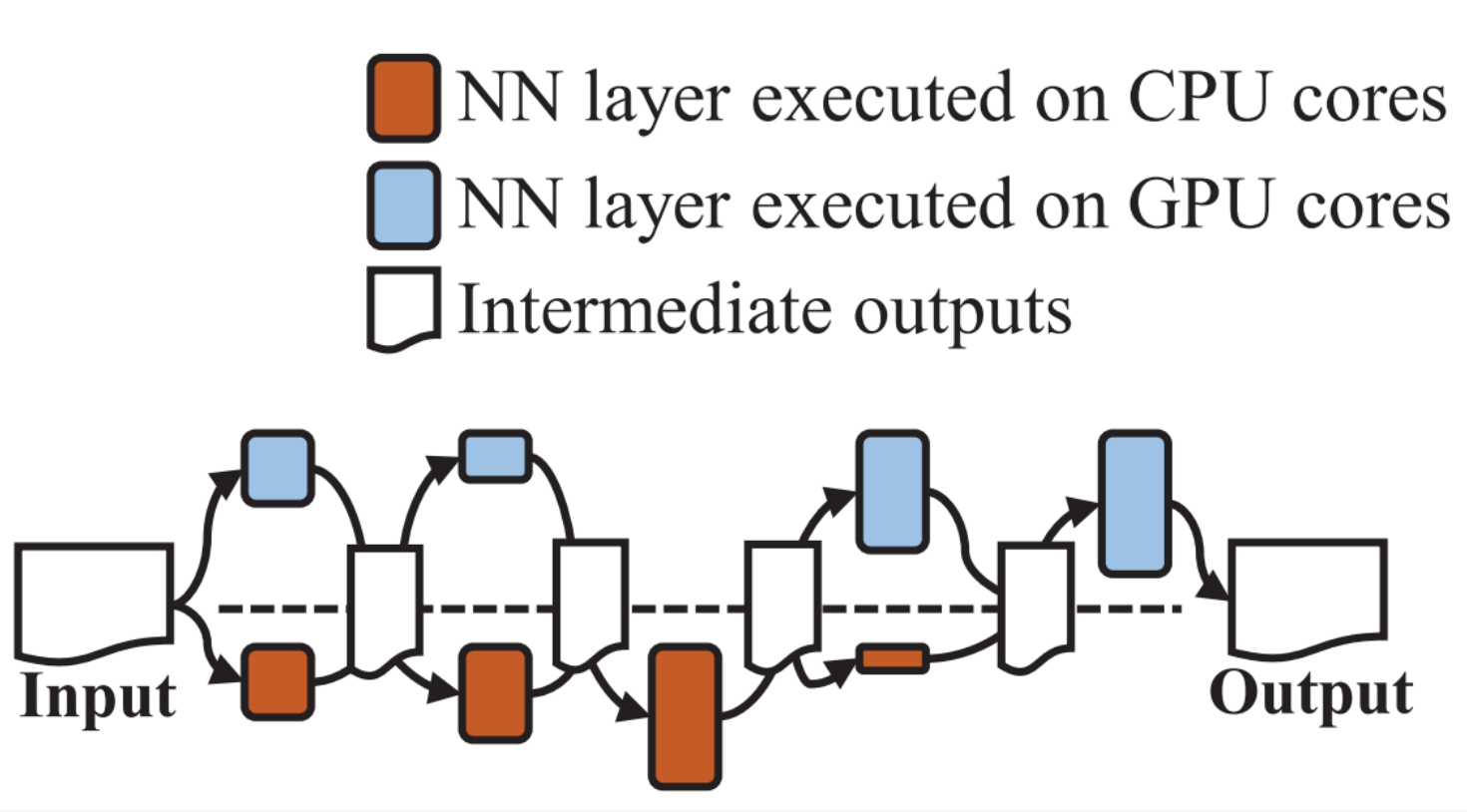
μLayer

: Low Latency On-Device Inference Using Cooperative Single-Layer Acceleration and Processor-Friendly Quantization
Martin Hwang



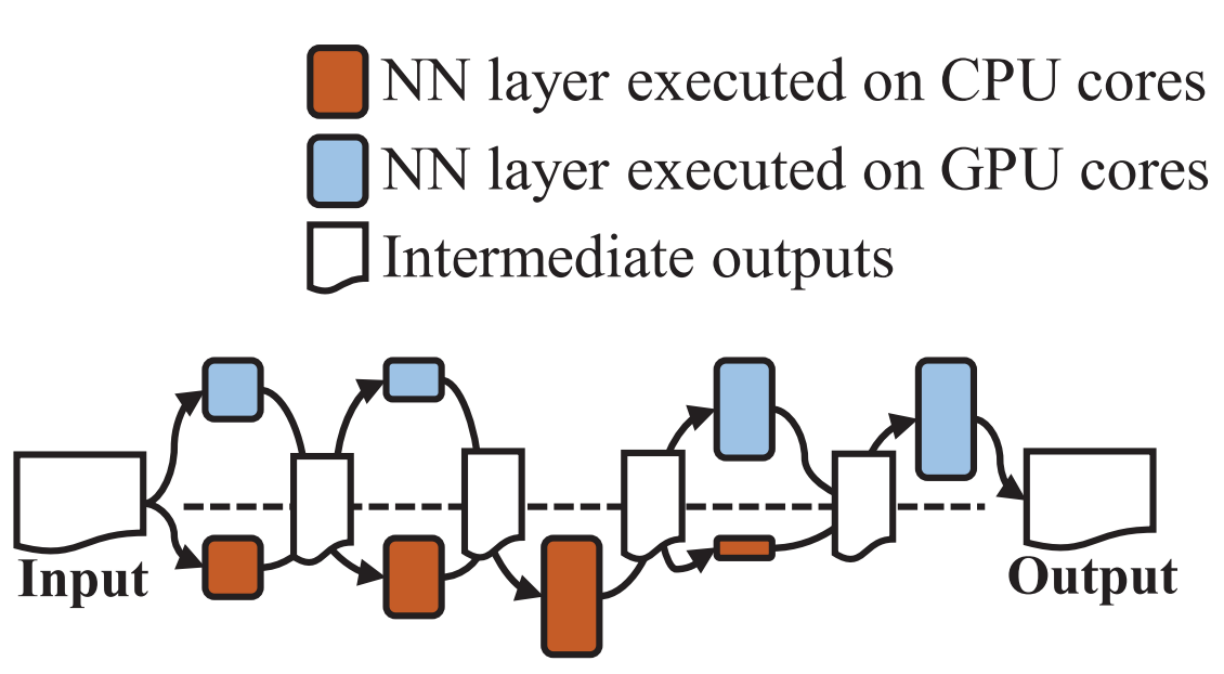
μLayer

핵심 개념: Concept

동기: Motivation

방법론: Detail

실험결과: Experiment



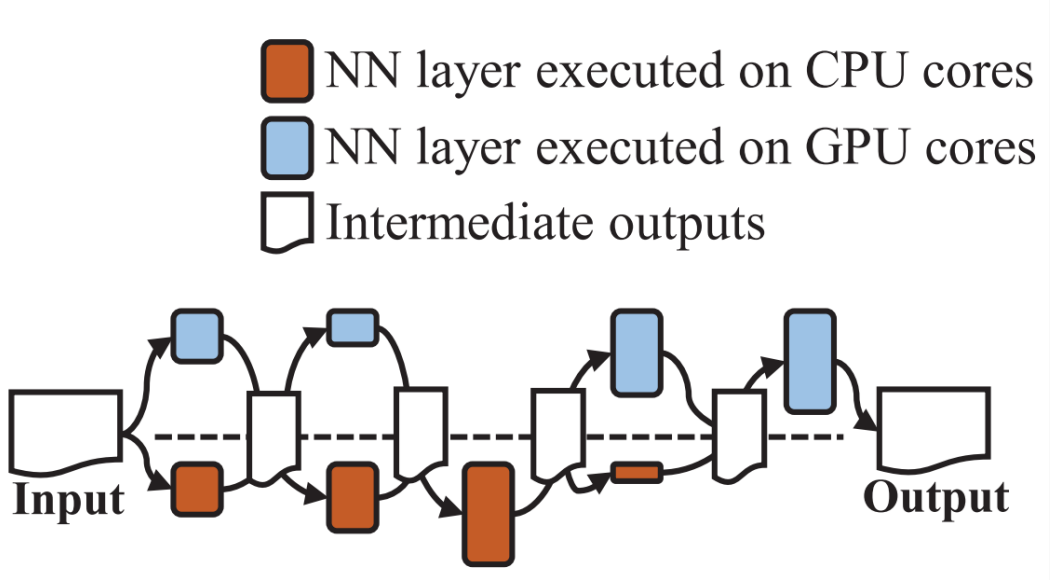
핵심 개념: Concept

이기종 프로세서(CPU, GPU)가 있는 시스템에서

뉴럴 네트워크의 레이어 연산을 채널별로 분리하여

분리된 채널들을 이기종 프로세서가 병렬로 연산하자

*논문의 초첨은 추론 시 연산 지연 최소화



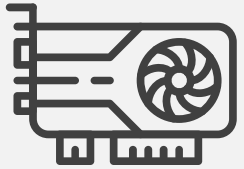
동기: Motivation

실시간 서비스 요구

서비스 실시간 응답 요구

통신 실패

통신 지연에 따른 실시간 응답에 한계
보안 문제



GPU 자원 크게 의존

단일 프로세서 연산 의존 / 클라우드 기반 GPU 의존
연산 지연으로 인한 추론 실패

On-Device에서 Inference를 위해 연산 최적화를 하자

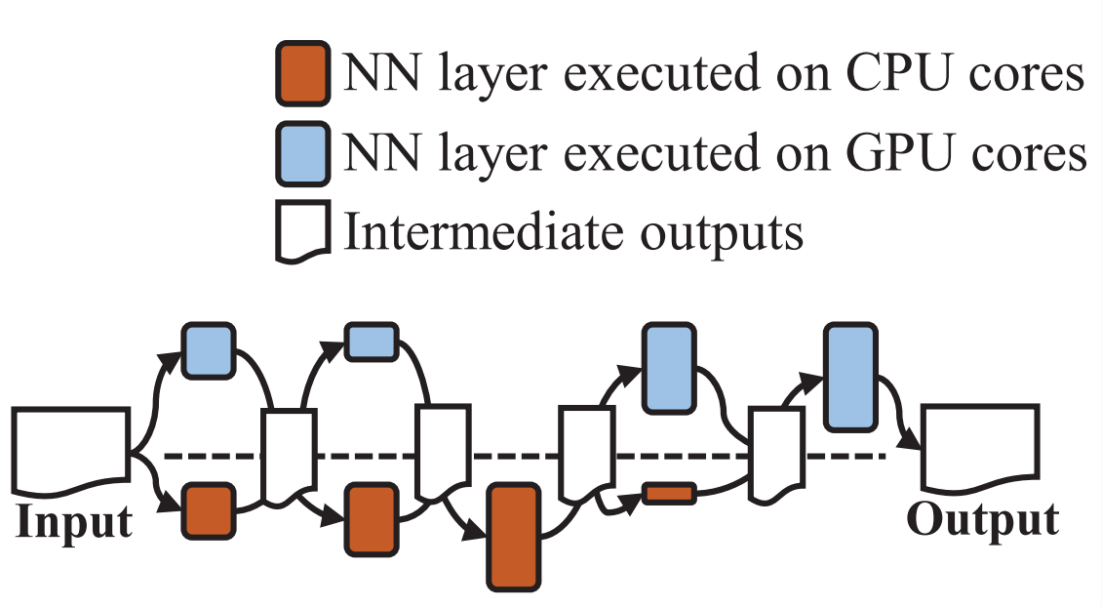
μLayer Detail

Network to Processor Mapping

Layer to Processor Mapping

Cooperative Single Layer Acceleration

Processor Friendly Quantization

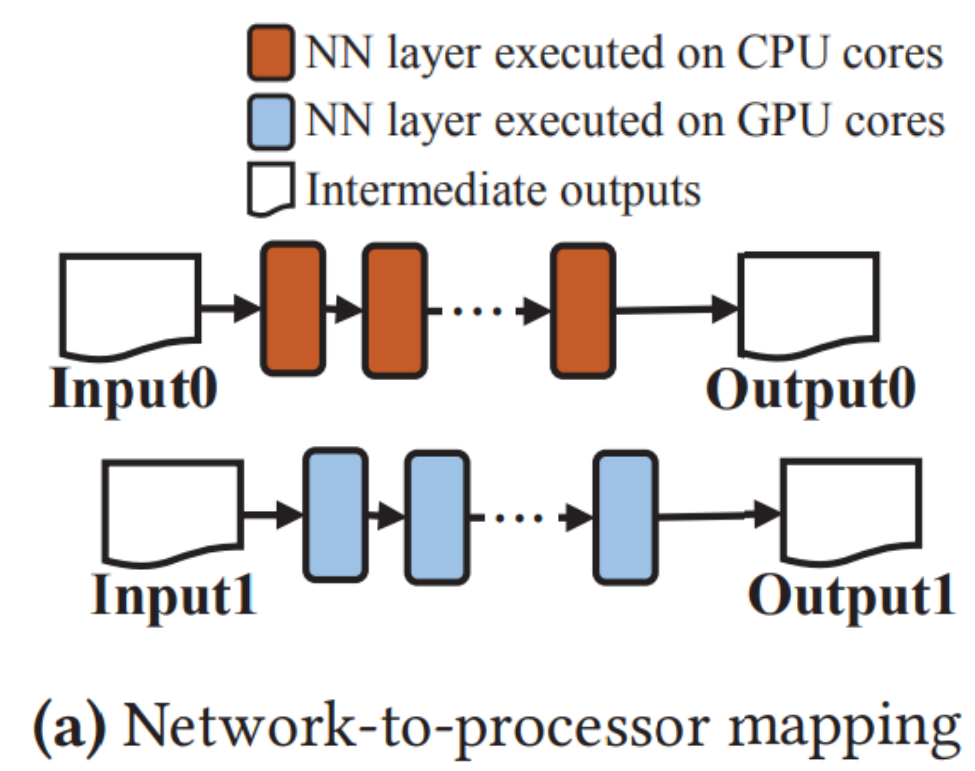


Network to Processor Mapping

- 인공지능 프레임워크가 사용하고 있는 메커니즘

  - 영상 여러 입력을 서로 다른 프로세서에서 처리하는 방식을 Network to Processor Mapping이라고 부름

  - Network to Processor Mapping 메커니즘은 단일 입력에서의 추론 지연에 대한 이득이 없음



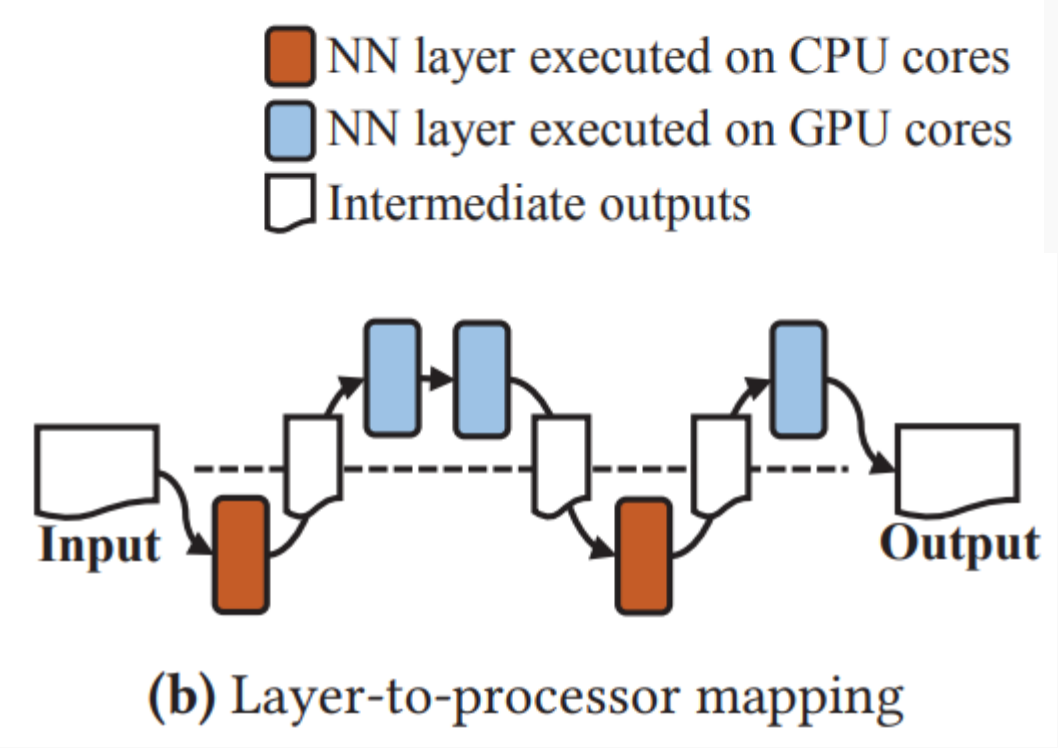
Layer to Processor Mapping

- 거대한 크기의 모델을 분산 학습할 때 사용하는 메커니즘

  - Layer 단위를 서로 다른 프로세서에서 연산하는 방식을 Layer to Processor Mapping이라고 부름

  - Layer to Processor Mapping 메커니즘은 단일 입력에서 Network to Processor Mapping에 비교했을 때 추론 시간에 대한 이득

  - 하지만, 단일 프로세서 성능에 의존



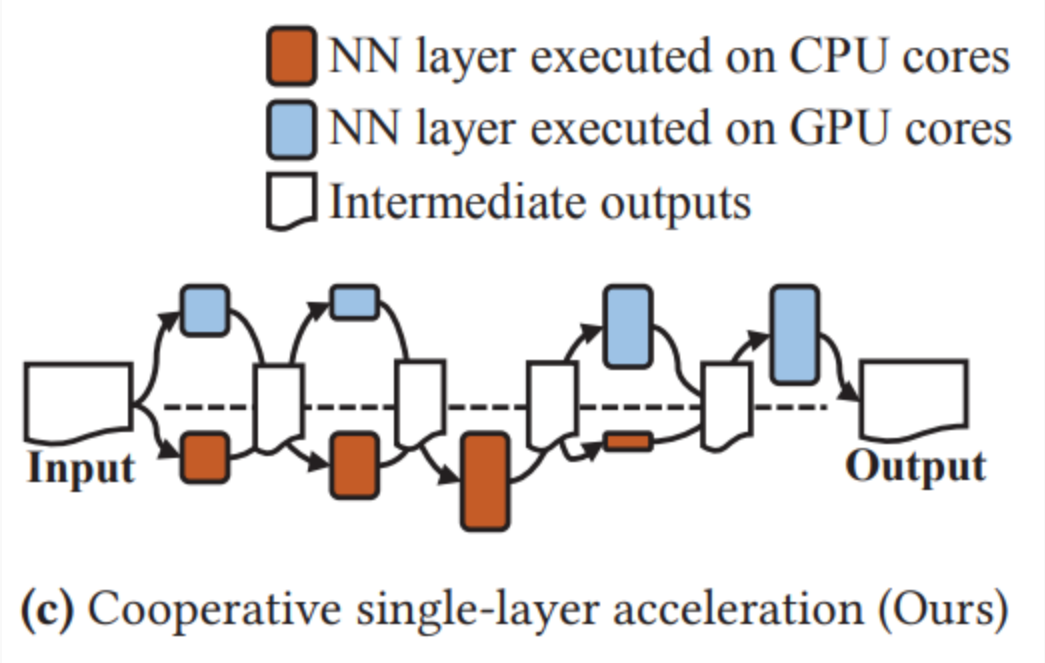
Cooperative Single Layer Acceleration

- Cooperative Single Layer Acceleration

  - 논문에서 제안하는 가속화 방법.

  - 컨볼루션 네트워크에서 나오는 출력 채널을 분할해서 이기종 프로세서별로 계산하는 방식

  - 효율적이지만, 이기종 프로세서의 메모리를 동기화, GPU 명령어 이슈 등과 같은 멀티 프로세서를 관리해야하는 오버헤드가 있음



(c) Cooperative single-layer acceleration (Ours)

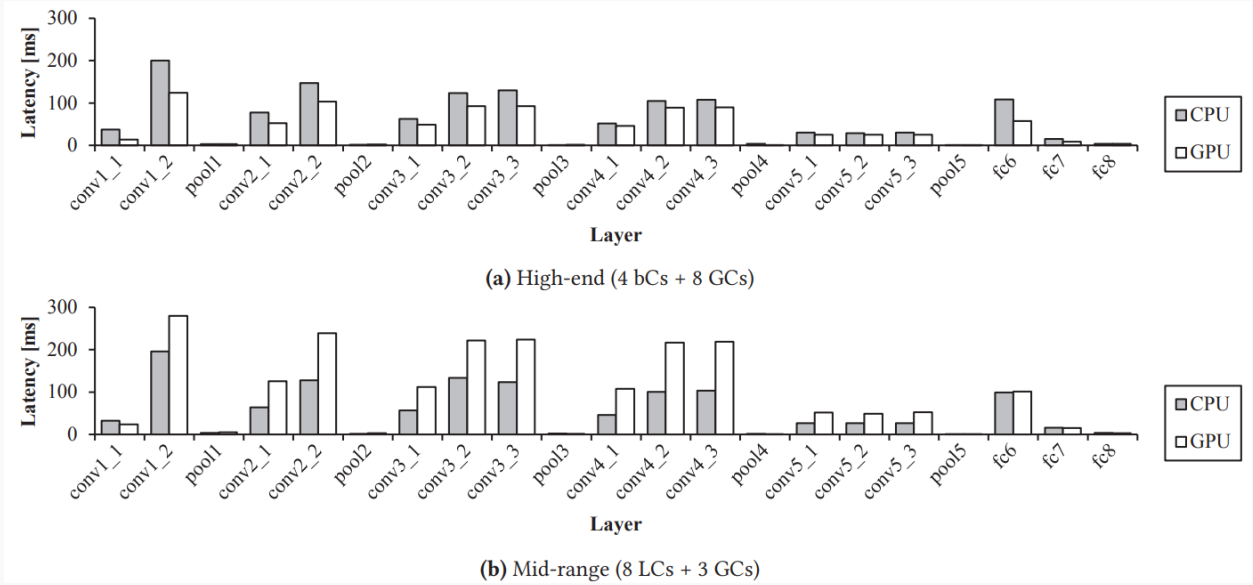
Cooperative Single Layer Acceleration

- Profile the per layer execution latency

  - 멀티 프로세서 오버헤드가 추론성능에 악영향을 끼치는지 확인하기 위해 VGG-16의 레이어 별 실행 시간을 확인(Samsung Exynos 7420, 7880)

  - 각 SoC(CPU, GPU)마다 레이어 별 연산속도는 크게 차이 나지 않음(ARM Compute Library 기준)

  - 저전력 GPU로 인해 high-end SoC(Exynos 7420)의 경우, GPU는 CPU보다 1.40배 빠르며, mid-range SoC (Exynos 7880)의 경우 octa-core CPU가 26.1%정도 연산이 더 빠른 것을 확인할 수 있었음



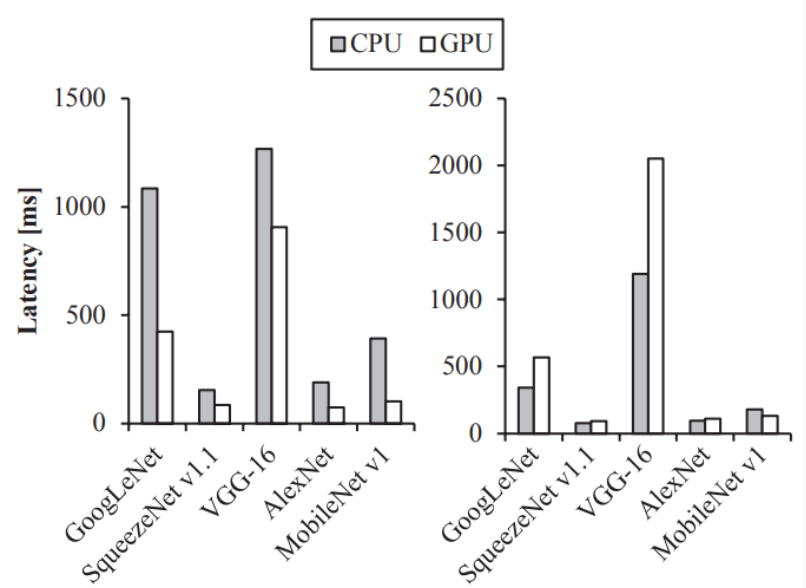
(a) High-end (4 bCs + 8 GCs)

(b) Mid-range (8 LCs + 3 GCs)

Cooperative Single Layer Acceleration

- Profile the per layer execution latency

  - high-end SoC(Exynos 7420) 특성과 mid-range SoC (Exynos 7880)의 특성은 다른 뉴럴 네트워크(GoogLeNet, SqueezeNet, AlexNet, MobileNet)에서도 같다는 것을 확인



(a) High-end (4 bCs + 8 GCs) **(b)** Mid-range (8 LCs + 3 GCs)

Figure 6. NN execution latency on modern SoCs

Cooperative Single Layer Acceleration

- Channel Wise Workload Distribution
  - 컨볼루션 네트워크의 필터는 모든 입력 데이터 Channel 간 공유됨
  - CPU, GPU간 연산을 담당할 채널의 비율은 $p_{cpu} = (1 - p_{gpu})$로 계산
  - 각 SoC에서 연산된 출력 채널은 병합됨
  - 풀링 레이어도 같은 방식으로 동작

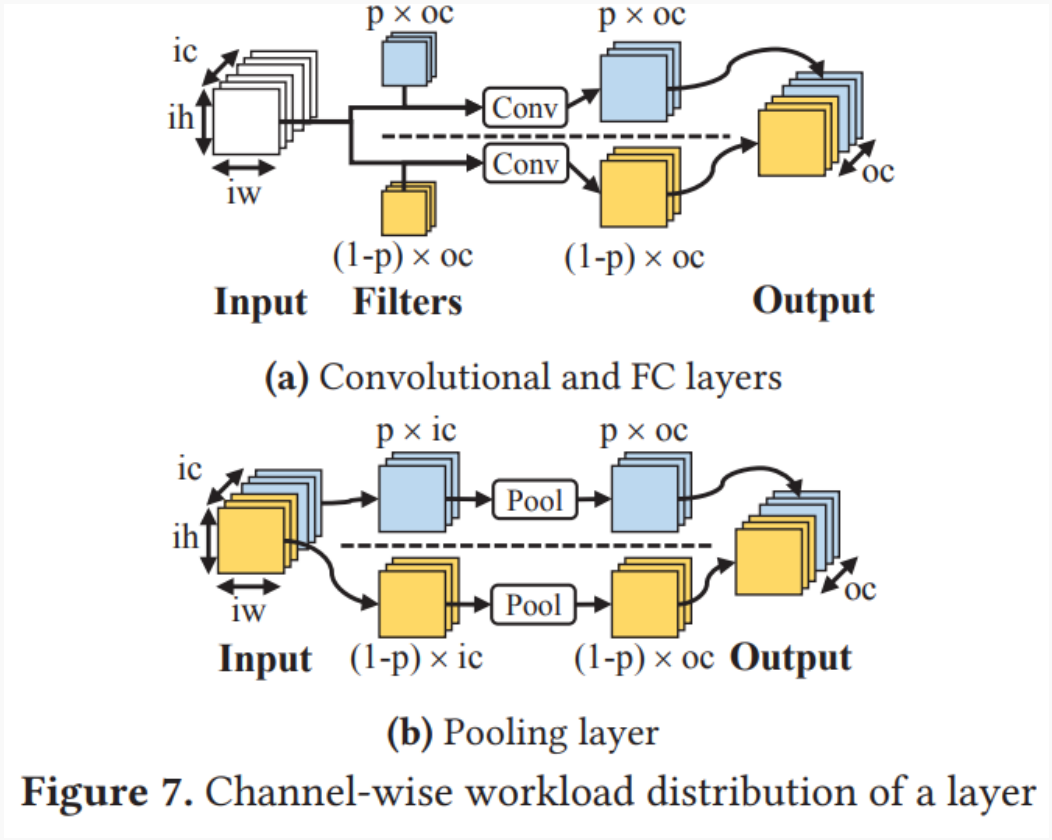


Figure 7. Channel-wise workload distribution of a layer

Processor Friendly Quantization

- **Bit Quantization**

  - SoC와 상관없이 네트워크의 Weights가 Quantization이 되었을 때 연산속도가 증가하고, 정확도는 많이 떨어지지 않는다는 사실이 알려져있음

  - 하지만 각각의 SoC에 따라서 적합한 Bit Quantization 기법이 다름

  - 프로세서의 성능을 최적화하기 위해 16-bit half precision floating point(F16) / 8-bit linear quantization(QUInt8)에 대해 분석함

  - GPU는 그래픽 렌더링 연산이라는 장치 특성으로 인해 F16이, CPU는 8-bit integer를 병렬 처리할 수 있는 ALUs를 가지고있어 bit quantization마다 성능 특성이 다름

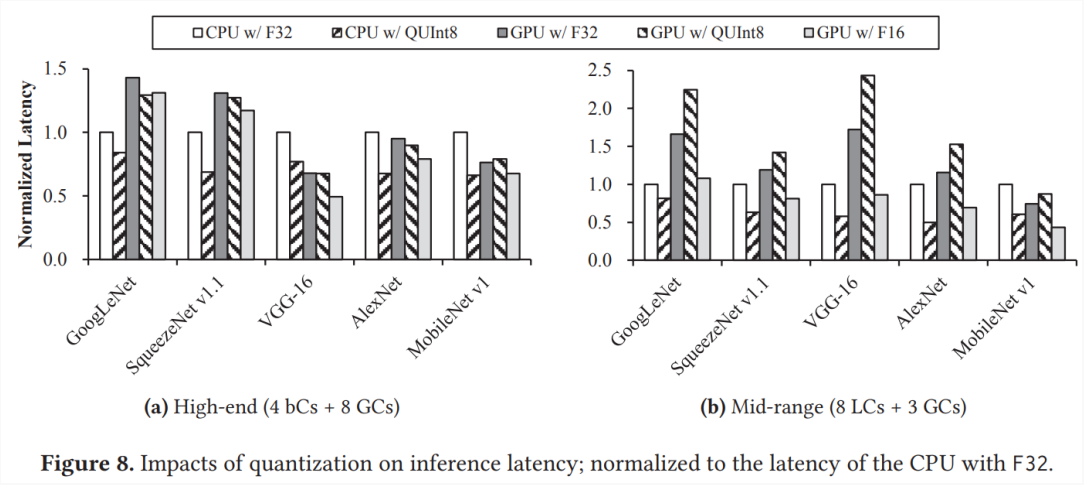


Figure 8. Impacts of quantization on inference latency; normalized to the latency of the CPU with F32.

Processor Friendly Quantization

- 16 / 8 Bit Quantization

  - Bit Quantization이 데이터를 표현하는 공간을 줄여주나, 8-bit Quantization의 경우 부작용으로 Fully Connected Layer에서 8-bit간의 연산으로 32-bit integer를 필요로 하게 됨(multiply, accumulating)

  - GPU는 8-bit Quantization을 하게 될 경우 앞서 이야기한 이유로16-bit integer를 32-bit integer를 사용해 누적연산을 하며, 이는 16bit operation에 비해 동시성이 절반으로 떨어져 비효율적

  - CPU의 경우 F16을 연산하는 ALU 지원부족과 QUInt8에 대해 지원되는 벡터 넓이가 부동 소수점 보다 넓어 QUInt8에서 큰 이득이 있음

  - CPU에서 F16기반으로 연산을 하게 되면 F16에 대한 ALU지원 부족으로 F32으로 강제 변환하여 연산하게 되며 이는 뉴럴 네트워크 연산 단위인 F32라 크게 다를게 없음

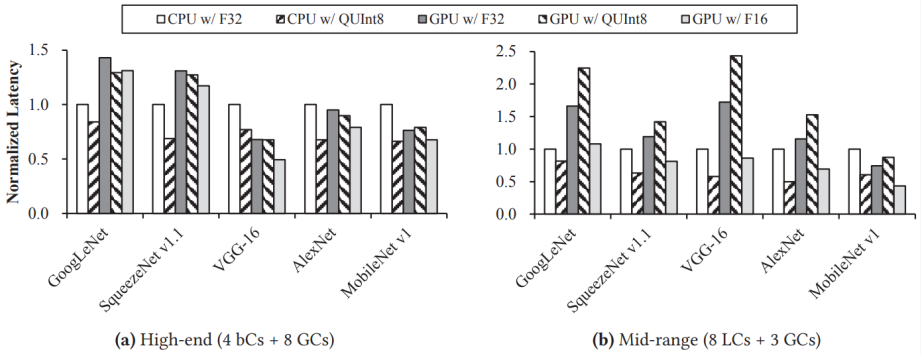


Figure 8. Impacts of quantization on inference latency; normalized to the latency of the CPU with F32.

Processor Friendly Quantization

- Maximizing Per Processor Throughput
  - 입력 데이터와 필터, 출력 데이터는 QUInt8로 저장하여 메모리 크기를 최소화함
  - CPU는 8-bit 연산을 GPU는 F16으로 변환하여 계산함

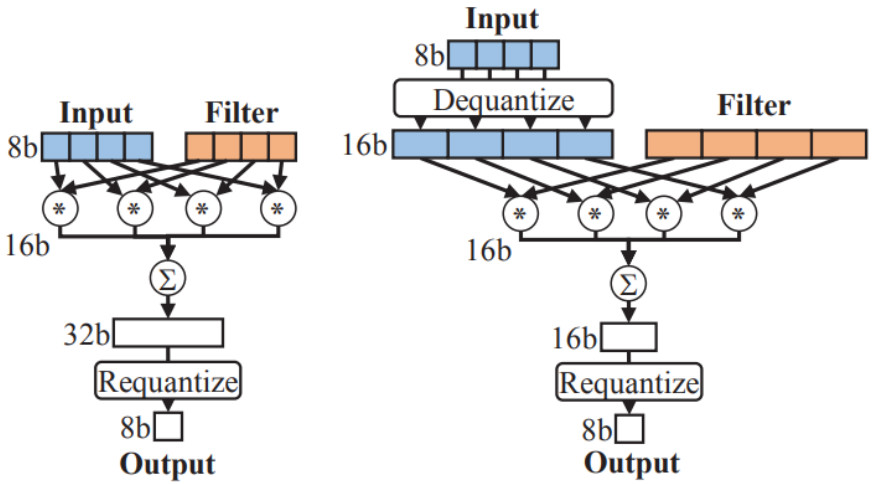


Figure 9. Processor-friendly quantization

Processor Friendly Quantization

- Impacts on Inference Accuracy
  - Quantization시 발생하는 Accuracy 하락에 대해서 네트워크별로 분석함
  - 크게 발생하는 Accuracy Loss를 막기 위해 8-bit linear quantization으로 네트워크를 재학습함
  - 그 결과 최대 Accuracy Loss가 2.7%를 달성함

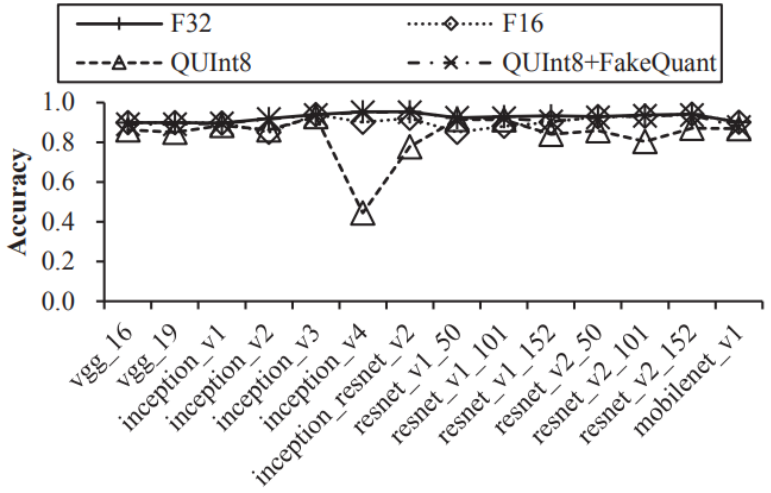
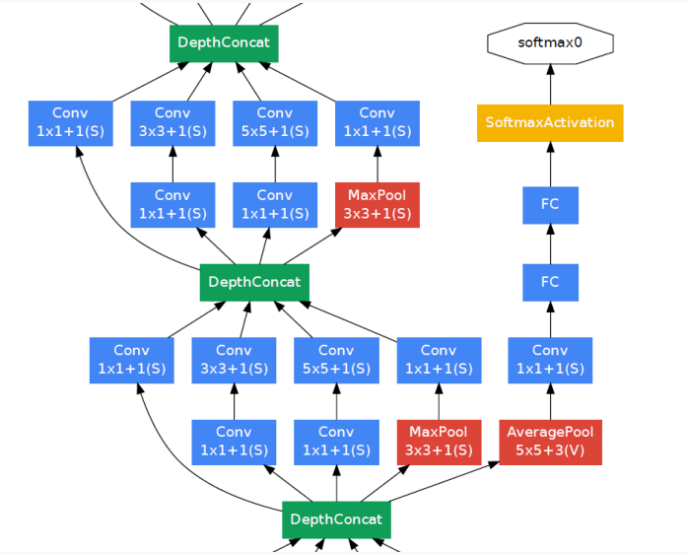


Figure 10. Impacts of quantization on the top-5 classification accuracy with the ImageNet dataset [62]

Branch Distribution

- GoogLeNet
  - 몇몇 뉴럴 네트워크 아키텍쳐들은 서브 매크로 아키텍쳐를 가지고있음
  - 이런 매크로 아키텍쳐에서 일부 브랜치들은 컨볼루션 레이어의 연산 비용이 비쌈
  - 이로 인해 레이어간 연산 지연시간이 서로 다르게 되며, 이는 Channel wise workload distribution이 CPU, GPU간 동기화 오버헤드를 증가 시킴



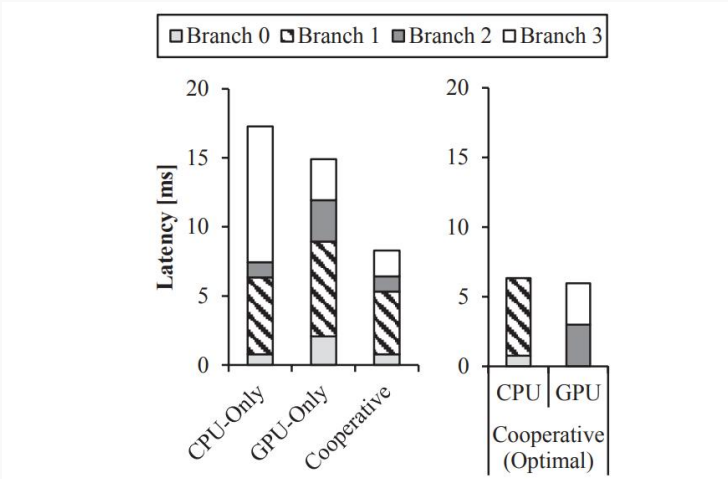
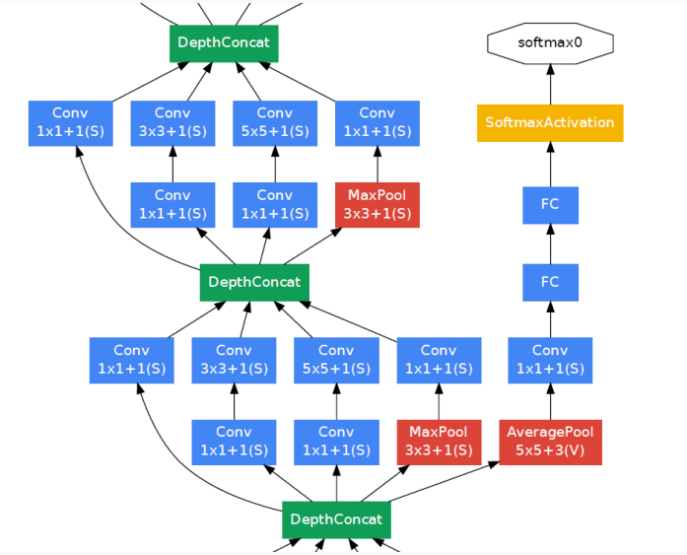


Figure 12. Potential latency benefits of branch distribution

Branch Distribution

- Branch Distribution
  - 연산 지연시간이 서로 다른 브랜치들에서 발생하는 CPU, GPU간 동기화 오버헤드를 최적화하기 위해 Branch Distribution을 적용
  - 먼저, 분기가 가능한 브랜치들을 식별함
  - 식별된 브랜치들을 CPU와 GPU로 분할해서 병렬처리를 진행



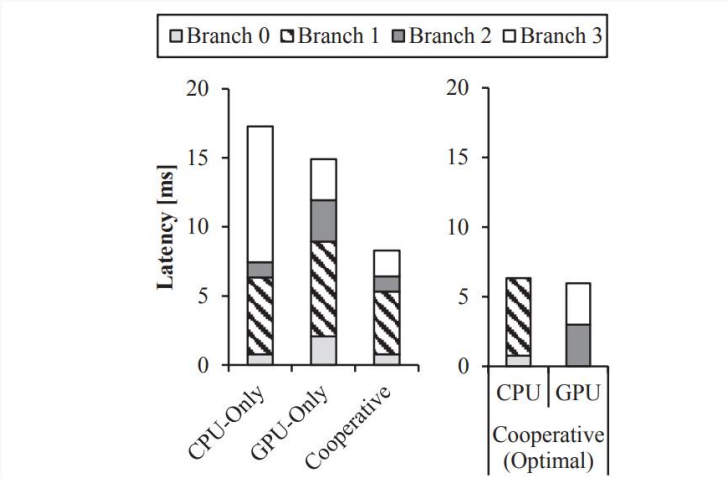


Figure 12. Potential latency benefits of branch distribution

Implementation

- **µLayer Architecture**
  - µLayer의 아키텍쳐는 Partitioner / Latency Predictor / Executor로 총 세 가지 컴포넌트로 구성됨
  - Paritioner는 Cooperative execution plan을 위해 네트워크의 레이어의 파라미터에 기반해서 CPU와 GPU가 담당할 Layer의 비율을 계산함
  - Latency Predictor는 레이어의 연산 지연시간을 추정함. 이를 위해 Neurosurgeon을 확장함
  - NN Partitioner의 Execution plan을 받아 컨볼루션 필터의 값을 CPU, GPU에 로드 / 뉴럴 네트워크 연산 수행을 진행

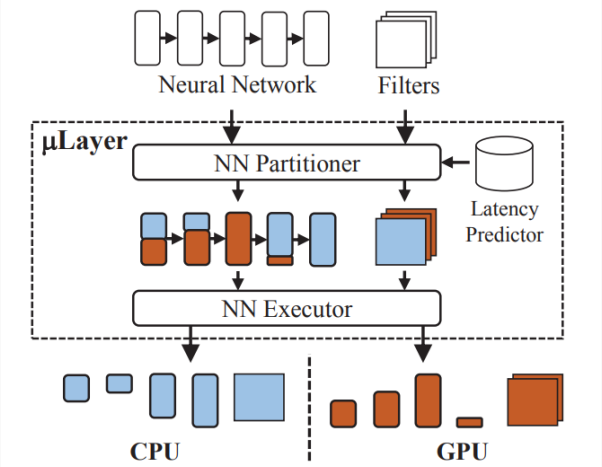


Figure 13. µLayer runtime architecture

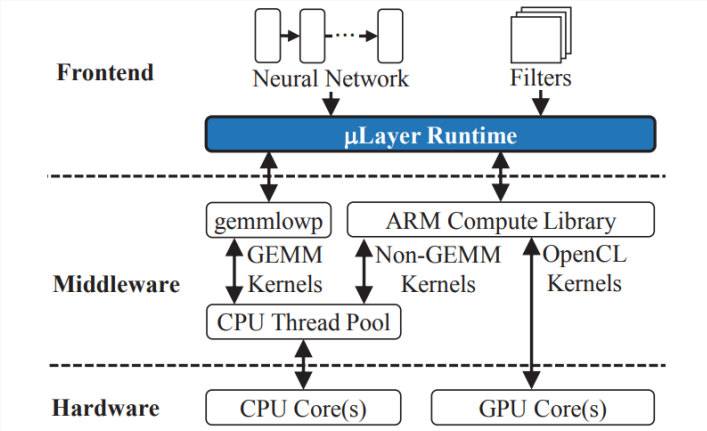


Figure 14. µLayer-augmented mobile NN framework

Implementation

- **µLayer Architecture**

  - CPU-GPU간 메모리 복사를 최소화하기 위해 Mobile SoC의 CPU-GPU가 함께 공유할 수 있는 공유 메모리를 사용

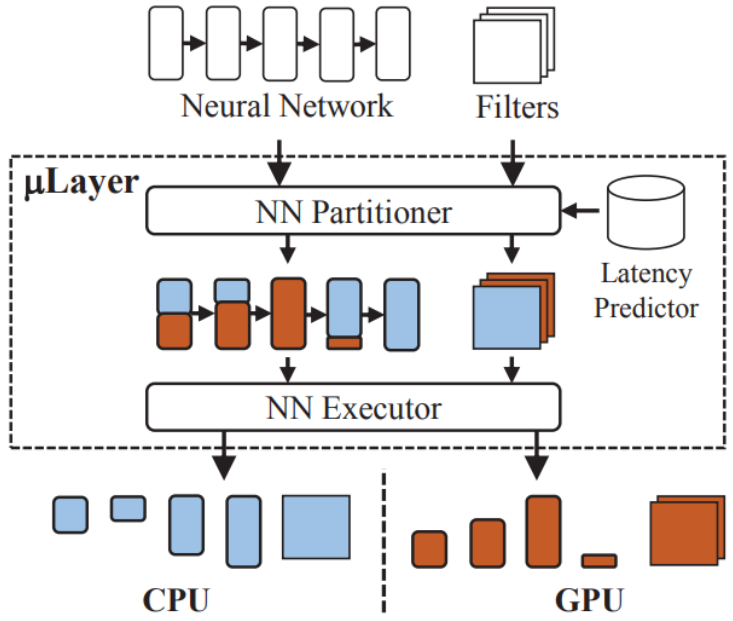


Figure 13. µLayer runtime architecture

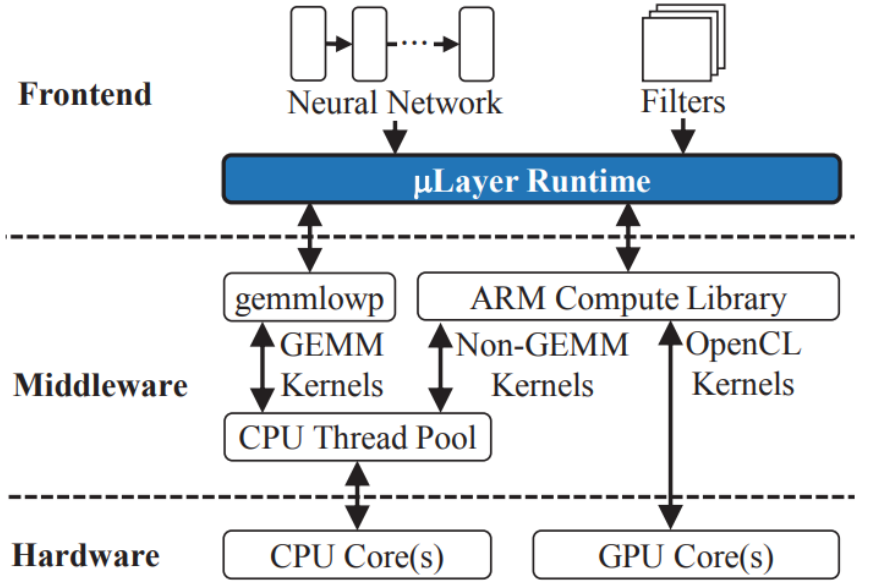
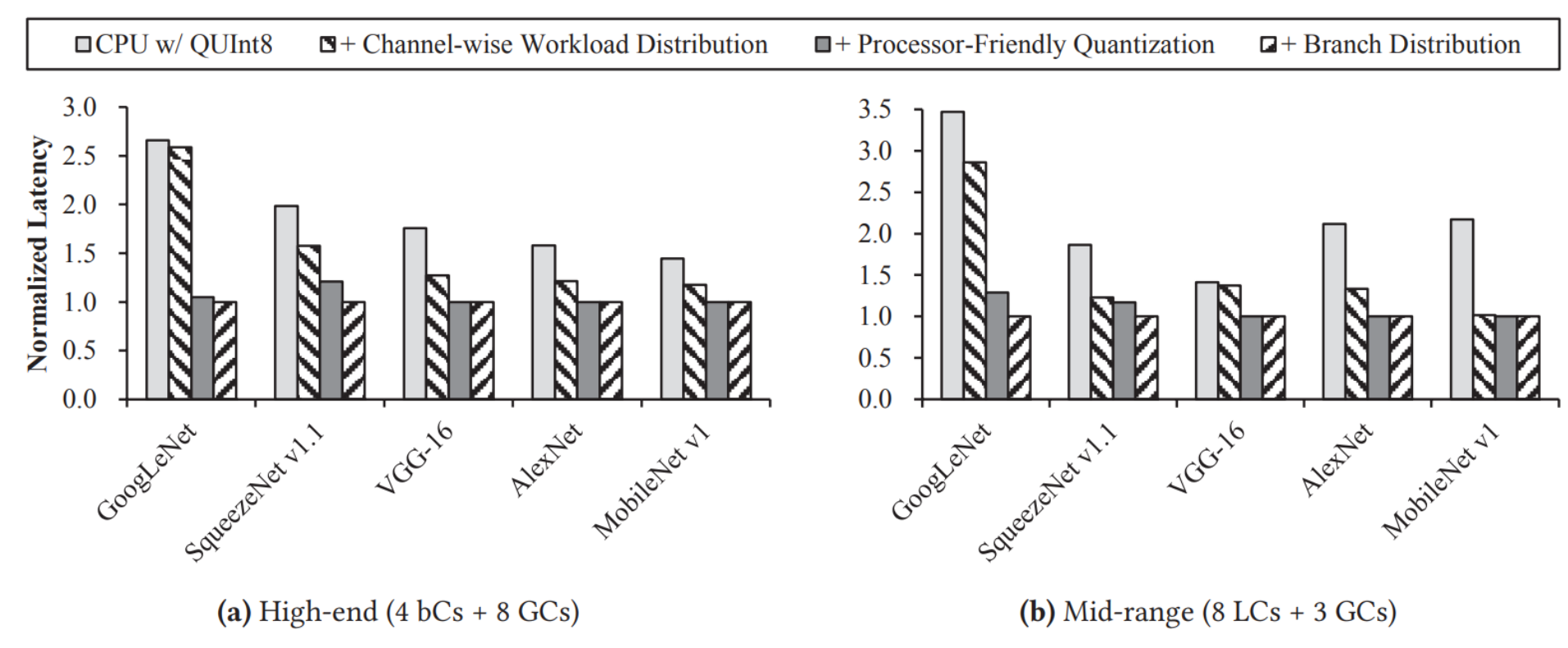


Figure 14. µLayer-augmented mobile NN framework

Experiment

- Execution Latency

  - µLayer가 high-end SoC에서는 59.9%, mid-range SoC에서는 69.6%의 실행 시간 감소가 있었음



(a) High-end (4 bCs + 8 GCs)

(b) Mid-range (8 LCs + 3 GCs)

Experiment

- Low Energy Consumption

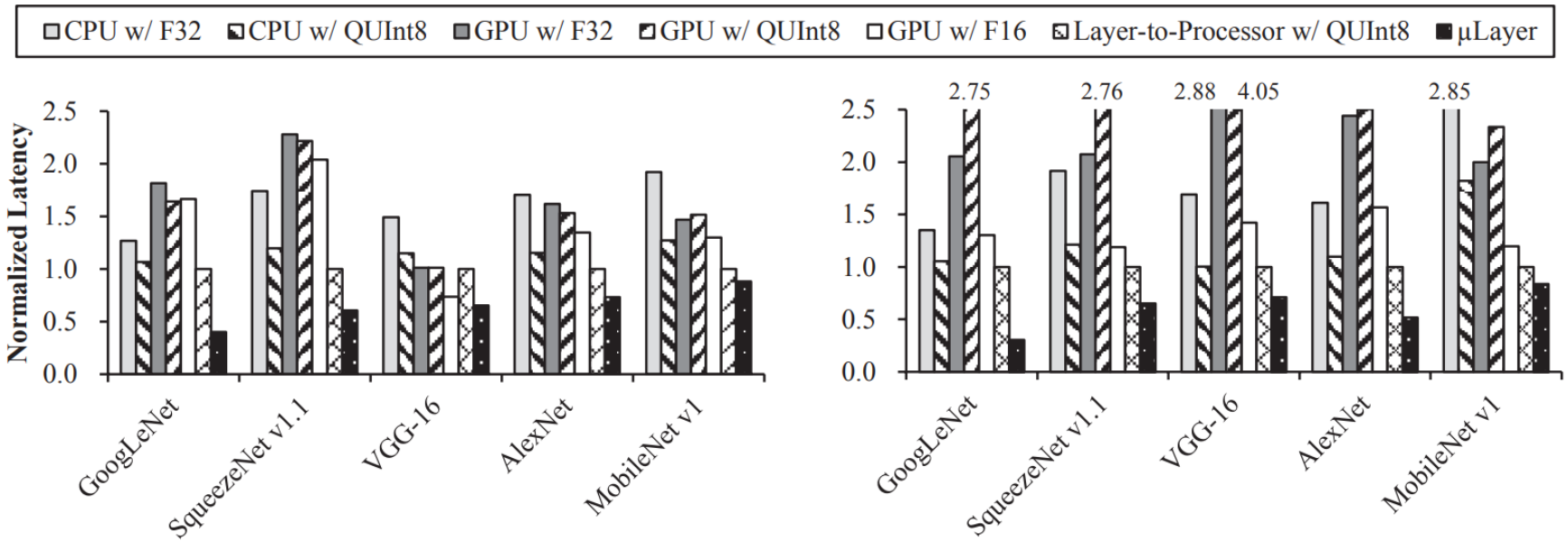
  - 전력 소모를 측정하기 위해 Monsoon Solutions의 HVPM(High Voltage Power Monitor)를 핸드폰의 배터리부와 납땜

Figure 15. Our experimental setup to measure the energy consumption of mobile SoCs

Experiment

- **Low Energy Consumption**

  - µLayer가 적은 실행시간, QUInt8로 인한 작은 메모리 대역폭 사용으로 인해 geometric mean으로 high-end SoC에서 1.26배, mid-range SoC에서1.34배 효율적



(a) High-end (4 bCs + 8 GCs) **(b)** Mid-range (8 LCs + 3 GCs)

Figure 16. NN execution latency of the single-processor mechanism, the layer-to-processor mechanism, and µLayer; normalized to the latency of the layer-to-processor mechanism.

감사합니다