7. Accuracy degradation of vision transformers after quantization

Model	With quantization	Without quantization	Accuracy drop
DeiT-tiny	74.5	69.5	-5.0
DeiT-small	81.2	57.9	-23.3
LeViT-128s	76.5	53.3	-23.2
LeViT-128	78.5	72.4	-6.1
LeViT-192	80.0	76.7	-3.3
LeViT-256	81.5	80.0	-1.5
LeViT-384	82.6	81.5	-1.1
MobileViT-xxs	68.9	0.2	-68.7
MobileViT-xs	74.6	0.1	-74.5
MobileViT-s	78.3	16.0	-62.3

표 7은 각 비전 트랜스포머 모델의 양자화 전후 추론 정확도를 비교하여 제시한다. 전통적인 CNN 기반 아키텍처에서는 모델의 크기가 클수록 양자화에 의한 정확도 손실이 감소하지만, 비전 트랜스포머에서는 그러한 경향이 나타나지 않았다. 레이어 정규화와 depth-wise 컨볼루션은 양자화 시 상당한 정확도 손실을 유발하는 것으로 알려져 있으며 [20, 46], 본 실험 결과에 나타난 DeiT과 MobileViT 아키텍처의 정확도 하락은 이러한 사실을 잘 보여 준다. 반면에 레이어 정규화와 depth-wise 컨볼루션을 모두 사용하지 않는 LeViT 아키텍처는 실험 결과에서 양자화 후 정확도 측면에서도 다른 두 아키텍처와 달리 큰 손실 없이 가장 좋은 성능을 보였으며 모델의 스케일이 커질수록 정확도의 손실이 점점 감소하여 LeViT-256 모델과 LeViT-384 모델에서는 각 1.5%, 1.1%만큼의 정확도 손실만이 발생하였다.

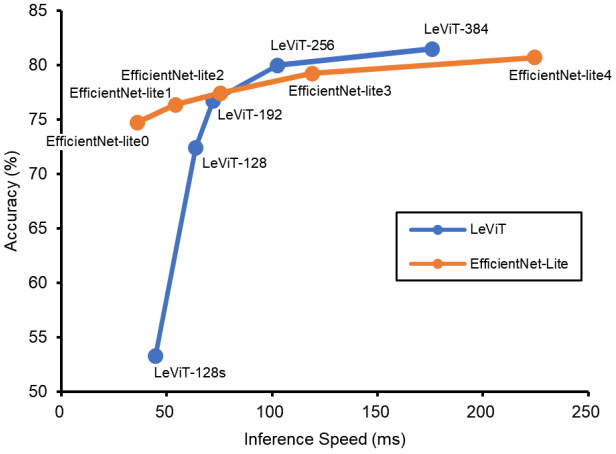


그림 10. 양자화된 LeViT과 EfficientNet-Lite의 Raspberry Pi 4B에서의 추론 성능 비교
Fig. 10. Performance comparison of quantized LeViT and EfficientNet-Lite on Raspberry Pi 4B

표 8. Jetson Nano에서의 각 모델의 추론 속도
Table 8. Model inference speed on Jetson Nano

Model	Raspberry Pi inference speed (ms)	Jetson Nano inference speed (ms)	Speedup ratio
DeiT-tiny	243	26	9.31x
DeiT-small	754	28	26.48x
LeViT-128s	162	81	1.98x
LeViT-128	217	99	2.18x
LeViT-192	303	98	3.07x
LeViT-256	504	99	5.07x
LeViT-384	1,030	111	9.21x
MobileViT-xxs	351	75	4.68x
MobileViT-xs	699	90	7.72x
MobileViT-s	1,027	107	9.59x
EfficientNet-Lite0	59	5	11.66x
EfficientNet-Lite1	84	5	16.21x
EfficientNet-Lite2	123	5	22.58x
EfficientNet-Lite3	196	6	33.99x
EfficientNet-Lite4	353	7	52.6x
MobileNetV3-Large-100	14	8	1.76x
MobileNetV3-Small-100	41	17	2.4x

CPU 기반의 엣지 디바이스 상에서 양자화를 적용한 결과를 종합적으로 볼 때, 평가 대상인 비전 트랜스포머 아키텍처 중 LeViT 아키텍처가 양자화 적용 후 추론 속도 향상과 추론 정확도 하락의 최소화 측면에서 가장 우수한 모델이라고 할 수 있다. 그림 10과 같이 LeViT-256 모델과 LeViT-384 모델은 각각 정확도 손실을 최소화하기 위해 양자화 인식 훈련을 사용한 CNN 모델인 EfficientNet-Lite3 모델과 EfficientNet-Lite4 모델보다도 양자화 후 더 높은 추론 정확도와 빠른 속도를 보이며 온-디바이스 AI 솔루션에 적용하기 충분한 모델임을 보여주고 있다.

3. 하드웨어 가속기 사용 플랫폼에서의 성능평가

다음 실험으로는 양자화되지 않은 모델을 그림 4와 같이 TF-TRT 모듈을 활용하여 GPU 장치에 적합하도록 변환한 모델의 성능을 Jetson Nano 상에서 비교, 평가해 보았다. 표 8은 기존 CPU 기반 엣지 디바이스인 Raspberry Pi 4B 상에서 최적화되지 않은 모델과 Jetson Nano에서 GPU 사용에 최적화된 모델의 추론 속도를 비교한 결과를 나타낸다.

GPU 기반의 엣지 디바이스에서는 모든 모델이 약 2배 이상의 추론 속도 증가율을 나타내었다. 특히 DeiT과 MobileViT 아키텍처는 CPU 기반 엣지 디바이스에서 정확도 대비 느린 추론 속도를 보였으나 GPU 기반 엣지 디바이스에서는 TF-TRT 모듈의 최적화를 기반으로 LeViT 아키텍처에 근접하거나 더욱 빠른 추론 속도를 보였다. 이 실험에서는 양자화를 적용하지 않았으므로, 추론 정확도는 그림 5에 제시한 결과와 동일하다. 그런 측면에서 본다면, GPU 최적화가 적용된 플랫폼 상에서는 비전 트랜스포머 모델 중 DeiT 모델이 CNN 아키텍처와 비교할 때 크게 뒤지지 않은 추론 정확도를 보이면서도 가장 추론 속도가 빠른 것을 알 수 있다. DeiT 아키텍처의 TF-TRT 모듈에 의한 최적화 비율은 두 아키텍처에 비해 높은 수치로 나타났다. 예를 들어, DeiT-small 모델은 전체 연산 중 74.81%에 해당하는 연산이 최적화되어 총 50개의 TRT 엔진으로 통합된 반면, MobileViT-s 모델은 전체 연산 중 45.36%의 연산만이 최적화되어 총 43개의 TRT 엔진으로 통합되었으며 LeViT-384 모델은 18.59%의 연산만이 최적화되어 48개의 TRT 엔진으로 통합되었다.

마지막으로, CPU 기반 엣지 디바이스에 EdgeTPU 장치를 연동하여 각 모델의 추론 속도를 평가해 보았으며 그 결과는 표 9와 같다. 그 결과로 볼 때, MobileViT 아키텍처만

표 9. EdgeTPU에서의 각 모델의 추론 속도
Table 9. Model inference speed on EdgeTPU

Model	Raspberry Pi inference speed (ms)	EdgeTPU inference speed (ms)	Speedup ratio
DeiT-tiny	270	350	0.77x
DeiT-small	600	733	0.82x
LeViT-128s	44	56	0.80x
LeViT-128	63	85	0.75x
LeViT-192	72	73	0.99x
MobileViT-xxs	243	146	1.67x
MobileViT-xs	368	225	1.64x
MobileViT-s	509	326	1.56x
EfficientNet-Lite0	36	6	6.12x
EfficientNet-Lite1	54	8	6.65x
EfficientNet-Lite2	75	12	6.49x
EfficientNet-Lite3	119	20	5.88x
EfficientNet-Lite4	225	43	5.28x
MobileNetV3-Large-100	12	30	0.40x
MobileNetV3-Small-100	30	53	0.58x

이 추론 속도 향상 효과를 얻었고 다른 비전 트랜스포머 모델들은 오히려 추론 속도가 느려졌음을 알 수 있다.

EdgeTPU 사용으로 인한 성능 이득을 최대화하기 위해서는 모델을 구성하는 모든 연산을 EdgeTPU로 오프로딩(Offloading)해야 한다. 연산의 오프로딩 여부는 모델 컴파일 시에 결정되며, 컴파일 중 EdgeTPU에서 지원되지 않는 연산이 있거나 지원되는 연산이지만 연산의 메모리 사용량 또는 연산이 수행되는 텐서의 차원 수와 같은 제약 사항을 충족시키지 못하면 해당 연산은 CPU에 할당되고 그 이전까지의 연산들은 EdgeTPU에 오프로딩될 수 있도록 서브 그래프로 묶인다. 컴파일 후, 각 모델은 구성하는 연산들의 EdgeTPU 오프로딩 여부에 따라 여러 개의 서브 그래프로 나뉠 수 있는데 이 경우 CPU와 EdgeTPU 간 데이터 전송에 의한 추가 오버헤드가 발생하므로 EdgeTPU 사용에 의한 성능 향상 효과가 크게 저하된다. 본 연구에서는, 사용된 각 비전 트랜스포머 아키텍처를 EdgeTPU로 이식하기 위해 메모리 제약을 만족시키지 못하는 큰 연산을 여러 개의 작은 연산들로 분할하여 수행하도록 각 모델의 구현을 수정하였으나, 그럼에도 지원되지 않는 연산으로 인하여 각 모델은 여러 개의 서브 그래프로 나뉘어 컴파일되었으며 이로 인해 성능 향상 효과가 크게 저하되었다.

MobileViT은 유일하게 추론 속도가 향상되었는데, 이러한 결과는 EdgeTPU의 파라미터 캐싱과 관련이 있다. EdgeTPU는 내부 8 MB 크기의 SRAM에 연산을 위한 공간을 제외한 나머지 공간에 모델의 파라미터를 캐싱하여 추론 성능을 극대화한다. 만약 연산에 사용될 파라미터가 캐시되어 있지 않다면 이를 교체하기 위한 추가적인 오버헤드가 발생하는데, MobileViT 아키텍처는 LeViT, DeiT에 비해 파라미터 수가 적어 동시에 캐시될 수 있는 서브 그래프의 수가 많으므로 상대적으로 캐시 친화적이라고 할 수 있으며 따라서 파라미터 캐시 교체에 의한 오버헤드가 적기 때문에 성능이 향상된 것으로 판단된다. 이러한 결과로 볼 때, EdgeTPU 사용을 통한 성능 이득을 최대화하기 위해서는 모델의 높은 EdgeTPU 오프로딩 (Offloading) 비율과 함께 캐시 친화적인 구조 역시 필요함을 알 수 있다. 한편, CNN 기반의 EfficientNet-Lite 아키텍처는 모든 연산이 EdgeTPU에서 실행되어 추론 속도가 큰 폭으로 증가하였다.

V. 결 론

본 연구는 비전 트랜스포머의 온-디바이스 적용 가능성을 확인하는 것을 목표로 ImageNet 검증 데이터셋에 대한 DeiT, LeViT, MobileViT 아키텍처의 이미지 분류 성능을 다양한 엣지 디바이스에서 평가하였다.

CPU 기반 엣지 디바이스에서는 구조적 효율성과 엣지 친화적 연산 구성을 기반으로 LeViT 아키텍처가 비전 트랜스포머 아키텍처 중 가장 좋은 추론 속도 대비 정확도를 보였으며 양자화 적용 시 엣지 친화적 CNN 아키텍처인 EfficientNet-Lite보다도 높은 정확도와 빠른 추론 속도를 나타냈다. 반면 DeiT, MobileViT 아키텍처는 병목이 되는 연산이 있어 추론 속도가 느렸다. GPU 기반 엣지 디바이스에서는 전반적으로 비전 트랜스포머 아키텍처의 추론 속도가 CNN 기반 아키텍처보다 느렸으나, DeiT-small 모델은 GPU 장치에 최적화되어 동작할 수 있는 연산의 비율이 높아 CPU 대비 성능이 크게 향상되어 실시간 응용까지 가능한 수준의 추론 속도를 나타내었다. 마지막으로, TPU 기반 엣지 디바이스에서는 모델 내 연산의 TPU 오프로딩 여부와 파라미터 캐싱이 모델 성능에 가장 큰 영향을 미쳤으며 캐시 친화적 구조를 지닌 MobileViT만 성능 향상 효과를 얻었고 파라미터 수가 많은 DeiT, LeViT 아키텍처는 CPU 기반 엣지 디바이스 대비 오히려 성능이 저하됐다. 이러한 성능 평가 결과들은 비전 트랜스포머 아키텍처의 성능이 각 하드웨어 플랫폼에 의해 최적화될 수 있는 연산 구성과 비율에 의존적임을 나타내는 동시에, 타겟 하드웨어 플랫폼에 따른 아키텍처 및 연산 구성의 추가적 최적화를 통한 비전 트랜스포머의 온-디바이스 AI 솔루션 적용 가능성을 함께 제시한다.

References

[1] W. Ahmad, A. Rasool, A. R. Javed, T. Baker, Z. Jalil, “Cyber Security in Iot-based Cloud Computing: A Comprehensive Survey,” Electronics, Vol. 11, No. 1, pp. 16, 2022.

[2] M. Ham, J. Moon, G. Lim, J. Jung, H. Ahn, W. Song, S. Woo, P. Kapoor, D. Chae, G. Jang, Y. Ahn, J. Lee, “NNStreamer: Efficient and Agile Development of On-Device AI Systems,” Proc. of the IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), pp. 198 - 207, 2021.

[3] S. Dhar, J. Guo, J. Liu, S. Tripathi, U. Kurup, M. Shah, “A Survey of On-device Machine Learning: An Algorithms and Learning Theory Perspective,” ACM Transactions on Internet of Things, Vol. 2, No. 3, pp. 1-49, 2021.

[4] D. Kong, “Science Driven Innovations Powering Mobile Product: Cloud AI vs. Device AI Solutions on Smart Device,” arXiv preprint arXiv:1711.07580, 2017.

[5] “Nvidia Embedded Systems for Next-Gen Autonomous Machines,” NVIDIA. [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/. [Accessed: 30-Jan-2023].

[6] “Edge TPU - run Inference at the Edge | Google Cloud,” Google. [Online]. Available: https://cloud.google.com/edge-tpu. [Accessed: 30-Jan-2023].

[7] W. Vijitkunsawat, P. Chantngarm, “Comparison of Machine Learning Algorithm’s on Self-driving Car Navigation Using Nvidia Jetson Nano,” Proc. of the International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), pp. 201 - 204, 2020.

[8] A. Basulto-Lantsova, J. A. Padilla-Medina, F. J. Perez-Pinal, A. I. Barranco-Gutierrez, “Performance comparative of OpenCV Template Matching method on Jetson TX2 and Jetson Nano Developer Kits,” Proc. of the Annual Computing and Communication Workshop and Conference (CCWC), pp. 0812 - 0816, 2020.

[9] K. Alibabaei, E. Assunção, P. D. Gaspar, V. N. Soares, J. M. Caldeira, “Real-Time Detection of Vine Trunk for Robot Localization Using Deep Learning Models Developed for Edge TPU Devices,” Future Internet, Vol. 14, No. 7, pp. 199, 2022.

[10] Y. H. Tseng, S. S. Jan, “Combination of Computer Vision Detection and Segmentation for Autonomous Driving,” Proc. of the IEEE/ION Position, Location and Navigation Symposium (PLANS), pp. 1047 - 1052, 2018.

[11] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, K. Zieba, “End to end Learning for Self-driving Cars,” arXiv preprint arXiv:1604.07316, 2016.

[12] D. N. N. Tran, H. H. Nguyen, L. H. Pham, J. W. Jeon, “Object Detection with Deep Learning on Drive PX2,” Proc. of the IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), pp. 1 - 4, 2020.

[13] K. He, X. Zhang, S. Ren, J. Sun, “Deep Residual Learning for Image Recognition,” Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770 - 778, 2016.

[14] M. Tan, Q. Le, “Efficientnet: Rethinking Model Scaling for Convolutional Neural Networks,” Proc. of the International Conference on Machine Learning, pp. 6105 - 6114, 2019.

[15] A. Howard, M. Sandler, G. Chu, L. C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, H. Adam, “Searching for Mobilenetv3,” Proc. of the IEEE/CVF International Conference on Computer Vision, pp. 1314 - 1324, 2019.

[16] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale,” arXiv preprint arXiv:2010.11929, 2020.

[17] M. Wortsman, G. Ilharco, S. Y. Gadre, R. Roelofs, R. Gontijo-Lopes, A. S. Morcos, H. Namkoong, A. Farhadi, Y. Carmon, S. Kornblith, L. Schmidt, “Model Soups: Averaging Weights of Multiple Fine-tuned Models Improves Accuracy Without Increasing Inference Time,” Proc. of the International Conference on Machine Learning, pp. 23965 - 23998, 2022.

[18] X. Zhai, A. Kolesnikov, N. Houlsby, L. Beyer, “Scaling Vision Transformers,” Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12104 - 12113, 2022.

[19] H. Bao, L. Dong, S. Piao, F. Wei, “Beit: Bert Pre-training of Image Transformers,” arXiv preprint arXiv:2106.08254, 2021.

[20] Z. Liu, Y. Wang, K. Han, W. Zhang, S. Ma, W. Gao, “Post-training Quantization for Vision Transformer,” Advances in Neural Information Processing Systems, Vol. 34, pp. 28092 - 28103, 2021.

[21] Y. Tang, K. Han, Y. Wang, C. Xu, J. Guo, C. Xu, D. Tao, “Patch Slimming for Efficient Vision Transformers,” Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12165 - 12174, 2022.

[22] L. Song, S. Zhang, S. Liu, Z. Li, X. He, H. Sun, J. Sun, N. Zheng, “Dynamic Grained Encoder for Vision Transformers,” Advances in Neural Information Processing Systems, Vol. 34, pp. 5770 - 5783, 2021.

[23] B. Chen, P. Li, B. Li, C. Li, L. Bai, C. Lin, M. Sun, J. Yan, W. Ouyang, “Psvit: Better Vision Transformer Via Token Pooling and Attention Sharing,” arXiv preprint arXiv:2108.03428, 2021.

[24] B. Graham, A. El-Nouby, H. Touvron, P. Stock, A. Joulin, H. Jégou, M. Douze, “Levit: A Vision Transformer in Convnet’s Clothing for Faster Inference,” Proc. of the IEEE/CVF International Conference on Computer Vision, pp. 12259 - 12269, 2021.

[25] S. Mehta, M. Rastegari, “Mobilevit: Light-weight, General-purpose, and Mobile-friendly Vision Transformer,” arXiv preprint arXiv:2110.02178, 2021.

[26] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, L. Fei-Fei, “Imagenet Large Scale Visual Recognition Challenge,” International Journal of Computer Vision, Vol. 115, pp. 211 - 252, 2015.

[27] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, H. Jégou, “Training Data-efficient Image Transformers & Distillation Through Attention,” Proc. of the International Conference on Machine Learning, pp. 10347 - 10357, 2021.

[28] B. Pan, R. Panda, Y. Jiang, Z. Wang, R. Feris, A. Oliva, “IA-RED2: Interpretability-Aware Redundancy Reduction for Vision Transformers,” Advances in Neural Information Processing Systems, Vol. 34, pp. 24898 - 24911, 2021.

[29] A. Ignatov, R. Timofte, W. Chou, K. Wang, M. Wu, T. Hartley, L. V. Gool, “Ai Benchmark: Running Deep Neural Networks on Android Smartphones,” Proc. of the European Conference on Computer Vision (ECCV) Workshops, pp. 0 - 0, 2018.

[30] A. Ignatov, R. Timofte, A. Kulik, S. Yang, K. Wang, F. Baum, M. Wu, L. Xu, L. V. Gool, “Ai Benchmark: All About Deep Learning on Smartphones in 2019,” Proc. of the IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), pp. 3617 - 3635, 2019.

[31] A. A. Süzen, B. Duman, B. Şen, “Benchmark Analysis of Jetson tx2, Jetson Nano and Raspberry pi Using Deep-cnn,” Proc. of the International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), pp. 1 - 5, 2020.

[32] P. Kang, J. Jo, “Benchmarking Modern Edge Devices for Ai Applications,” IEICE TRANSACTIONS on Information and Systems, Vol. 104, No. 3, pp. 394 - 403, 2021.

[33] X. Wang, L. L. Zhang, Y. Wang, M. Yang, “Towards Efficient Vision Transformer Inference: A First Study of Transformers on Mobile Devices,” Proc. of the Annual International Workshop on Mobile Computing Systems and Applications, pp. 1 - 7, 2022.
[34] T. Xiao, M. Singh, E. Mintun, T. Darrell, P. Dollár, R. Girshick, “Early Convolutions Help Transformers See Better,” Advances in Neural Information Processing Systems, Vol. 34, pp. 30392 - 30400, 2021.
[35] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, “Mobilenetv2: Inverted Residuals and Linear Bottlenecks,” Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4510 - 4520, 2018.
[36] “Higher Accuracy on Vision Models with EfficientNet-Lite,” The TensorFlow Blog. [Online]. Available: https://blog.tensorflow.org/2020/03/higher-accuracy-on-vision-models-with-efficientnet-lite.html. [Accessed: 30-Jan-2023].
[37] R. Wightman, PyTorch Image Models. GitHub, 2019. doi: 10.5281/zenodo.4414861.
[38] Raspberry Pi, “Raspberry pi 4 Model B Specifications,” Raspberry Pi. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/. [Accessed: 30-Jan-2023].
[39] “CUDA Toolkit Documentation,” CUDA Toolkit Documentation v12.0 - landing 12.0 documentation, 09-Dec-2022. [Online]. Available: https://docs.nvidia.com/cuda/index.html. [Accessed: 30-Jan-2023].
[40] “Jetson Nano Developer Kit,” NVIDIA Developer, 28-Sep-2022. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit. [Accessed: 30-Jan-2023].
[41] “EdgeTPU USB Accelerator,” Coral. [Online]. Available: https://coral.ai/products/accelerator/. [Accessed: 30-Jan-2023].
[42] “Tensorflow,” TensorFlow. [Online]. Available: https://www.tensorflow.org/. [Accessed: 30-Jan-2023].
[43] “Tensorflow Lite,” TensorFlow. [Online]. Available: https://www.tensorflow.org/lite/guide. [Accessed: 30-Jan-2023].
[44] “TensorFlow-TensorRT (TF-TRT),” NVIDIA Documentation Center. [Online]. Available: https://docs.nvidia.com/deeplearning/frameworks/tf-trt-user-guide/index.html. [Accessed: 30-Jan-2023].
[45] “Edge Tpu Compiler,” Coral. [Online]. Available: https://coral.ai/docs/edgetpu/compiler/. [Accessed: 30-Jan-2023].
[46] T. Sheng, C. Feng, S. Zhuo, X. Zhang, L. Shen, M. Aleksic, “A Quantization-friendly Separable Convolution for Mobilenets,” Proc. of the Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications (EMC2), pp. 14 - 18, 2018.

Minha Lee (이 민 하)

2022 Mechanical and Information Engineering from University of Seoul (B.S.)
2022~Mechanical and Information Engineering/Smart Cities from University of Seoul (M.S.)

Field of Interests: On-device AI, Parallel Computing
Email: lmh970329@gmail.com

Seongjae Lee (이 성 재)

2019 Mechanical and Information Engineering from University of Seoul (B.S.)
2019~Mechanical and Information Engineering/Smart Cities from University of Seoul (Ph.D. Candidate)

Field of Interests: On-device AI, Parallel Computing, Machine Fault Diagnosis
Email: seongjae.lee.1118@gmail.com

Taehyoun Kim (김 태 현)

1994 Computer Engineering from Seoul National University (B.S.)
1996 Computer Engineering from Seoul National University (M.S.)
2001 Electrical and Computer Engineering from Seoul National University (Ph.D.)

Career:
2001~2005 R&D Manager, GCT Research, Inc.
2005~ Professor, Dept. of Mechanical & Information Eng./Smart Cities, University of Seoul

Field of Interests: Embedded Real-Time Systems, Edge AI Solution, Industrial Automation
Email: thkim@uos.ac.kr