MNN: A Universal and Efficient Inference Engine

Published in MLSys Conference 2020

Background

The **major challenges** for mobile inference engines

- Model compatibility
  - should have the model compatibility for different formats and different operators

- Device diversity
  - should take hardware architectures or device vendors into consideration
  - should take care of the software diversity problem

- Resource limitation
  - memory and computation power are still constrained on mobile devices

Background

The **properties** a good mobile inference engine should have

- Universality
  - address both model compatibility and device diversity

- Efficiency
  - inference models on devices with great performance and to use as little memory and energy consumption as possible

- Scalability
  - support new operators emerging in the future

Overview

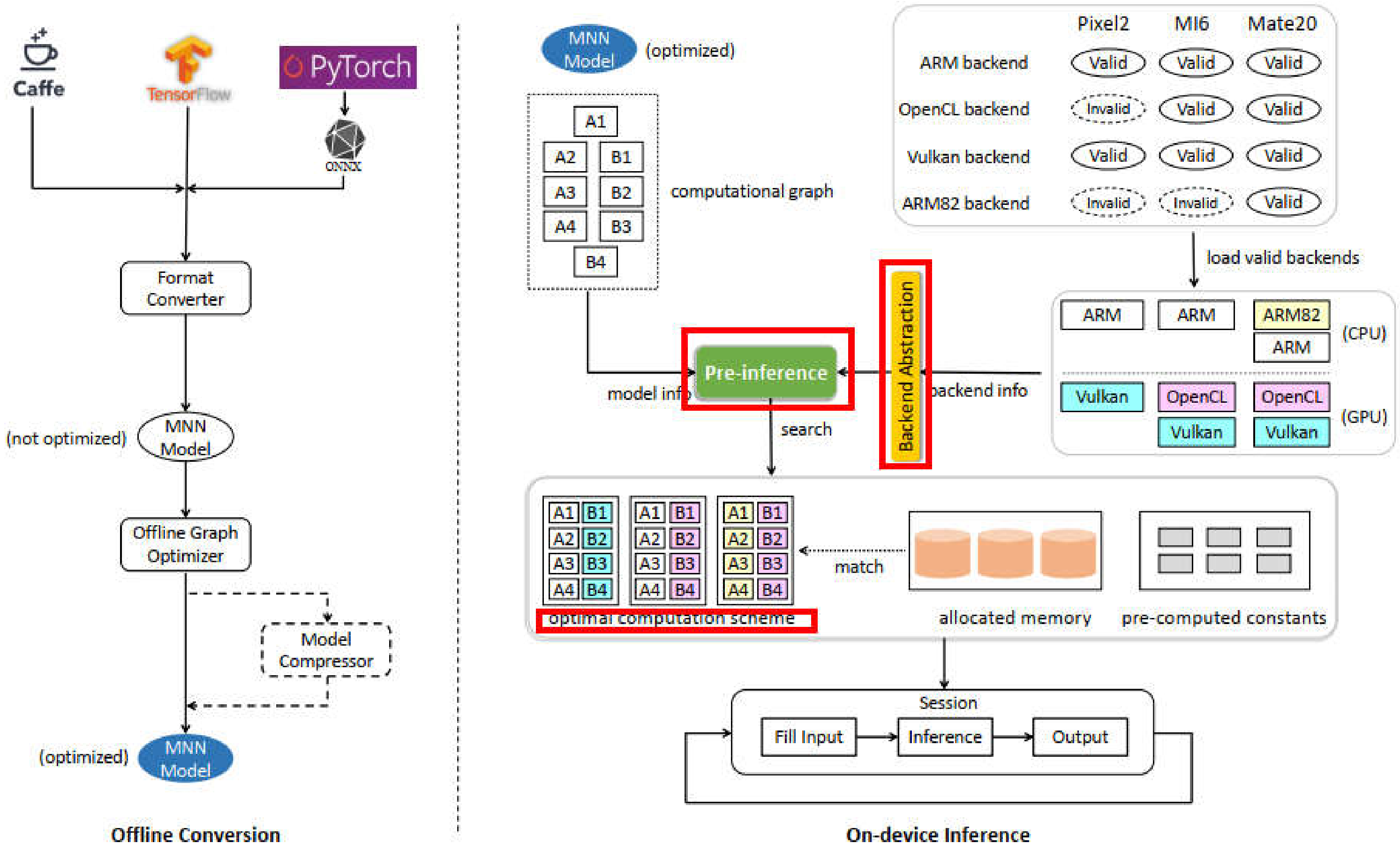


Figure 2. Architecture detail of the proposed mobile inference engine Mobile Neural Network (MNN).

Pre-Inference

Goal - Pre-Inference to optimize computational cost and memory usage ahead of formal inferences

Computation Scheme Selection

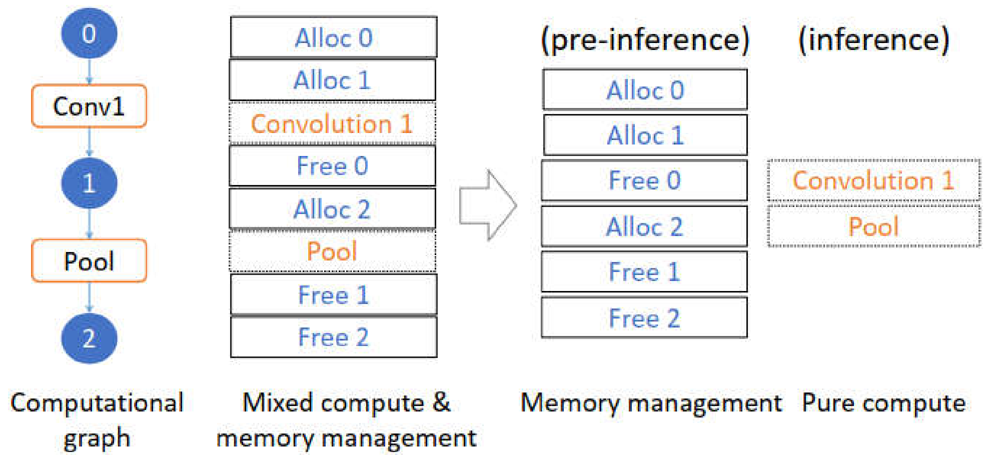
- propose a cost evaluation mechanism to select the optimal scheme from the scheme pool
- define a cost model: C_total = C_algorithm + C_backend
- C_algorithm: theoretical cost of different algorithms used to implement the workload (sliding window / FFT / winograd for conv)
- C_backend: theoretical cost of executing workload on different software / hardware stacks (CPU / GPU / NPU)
- search the best scheme in algorithm level and backend level separately

Pre-Inference

Goal - Pre-Inference to optimize computational cost and memory usage ahead of formal inferences

Preparation-execution decoupling

- infer the exact required memory for the entire graph by virtually walking through all operations
- pre-allocates required memory as a memory pool during the pre-inference stage
- reuses memory pool in the following inference sessions



Kernel Optimization

Winograd

- Block division and pipelining
  - the tile size of existing framework is fixed, MNN propose a flexible tiling strategy

$$T = \left\lfloor \frac{o_w o_h}{\hat{n}^2} \right\rfloor$$

- Hadamard product optimization
  - transform the Hadamard product to matrix multiplication building upon data layout re-ordering

- Winograd generator
  - existing framework using Winograd hardcode the *A; B; G* matrices for common kernel and input sizes in source codes
  - propose Winograd generator to support high scalability in face of new cases

Kernel Optimization

Strassen

- Strassen is a fast matrix multiplication algorithm that trades expensive multiplications with cheap additions

- Strassen can be applied *recursively*

- When matrix size is small enough, Strassen is slower than GEMM

- Give the rules about when to stop the recursion

$$mnk - 7 \cdot \frac{m}{2}\frac{n}{2}\frac{k}{2} > 4 \cdot \frac{m}{2}\frac{k}{2} + 4 \cdot \frac{n}{2}\frac{k}{2} + 7 \cdot \frac{m}{2}\frac{n}{2}.$$

Backend Abstraction

- make all the hardware platforms and software solutions encapsulated into a uniform Backend class
- resource management, memory allocation, and scheduling are disentangled with the concrete operator implementations

Advantages

- Reduce complexity
- Enable hybrid Scheduling
- More Lightweight