엣지 컴퓨팅 환경에서 FastSAM-s 모델의 구현 및 성능비교에 관한 연구

김동우[1], 황준석[1], 강인경[1]
[1]동서울대학교 컴퓨터 소프트웨어과 학부생
gimdongu19@gmail.com, hjstone3835@gmail.com, ingyeong7161@gmail.com

A Study on the Implementation and Performance Comparison of the FastSAM-s Model in Edge Computing Environments

Dong-Woo Kim[1], Joon-Suk Hwang[1], In-Gyeong Kang[1]
[1]Dept. of Computer Software, Dong-Seoul University

요 약

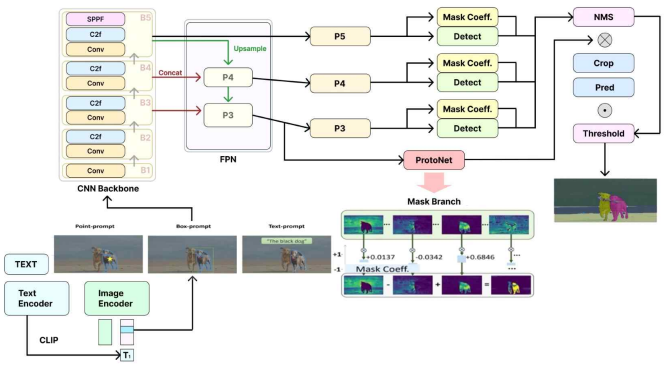
본 연구는 FastSAM 모델의 기존 PyTorch(.pt)형식을 엣지 컴퓨팅(Edge Computing) 환경에 적합하도록 8-bit 정수 양자화를 적용한 TensorFlow Lite(.tflite)형식으로 변환하여 Raspberry Pi 5에 포팅(Porting)하였다. Raspberry Pi 5 자체의 CPU만 사용한 경우와 Coral Edge TPU 가속기를 붙여 사용한 두 가지 경우에 대해 기존 PyTorch 형식을 기준으로 추론성능 비교평가를 수행한 결과, 객체 분할 정확도를 측정하는 지표인 IoU(Intersection over Union)는 기존의 PyTorch(.pt)형식과 비교하여 약 3배 떨어졌으나, FPS(Frame Per Second) 측정치는 단독 CPU 환경에서는 약 3배, Coral Edge TPU 장착 환경에서는 약 5배 향상되었음을 확인하였다.

1. 서론

범용 컴퓨터 환경은 높은 연산 자원을 바탕으로 대규모 AI 모델을 안정적으로 실행할 수 있으나, 휴대성이나 저전력을 필요로 하는 엣지 AI 분야에서는 한계가 존재한다. 이러한 제약을 극복하기 위해 최근에는 엣지 컴퓨팅에서 인공지능 모델을 구동하는 연구가 활발히 이루어지고 있다. 그러나 엣지 컴퓨팅 환경에서는 연산 성능의 한계로 인해 모델 크기 축소 및 경량화가 필수적이며, 이 과정에서 성능의 저하가 불가피하다. 본 연구에서는 데스크톱 환경에서 FastSAM PyTorch 형식의 모델을 수행한 결과를 기준으로, Raspberry Pi 5 환경에서 CPU만을 이용하여 수행한 TensorFlow Lite 변환 모델의 경우와 엣지 AI 가속기인 Coral Edge TPU을 사용한 Edge TPU 최적화 모델의 성능을 비교평가 하였다. 성능 평가 측정치는 객체 분할 정확도를 나타내는 IoU와 처리 속도를 나타내는 FPS를 지표로 수행하였다.

2. FastSAM 모델

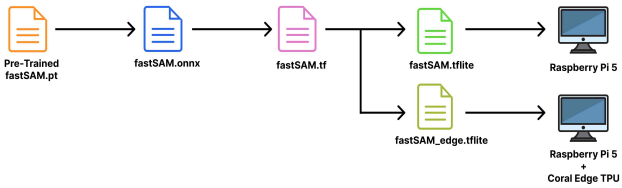
FastSAM은 SAM(Segment Anything Model)의 높은 계산 비용 문제를 해결하기 위해 제안된 경량 대안 모델로, YOLOv8-seg 구조를 기반으로 한다[1]. 본 모델은 객체 분할과 프롬프트 기반 선택의 두 단계로 구성되며, CNN 기반 탐지기와 인스턴스 분할 브랜치를 활용해 SAM 대비 약 50배 빠른 추론 속도를 가지면서도 유사한 수준의 성능을 유지할 수 있다. 본 연구에서는 이러한 FastSAM의 경량성과 처리 능력에 주목하여 FastSAM 모델 중 경량화 모델인 FastSAM-s 모델을 선택하여 연구를 진행하였다. 그림1은 FastSAM 모델의 구조도이다.



[그림 1] FastSAM 모델 구조도

3. FastSAM-s 모델의 파일형식 변환

본 연구에서는 FastSAM 모델의 기존 PyTorch(.pt) 형식을 엣지 컴퓨팅 환경에 포팅하기 위해 8-bit 정수 양자화를 적용한 TensorFlow Lite(.tflite)형식으로 변환하였다. 변환 과정은 먼저 PyTorch 형식의 모델을 ONNX(Open Neural Network Exchange) 형식으로 내보낸 뒤, 이를 TFLite 형식으로 변환하는 단계로 구성된다[2]. 이 과정에서 모든 연산을 8-bit 정수 연산으로 변환하여 모델 크기를 축소하고, Edge TPU 환경에서 실행이 가능한 edge.tflite형식으로 변환하였다. 그림 2는 FastSAM-s모델의 파일 형식 변환 과정을 나타낸 것이다.



[그림 2] FastSAM-s 모델 형식의 변환 과정

4. 실험 환경 및 결과

표 1은 본 연구의 실험 환경 규격 내용이다.

<표 1> 실험 환경 규격

구분	항목		내용
Raspberry Pi 5	H/W	CPU	Broadcom BCM2712
		RAM	8GB LPDDR4X-4267
	S/W		OS==Debian12(BookWorm) Python == 3.9.12, 3.11.2 opencv-python == 4.10.0.84 torch==2.0.1, torchvision==0.15.2
Google Coral Edge TPU	H/W		8TOPS (Google Edge TPU Process)
	S/W		Python == 3.9.12 edge-tpu-silva == 1.0.4
	power consumption		2W

표 2는 실험에서 사용한 성능 측정 방식에 관한 내용이다.

<표 2> 성능측정 방식

구분	내용
IoU (Intersection over Union)	객체 분할 모델의 성능을 평가하는 지표로 모델이 생성한 마스크의 정확도를 측정한 값
FPS (Frame Per Second)	처리 속도를 평가하는 측정치로 /ms 기준으로 계산한 값

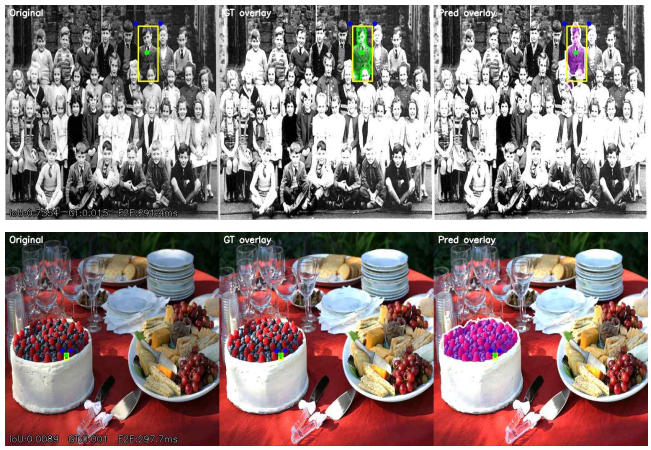
표 3은 모델의 형식별 정확도와 처리 속도를 비교한 것이다. 본 연구에서 사용한 이미지의 총 개수는 50장이며, 총 마스크의 개수는 2119개다. 포인트, 박스, 텍스트, 마스크 네 가지의 프롬프트 입력 방식 중 포인트와 박스를 혼합한 한 가지의 프롬프트를 입력으로 사용하여 성능을 측정 및 비교하였다. 전체 마스크의 객체 분할 성능측정 결과는 평균 0.33으로 기존 PyTorch 형식의 모델에 비해 약 3배 낮은 수치를 보이지만, 이미지 전체 크기 512를 기준으로 마스크의 크기가 0.05 이상일 때의 값은 약 0.7, 그 이하일 때는 약 0.3의 정확도를 갖는다.

<표 3> 형식별 성능 비교

구분	IoU	FPS (/ms)
PyTorch (.pt)	1	1545.8
TFLite (.tflite)	0.3343	465.8
Edge TFLite (.tflite)	0.3344	289.4

그림 3과 그림 4는 TFLite 형식과 Edge TFLite 형식의 모델을 수행한 결과이다. 두 형식의 모델 모두 큰 물체에 대한 세그멘트(segment) 성능은 기존 PyTorch 형식의 모델과 비교하여 큰 차이가 발생하지 않았으나, 작은 물체에 대한 정확도 성능은 두 형식의 모델 모두 기존 모델에 비하여 현저히 떨어지는 것을 확인할 수 있다. 본 연구에서 측정한 결과, 추론 속도 성능 면에서 기존 모델에 비교하여 TFLite 형식 모델은 3.31배 빨랐으며, Edge TFLite 형식 모델은 5.34배 빠름을 확인할 수 있다.

[그림 3] Raspberry Pi 5 CPU 단독 환경에서의 TFLite 수행 결과



[그림 4] Coral Edge TPU를 장착한 환경에서의 Edge TFLite 수행 결과

5. 결론

본 연구에서는 데스크톱 환경에서 FastSAM-s모델의 PyTorch 형식을 수행한 결과를 기준으로, Raspberry Pi 5 환경에서 CPU만을 이용하여 수행한 TensorFlow Lite 변환 모델과 엣지 AI 가속기인 Coral Edge TPU을 사용한 Edge TPU 최적화 모델의 성능을 비교 평가한 결과를 보면, 평균 처리 속도 면에서는 기존 모델에 비하여 TFLite 형식 모델은 3.31배, Edge TFLite 형식 모델은 5.34배 향상되었다. 객체 분할 정확도 면에서는 두 형식의 모델 모두 평균 0.33으로 확인되었다. 큰 객체에 대한 분할 성능은 기존 모델과의 큰 차이를 보이지 않는 결과를 얻었으나, 양자화에 따른 수치 표현 축소와 입력 해상도를 1024에서 512로 축소함에 따라 작은 객체에 대한 분할 정확도는 떨어졌다. 향후 동일한 실험 환경에서 캘리브레이션 데이터셋을 구축하고 작은 객체 분포에 맞춘 활성값 범위를 추정하는 것으로 스케일과 제로포인트 산정 절차를 최적화하는 연구를 진행하고자 한다.

REFERENCES

[1] Xu Zhao et al, “Fast Segment Anything”, arXiv:2408.00714, 2023

[2] Pradyun Gedam (2024). Introduction to ONNX.https://onnx.ai/onnx/intro/