Machine Learning at Facebook: Understanding Inference at the Edge

ModuLABs, Neural Acceleration LAB
이상록

Introduction

- 페이스북에서 사용되는 ML 모델들
  Ranking, Content understanding, Object detection, Tracking, VR etc
- 90%의 광고수익이 모바일에서 나오기 때문에 모바일 사용자 경험이 매우 중요
- 딥러닝을 안정적으로 인퍼런스하고 응답시간을 줄이는 것 등

Introduction

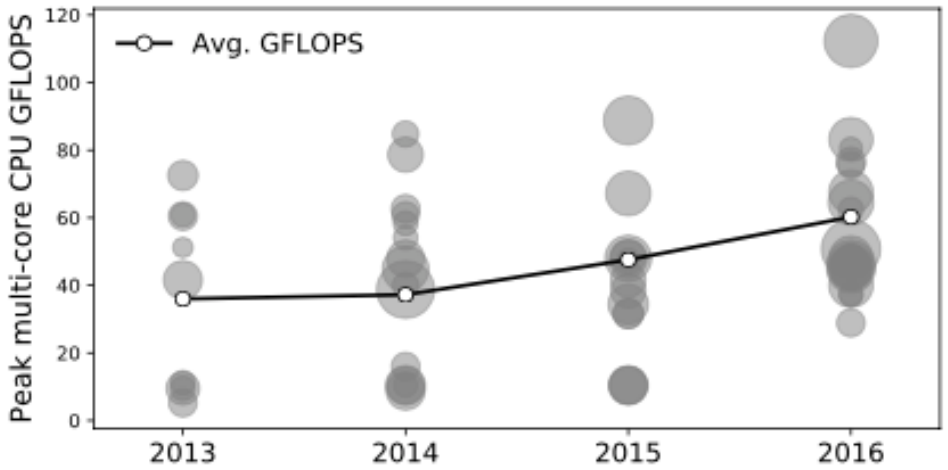


Figure 1: The distribution of peak performance of smartphone SoCs running Facebook mobile app exhibit a wide spread. The data samples represents over 85% of the entire market share and are sorted by the corresponding SoC release year. Peak performance can vary by over an order of magnitude, increasing the design challenge of performance optimization.

연도별 System on a Chip(SoC) Peak performance 분포

분산이 크다
최대한 모든 칩에서 좋은 퍼포먼스를 내야 한다

Introduction

- 모든 Optimization 수단을 동원
  Model architecture search
  Weight compression
  Quantization, algorithmic complexity reduction
- 현실
  SOTA 모델을 쓸 여유가 없다
  매우 적은 수의 Device만 gpu가지고 있음
  GPU가 CPU보다 성능이 잘 나오는 경우도 20% 미만

Introduction

- 대부분이 오래된 low-end cpu 사용하며 gpu 성능이 cpu에 미치지 못함
- 시스템이 다양한 상황에서 co-processor를 고려해 포팅하기는 매우 어렵다. 하지만 시스템 환경을 컨트롤 할 수 있거나 iphone처럼 다양성이 적다면 시도해 볼 만 하다
- Co-processor를 사용하는 이유는 안정성과 에너지 효율성 때문이다(속도는 부차적인 효과)
- 실제 퍼포먼스 평가를 하면 성능이 벤치마크보다 나빠지는데 유저가 사용하면서 겪는 다양한 문제들 때문

THE LAY OF THE LAND: A LOOK AT SMARTPHONES FACEBOOK RUNS ON

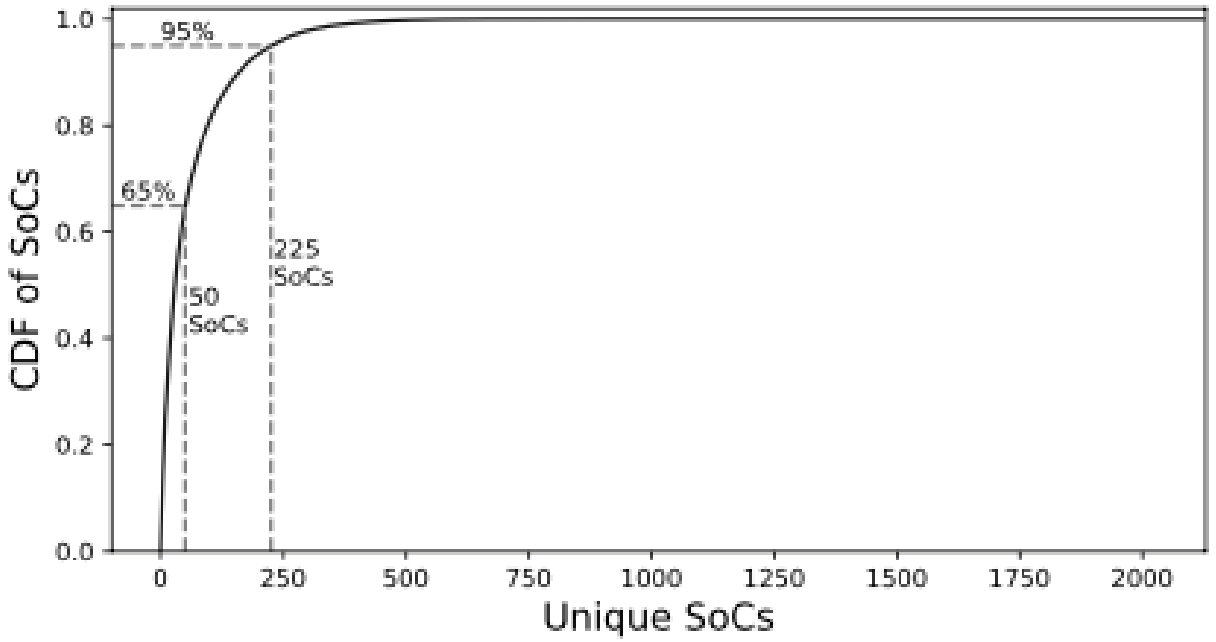


Figure 2: There is no standard mobile SoC to optimize for. The top 50 most common SoCs account for only 65% of the smartphone market.

- No standard: 칩들이 너무 다양하다
- 30개의 칩만 1%이상의 점유율을 갖고 있다
- 위의 30개 칩 점유율을 합해도 51%

Mobile CPUs show little diversity

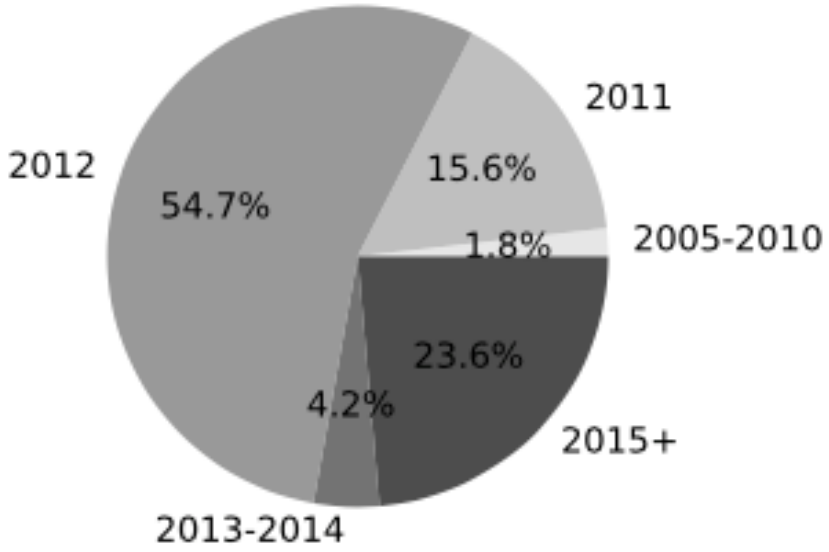


Figure 3: The most commonly-used mobile processors, Cortex A53, are at least six years old. In 2018, only a fourth of smartphones implemented CPU cores designed in 2013 or later.

- 오래된 모바일 기기가 대부분
- Cortex A53 ip가 48%, Cortes a7가 15%이상을 점유
- 대부분의 기기가 1~4코어
  1~4코어에서 머신러닝 모델을 돌려야 한다
- 하드웨어에 최적화 할 때 오래 쓰일 것을
  예상해야 함(long lifetime)

Mobile CPUs show little diversity

- 멀티코어 트랜드
- 안드로이드: 코어를 늘리고 성능은 낮게
- IOS: 코어를 줄이고 성능을 올리고

- Two CPU Cluster
- 하나는 고성능, 다른 하나는 효율성

-> 고성능 클러스터를 타겟으로 최적화

The performance difference between a mobile CPU and GPU/DSP is narrow

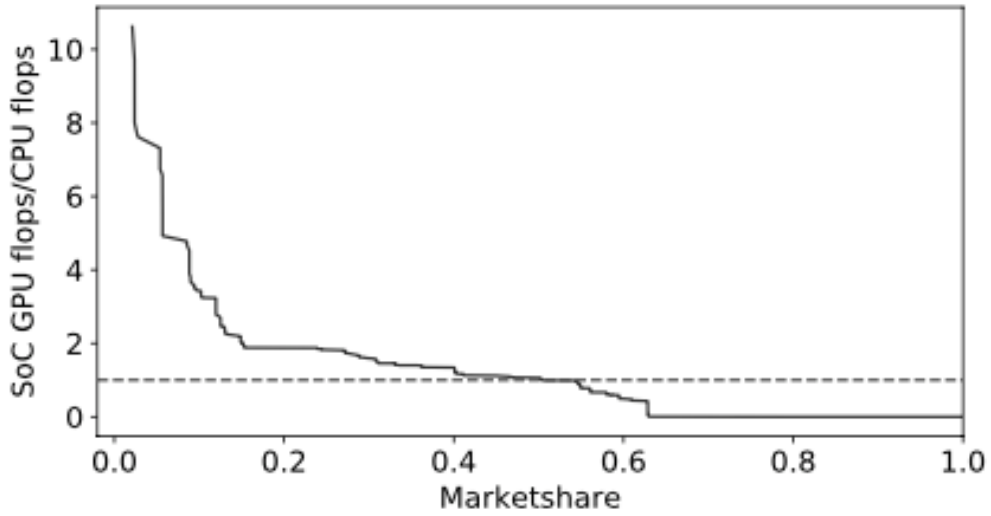


Figure 4: The theoretical peak performance difference between mobile CPUs and GPUs is narrow. In a median Android device, GPU provides only as much performance as its CPU. Only 11% of the smartphones have a GPU that is 3 times more performant than its CPU.

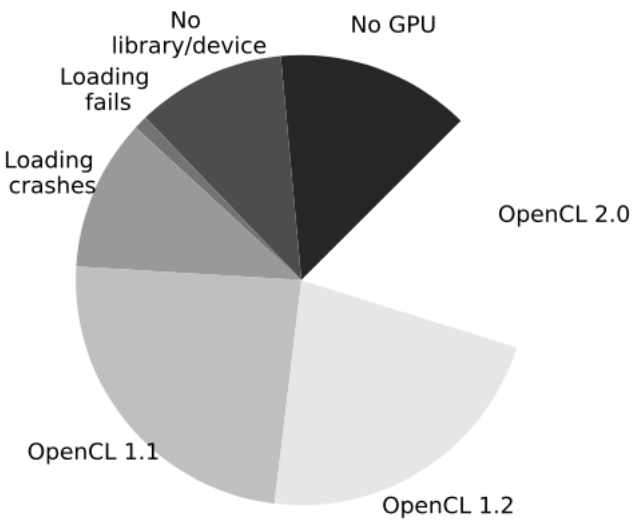
GPU가 Edge device에서도 중요할까?
-> NO

- 제한적인 퍼포먼스
- 23% 두배의 성능, 11% 세배 이상
- 프로그래밍이 어렵고 software challenge 성능은 별로
- 메모리 bandwidth, CPU GPU 메모리 공유

Available co-processors: DSPs and NPUs

- DSP(digital signal processor) 사용
  에너지 효율이 높음
  퀄컴 베이스 칩 중 5%만 장착
- NPU
  아직은 많이 사용하지 않고 오픈되어있지 않음
  - Cambrico 1A in Kirin970
  - Neural engine apple A12 Bionic soc

Programmability is a primary roadblock for using mobile co-processors



(a) OpenCL Support

OpenCL

- General purpose용, OpenGL같이 그래픽용으로 만들어진 API보다 계산을 표현하기 편하다
- Android에 드라이버가 장착되어 나오는데 공식적으로 지원하지 않음, 테스트 없이 나와서 고장난 경우 많음
  최악의 경우 로드하자 마자 Crash

Programmability is a primary roadblock for using mobile co-processors

OpenGL ES

- OpenGL의 모바일 & 임베디드 용으로 소형화된 api
- General purpose로 만들어지지 않음, 최근 버전에서 NN을 충분히 구현할 수 있게 API 만듬

2.0

- 프로그래밍으로 구현 가능한 첫 번째 api
- 페이스북에서는 모든 앱이 이 버전을 지원
- Render-to-texture 테크닉을 써서 NN을 구현할 수 있는데 메모리 제한이 있음
- 쉐이더 내에서 계산해야 해서 16bits의 아웃풋으로 제한

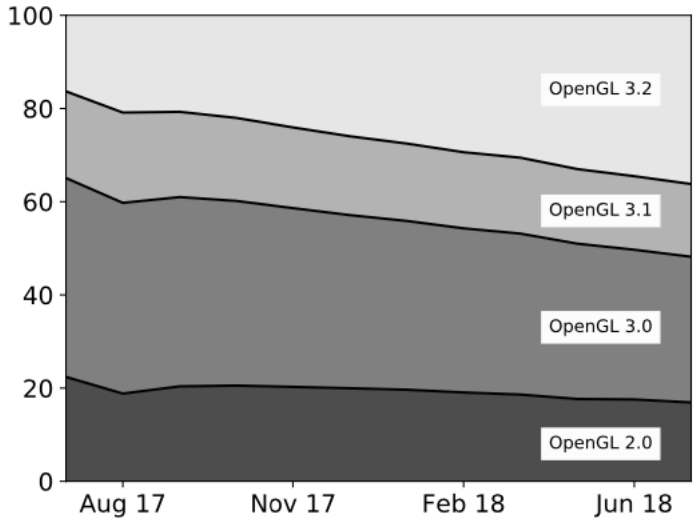
3.0 or newer

- 83%의 안드로이드 앱이 지원
- NN구현을 처음으로 지원
- 여전히 shader로 구현해야 하지만 128bits까지 가능

Programmability is a primary roadblock for using mobile co-processors

3.1 or newer

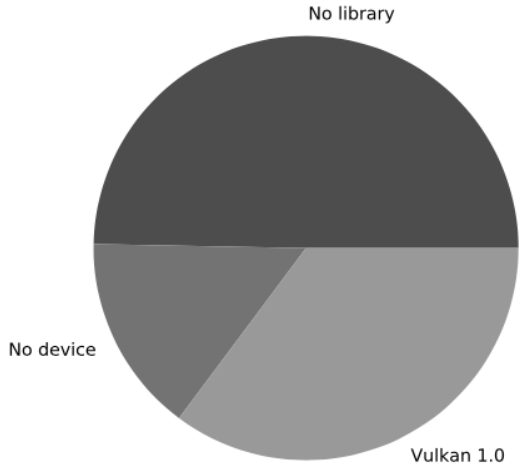
- 52% 안드로이드 앱이 지원
- 초기 cuda, Open CL과 비슷한 기능 제공
- Kernel로딩 오버헤드가 적음, 빠른 synchronization 등



(b) OpenGL ES Support

Programmability is a primary roadblock for using mobile co-processors

Vulkan
- Open GL의 후속작
- 36% 안드로이드만 지원함, 늘어날 예정
- Promising



(c) Vulkan Support

Metal
- Apple의 gpu언어
- 애플은 시스템 스택이 아이폰을 위해 강하게 통합되어 있음
- 2013년부터 모든 프로세서가 Metal 지원, 95% 기기가 지원
- Gpu를 실험하기에 최적의 환경, 성능 cpu대비 3~4배

MACHINE LEARNING AT FACEBOOK

- 페이스북은 어떻게 Edge device에서 real-time inference를 할까?
- 페이스북은 머신러닝을 프로덕트에 넣는 프래임워크와 플랫폼이 있음(FBLearner)

MACHINE LEARNING AT FACEBOOK

- Caffe2와 pytorch
- Caffe2: 많은 플랫폼을 지원하는 프로덕트 스케일을 위한 프레임워크
- Pytorch: 유연하고 표현력이 좋은 프레임워크

-> 통합 & onnx export 지원

Important Design Aspects for Mobile Inference and Potential Approaches

- CPU를 타겟으로 인퍼런스하게 만들면 되지만, 퍼포먼스 손실이 있다
- 엣지 디바이스에서는 퍼포먼스가 가장 중요하기 때문에 최적화가 중요하다. 하지만 파편화 때문에 퍼포먼스 측정과 분석이 어렵다

Important Design Aspects for Mobile Inference and Potential Approaches

- Model size, Code size를 고려해야 함

Model size

- Prunning, Quantization, Compression

Code size

- ML 모델을 플랫폼에 특화된 오브젝트 코드로 컴파일
  Glow, XLA, TVM
- Core ML처럼 벤더가 제공하는 api 사용
- Generic한 인터프리터 (Caffe2, TFLite) 사용

Mobile Inference Workflow

- FBLearner
  Flow(학습, 관리), Store(저장), AutoML, Predictor(real-time)

- Store에서 Feature 로딩
- Flow로 학습, 평가
- Export
- Optimization
- Iteration(Increase data, refining feature set, change model arch)

HORIZONTAL INTEGRATION: MAKING INFERENCE ON SMARTPHONES

Caffe2 features

- Compact image representation
- Channel pruning
- Quantization

Prunning, quantization, cut-down size 적용

HORIZONTAL INTEGRATION: MAKING INFERENCE ON SMARTPHONES

Caffe2에 통합된 라이브러리

- NNPACK: winograd, 퓨리에 변환 등 알고리즘 사용
- QNNPACK: 1 * 1, grouped, depth-wise, dilated conv에 사용 convolution 알고리즘을 효율적으로 구현

Performance optimization versus accuracy tradeoff

- Reduced precision computation benefits
  ex) FP32 -> 8-bit fixed-point

1. Reduced memory footprint
2. Higher computation efficiency
3. Improved performance for bandwidth bounded operation
   depthwise conv, small conv

Performance optimization versus accuracy tradeoff

- Quantization Experiences

1. Unet: small spatial

No Winograd (NNPACK), 16bits extend from computation

2. Style transfer: large spatial, small channel

QNNPACK, reduced memory bandwidth(large spatial)

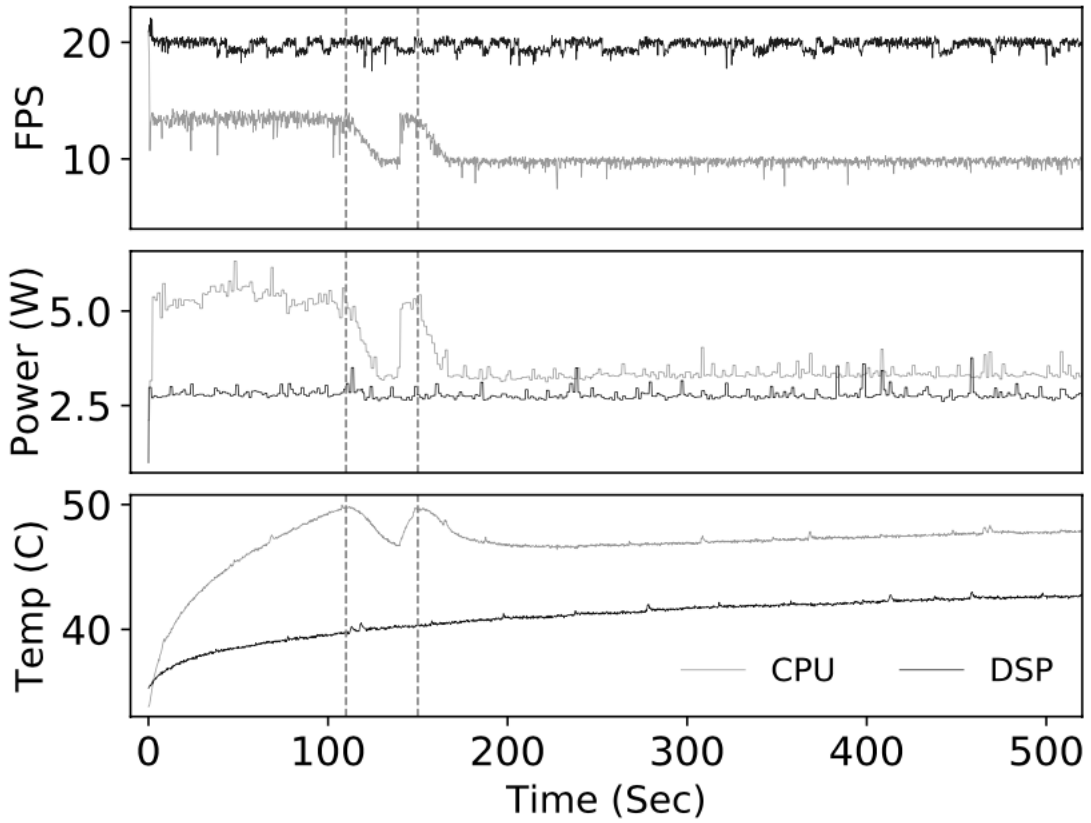
3. ShuffleNet: 1x1 conv, depthwise conv

Performance optimization versus accuracy tradeoff

- Quantization, algorithmic optimization
- Quantization을 사용하면 Winograd 사용불가

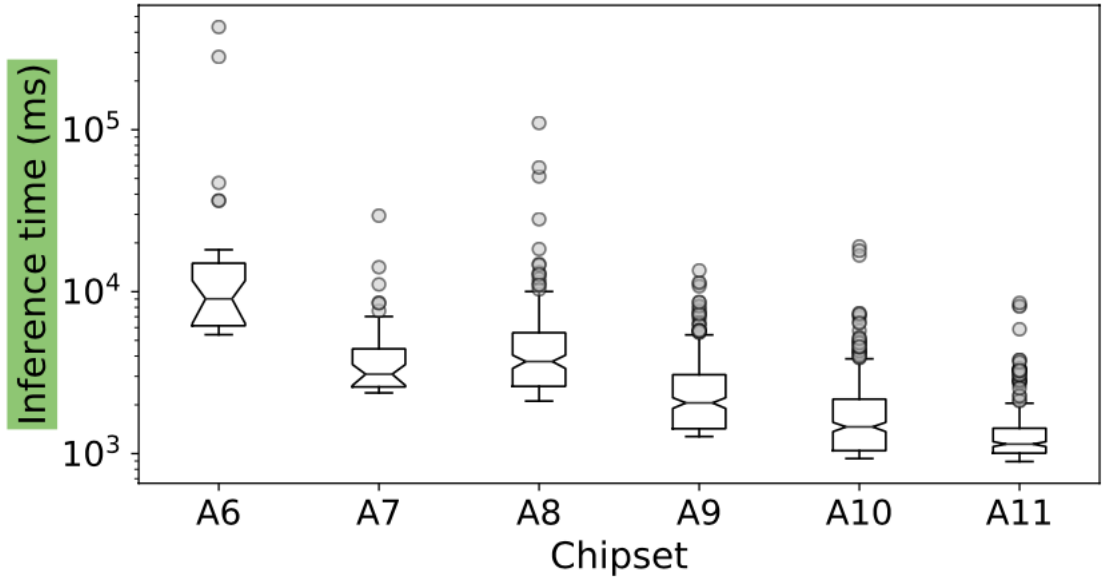
VERTICAL INTEGRATION: PROCESSING VR INFERENCE FOR OCULUS

- Hand Tracking, Image classification, Pose estimation, Action Segmentation
- CPU vs DSP
- DSP 고성능
  프로그래밍이 어렵다
  Quantization 필요
  (only fixed-point)
  메모리 레이아웃에 민감하다



MAKING INFERENCE IN THE WILD: PRACTICAL CONSIDERATIONS FOR PROCESSING ON MOBILE DEVICES

- 동일한 기기, 소프트웨어에서도 성능의 변동성이 높음
- 언제, 어디서, 어떻게 테스트하는지 상당한 영향
- 신뢰할만한 테스트는 in-field, mobile-space에서 해야한다



Do the performance variability characteristics follow certain trends or statistical distributions?

- 분포가 매우 다양하다는 것은 자명한 사실
- 평균을 기준으로 설계하면
  필요한 퍼포먼스 수준을 만족하지
  못할 수 있다
  최소 최대 퍼포먼스 값 고려
- In-field study 중요

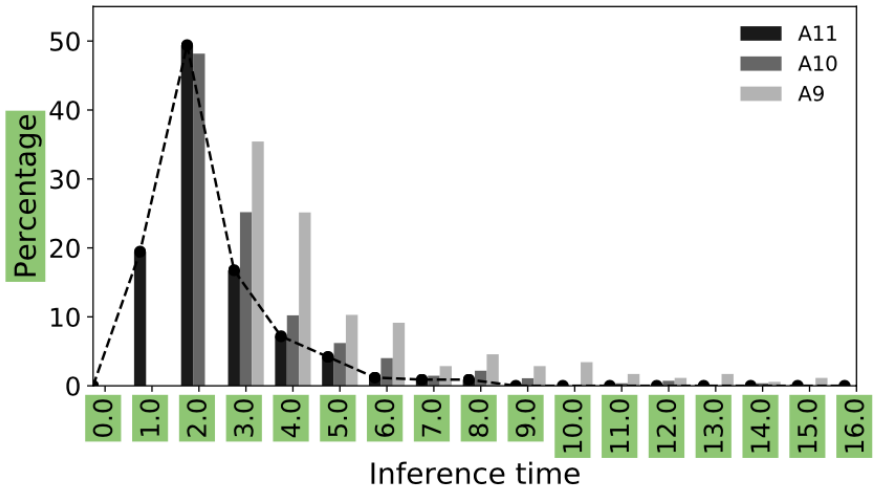


Figure 11: The inference time follows an approximate Gaussian distribution with the mean centered at 2.02ms and the standard deviation of 1.92ms.