모빌리티 환경 적용을 위한 Depth-Anything V2 모델의 다각적 최적화 전략 분석*

한승우[0], 임희진, 강영희
충남대학교 전파정보통신공학과, 목원대학교 컴퓨터공학과, 대전대학교 컴퓨터공학과
hsw_2266@naver.com, gabriel227@mokwon.ac.kr, zmfltm7894@gmail.com

Multi-faceted Optimization Strategies for Depth-Anything V2 in Mobility Applications

Seungwoo Han[0], Huijin Lim, Younghui Gang
Department of Radio and Information Communications Engineering, Chungnam National University
Computer Engineering, Mokwon University
Computer Engineering, Daejeon University

요 약

본 논문은 고성능 범용 깊이 추정 모델인 Depth-Anything V2의 실시간 적용 가능성을 높이기 위한 경량화 및 최적화 방안을 제시한다. 먼저, 지식 증류 기법을 활용하여 MobileNetV2 기반 모델과 축소형 Depth-Anything(ViT-S) 구조를 설계하였으며, 실험을 통해 정량적 성능을 비교하였다. 이후, 원본 모델에 L1 비정형 가지치기와 동적 양자화를 각각 적용하고 두 기법을 결합하는 모델을 구성하여, 추론 정확도 및 효율성을 심층 분석하였다. 마지막으로, TensorRT 기반 최적화 조합(FP16 정밀도, 해상도 및 배치 크기 변화)에 따른 추론 속도 및 정확도 변화를 평가하였다. 분석 결과, 단순한 구조 변경보다 동적 양자화 및 FP16 최적화 조합이 가장 효과적인 성능 개선을 보였으며, 이는 실시간 시스템에 적합한 현실적인 적용 전략임을 시사한다.

1. 서 론

최근 단안 영상 기반 깊이 추정(Monocular Depth Estimation, MDE)은 자율주행, 드론, 로봇 비전, 증강현실 등 다양한 응용 분야에서 핵심 기술로 주목받고 있다. 이러한 환경에서는 조명, 날씨, 장면 구성 등 다양한 조건에서도 안정적인 성능을 유지할 수 있는 범용(Foundation) 모델의 필요성이 높아지고 있다.

Depth Anything은 이러한 요구를 반영하여 개발된 범용 MDE 모델로, 사전 학습된 Vision Transformer(ViT) 인코더를 사용한다. 그림 1에 나타난 바와 같이, 그 학습 과정은 다음과 같다: (1) 고품질 합성 이미지로 DINOv2-G 기반의 교사 모델을 먼저 학습시킨다. (2) 이 교사 모델을 이용해, 약 6,200만 장에 달하는 비라벨 이미지로부터 의사 라벨(Pseudo-Label)을 생성한다. (3) 마지막으로 생성된 의사 라벨을 바탕으로 자기지도학습(Self-Training)을 수행하여 최종 모델을 완성한다. 이러한 방식을 통해 기존 깊이 데이터셋의 라벨 부족 문제를 극복하고 다양한 도메인에 대한 일반화 능력을 확보하였으며, 특히 DINOv2 인코더[1]의 표현을 정렬(Feature Alignment)함으로써 의미 보존 기반의 고수준 깊이 추정을 가능하게 하였다.

이러한 기술적 구성은 Depth Anything이 대표적인 범용 모델인 MiDaS[2][3]보다 뛰어난 제로샷 성능(Zero-shot Performance)을 보이게 하였으며, 정량 깊이 추정 모델인 ZoeDepth와 비교해도 우수한 성능을 입증하였다. 예를 들어, NYU-D 데이터셋에서 Depth Anything V2 (ViT-L)는 AbsRel 0.056, RMSE 0.206으로, ZoeDepth(AbsRel 0.077, RMSE 0.282) 대비 더 낮은 오차율을 기록하였다[4].

그러나 자율주행이나 드론과 같은 실제 모빌리티 환경에서는 실시간 의사결정과 저전력 연산이 필수적이므로, 고성능 모델인 Depth Anything은 추론 시간과 연산 자원 측면에서 제약이 존재한다. 이 문제를 해결하기 위해, 본 연구에서는 세 가지 차원의 최적화 전략을 탐색하고 각 접근법의 실효성을 비교 분석하였다.

첫째, '아키텍처 변경' 관점에서 지식 증류(Knowledge Distillation)[5]를 통해 새로운 소형 학생 모델을 설계하여 경량화 가능성을 타진하였다. 둘째, '하드웨어 범용적 최적화' 관점에서 원본 모델에 직접 양자화(Quantization)와 가지치기(Pruning)를 적용하여 CPU 등 일반 환경에서의 효율성을 분석하였다. 셋째, '플랫폼 특화 가속' 관점에서 NVIDIA TensorRT 프레임워크[6]를 활용하여 GPU 환경에서의 추론 성능을 극한까지 최적화하는 방안을 실험하였다.

이러한 다각적인 연구를 통해, 원본 모델의 성능을 가능한 한 유지하면서도 모빌리티 분야 실시간 적용이 가능한

* 본 논문은 주식회사 카카오모빌리티에서 ETRI AI 나눔을 통해 공개한 [자율주행] 3D 동적객체 검출/추적 학습데이터 셋을 사용함

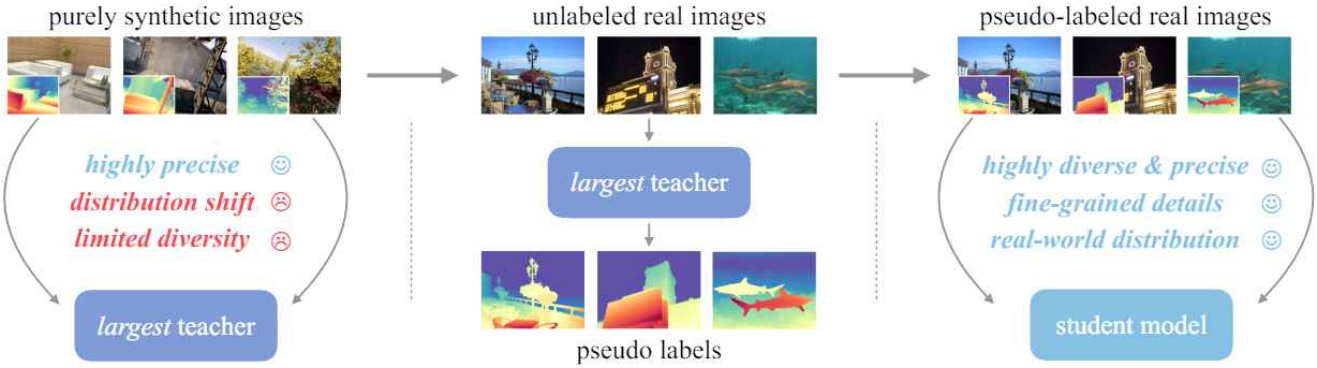

그림 1. Depth-Anything V2의 프레임워크 구조

효율적인 추론 환경을 구축하고자 하였다.

2. 지식 증류 기반으로 MobileNet 및 축소형 Depth Anything 구조의 학생 모델 설계

지식 증류(Knowledge Distillation)는 대규모 모델(Teacher)이 학습한 정보를 소형 모델(Student)에 전달하여 모델을 경량화하는 기법이다. 일반적으로 고성능 모델은 연산 비용이 크고 모바일 환경에 적합하지 않다는 한계가 있다.

지식 증류는 Teacher 모델의 예측값(Soft Target)을 기반으로 Student 모델을 학습시켜, 단순한 정답 레이블보다 더 풍부한 정보로 일반화 성능을 향상시킨다. 이 방식은 성능 저하를 최소화하면서도 모델의 크기와 연산량을 줄일 수 있어, 자원이 제한된 환경에서 효과적으로 활용된다.

본 연구의 학습 과정은 다음과 같다. 먼저, 사전 학습된 Depth-Anything V2 모델을 교사 모델로 준비한다. 다음으로, 이 교사 모델을 이용해 ETRI 데이터셋 이미지에 대한 깊이맵(의사 라벨)을 생성한다. 마지막으로, 이 '이미지-의사 라벨' 쌍을 가지고 학생 모델(MobileNetV2[7], 축소형 Depth-Anything)을 학습시킨다.

2.1. MobileNet 기반 지식 증류 구조

본 절에서는 Depth-Anything의 전체 구조에서 일부 모듈과 채널 수를 축소하여, 메모리 사용량과 연산 복잡도를 줄이면서도 성능 저하를 최소화한 구조의 학생 모델을 설계하였다.

학습은 518×518 해상도의 RGB 이미지와 해당 Pseudo-Depth 라벨을 입력으로 수행되었으며, 정규화된 L1 손실(Scale-Shift Invariant Loss)과 경사도 일치 손실(Gradient Matching Loss)을 조합하여 모델을 최적화하였다. RGB 입력에 대한 MobileNetV2 모델의 깊이 추정 결과는 그림 2에 시각화하였다.

2.2. 축소형 Depth-Anything (ViT-S)을 통한 지식 증류 구조

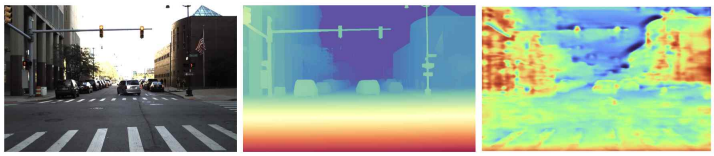

그림 2. RGB 입력에 대한 원본, 선생(Depth-Anything V2), 학생(MobileNetV2) 모델의 깊이 추정 결과

본 절에서는 Depth-Anything의 전체 구조에서 일부 모듈과 채널 수를 축소하여, 메모리 사용량과 연산 복잡도를 줄이면서도 성능 저하를 최소화한 구조의 학생 모델을 설계하였다.

학습은 518×518 해상도의 RGB 이미지와 해당 pseudo-depth 라벨을 입력으로 수행되었으며, 정규화된 L1 손실(Scale-Shift Invariant Loss)과 경사도 일치 손실(Gradient Matching Loss)을 조합하여 모델을 최적화하였다. RGB 입력에 대한 축소형 Depth-Anything 모델의 깊이 추정 결과는 그림 3에 시각화하였다.

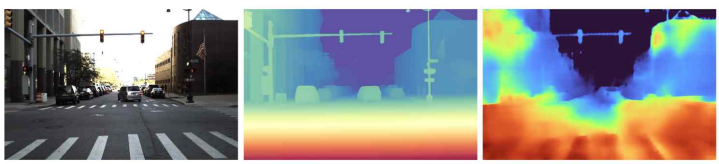

그림 3. RGB 입력에 대한 원본, 선생(Depth-Anything V2), 학생(축소형 Depth-Anything) 모델의 깊이 추정 결과

2.3. 지식 증류 기반 두 가지 모델의 RMSE 비교 평가

지식 증류를 적용한 두 가지 경량 모델(MobileNetV2 기반 모델, 축소형 Depth-Anything 모델)의 성능을 비교 평가하기 위해, Depth-Anything V2 Base (ViT-B)의 추론 결과를 정답(Ground Truth)으로 간주하고 총 400개의 검증 샘플에 대해 RMSE (평균 제곱근 오차) 및 MAE(평균 절대 오차)를 측정하였다. 두 모델의 정량적 성능은 표 1에 정리하였다. 이때 GT의 평균값은 5.8907로 계산되었다.

표 1. 지식 증류 기반 두 가지 모델의 RMSE와 MAE

모델	RMSE	MAE
MobileNetV2 기반 모델	8.095	5.8877
축소형 Depth-Anything 모델	4.1590	2.9190

MobileNetV2 기반 모델은 RMSE 8.0958, MAE 5.8877을 기록하였으며, 예측값이 GT의 평균값과 유사한 스케일로 정규화되었음에도 불구하고 상대적으로 높은 오차를 보였다. 이는 기존 CNN 아키텍처가 ViT 기반 교사 모델의 복잡한 깊이 표현을 학습하는 데 어려움이 있음을 시사한다.

반면, 축소형 Depth-Anything(ViT-S) 모델은 평균 RMSE 4.1590, MAE 2.9190을 기록하였으며, GT의 평균값 5.8907과 비교할 때 상대적으로 낮은 오차를 보였다. 전체적으로 안정적인 추정 성능을 나타냈으나, 일부 샘플에서 높은 RMSE가 관측되어 특정 조건에서의 예측 안정성 향상이 과제로 남았다. 결론적으로, ViT와 같은 대규모 Transformer 아키텍처의 복잡한 표현력을 단순한 CNN(MobileNetV2)이나 축소된 모델로 온전히 전달하는 데에는 단순 지식 증류만으로는 명백한 한계가 있음을 확인하였다. 이는 3장에서 다룰 원본 모델 직접 경량화 연구의 필요성을 뒷받침한다.

3. Depth-Anything V2 모델의 직접 경량화

본 절에서는 원본 모델을 직접 경량화하는 방식으로 접근하였다. 정량적인 성능 평가를 위해, DDAD 데이터셋과 같은 실제 거리(Metric) 스케일의 데이터로 사전 학습된 Depth-Anything V2 Metric-Outdoor-Base 모델을 기준선(Baseline)으로 설정하고, 이를 기반으로 연산량과 메모리 사용량을 줄이는 방안을 실험하였다. 실시간 시스템에 적합한 모델을 구현하기 위해, 본 연구는 L1 비정형 가지치기(Unstructured Pruning)와 동적 양자화(Dynamic Quantization) 기법을 적용하여 모델의 효율성을 극대화하고자 하였다.

3.1. 경량화 모델 설계

본 연구에서는 각 경량화 기법의 독립적 효과와 통합효과를 비교 분석하기 위해 다음과 같이 네가지 모델을 설계하였다.

1. 원본 모델 (FP32 Baseline): 모든 성능 비교의 기준이 되는, 수정되지 않은 Depth-Anything V2 모델이다.

2. 가지치기 모델 (Pruned): L1 Norm을 기준으로 가중치의 중요도를 평가하여, 전체 네트워크에서 중요도가 낮은 파라미터 50%를 제거(Pruning)한 모델이다.

3. 통합 모델 (Pruned + Dyn INT8): 30% 비율의 가지치기를 먼저 적용한 모델에 동적 양자화를 추가로 적용한 모델이다. 이는 두 기법의 조합이 시너지 효과를 내는지, 혹은 성능을 저해하는지를 분석하기 위해 설계되었다.

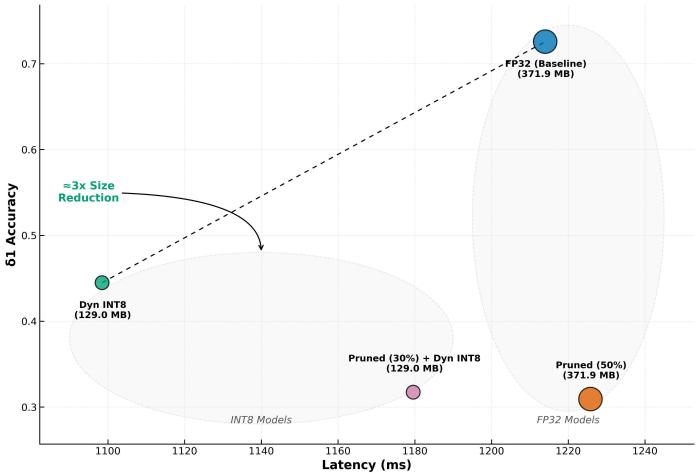

그림 4. 경량화 모델별 성능-지연시간 트레이드오프

4. 동적 양자화 모델 (Dyn INT8): PyTorch의 quantize_dynamic() 함수를 사용하여, 학습 데이터 없이도 모델의 선형(Linear) 계층 가중치를 FP32에서 INT8 정밀도로 변환한 모델이다.

가지치기 후 별도의 Fine-Tuning을 수행하지 않은 것은, 추가적인 학습 비용 없이 경량화 기법 자체의 즉각적인 효과를 측정하기 위함이다.

3.2. 각 경량화 모델의 정량적 성능 비교

본 실험의 정량 평가를 위해, DDAD 데이터셋을 전처리하여 RGB-깊이 쌍으로 구성하고 학습 및 검증 세트로 분할하여 사용하였다. DDAD 데이터셋은 실제 자율주행 환경에서 수집된 대규모 데이터로, 다양한 주행 시나리오를 포함하고 있어 본 연구의 목표인 모빌리티 환경에서의 깊이 추정 성능을 평가하기에 가장 적합하다. 각 모델은 동일한 입력 이미지와 조건 하에서 추론을 수행하였으며, 이 과정에서 δ1 정확도, RMSE, AbsRel, Latency(ms), FPS, 모델 크기(MB) 등의 지표를 정량적으로 비교하였다. 측정은 Intel i7-14700 CPU 기반의 데스크탑 환경에서 이루어졌으며, Latency는 단일 이미지 추론 시간을 기준으로, FPS는 초당 추론 가능한 프레임 수를 의미한다.

표 2-1. 정량적 성능 비교 (정확도 및 효율 지표)

모델 버전	Precision	δ1 (↑)	RMSE (↓)	AbsRel (↓)
FP32 (Baseline)	FP32	0.7257	8.4507	0.1848
Pruned (50%)	FP32 (Sparse)	0.3092	14.4092	0.6065
Pruned (30%) + Dyn INT8	INT8 (Sparse)	0.3172	14.3654	0.5764
Dyn INT8	INT8	0.4448	12.0005	0.3590

표 2-2. 경량화 및 속도 지표

모델 버전	모델 크기 (MB)	Latency (ms)	FPS
FP32 (Baseline)	371.93	1214.0	0.82
Pruned (50%)	371.93	1225.8	0.82
Pruned (30%) + Dyn INT8	128.98	1179.6	0.85
Dyn INT8	128.98	1098.5	0.91

표 2의 결과에 따르면, 동적 양자화(Dyn INT8) 모델이 경량화의 목표를 가장 효과적으로 달성했음을 알 수 있다. 모델 크기를 원본 대비 약 65.3% 줄이면서도 가장 낮은 Latency(1098.5ms)를 기록했다. δ1 정확도는 0.4448로 하락했지만, 다른 경량화 모델 중에서는 가장 양호한 수준을 보였다.

반면, Pruned(50%) 모델은 Latency가 오히려 1%가량 증가하고 δ1 정확도는 0.3092로 급락하여, CPU 환경에서 비정형 가지치기의 비효율성을 명확히 보여주었다. 가지치기와 양자화를 결합한 모델 역시 정확도 회복이 미미하여, 단순한 기법의 조합이 항상 최적의 결과를 보장하지는 않는다는 점을 시사한다.

3.3. 모델 간 Trade-off 관계 시각화

모델 간 Trade-off 관계를 시각적으로 분석하기 위해, δ1 정확도(높을수록 성능 우수), Latency(ms, 낮을수록 우수), 모델 크기(MB)를 시각화한 2D 그래프를 구성하였다. 그림 4는 네 가지 모델 버전(FP32, Pruned, Pruned(30%) + Dyn INT8, Dyn INT8)의 성능을 직관적으로 비교한 결과이다.

해당 시각화에서는 Dyn INT8 모델이 좌측 상단 영역에 위치하여, 가장 빠른 추론 속도(Latency)와 상대적으로 높은 δ1 정확도, 129.0MB의 소형화된 모델 크기를 동시에 달성한 균형 잡힌 성능을 보여준다. 반면, Pruned+INT8 조합은 모델 크기는 감소했지만, 정확도는 더 낮고 Latency는 오히려 증가한 형태를 보인다. 이는 비정형 가지치기 적용 시 연산 최적화가 하드웨어 수준에서 비효율적으로 작용할 수 있음을 시사한다.

또한, 그림 5는 각 모델의 깊이 추정 결과를 시각적으로 비교한 예시이다. FP32 모델이 정밀한 윤곽과 깊이 차이를 표현한 반면, Pruned 기반 모델들은 공통적으로 높은 RMSE 값(14.4 이상)이 시사하듯, 객체의 형태가 심하게 왜곡되고 다량의 노이즈가 발생하는 등 질적으로 크게 저하된 결과를 보였다. Dyn INT8 모델은 원본 대비 디테

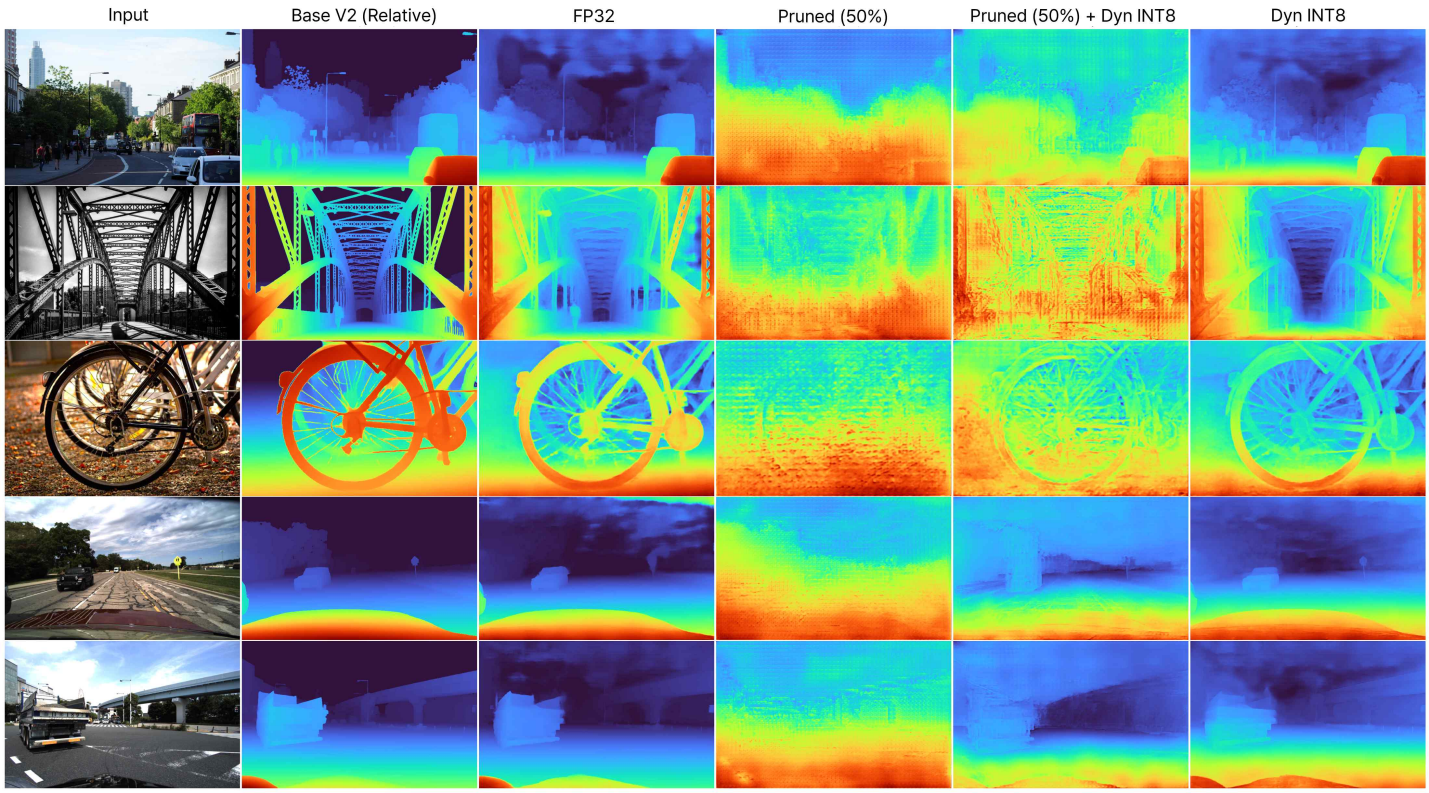

그림 5. 데모 이미지 및 DDAD 검증 이미지 모델별 깊이 추정 결과 시각화

일이 일부 손실되었으나, 전반적인 구조는 유지하며 정량 지표와 일치하는 경향을 나타냈다.

이러한 결과는 단순히 파라미터 수를 줄이는 것(Pruning)이 아닌, 연산 자체를 하드웨어 친화적으로 바꾸는 것(Quantization)이 범용 CPU 환경에서 더 실용적인 경량화 전략임을 시각적으로 뒷받침한다.

3.4. 경량화 기법 비교 결과 요약

본 연구에서는 Depth-Anything V2 모델을 기반으로 L1 비정형 가지치기, 동적 양자화, 그리고 두 기법의 결합 모델을 설계하고, 다양한 정량 지표를 통해 성능을 비교하였다.

실험 결과, 동적 양자화(Dyn INT8)는 모델 크기, 추론 속도, 정확도 간의 트레이드오프 관계에서 가장 균형 잡힌 해법임을 입증하였다. 이는 추가 학습 없이도 즉각적인 효율성 개선을 가져와 실시간 시스템 적용에 가장 적합한 경량화 방식으로 평가된다.

반면, 비정형 가지치기(Pruning)는 CPU 환경에서 실질적인 속도 이점을 제공하지 못하고 심각한 정확도 하락만을 초래하였다. 특히, 두 기법을 단순히 결합했을 때 발생하는 부정적 시너지는 경량화 전략 설계 시 각 기법의 특성과 타겟 하드웨어의 아키텍처를 반드시 함께 고려해야 함을 보여준다.

이러한 분석은 향후 효율적인 깊이 추정 모델을 개발하는 데 있어 실용적인 가이드라인을 제공할 수 있을 것이다.

4. TensorRT 기반 모델 최적화를 통한 파라미터 구성

본 절에서는 PyTorch 기반의 Depth-Anything V2 Base (ViT-B) 모델을 TensorRT로 변환하여, 다양한 경량화 설정 조합에 따른 추론 성능을 정량적으로 평가하였다. TensorRT는 NVIDIA에서 제공하는 고속 추론 최적화 프레임워크로, 다양한 하드웨어 환경에 대응할 수 있도록 설계되어 있다. 본 연구에서는 정확도 손실을 최소화하면서도 실행 효율을 개선할 수 있는 조합을 분석 대상으로 설정하였다.

특히 TensorRT 기반의 FP16 및 FP32 정밀도 설정은 모델의 추론 속도와 메모리 효율을 극대화할 수 있다는 점에서 채택되었다. 이는 실시간 서비스 적용 가능성을 고려하여 연산량을 줄이고, 다양한 하드웨어 환경에서도 최적의 성능을 확보하려는 목적에 기반한다.

Jetson Nano나 Orin과 같은 임베디드 환경에서의 실측 테스트가 이상적이나, 본 평가에서는 하드웨어 제약으로 인해 NVIDIA RTX 3060 Ti 환경에서 실험을 수행하였다. 이에 따라 전력 소비량, 발열, Jetson 특화 기능(DLA 오프로드 등)은 평가 범위에서 제외되었으며, 정확도, 추론 지연 시간, 메모리 사용량 등 하드웨어 중립적인 성능 지표를 중심으로 분석하였다. 평가에 사용된 주요 라이브러리 및 도구 버전은 표 3에 정리하였다.

표 3. 평가에 사용된 주요 라이브러리 및 도구 버전

항목	버전
PyTorch	2.2.2 + CUDA 12.1
ONNX	1.16.0
TensorRT	10.6.0
xFormers	0.0.30

4.1. 변환 및 최적화 절차

Depth-Anything V2 Base (ViT-B) 모델에 대한 최적화는 다음의 세 단계로 구성된다.

1. **모델 변환 단계**
   PyTorch 환경에서 학습된 원본 모델을 ONNX(Open Neural Network Exchange) 포맷으로 변환하였다[8]. 이는 TensorRT가 PyTorch 모델을 직접 처리할 수 없기 때문에 필수적인 전처리 과정이다. 변환 과정에서는 모델 구조와 가중치가 온전히 유지되도록 연산자 호환성을 검토하고, 불필요한 연산은 제거되도록 최적화하였다.

2. **TensorRT 엔진 생성 단계**
   ONNX로 변환된 모델을 기반으로, 다양한 추론 환경에 적합한 TensorRT 엔진(.plan 파일)을 생성하였다. 이 과정에서는 연산 정밀도(Precision: FP32, FP16), 입력 해상도(4종), 배치 크기(3종)를 조합하여 총 24개의 엔진 구성을 제작하였다.

3. **성능 평가 및 시각화**
   생성된 TensorRT 엔진은 별도 구현된 평가 모듈을 통해 추론 지연 시간, 메모리 사용량, 정량 정확도(RMSE 등)를 측정하였다. 또한, 입력 이미지에 대한 깊이 추정 결과를 시각화하여 모델별 품질 차이를 직관적으로 비교하였다.

평가에는 FP32, FP16의 두 가지 연산 정밀도, 네 가지 입력 해상도, 세 가지 배치 크기 조합을 적용하였으며, 총 24개의 엔진 구성을 생성하여 성능을 비교하였다. TensorRT 엔진 조합 설정의 세부 항목은 표 4에 정리하였다. 해당 조합은 추론 속도, 메모리 사용량, 정확도 간의 균형을 분석하기 위한 기준으로 설정되었으며, 이후 절에서 각 조합에 대한 정량적 평가 및 시각화를 통해 비교 분석을 수행한다.

4.2. TensorRT 최적화 조합에 따른 성능 평가

그림 6. 성능 최적 조합(FP16, 560×560, Batch 1)의 출력 시각화 예시

표 4. TensorRT 엔진 조합 설정

실험 변수	값
Precision	FP32 / FP16
Input Size	672×672 / 560×560 / 504×672 / 448×896
Batch Size	1 / 2 / 4

본 절에서는 TensorRT 기반의 다양한 경량화 조합에 대해 추론 정확도와 연산 효율 측면에서의 성능을 평가하였다. 비교 항목은 평균 RMSE, 상대 오차(AbsRel), 평균 추론 지연 시간(Latency Mean, ms)이며, 정밀도(Precision), 입력 해상도(Input Resolution), 배치 크기(Batch Size)를 기준으로 분석하였다.

총 24개의 엔진 조합 중 최적 성능을 보인 조합은 FP16 정밀도, 560×560 해상도, 배치 크기 1인 설정이었다. 이 조합은 평균 RMSE 22.9547, AbsRel 0.7815, 추론 지연 시간 33.78ms로, 정확도와 속도 모두에서 균형 잡힌 결과를 보였다. 해당 조합은 표 5에 정리하였다.

표 5. 최적 조합에 따른 TensorRT 조합의 성능 지표

항목	값
Precision	FP16
Input Resolution	560×560
Batch Size	1
RMSE	22.9547
AbsRel	0.7815
Latency (ms)	33.78

그림 6은 FP16 정밀도, 560×560 해상도, 배치 크기 1 설정에서 추론한 깊이 추정 결과를 시각화한 예시로, 모델의 실질적인 예측 품질을 보여준다.

아울러, 입력 해상도별로 가장 우수한 조합을 추출하여 성능을 비교하였다. 그 결과, 모든 해상도에서 공통적으로 FP16 정밀도가 선택되었으며, 해상도가 높아질수록 평균 추론 지연 시간(Latency)은 증가하는 경향을 보였다. 반면 RMSE는 해상도 변화에 크게 영향을 받지 않고 일정 수준을 유지하였다. 각 해상도에서의 최적 조합 및 성능 지표는 표 6에 정리하였다.

표 6. 입력 해상도별 최적 조합 및 성능 지표

Input Resolution	Precision	Batch Size	RMSE (mean)	Latency (ms)
448×896	FP16	4	22.9895	76.8004
504×672	FP16	2	22.9633	56.1861
560×560	FP16	1	22.9547	33.7817
672×672	FP16	1	23.0079	35.4844

이어서, 배치 크기별로 가장 우수한 성능을 보이는 조합을 정리하였다. 모든 배치 크기에서 공통적으로 FP16 정밀도와 560×560 해상도가 선택되었으며, 배치 크기가 증가할수록 추론 지연 시간은 다소 증가하는 경향을 나타냈다. RMSE는 배치 크기에 따라 큰 차이는 없었으나, 실시간성 요구가 있는 환경에서는 낮은 배치 크기가 유리할 수 있다. 해당 결과는 표 7에 정리하였다.

표 7. 배치 크기별 최적 조합 및 성능 지표

Batch Size	Input Resolution	Precision	RMSE (mean)	Latency (ms)
1	560×560	FP16	22.9547	33.7817
2	560×560	FP16	22.9550	59.6572
4	560×560	FP16	22.9554	64.8085

이러한 결과를 종합적으로 보면, FP16 기반의 560×560 해상도 조합이 정확도와 연산 효율 간의 Trade-off 측면에서 가장 안정적인 선택임을 확인할 수 있었다.

5. 종합 논의 및 결론

본 논문은 고성능 깊이 추정 모델인 Depth-Anything V2를 실제 모빌리티 환경에 적용하기 위한 현실적인 최적화 방안을 탐색하고자, 지식 증류, 직접 경량화, TensorRT 가속이라는 세 가지 다각적 접근법을 실험적으로 비교 및 분석하였다.

실험 결과들을 종합하면, 최적의 전략은 타겟 하드웨어

와 요구 성능 수준에 따라 달라져야 한다는 명확한 결론을 얻을 수 있었다.

GPU 기반의 고성능 실시간 추론 환경이 목표라면, 모델 구조 변경 없이 TensorRT를 활용한 FP16 최적화가 가장 효과적이었다. 이는 원본 모델의 정확도를 거의 손상시키지 않으면서 가장 극적인 속도 향상을 달성하여, 고신뢰성 시스템에 즉시 적용 가능한 솔루션임을 입증하였다.

반면, 범용 CPU 환경이나 자원 제약이 있는 엣지 디바이스를 고려할 경우, 동적 양자화(Dynamic Quantization)가 모델 크기, 추론 속도, 정확도 간의 가장 현실적이고 균형 잡힌 해법임을 확인하였다. 모델 크기를 획기적으로 줄이면서도 다른 경량화 기법 대비 가장 양호한 정확도를 유지하여, 범용성 높은 경량화 전략으로서의 가치를 보여주었다.

이와 대조적으로, 새로운 경량 아키텍처를 설계하는 접근법(지식 증류)과 비정형 가지치기(Unstructured Pruning)는 본 연구의 조건 하에서는 상당한 성능 저하를 초래하였다. 이는 복잡한 깊이 추정 태스크에서 원본 모델의 표현력을 유지하는 것이 얼마나 중요한지, 그리고 하드웨어 비친화적인 최적화가 실제 속도 개선으로 이어지지 않는다는 '이론과 현실의 간극'을 명확히 보여주는 결과이다.

결론적으로 본 연구는, 특정 응용 분야에 최첨단 모델을 배포하고자 할 때 고려해야 할 다양한 트레이드오프를 체계적으로 분석하고, 각 시나리오에 맞는 최적화 전략에 대한 실용적인 가이드라인을 제시했다는 점에서 의의를 갖는다. 향후 연구로는 정확도 저하를 최소화하기 위한 정적 양자화(Post-Training Static Quantization)나 양자화 인식 학습(Quantization-Aware Training, QAT)을 적용하거나, 필터/채널 단위의 구조적 가지치기(Structured Pruning) 기법을 통해 하드웨어 효율성을 극대화하는 방안을 탐색해 볼 수 있을 것이다.

참고 문헌

[1] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Formal, et al., "DINOv2: Learning Robust Visual Features without Supervision," arXiv preprint arXiv:2304.07193, 2023.

[2] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, "Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 44, no. 3, pp. 1623-1637, 2022.

[3] R. Ranftl, A. Bochkovskiy, and V. Koltun, "Vision Transformers for Dense Prediction," in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 12079-12090, 2021.

[4] Y. Wang, Z. Li, H. Wang, J. Wang, J. Shi, and Z. Qiao, "Depth Anything V2," arXiv preprint arXiv:2406.09414, 2024.

[5] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," arXiv preprint arXiv:1503.02531, 2015.

[6] NVIDIA Corporation, "NVIDIA TensorRT Documentation," NVIDIA Developer Zone, 2024. [Online]. Available: https://developer.nvidia.com/tensorrt

[7] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4510-4520, 2018.

[8] B. Ashbaugh, S. Chernova, S. Farhadi, et al., "ONNX: Open Neural Network Exchange," GitHub, 2019. [Online]. Available: https://github.com/onnx/onnx