Distributed Perception by Collaborative Robots

Ramyad Hadidi, Jiashen Cao, Matthew Woodward, Michael S. Ryoo, and Hyesoon Kim

in IEEE Robotics and Automation Letters,
vol. 3, no. 4, pp. 3709-3716, Oct. 2018.

Introduction

- 데이터의 양의 증가, 알고리즘의 발전, 연산 성능의 향상을 통해
  DNN 응용 역시 빠르게 발전.
- 로봇은 DNN 응용을 통해 데이터 처리에 이익을 얻을 수 있음.
- 로봇의 제한된 성능과 제한된 전력 공급 속에서
  데이터를 실시간으로 처리하는 것은 어려움.
- 논문은 실시간 DNN 기반 인식을 수행하기 위해
  분산 로봇 시스템에서 로봇들의 집계된 연산 능력을 활용할 것을 제안.
- 공동 연산을 통해 성능, 전력 문제 없이 수집된 데이터를 처리.

Related Works

- 연산을 분산 처리 하지만 고성능 하드웨어에 초점을 맞춤.
  추론보다는 훈련을 위한 방법을 연구.
  - J. Dean et al., "Large scale distributed deep networks,"
    in Proc. Proc. Conf. Neural Inf. Process. Syst., 2012, pp. 1223–1231.

- 단일 엣지 장치(Tegra TK1)와 클라우드 간에 DNN 모델을 동적으로 분할하여
  성능과 에너지 효율을 향상시킴.
  - Y. Kang et al., "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge,"
    in Proc. 22nd ACM Int. Conf. Architectural Support Program. Lang. Oper. Syst., 2017, pp. 615–629.

- 모바일 장치(갤럭시 S3)와 클라우드 간에 연산을 분산 처리.
  - J. Hauswald, T. Manville, Q. Zheng, R. Dreslinski, C. Chakrabarti, and T. Mudge,
    "A hybrid approach to offloading mobile image classification,"
    in Proc. Int. Conf. Acoust., Speech, Signal Process., 2014, pp. 8375–8379.

Related Works

- 모바일 장치(넥서스 5)를 이용한 로컬 분산 시스템(MoDNN).
  레이턴시가 길어 실시간 추론을 보장할 수 없음.
  - J. Mao, X. Chen, K. W. Nixon, C. Krieger, and Y. Chen,
    "MoDNN: Local distributed mobile computing system for deep neural network,"
    in Proc. Design Automat. Test Europe, 2017, pp. 1396–1401.

- 로컬 장치에서 연산을 분산 처리(DDNN).
  로컬에서 몇 개의 레이어를 수행하고 나머지 연산은 클라우드에서 처리.
  - S. Teerapittayanon, B. McDanel, and H. Kung,
    "Distributed deep neural networks over the cloud, the edge and end devices,"
    in Proc. Proc. Int. Conf. Distrib. Comput. Syst., 2017, pp. 328–339.

- 리소스가 제한된 플랫폼을 위한 라이브러리
  - Microsoft, "Embedded Learning Library(ELL)"
  - Google, "TensorFlow Lite"

분산 처리 방식

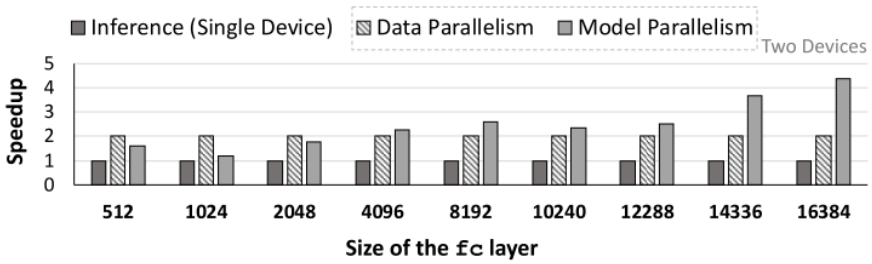
- Model Parallelism
  - 주어진 레이어나 레이어 그룹을 장치에 분배하는 방식
- Data Parallelism
  - 입력 데이터를 네트워크의 여러 장치에 보내는 방식
- Convolution Layer는 Data Parallelism을 활용할 때 이득
- FC Layer는 장치의 리소스 환경에 따라 다름

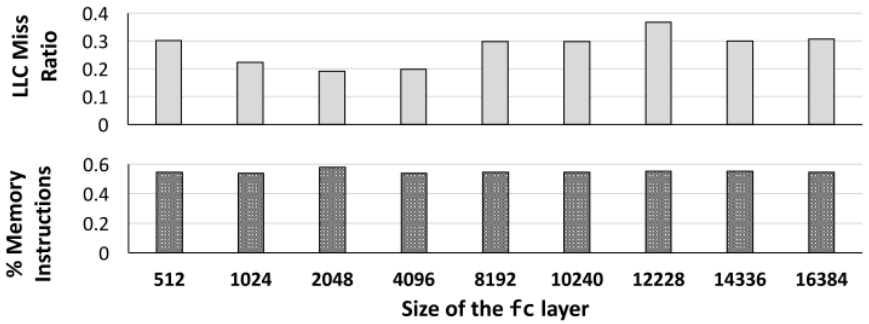
FC Layer

- 모든 입력 값을 각각의 장치에 복사해 각 하위 연산을 진행
- 출력 값을 다음 레이어의 입력으로 활용하기 위해 병합
- 동일한 모델을 사용하면서 장치 당 연산을 줄일 수 있음

FC Layer

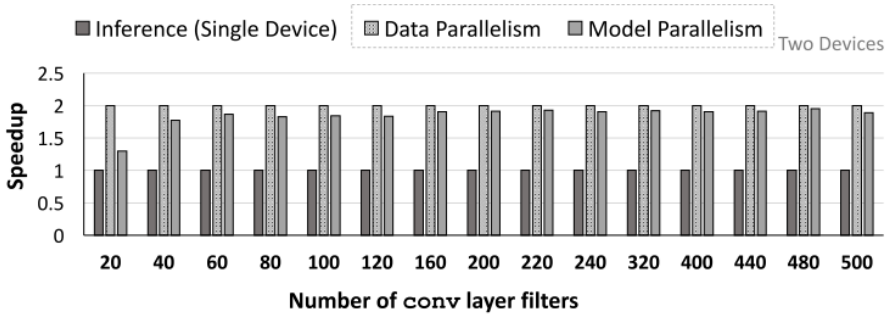
- 데이터 병렬 처리는 두 배의 성능 향상
- 모델 병렬 처리는 절반 크기의 fc layer의 성능에 따라 달라짐
- FC layer의 메모리 명령어 비율과 LLC 미스 비율이 높음
- FC layer의 크기가 10240 이상일 경우 메모리 스왑 공간을 사용
  - 모델 병렬 처리를 이용할 경우 스왑 공간 작업을 피할 수 있음

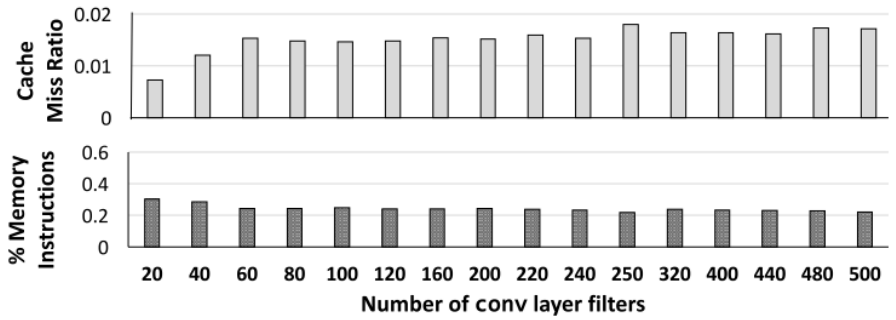




Convolution Layer

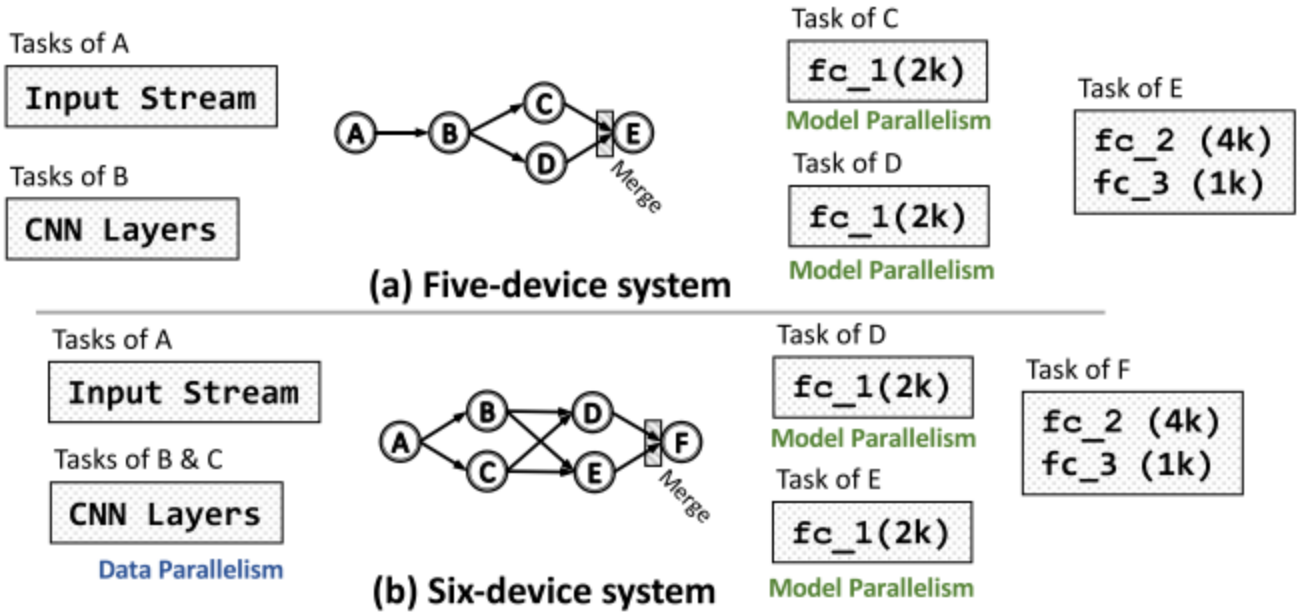
- 일반적으로 입력보다 필터의 수가 크므로
  필터를 분산시켜 병렬 처리를 진행
- LLC 미스와 메모리 명령어 비율이 상대적으로 낮음
- 데이터 병렬 처리가 모델 병렬처리보다 좋은 성능을 보임





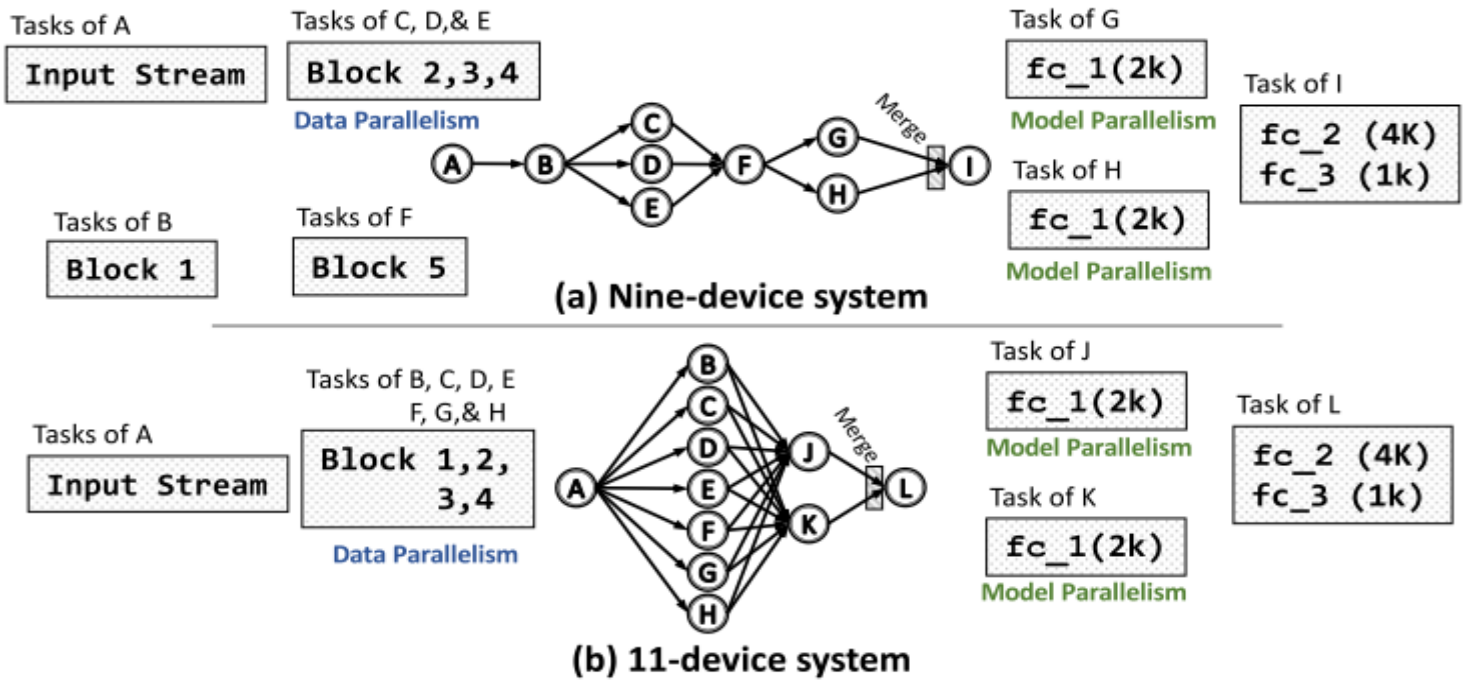
분산 처리 방식

- AlexNet



분산 처리 방식

- VGG16



Action Recognition

- Input Data
- Two-Stream CNN
- Temporal Pyramid(pooling)
- Final Dense Layer

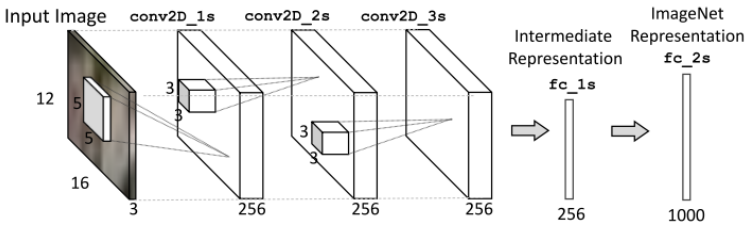


Figure 4: Spatial stream CNN.

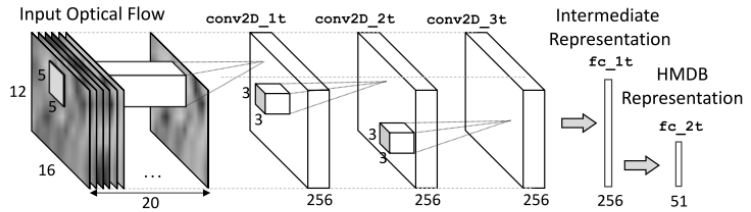


Figure 5: Temporal stream CNN

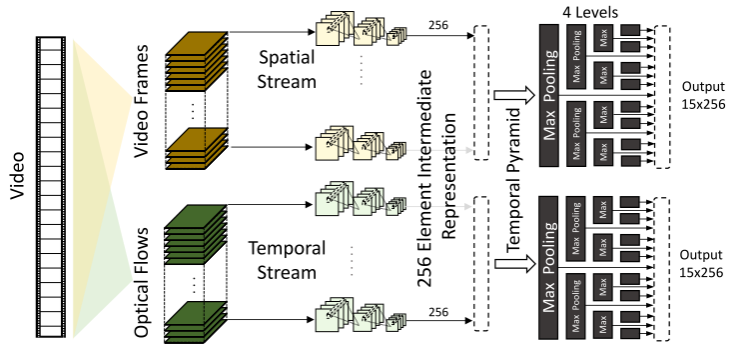
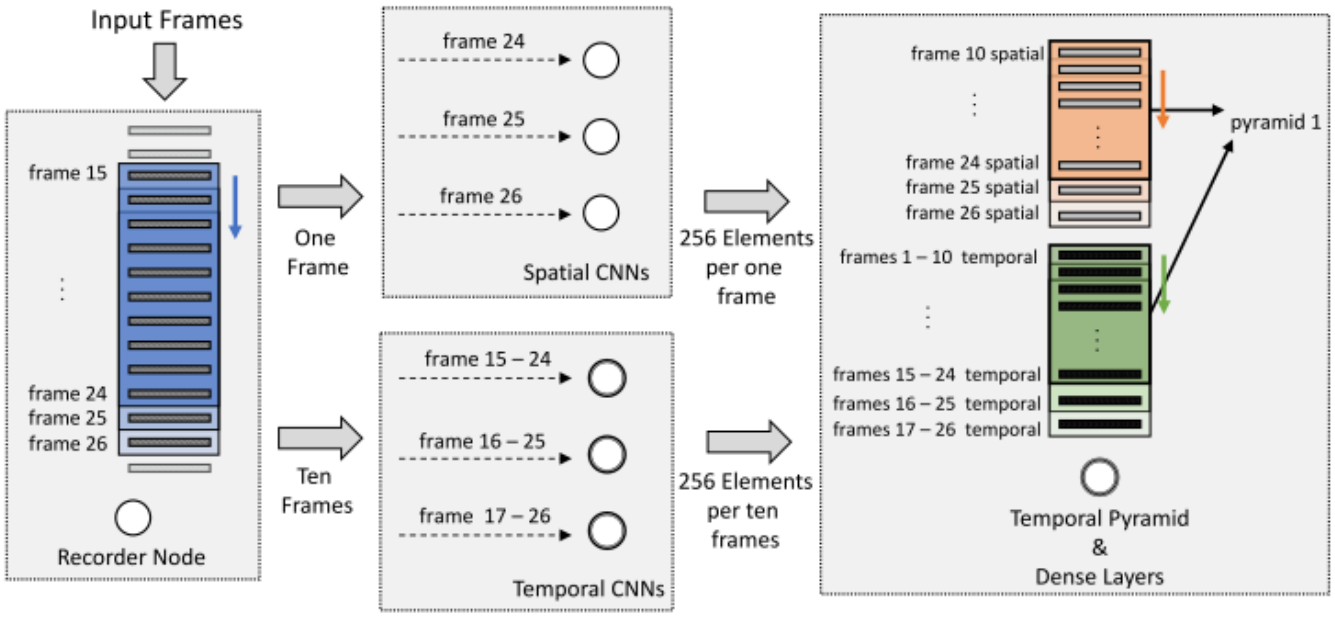


Figure 6: Four-level temporal pyramid generation in two-stream CNN.

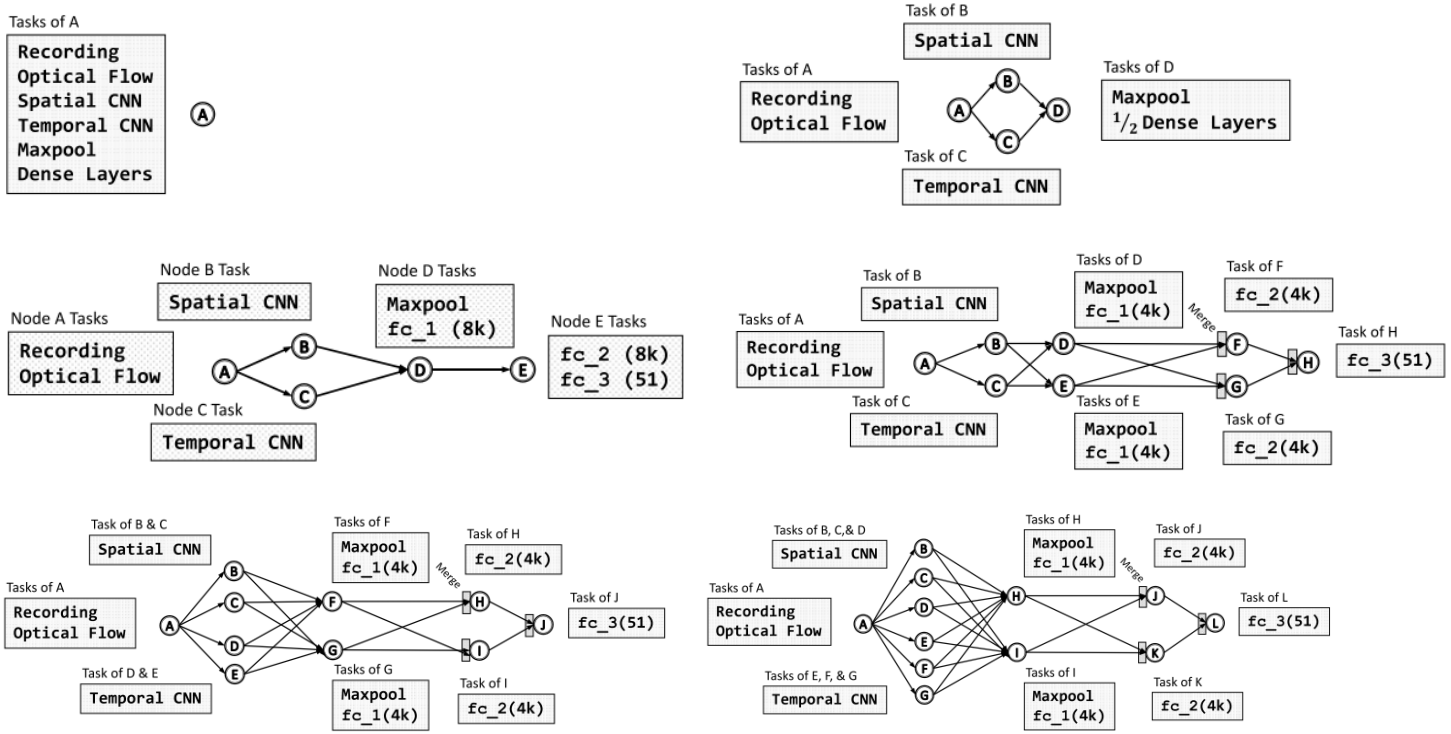
Action Recognition

- 영상의 실시간 처리를 위해 수집된 데이터를 슬라이딩 윈도우로 활용



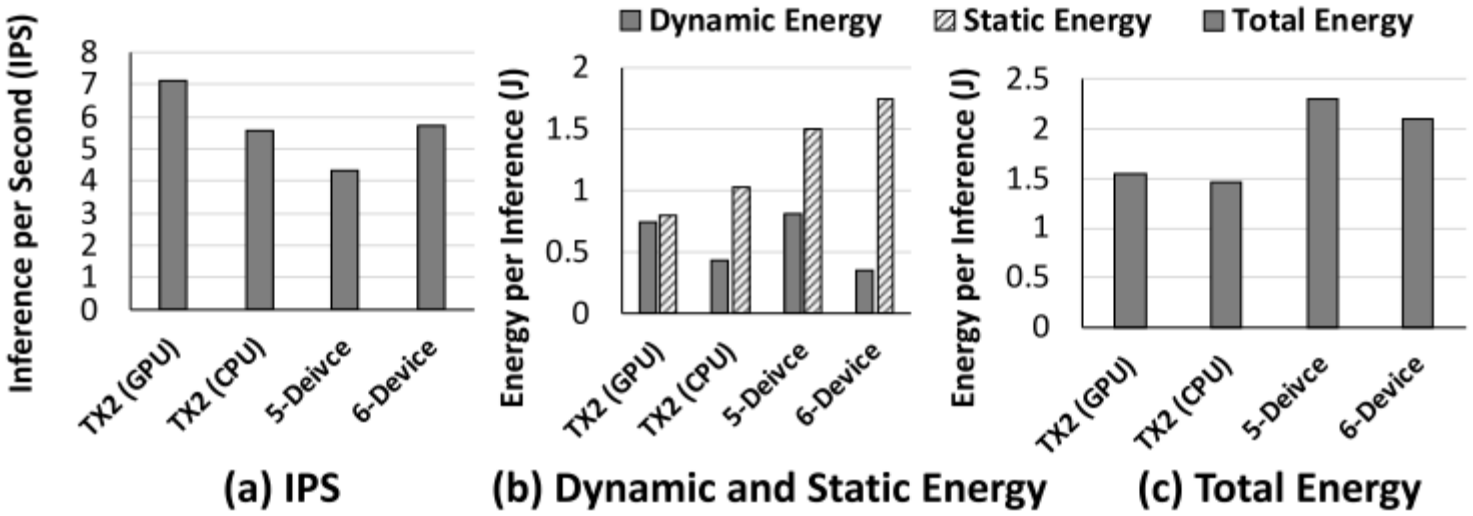
분산 처리 방식

- Action Recognition



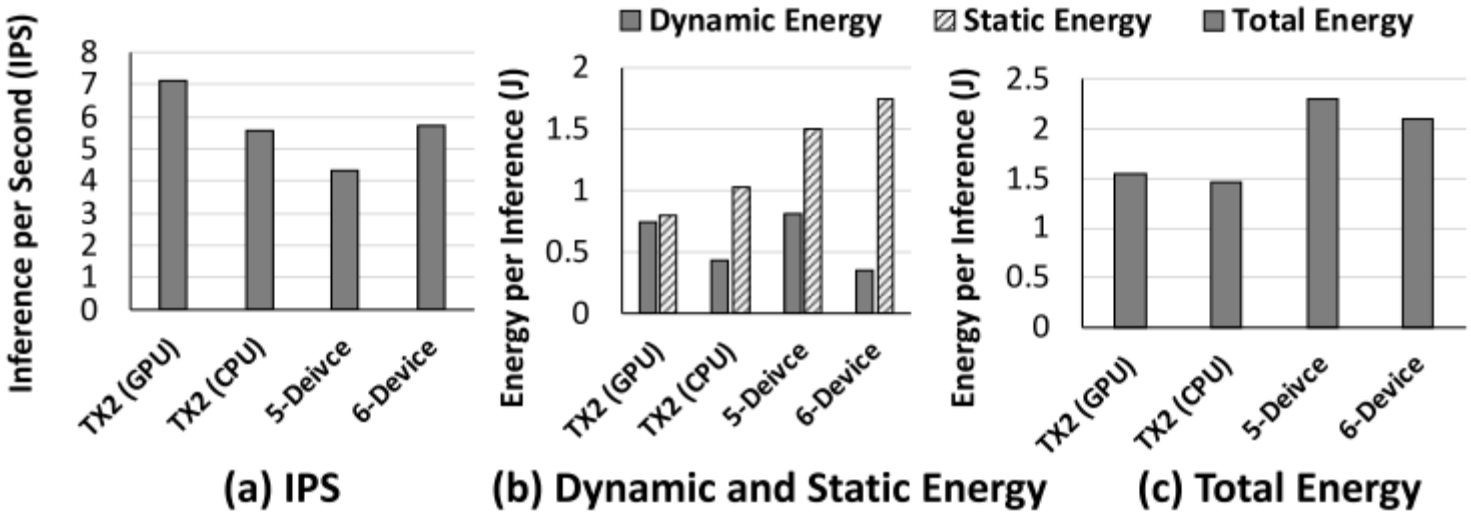
성능 평가

- AlexNet
  - 6개의 장치를 사용했을 때 TX2 CPU와 비슷한 성능
  - TX2 GPU보다는 30% 낮은 성능



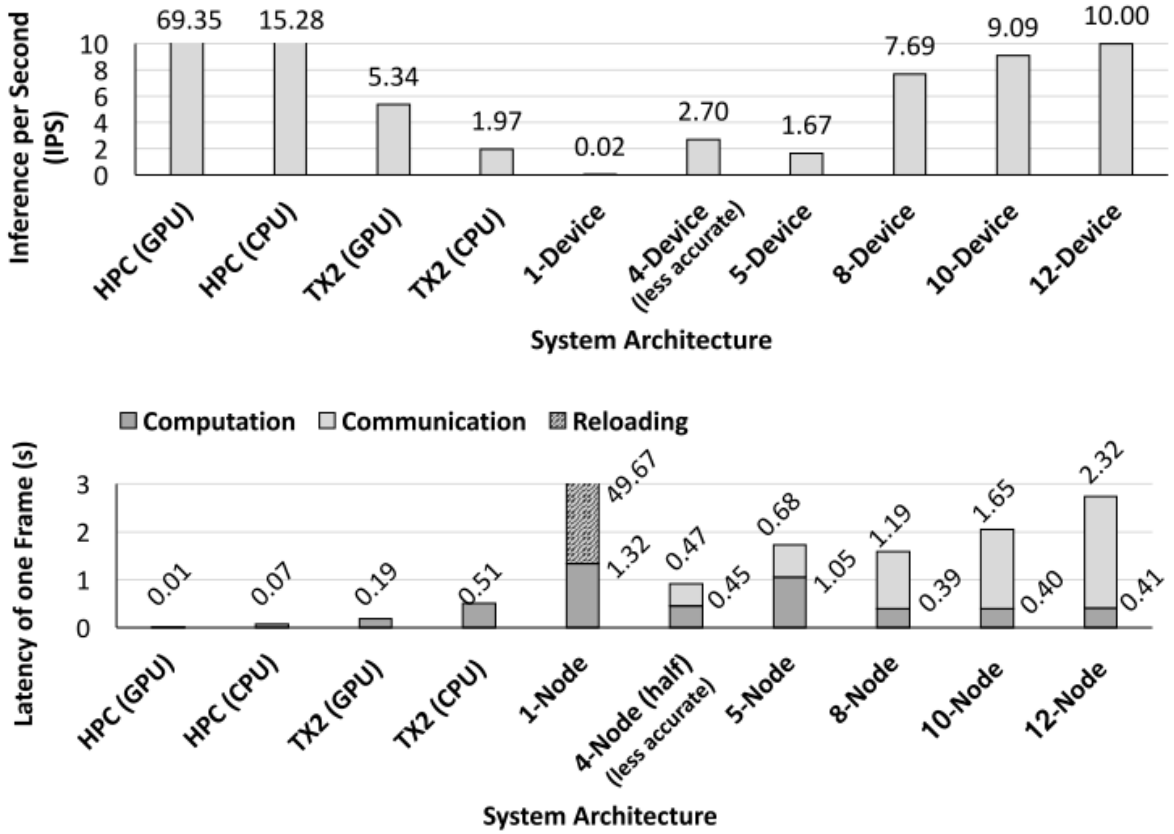
성능 평가

- VGG16
  - 더 많은 장치를 사용하여 CNN에 더 최적화된 data parallelism을 활용
  - 11개의 장치를 사용했을 때 TX2 GPU 대비 15% 낮은 성능



성능 평가

- Action Recognition



성능 평가

- Action Recognition