Mobile
Neural
Network

端侧推理引擎面临的挑战与应对

离青

淘宝无线开发专家

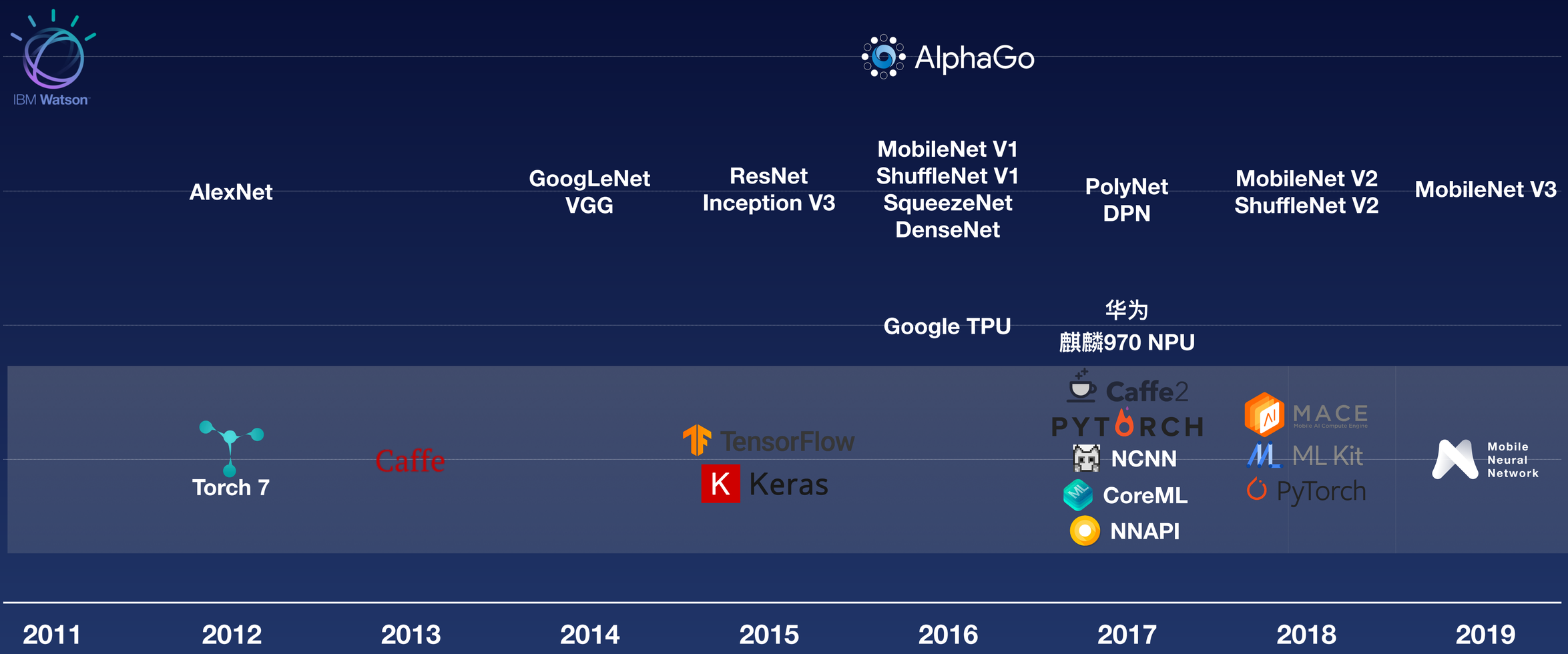
自我介绍

- 陈以鎏，花名离青
- 淘宝无线开发专家
- 2015 - 2019
  - 阿里百川
  - 手机淘宝架构
  - MNN

目录

- 现状与挑战
- 应对之道
- 编译方案与比较
- 应用场景
- 开源与规划

背景

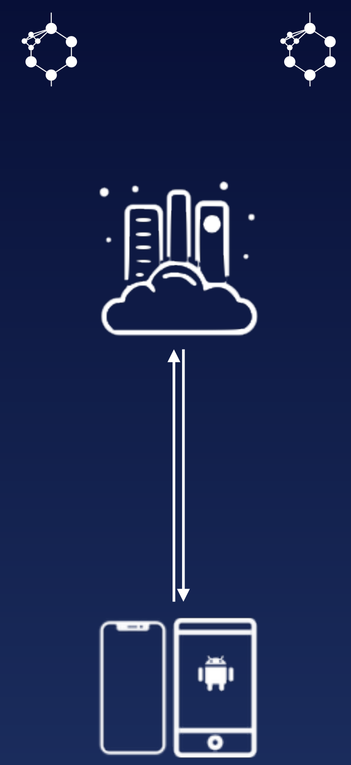

IBM Watson
AlphaGo
AlexNet
GoogLeNet
VGG
ResNet
Inception V3
MobileNet V1
ShuffleNet V1
SqueezeNet
DenseNet
PolyNet
DPN
MobileNet V2
ShuffleNet V2
MobileNet V3
Google TPU
华为
麒麟970 NPU
Torch 7
Caffe
TensorFlow
Keras
Caffe2
PYTORCH
NCNN
CoreML
NNAPI
MACE
Mobile AI Compute Engine
ML Kit
PyTorch
Mobile
Neural
Network
2011
2012
2013
2014
2015
2016
2017
2018
2019

移动AI

云端AI

端侧AI

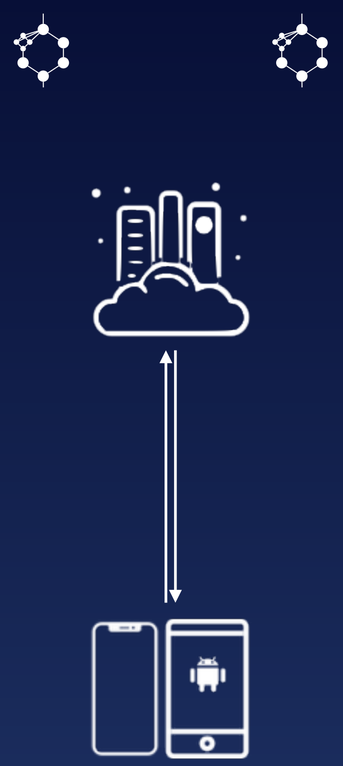
移动AI



云端AI

端侧AI

移动AI



云端AI

端侧AI

移动AI

云端AI

端侧AI

挑战

挑战

碎片化

- 训练框架
  - Caffe
  - TensorFlow
  - PyTorch
  - MXNet
  - ...
- 设备环境
  - CPU
  - GPU
  - NPU
  - DSP
  - ...
- 算子分支
  - kernel
  - stride
  - pad
  - dilation
  - ...

应对

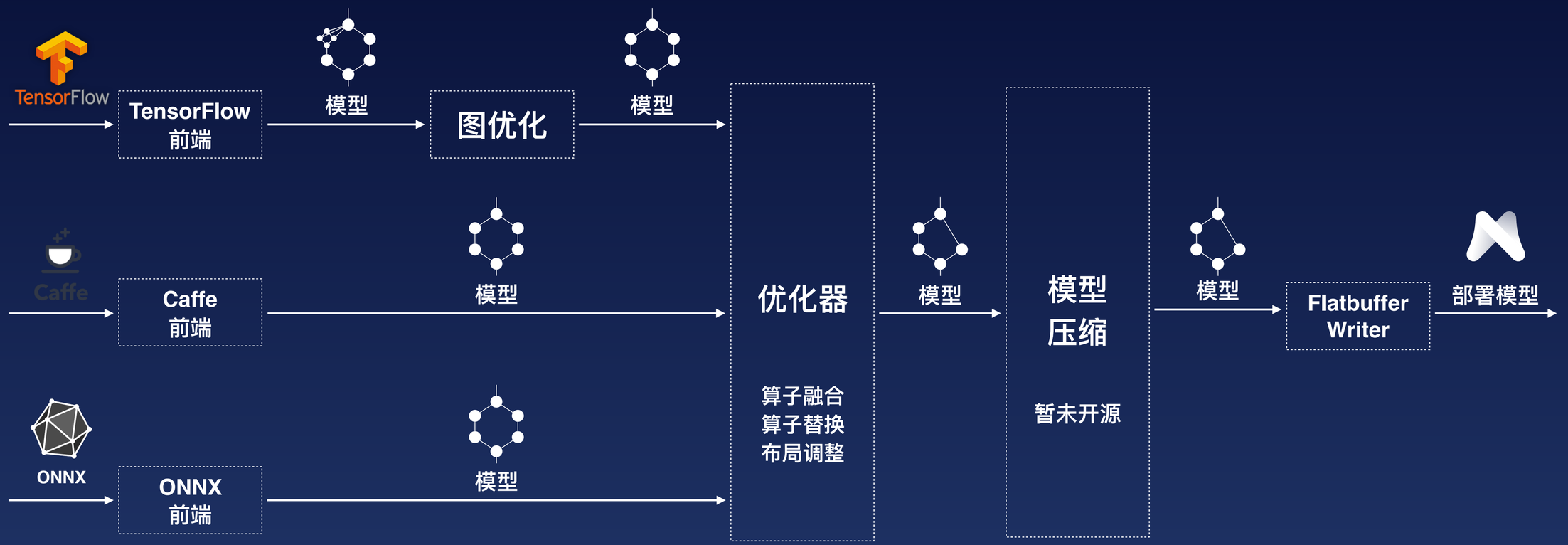
资源&性能

模型优化
- 统一可拓展描述
- 图优化/算子融合
- 布局优化
- 模型压缩

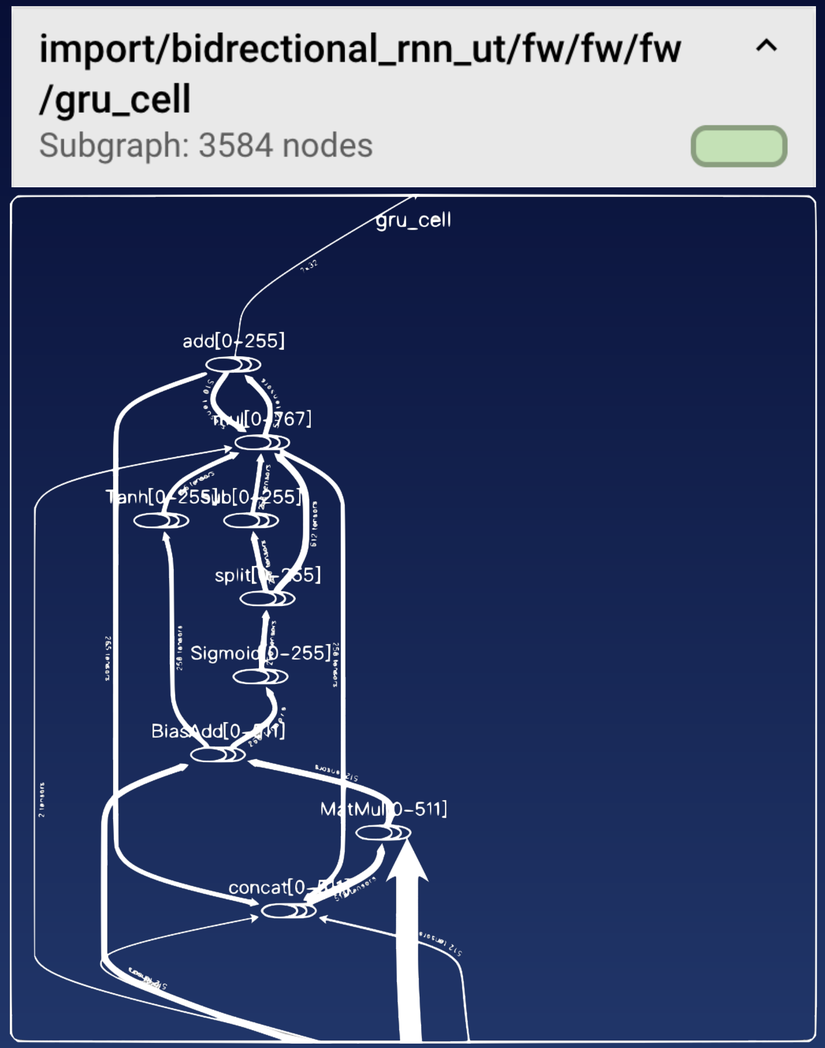
智能调度
- 常量识别
- 资源/缓存管理
- 混合/并发调度
- 算力/负载 自动选择

计算加速
- SIMT/SIMD
- 忽略异常/低精度计算/近似算法
- 带宽/缓存/参数/输入 自动选择
- 链路优化

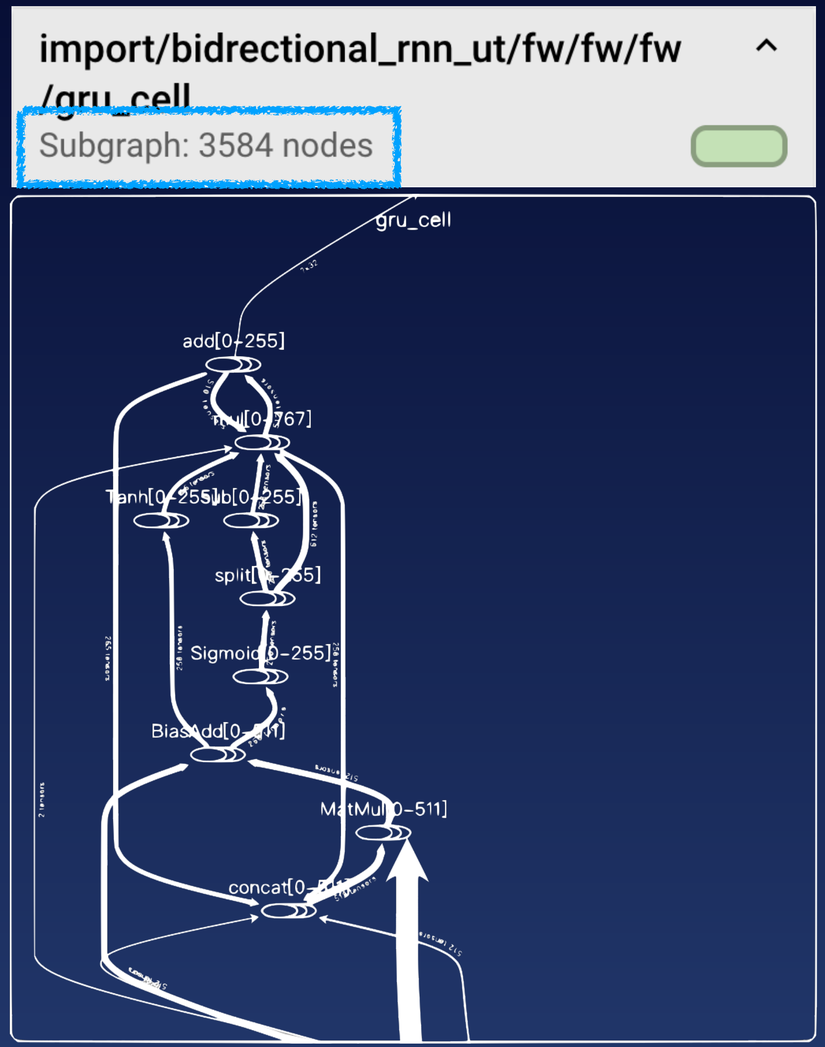
应对 - 模型优化


TensorFlow
TensorFlow
前端
模型
图优化
模型
Caffe
Caffe
前端
模型
ONNX
ONNX
前端
模型
优化器
算子融合
算子替换
布局调整
模型
模型
压缩
暂未开源
模型
Flatbuffer
Writer
部署模型

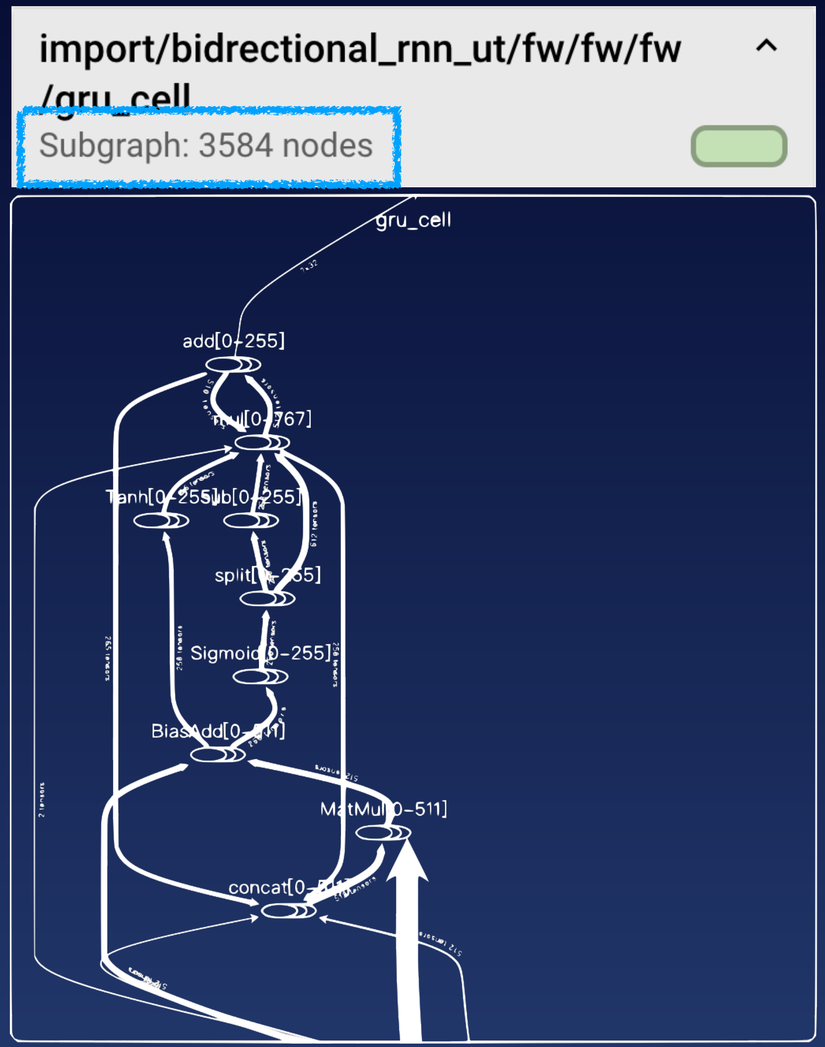
应对 - 图优化


import/bidirectional_rnn_ut/fw/fw/fw/gru_cell
Subgraph: 3584 nodes
gru_cell
add[0-255]
Tanh[0-255]
split[0-255]
Sigmoid[0-255]
BiasAdd[0-511]
MatMul[0-511]
concat[0-511]

应对 - 图优化

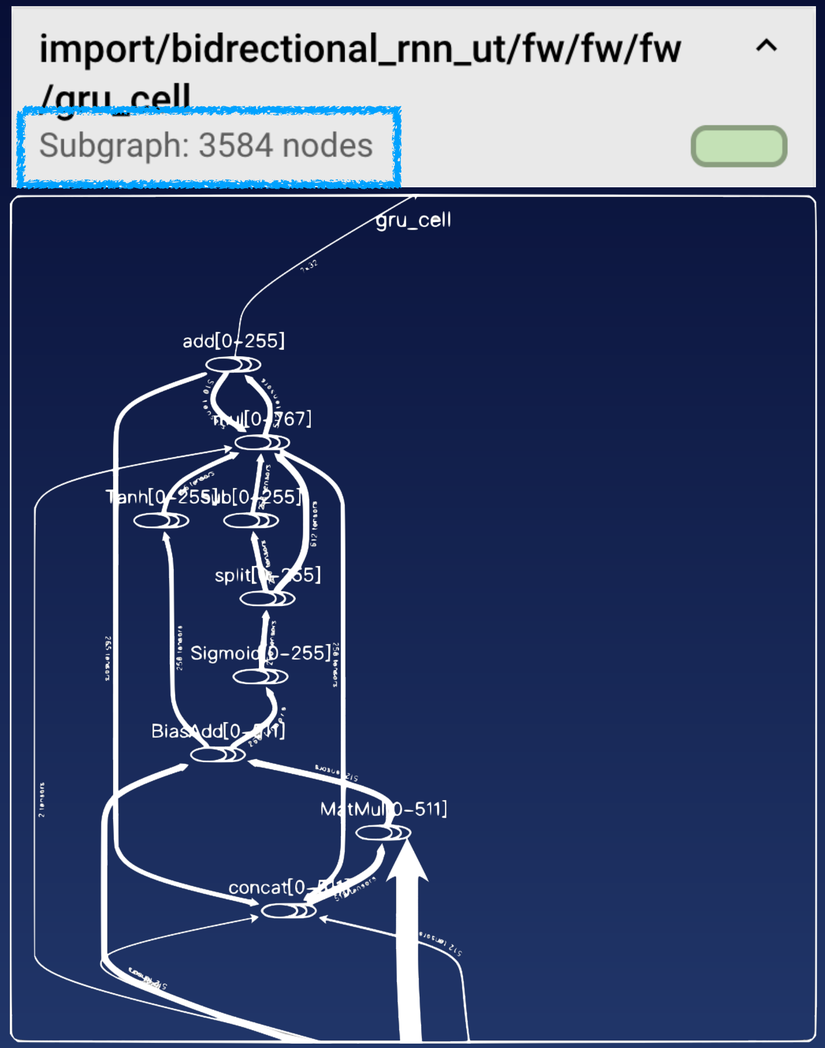

import/bidirectional_rnn_ut/fw/fw/fw
/gru_cell
Subgraph: 3584 nodes
gru_cell
add[0-255]
Tanh[0-255]
split[0-255]
Sigmoid[0-255]
BiasAdd[0-511]
MatMul[0-511]
concat[0-511]

应对 - 图优化

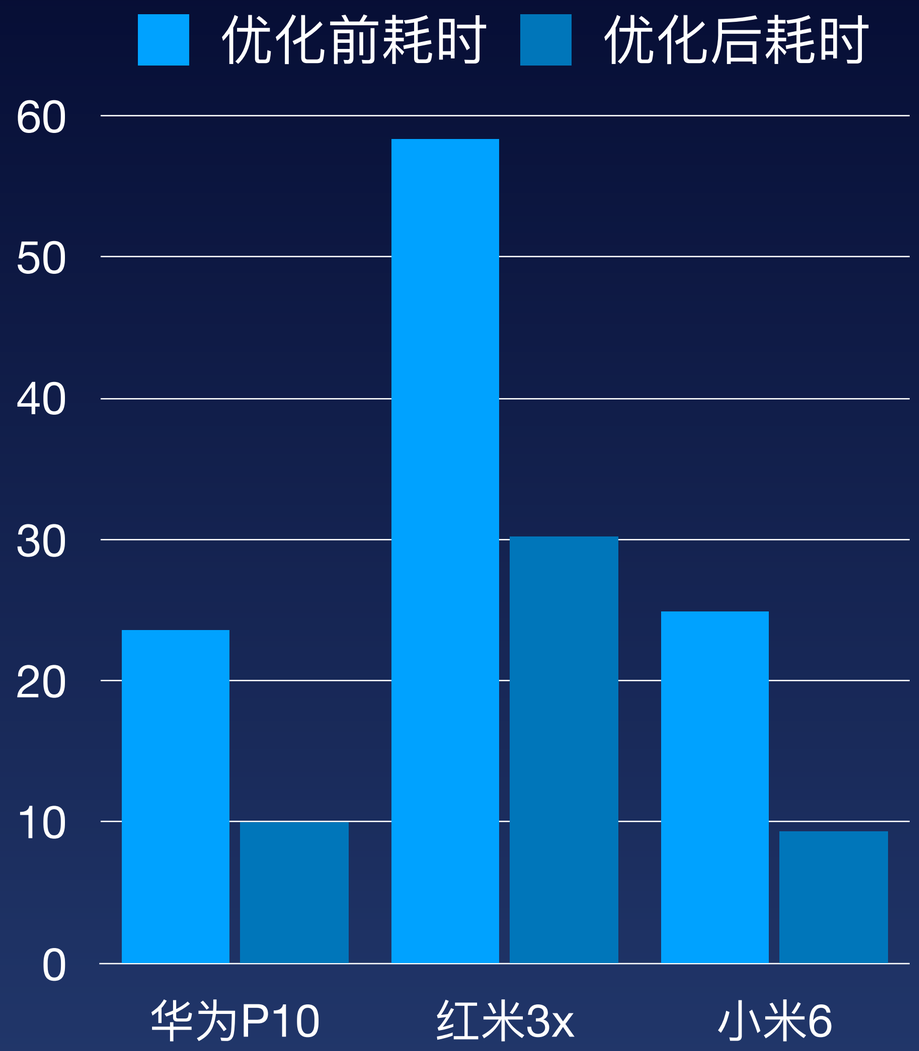

import/bidirectional_rnn_ut/fw/fw/fw
/gru_cell
Subgraph: 3584 nodes
gru_cell
add[0-255]
Tanh[0-255]
split[0-255]
Sigmoid[0-255]
BiasAdd[0-511]
MatMul[0-511]
concat[0-511]

GRU

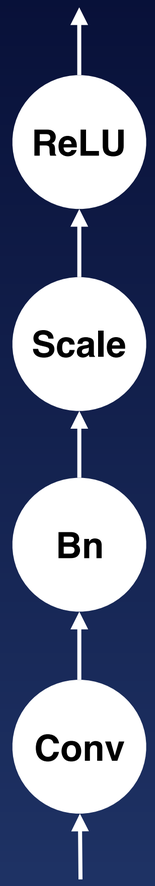
应对 - 图优化


import/bidirectional_rnn_ut/fw/fw/fw
/gru_cell
Subgraph: 3584 nodes
gru_cell
add[0-255]
MatMul[0-767]
Tanh[0-255]
b[0-255]
split[0-255]
Sigmoid[0-255]
BiasAdd[0-511]
MatMul[0-511]
concat[0-511]

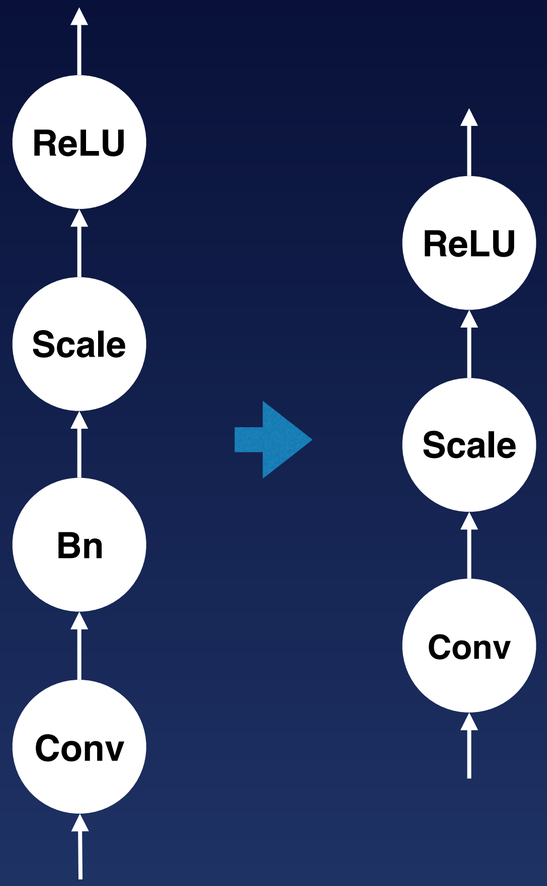
GRU


优化前耗时
优化后耗时
60
50
40
30
20
10
0
华为P10
红米3x
小米6

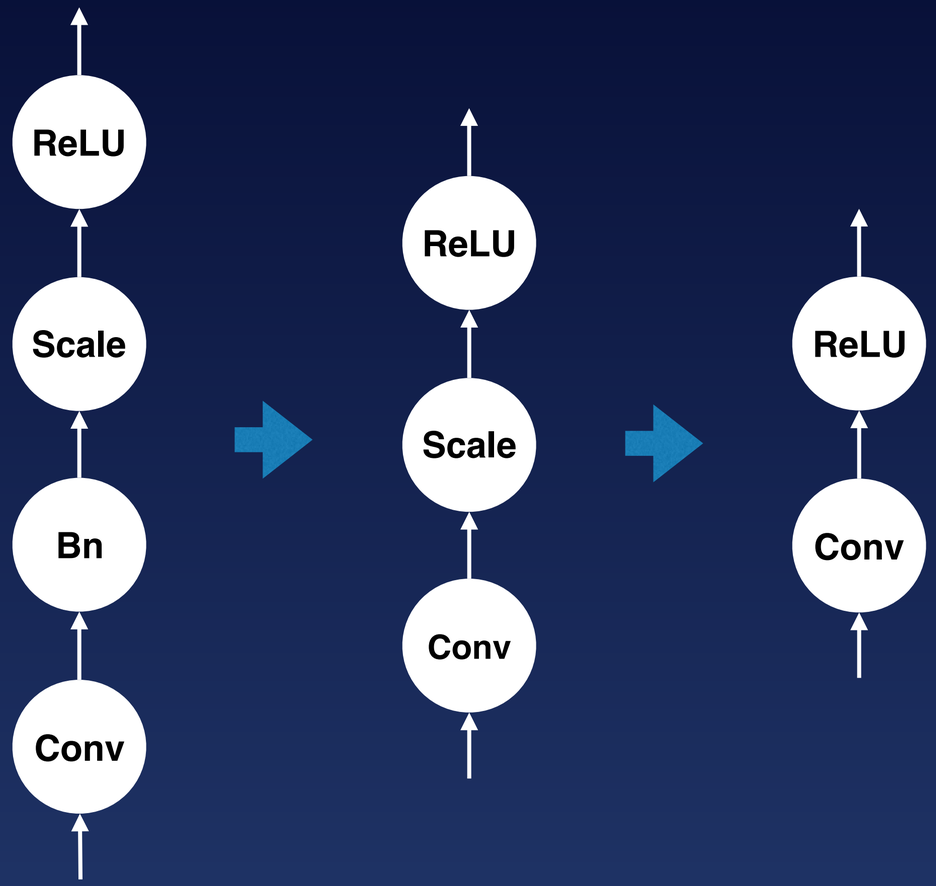
应对 - 算子融合



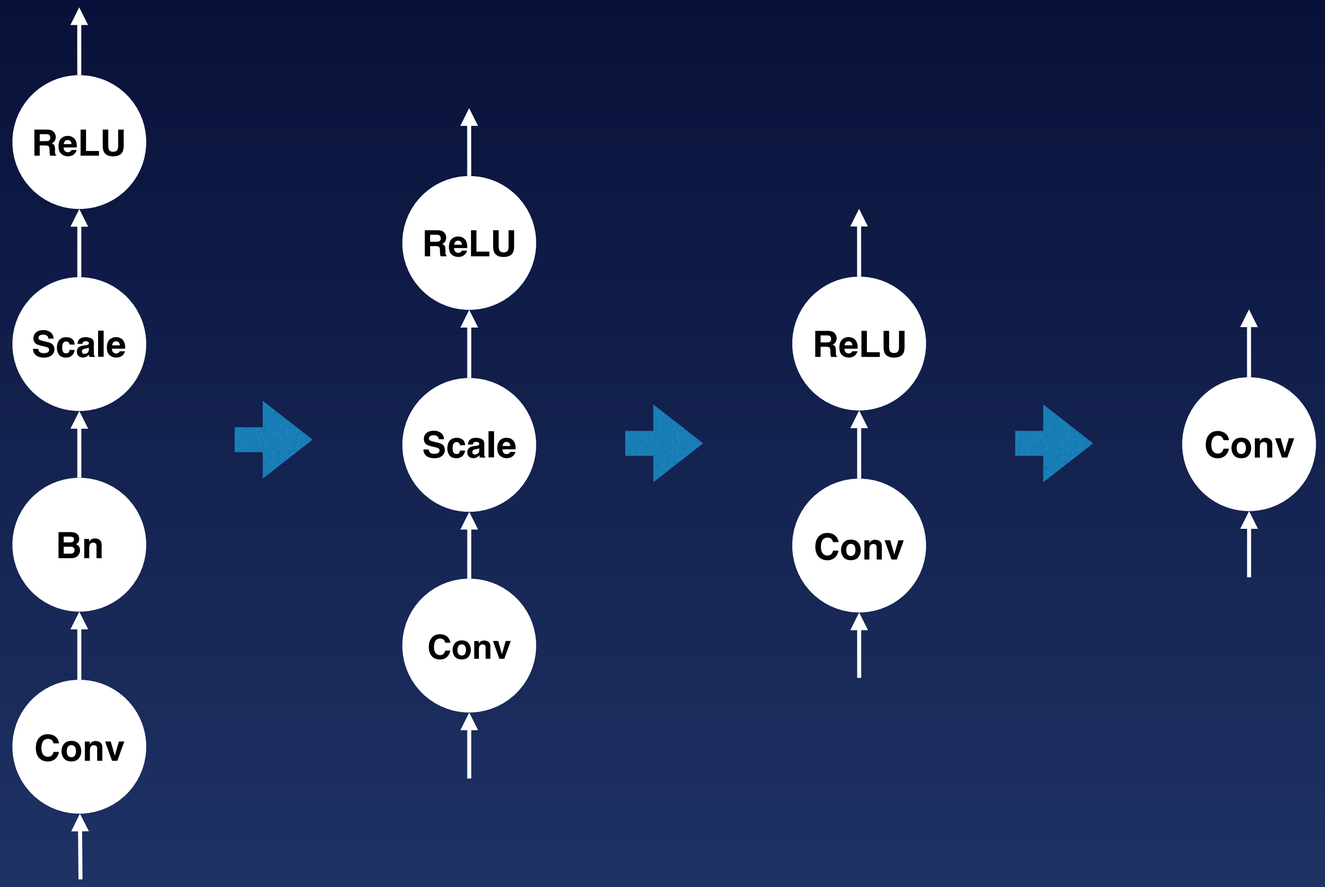
应对 - 算子融合



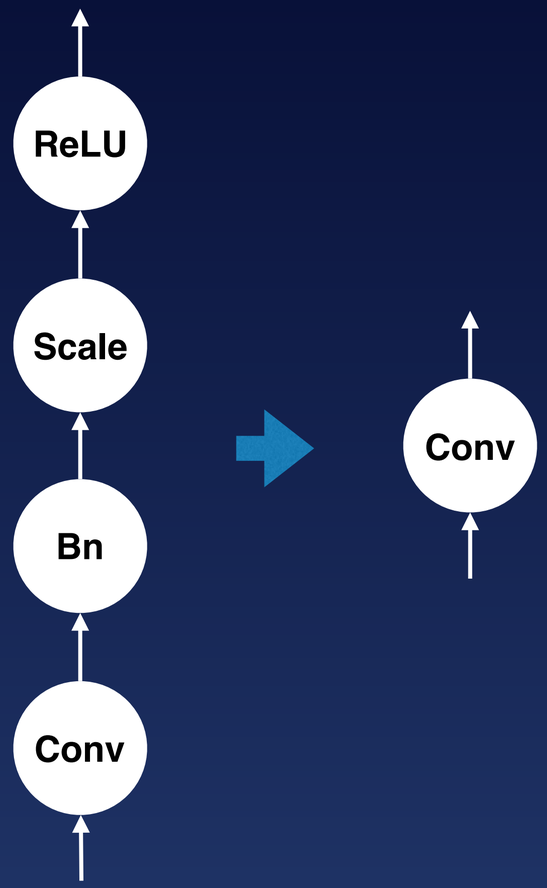
应对 - 算子融合



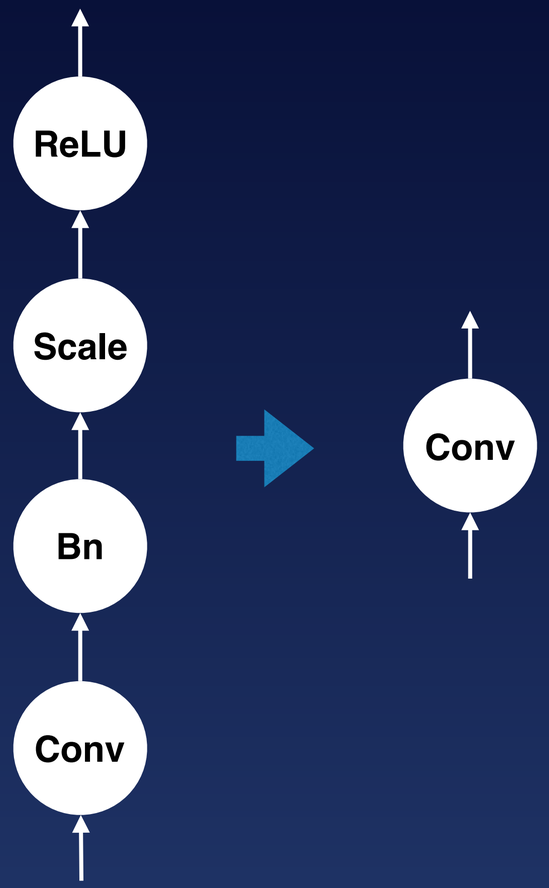
应对 - 算子融合

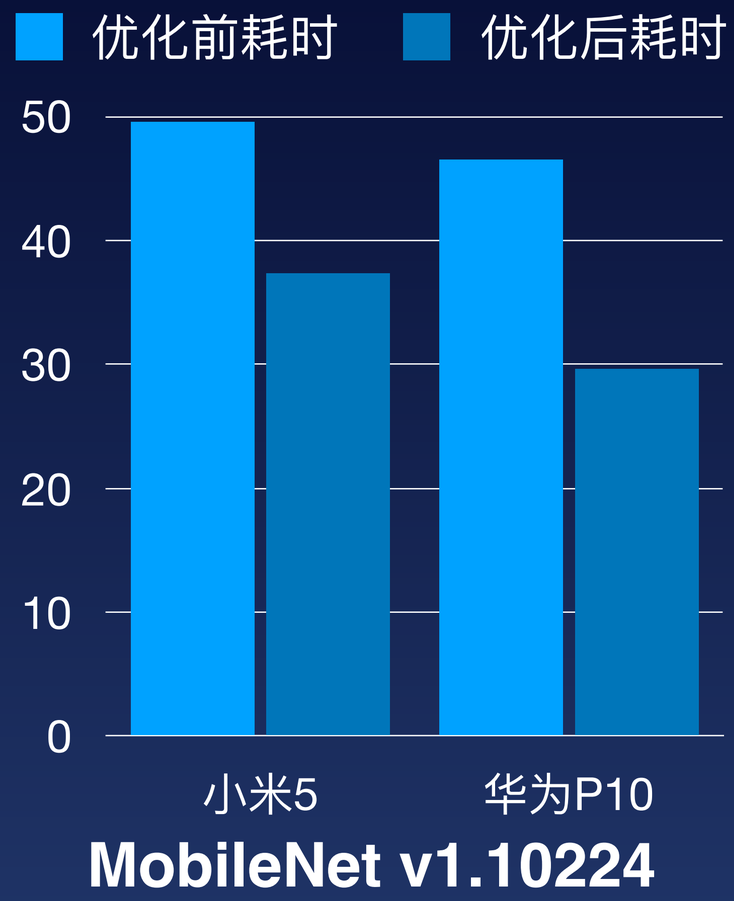


应对 - 算子融合



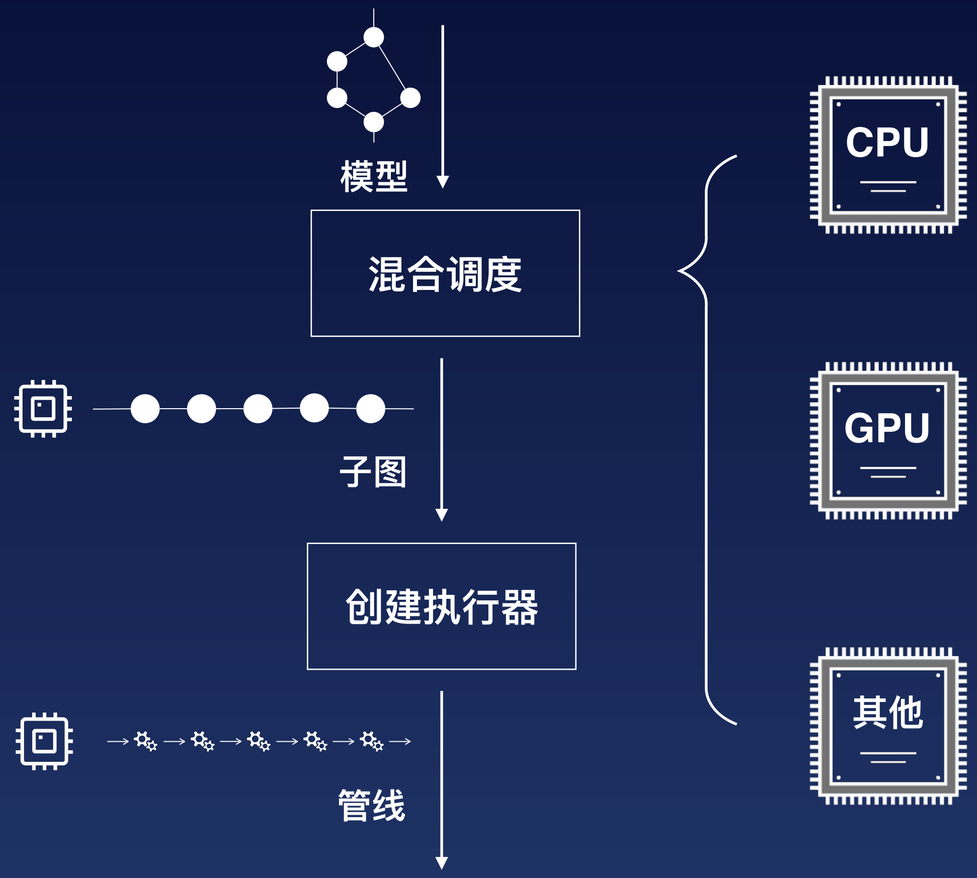
应对 - 算子融合


ReLU
Scale
Bn
Conv
Conv

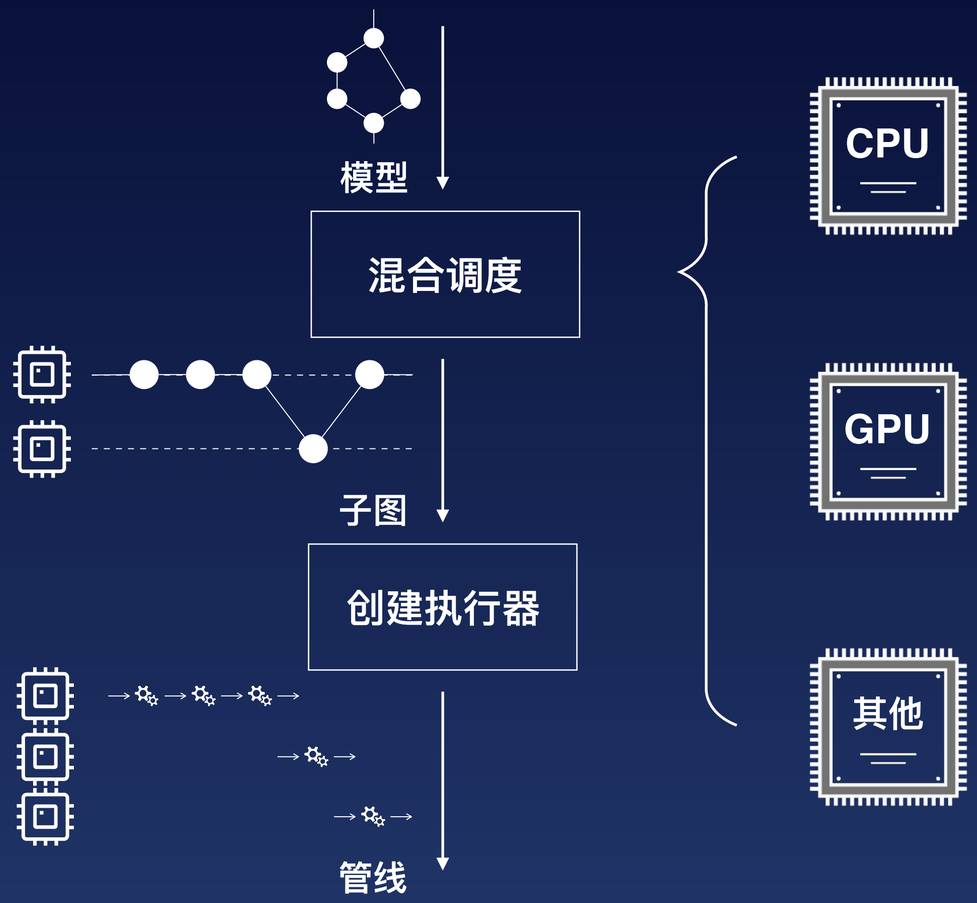

优化前耗时
优化后耗时
50
40
30
20
10
0
小米5
华为P10
MobileNet v1.10224

应对 - 智能调度

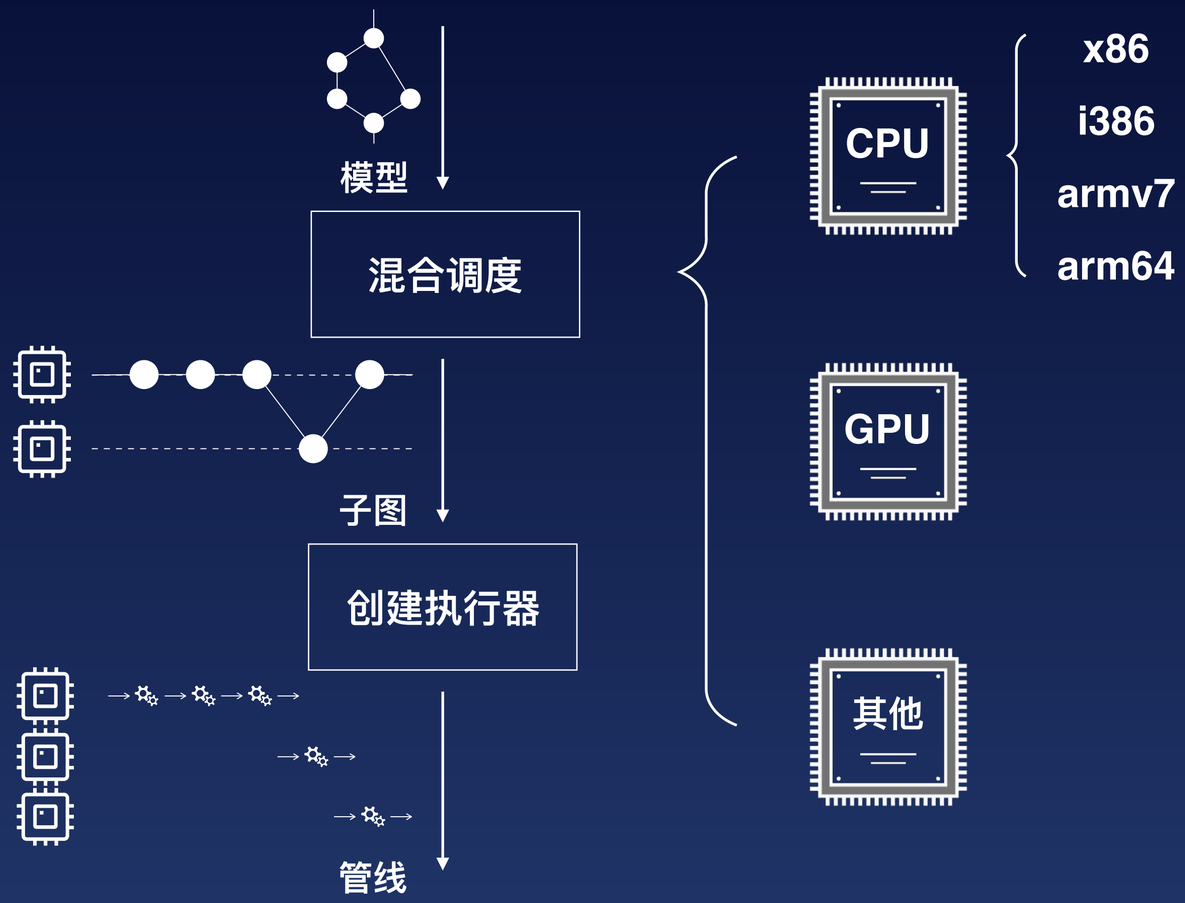
应对 - 智能调度



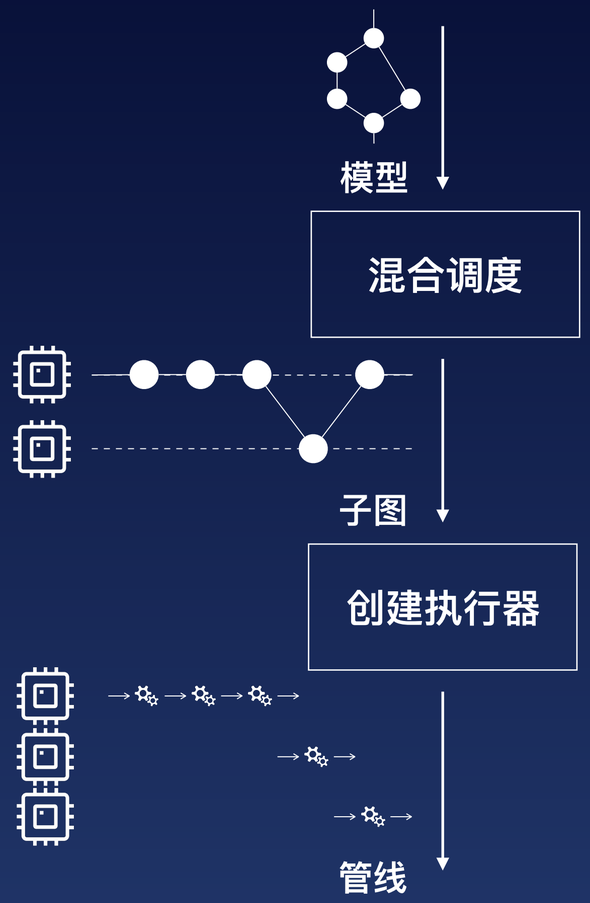
应对 - 智能调度

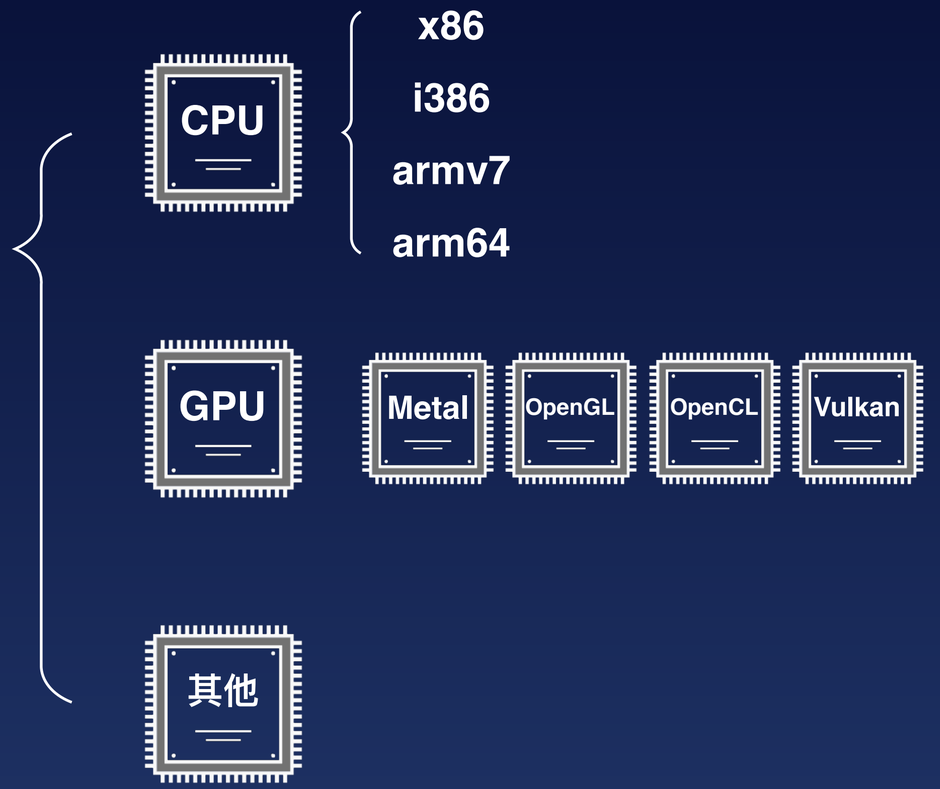


应对 - 智能调度

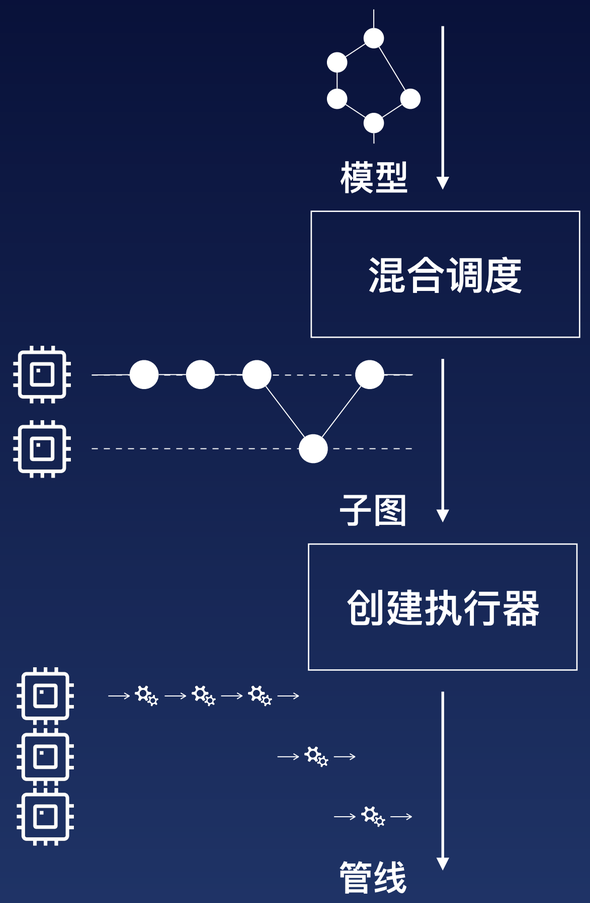


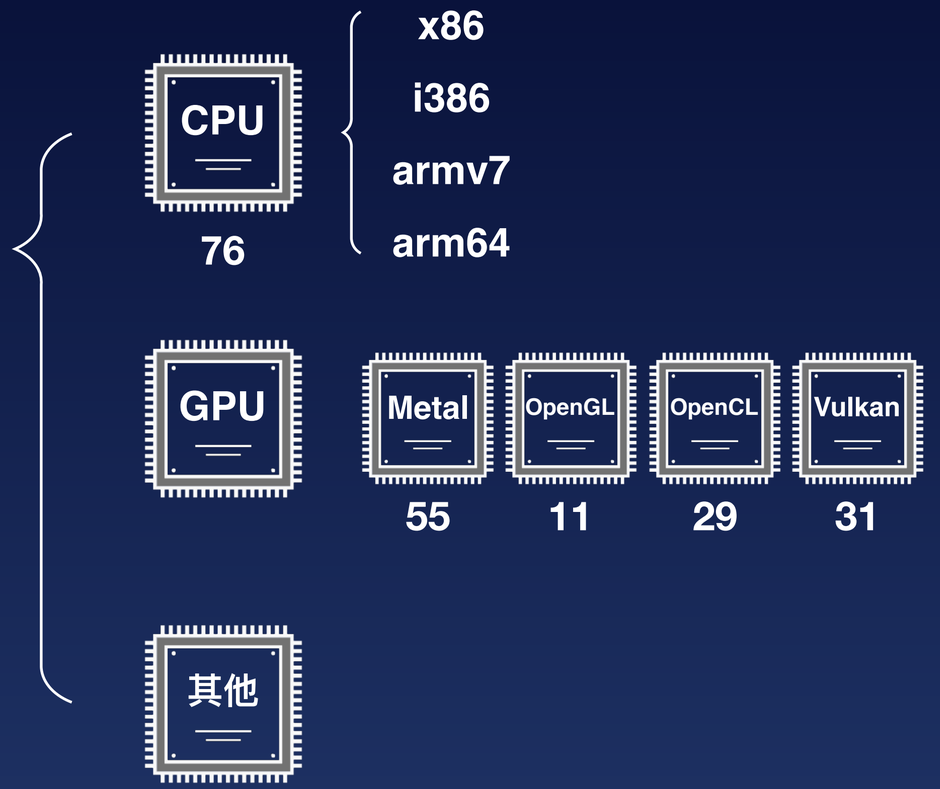
应对 - 智能调度


模型
混合调度
子图
创建执行器
管线

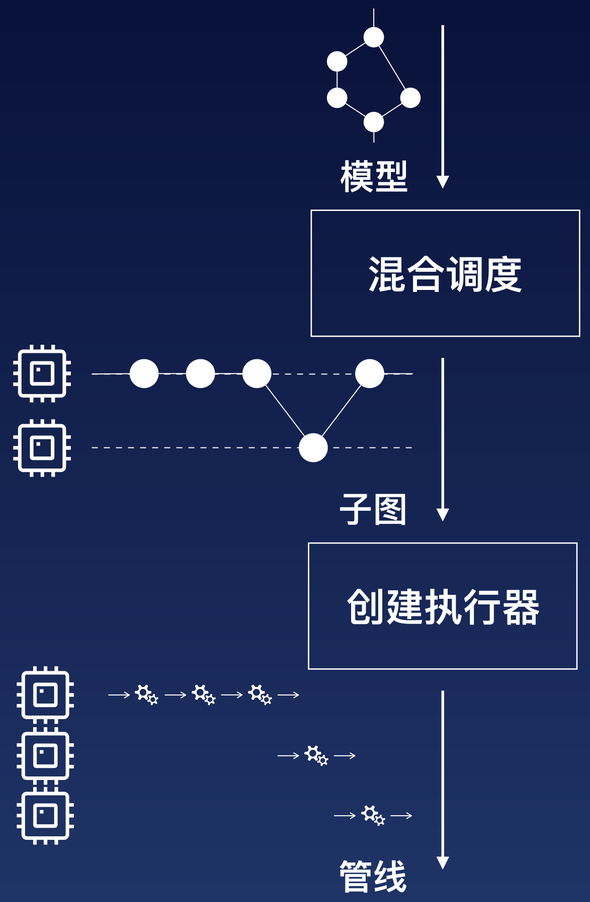

CPU
x86
i386
armv7
arm64
GPU
Metal
OpenGL
OpenCL
Vulkan
其他

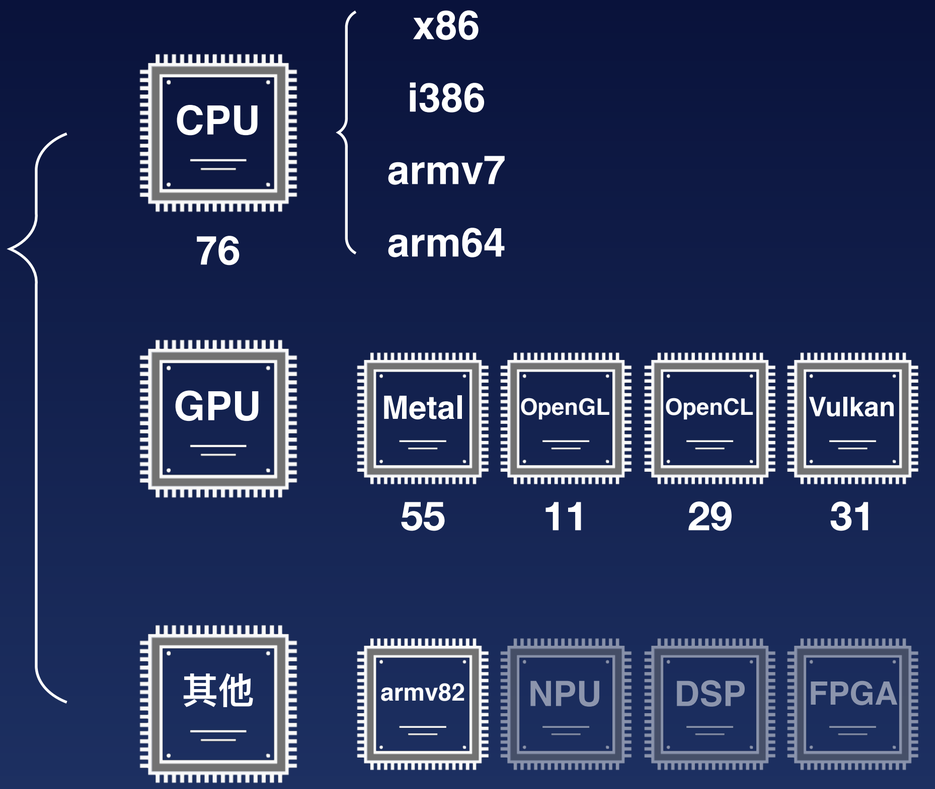
应对 - 智能调度


模型
混合调度
子图
创建执行器
管线


CPU
x86
i386
armv7
arm64
76
GPU
Metal
OpenGL
OpenCL
Vulkan
55
11
29
31
其他

应对 - 智能调度





- 平台隔离
- 易于拓展
- 扬长避短
- 混合调度
- 并发调度

应对 - 缓存管理

应对 - 缓存管理


3 1 5 7 4
0 1 0 0 4
4 4 1 2 5
tensors
(管线内串行)
(管线内串行)
管线
管线
(串行/并行)
(串行/并行)
(串行/并行)
模型
调度
子图
创建
执行器
子图
形状
计算
子图

应对 - 缓存管理


100
150
100
...
buffers
3 1 5 7 4
0 1 0 0 4
4 4 1 2 5
tensors
3 1 5 7 4
0 1 0 0 4
4 4 1 2 5
tensors
(管线内串行)
管线
(管线内串行)
管线
(管线内串行)
管线
(串行/并行)
(串行/并行)
(串行/并行)
模型
调度
子图
创建
执行器
子图
形状
计算
子图
准备
执行器
子图

应对 - 缓存管理


100
150
100
...
buffers
3 1 5 7 4
0 1 0 0 4
4 4 1 2 5
tensors
3 1 5 7 4
0 1 0 0 4
4 4 1 2 5
tensors
(管线内串行)
管线
(管线内串行)
管线
(管线内串行)
管线
模型
调度
(串行/并行)
子图
创建
执行器
(串行/并行)
子图
形状
计算
(串行/并行)
子图
准备
执行器
(串行/并行)
子图
运行
执行器
3 1 5 7 4
0 1 0 0 4
4 4 1 2 5
tensors

应对 - 缓存管理


100
150
100
...
buffers
3 1 5 7 4
0 1 0 0 4
4 4 1 2 5
tensors
3 1 5 7 4
0 1 0 0 4
4 4 1 2 5
tensors
(管线内串行)
管线
(管线内串行)
管线
(管线内串行)
管线
模型
调度
(串行/并行)
子图
创建
执行器
(串行/并行)
子图
形状
计算
(串行/并行)
子图
准备
执行器
(串行/并行)
子图
运行
执行器
3 1 5 7 4
0 1 0 0 4
4 4 1 2 5
tensors
后续推理

应对 - 缓存管理


100
150
100
...
buffers
3 1 5 7 4
0 1 0 0 4
4 4 1 2 5
tensors
3 1 5 7 4
0 1 0 0 4
4 4 1 2 5
tensors
(管线内串行)
管线
(管线内串行)
管线
(管线内串行)
管线
模型
调度
(串行/并行)
子图
创建
执行器
(串行/并行)
子图
形状
计算
(串行/并行)
子图
准备
执行器
(串行/并行)
子图
运行
执行器
3 1 5 7 4
0 1 0 0 4
4 4 1 2 5
tensors
响应形变

应对 - 缓存管理

- tensor / buffer 内存由后端分配
- 内存在后端内的执行器间复用
- 内存分配默认按照32位对齐

应对 - 工具集

1. 单元测试

```
# ./run_test.out matching substring
./run_test.out op/convolution
```

2. Profiler

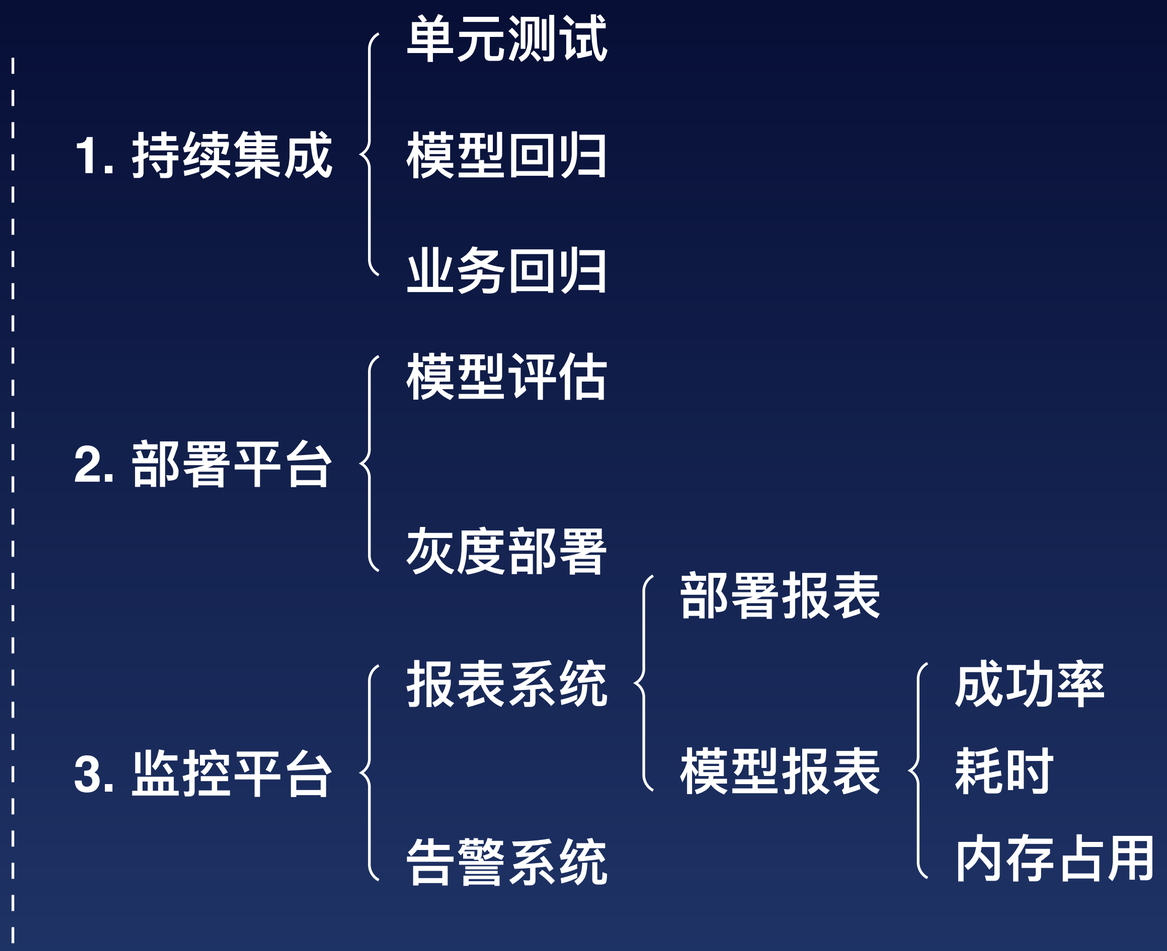
```
# ./timeProfile.out model loops backend dims
./timeProfile.out model.mnn 10 0 1x3x448x448
```

3. 单端比较

```
# ./MNNV2Basic.out model loops outputs backend dims
# ./checkDir.out output comparison tolerance
./MNNV2Basic.out model.mnn 10 1 0 1x3x224x224
./checkDir.out output comparison 1
```

4. 端间比较

```
# ./backendTest.out model, backend, tolerance
./backendTest.out model.mnn 3 0.15
```



应对 - 分析MetalPerformanceShaders

1. 下载固件并解压

2. 拆解metallib

```
MPSNeuralNetwork.framework $ pwd
/Volumes/PeaceF16F203.D22D221OS/System/Library/
Frameworks/MetalPerformanceShaders.framework/Frameworks/
MPSNeuralNetwork.framework
MPSNeuralNetwork.framework $ cp default.metallib path/
to/airs && cd path/to/airs
airs $ python3 unmetallib.py default.metallib
```

工具参考：

https://github.com/zhuowei/MetalShaderTools

3. 反汇编air

```
airs $ ls
out_cnnConvArray_3xIn_8xOut_3_1.air
out_cnnConvArray_3xIn_8xOut_4_1.air
out_cnnConvArray_4xIn_16xOut_1_1.air
out_cnnConvArray_4xIn_16xOut_1_2.air
out_cnnConvArray_4xIn_16xOut_1_3.air
out_cnnConvArray_4xIn_16xOut_1_4.air
out_cnnConvArray_4xIn_16xOut_2_1.air
out_cnnConvArray_4xIn_16xOut_2_2.air
out_cnnConvArray_4xIn_16xOut_3_1.air
out_cnnConvArray_4xIn_16xOut_4_1.air
out_cnnConvArray_4xIn_4xOut_1_1.air
out_cnnConvArray_4xIn_4xOut_1_2.air
…
airs $ llvm-dis out_cnnConvArray_3xIn_8xOut_3_1.air
airs $ cat out_cnnConvArray_3xIn_8xOut_3_1.air.ll
; Function Attrs: convergent nounwind
define void @cnnConvArray_3xIn_8xOut_3_1(
 %struct._texture_2d_array_t addrspace(1)*,
 %struct._texture_2d_t addrspace(1)*,
 %struct._texture_2d_array_t addrspace(1)*,
 %struct._texture_2d_t addrspace(1)*,
 %struct.cnnConvArrayParams addrspace(2)* noalias readonly
 dereferenceable(176),
 half addrspace(2)* noalias nocapture readonly,
 half addrspace(2)* noalias nocapture readonly,
 <3 x i16>, <3 x i16>, <3 x i16>,
 <3 x i16>, <3 x i16>, <3 x i16>,
```

应对 - 数据布局

NCHW布局下，如何利用SIMD加速卷积?

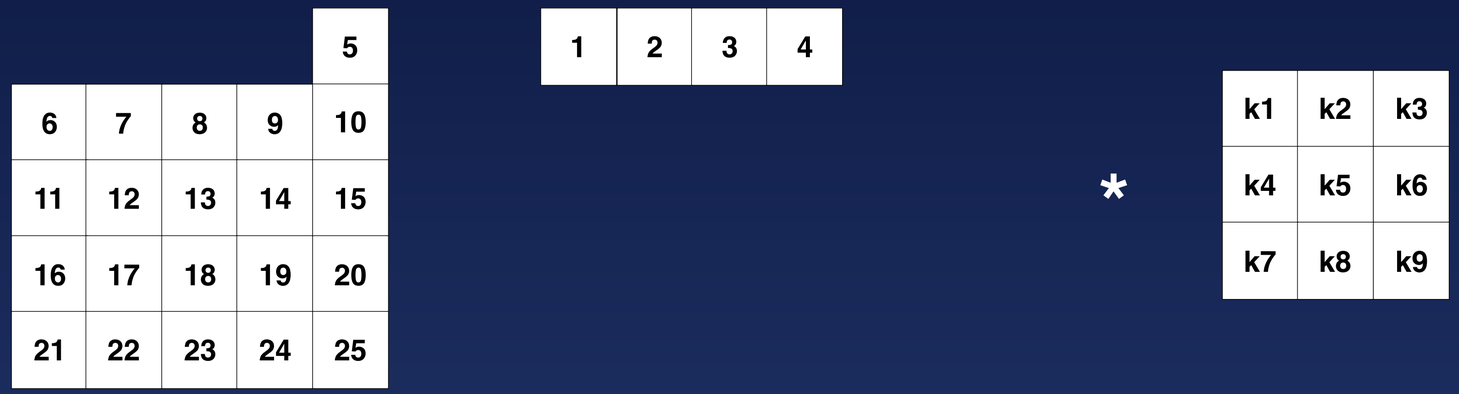
1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

*

k1	k2	k3
k4	k5	k6
k7	k8	k9

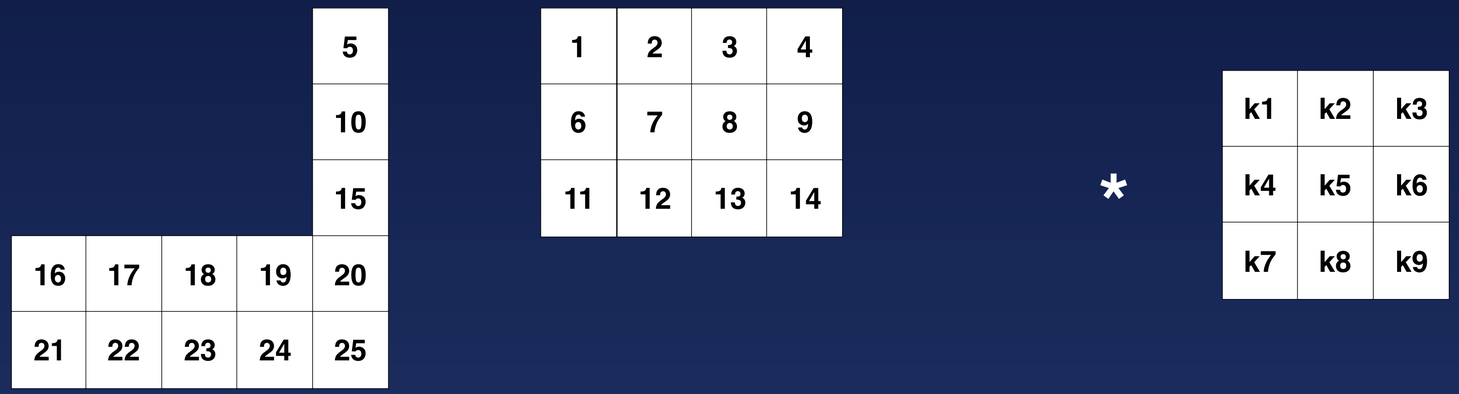
应对 - 数据布局

NCHW布局下，如何利用SIMD加速卷积?



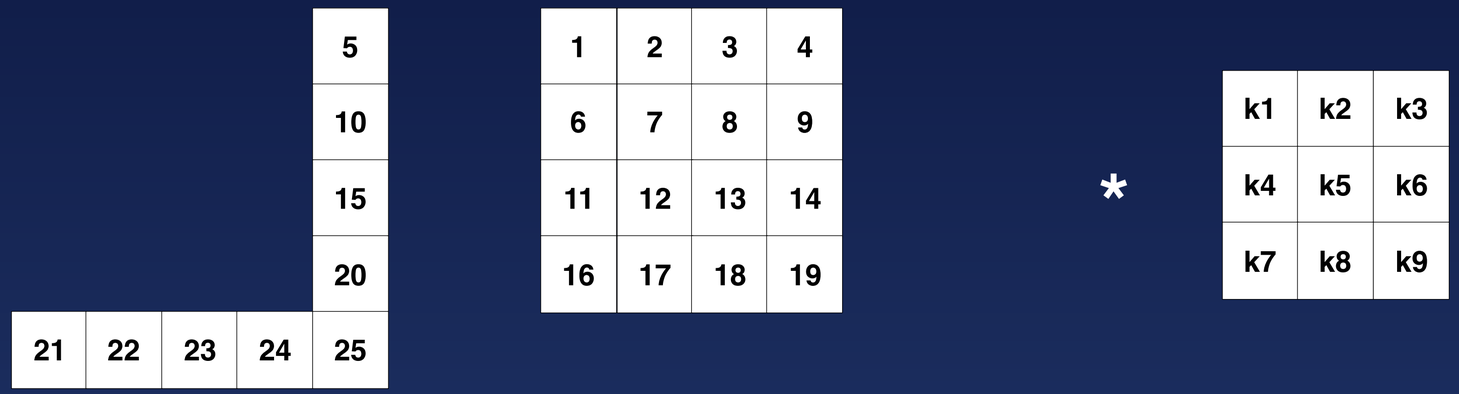
应对 - 数据布局

NCHW布局下，如何利用SIMD加速卷积?



应对 - 数据布局

NCHW布局下，如何利用SIMD加速卷积?



应对 - 数据布局

NCHW布局下，如何利用SIMD加速卷积?

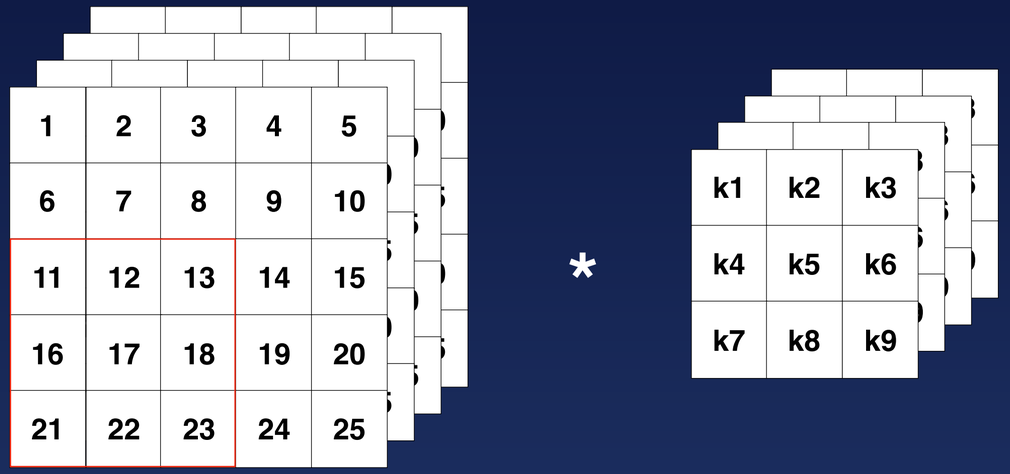
1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

*

k1	k2	k3
k4	k5	k6
k7	k8	k9

应对 - 数据布局

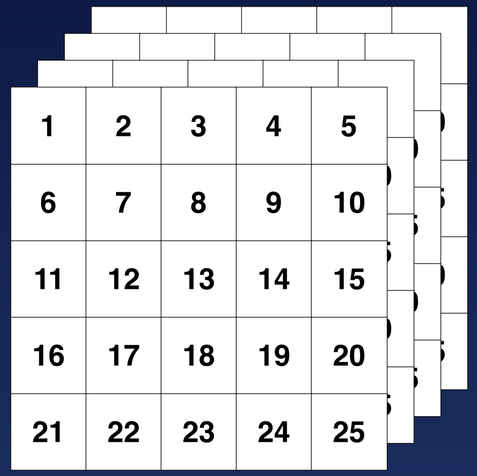
NCHW布局下，如何利用SIMD加速卷积?



- kernel、stride、dilation影响
  - 数据读写指令
  - 数据加载有效性
  - 数据复用逻辑
  - 寄存器开销
- 边缘数据无法利用SIMD加速
- 优化分支多，影响包大小

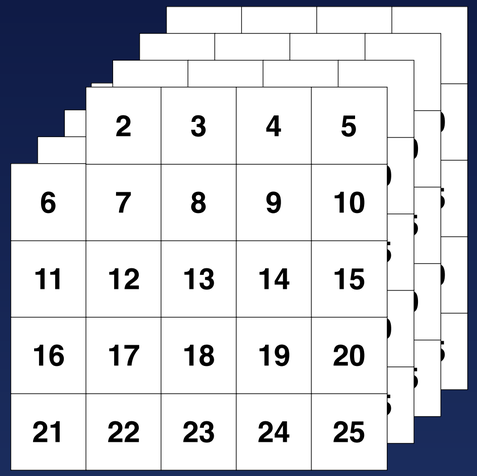
应对 - 数据布局

NC/4HW4布局下，如何利用SIMD加速卷积?



应对 - 数据布局

NC/4HW4布局下，如何利用SIMD加速卷积?



应对 - 数据布局

NC/4HW4布局下，如何利用SIMD加速卷积?

1	1	2	2	3	3	4	4	5	5
1	1	2	2	3	3	4	4	5	5
6	6	7	7	8	8	9	9	10	10
6	6	7	7	8	8	9	9	10	10
11	11	12	12	13	13	14	14	15	15
11	11	12	12	13	13	14	14	15	15
16	16	17	17	18	18	19	19	20	20
16	16	17	17	18	18	19	19	20	20
21	21	22	22	23	23	24	24	25	25
21	21	22	22	23	23	24	24	25	25

应对 - 数据布局

NC/4HW4布局下，如何利用SIMD加速卷积?

1	1	2	2	3	3	4	4	5	5
1	1	2	2	3	3	4	4	5	5
6	6	7	7	8	8	9	9	10	10
6	6	7	7	8	8	9	9	10	10
11	11	12	12	13	13	14	14	15	15
11	11	12	12	13	13	14	14	15	15
16	16	17	17	18	18	19	19	20	20
16	16	17	17	18	18	19	19	20	20
21	21	22	22	23	23	24	24	25	25
21	21	22	22	23	23	24	24	25	25

*

k1	k1	k2	k2	k3	k3
k1	k1	k2	k2	k3	k3
k4	k4	k5	k5	k6	k6
k4	k4	k5	k5	k6	k6
k7	k7	k8	k8	k9	k9
k7	k7	k8	k8	k9	k9

应对 - 数据布局

NC/4HW4布局下，如何利用SIMD加速卷积?

1	1	2	2	3	3	4	4	5	5
1	1	2	2	3	3	4	4	5	5
6	6	7	7	8	8	9	9	10	10
6	6	7	7	8	8	9	9	10	10
11	11	12	12	13	13	14	14	15	15
11	11	12	12	13	13	14	14	15	15
16	16	17	17	18	18	19	19	20	20
16	16	17	17	18	18	19	19	20	20
21	21	22	22	23	23	24	24	25	25
21	21	22	22	23	23	24	24	25	25

*

k1	k1	k2	k2	k3	k3
k1	k1	k2	k2	k3	k3
k4	k4	k5	k5	k6	k6
k4	k4	k5	k5	k6	k6
k7	k7	k8	k8	k9	k9
k7	k7	k8	k8	k9	k9

应对 - 数据布局

NC/4HW4布局下，如何利用SIMD加速卷积?

1	1	2	2	3	3	4	4	5	5
1	1	2	2	3	3	4	4	5	5
6	6	7	7	8	8	9	9	10	10
6	6	7	7	8	8	9	9	10	10
11	11	12	12	13	13	14	14	15	15
11	11	12	12	13	13	14	14	15	15
16	16	17	17	18	18	19	19	20	20
16	16	17	17	18	18	19	19	20	20
21	21	22	22	23	23	24	24	25	25
21	21	22	22	23	23	24	24	25	25

*

k1	k1	k2	k2	k3	k3
k1	k1	k2	k2	k3	k3
k4	k4	k5	k5	k6	k6
k4	k4	k5	k5	k6	k6
k7	k7	k8	k8	k9	k9
k7	k7	k8	k8	k9	k9

✓

- 适应任何kernel、stride、dilation
- 充分利用SIMD
- 数据复用更简单
- 适应任何pad、输入大小
- NEON、汇编编写难度下降
- 适应GPU数据布局
- 轻量化，包大小增量低

✗

- 最高3通道内存/算力浪费
- 部分Op需要额外布局转换

应对 - Winograd

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

* K

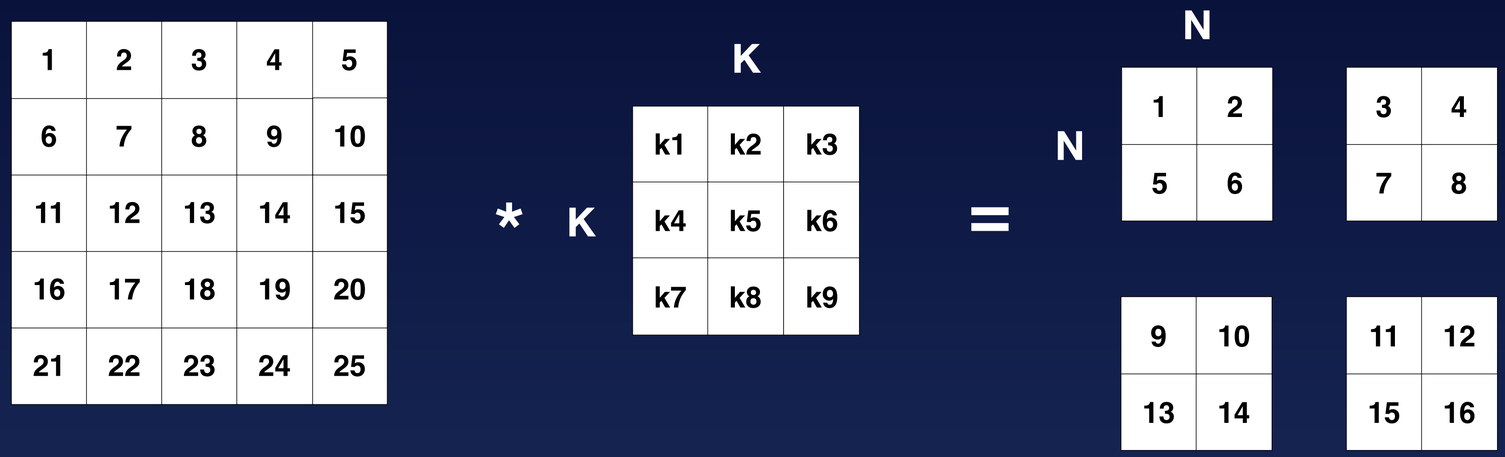
K

k1	k2	k3
k4	k5	k6
k7	k8	k9

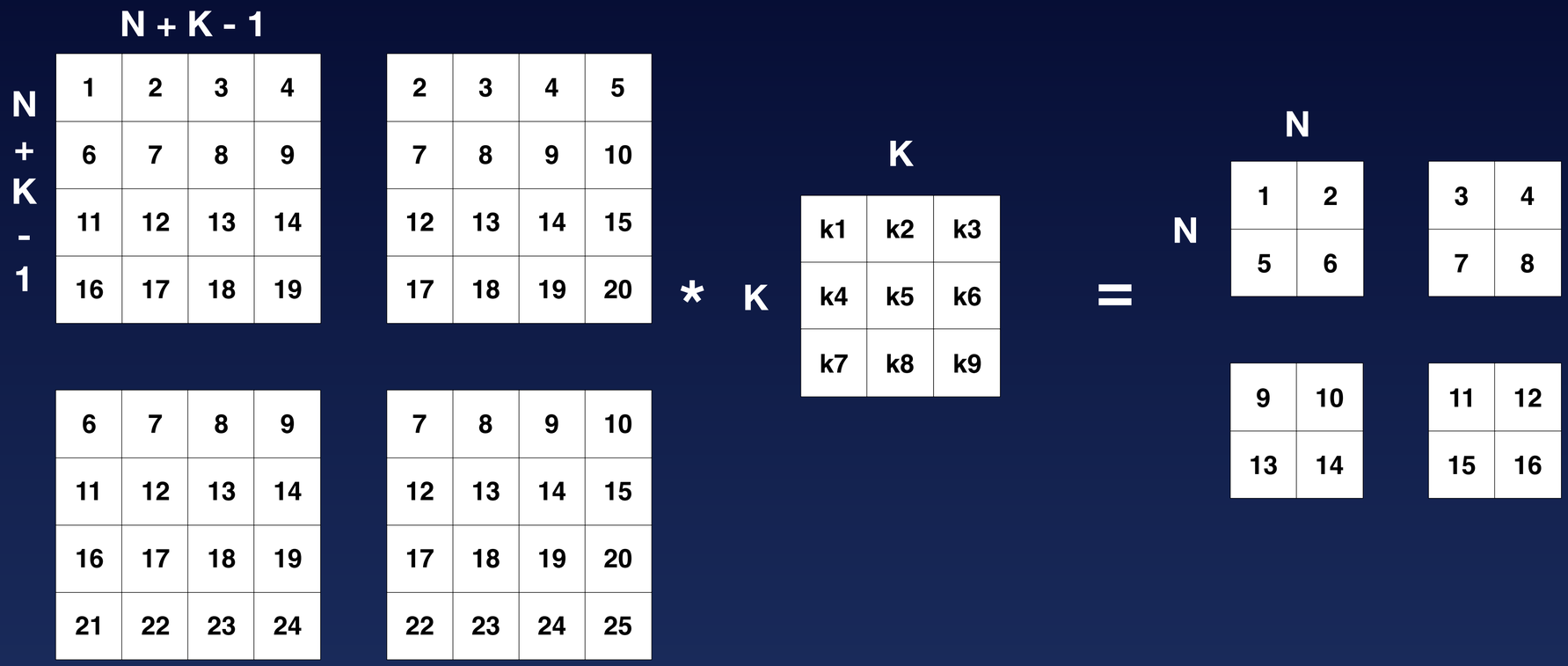
=

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

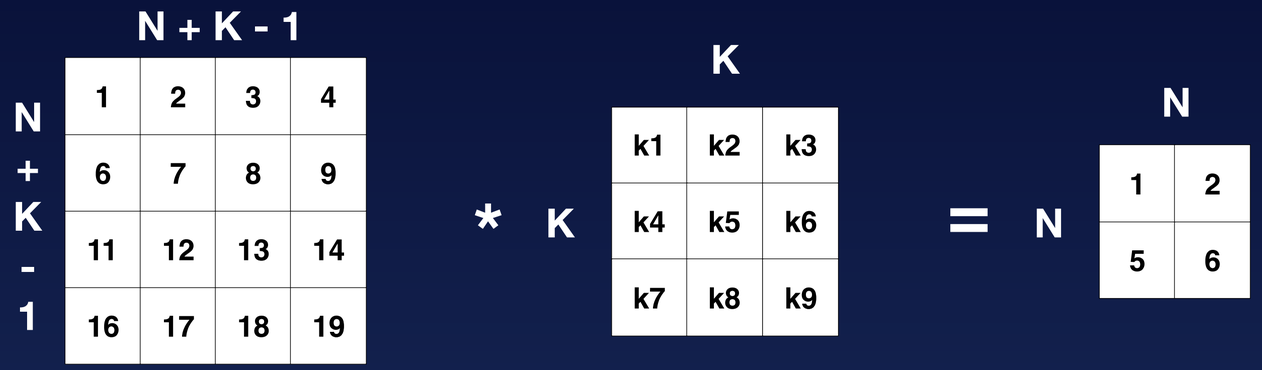
应对 - Winograd


1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25
*
K
K
k1 k2 k3
k4 k5 k6
k7 k8 k9
=
N
N
1 2
5 6
3 4
7 8
9 10
13 14
11 12
15 16

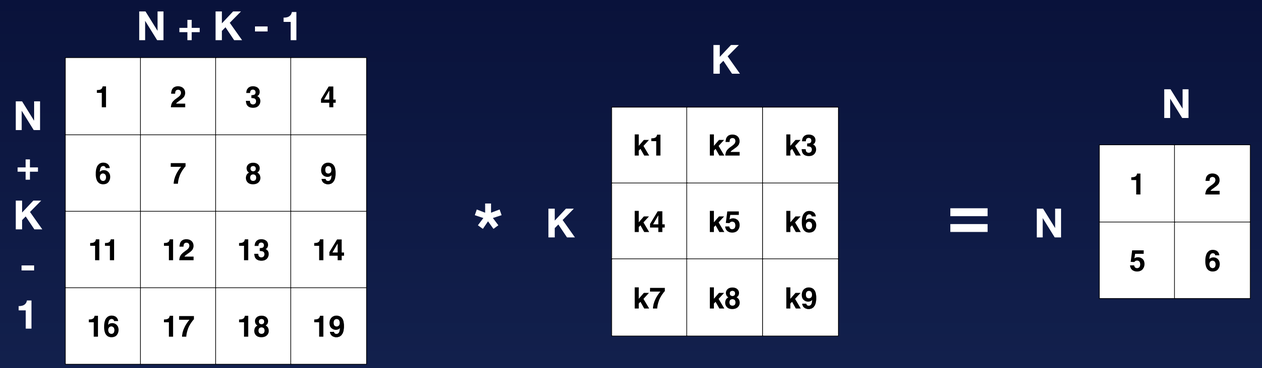
应对 - Winograd


N + K - 1
N + K - 1
1 2 3 4
6 7 8 9
11 12 13 14
16 17 18 19
2 3 4 5
7 8 9 10
12 13 14 15
17 18 19 20
6 7 8 9
11 12 13 14
16 17 18 19
21 22 23 24
7 8 9 10
12 13 14 15
17 18 19 20
22 23 24 25
*
K
K
k1 k2 k3
k4 k5 k6
k7 k8 k9
=
N
N
1 2
5 6
3 4
7 8
9 10
13 14
11 12
15 16

应对 - Winograd

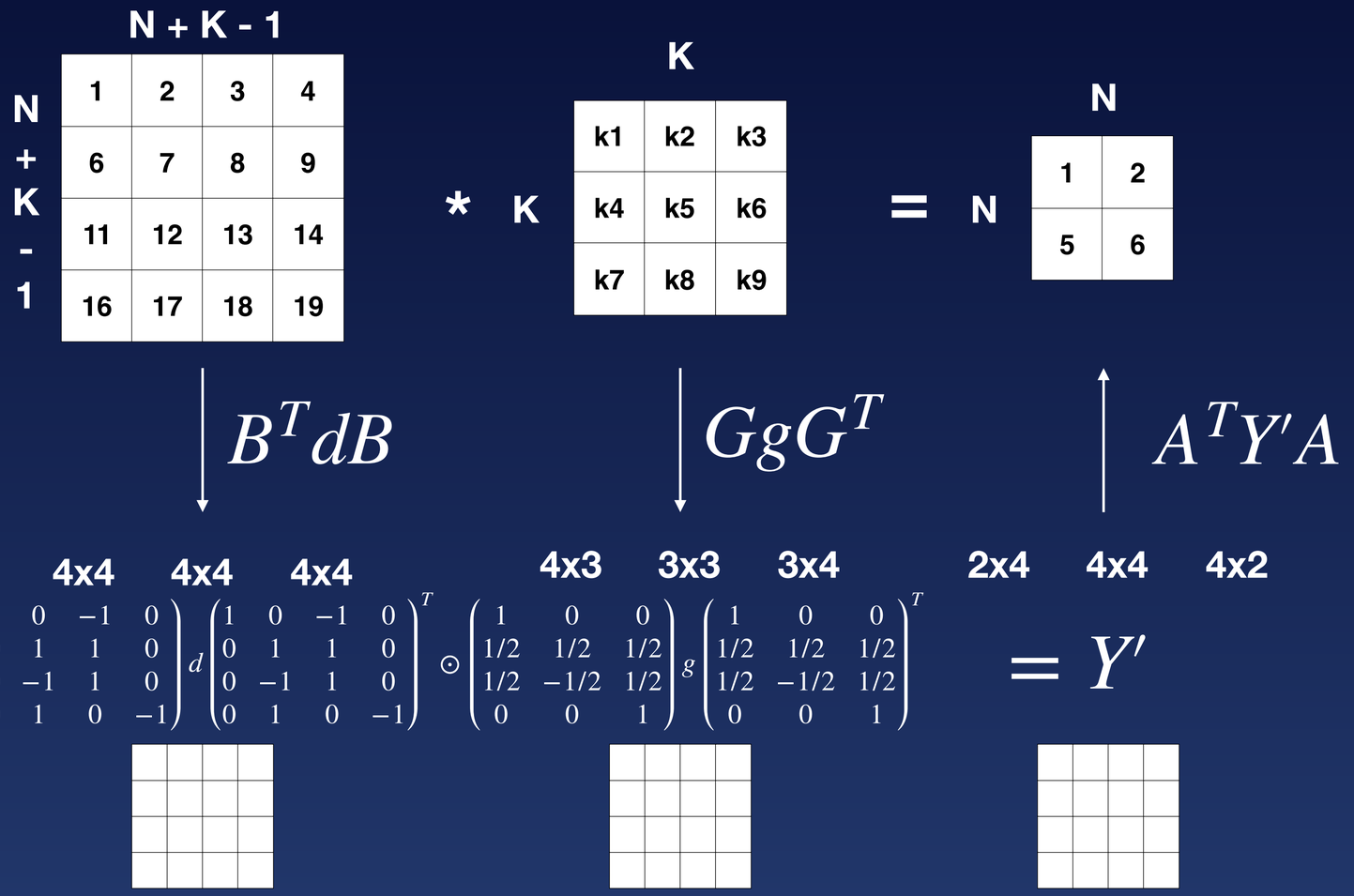

N + K - 1
N + K - 1
1 2 3 4
6 7 8 9
11 12 13 14
16 17 18 19
*
K
K
k1 k2 k3
k4 k5 k6
k7 k8 k9
=
N
N
1 2
5 6

应对 - Winograd


N + K - 1
N + K - 1
1 2 3 4
6 7 8 9
11 12 13 14
16 17 18 19
*
K
K
k1 k2 k3
k4 k5 k6
k7 k8 k9
=
N
N
1 2
5 6

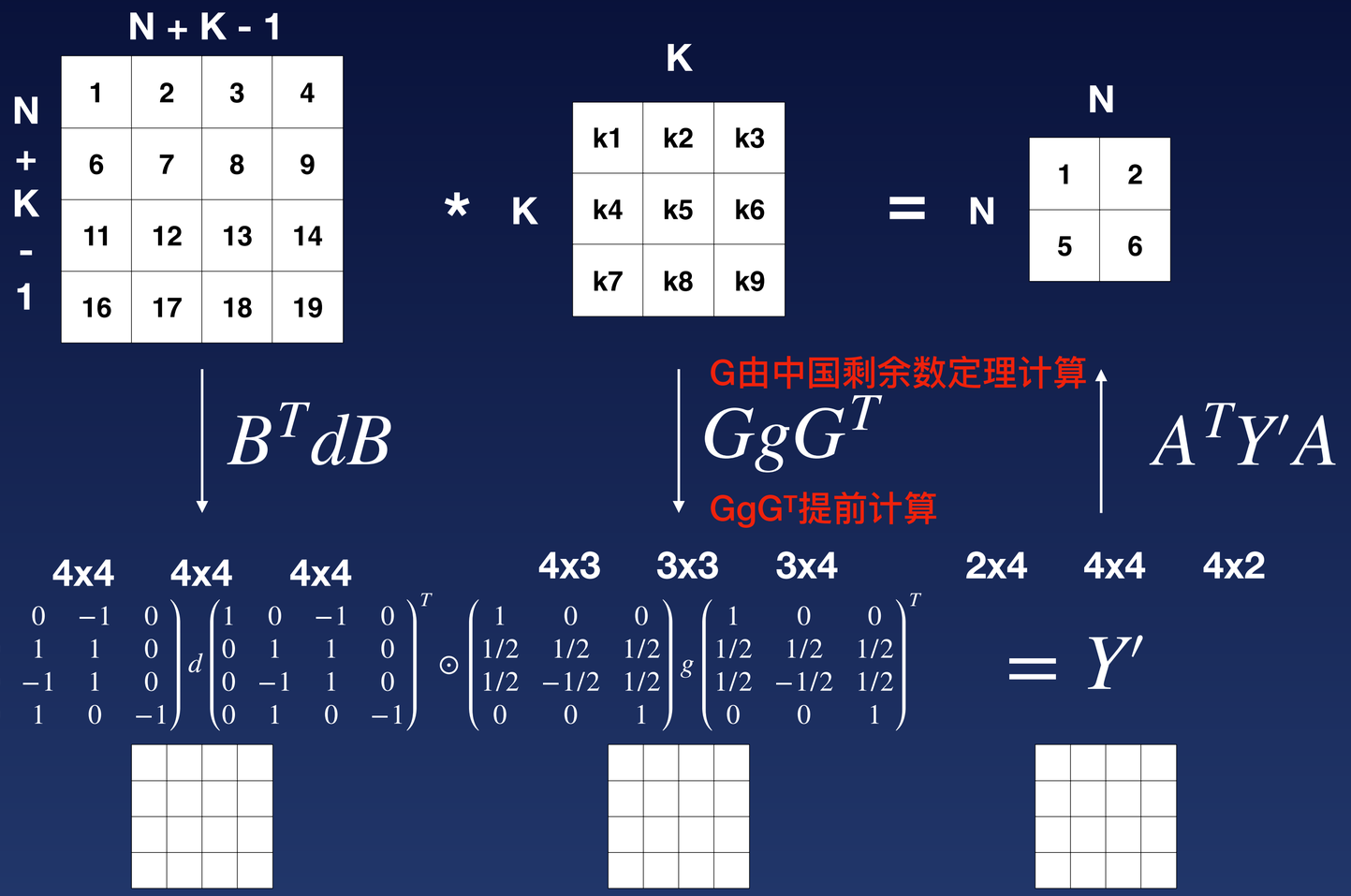
应对 - Winograd

$$Y = A^T[(GgG^T) \odot (B^TdB)]A$$



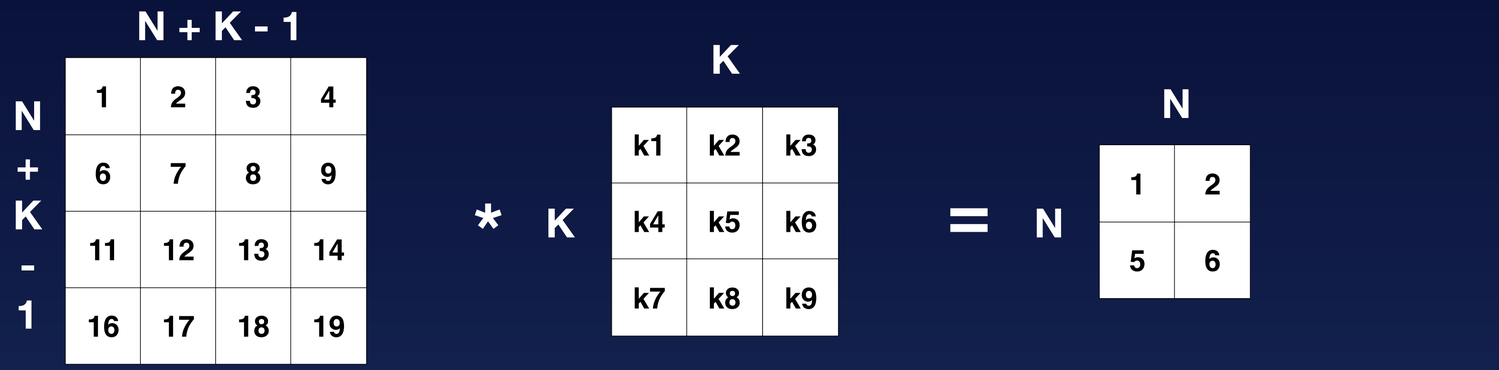
应对 - Winograd

$$Y = A^T[(GgG^T) \odot (B^TdB)]A$$



应对 - Winograd

$$Y = A^T[(GgG^T) \odot (B^T dB)]A$$



B根据N和K计算

$B^T dB$

BᵀdB即时计算

G由中国剩余数定理计算

GgG^T

GgGᵀ提前计算

A根据N和K计算

$A^T Y' A$

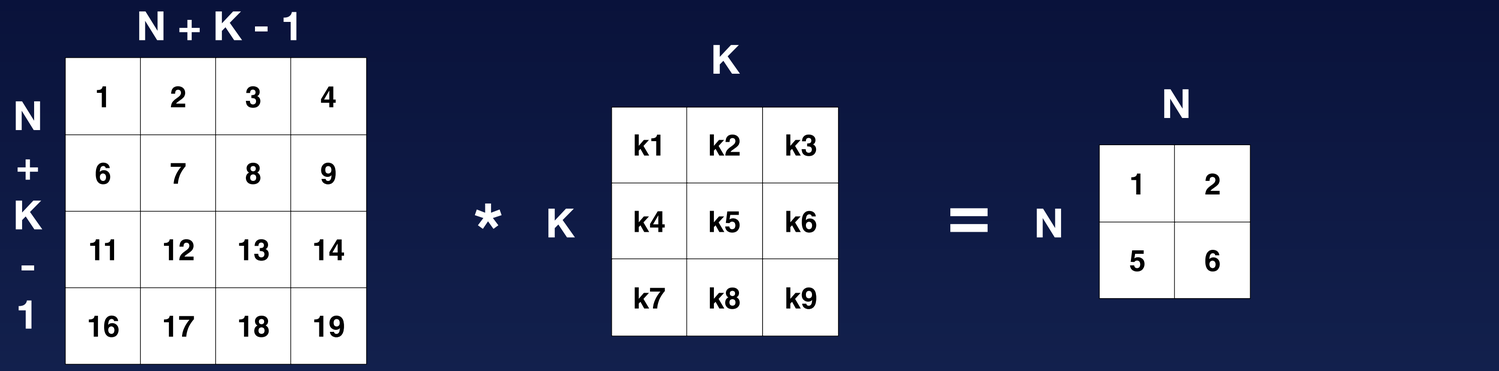
AᵀY`A即时计算

4x4 4x4 4x4 4x3 3x3 3x4 2x4 4x4 4x2

$$\begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix}^T d \begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} \odot \begin{pmatrix} 1 & 0 & 0 \\ 1/2 & 1/2 & 1/2 \\ 1/2 & -1/2 & 1/2 \\ 0 & 0 & 1 \end{pmatrix}^T g \begin{pmatrix} 1 & 0 & 0 \\ 1/2 & 1/2 & 1/2 \\ 1/2 & -1/2 & 1/2 \\ 0 & 0 & 1 \end{pmatrix} = Y'$$

应对 - Winograd

$$Y = A^T[(GgG^T) \odot (B^T dB)]A$$



B根据N和K计算

$B^T dB$

BᵀdB即时计算

G由中国剩余数定理计算

GgG^T

GgGᵀ提前计算

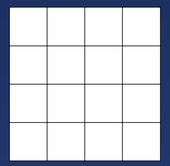
A根据N和K计算

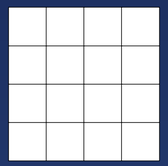
$A^T Y' A$

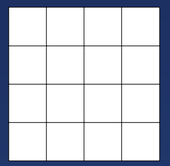
AᵀY`A即时计算

4x4 4x4 4x4 4x3 3x3 3x4 2x4 4x4 4x2

$$\begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} d \begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix}^T \odot \begin{pmatrix} 1 & 0 & 0 \\ 1/2 & 1/2 & 1/2 \\ 1/2 & -1/2 & 1/2 \\ 0 & 0 & 1 \end{pmatrix} g \begin{pmatrix} 1 & 0 & 0 \\ 1/2 & 1/2 & 1/2 \\ 1/2 & -1/2 & 1/2 \\ 0 & 0 & 1 \end{pmatrix}^T = Y'$$







应对 - Winograd

N + K - 1				
N + K - 1	1	2	3	4
	6	7	8	9
	11	12	13	14
	16	17	18	19

* K

K		
k1	k2	k3
k4	k5	k6
k7	k8	k9

= N

N	
1	2
5	6

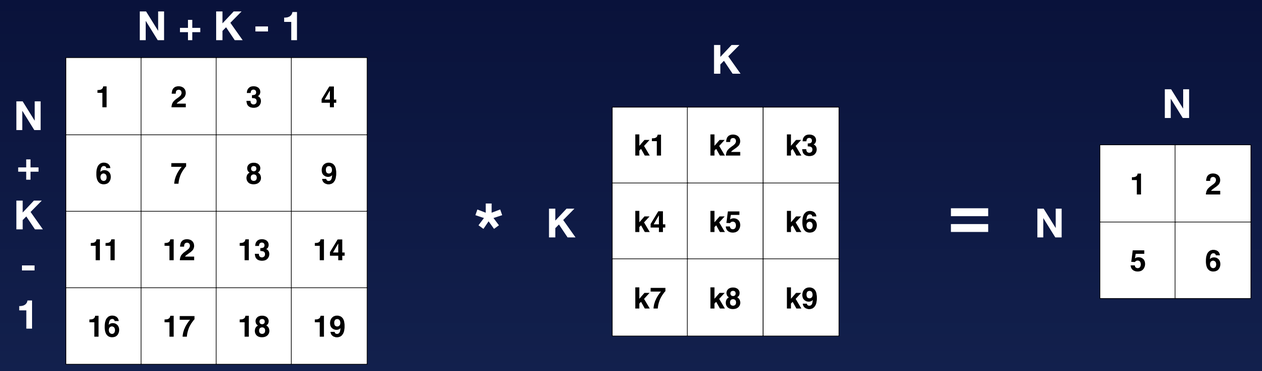
9 x 4 = 36 次乘法计算

$$\begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} d \begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix}^T \odot \begin{pmatrix} 1 & 0 & 0 \\ 1/2 & 1/2 & 1/2 \\ 1/2 & -1/2 & 1/2 \\ 0 & 0 & 1 \end{pmatrix} g \begin{pmatrix} 1 & 0 & 0 \\ 1/2 & 1/2 & 1/2 \\ 1/2 & -1/2 & 1/2 \\ 0 & 0 & 1 \end{pmatrix}^T = Y'$$

4 x 4 = 16 次乘法计算

36 / 16 = 2.25

应对 - Winograd



9 x 4 = 36 次乘法计算

$$\begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} d \begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix}^T \odot \begin{pmatrix} 1 & 0 & 0 \\ 1/2 & 1/2 & 1/2 \\ 1/2 & -1/2 & 1/2 \\ 0 & 0 & 1 \end{pmatrix} g \begin{pmatrix} 1 & 0 & 0 \\ 1/2 & 1/2 & 1/2 \\ 1/2 & -1/2 & 1/2 \\ 0 & 0 & 1 \end{pmatrix}^T = Y'$$

4 x 4 = 16 次乘法计算

36 / 16 = 2.25

以内存换取性能

N	K	加速倍率	内存增加倍数
2	3	2.25	0.78
4	3	4.00	3.00
6	3	5.06	6.11

应对 - Strassen

$$\begin{pmatrix} A1 & A2 & A3 & A4 \\ A5 & A6 & A7 & A8 \\ A9 & A10 & A11 & A12 \\ A13 & A14 & A15 & A16 \end{pmatrix} * \begin{pmatrix} B1 & B2 & B3 & B4 \\ B5 & B6 & B7 & B8 \\ B9 & B10 & B11 & B12 \\ B13 & B14 & B15 & B16 \end{pmatrix} = \begin{pmatrix} C1 & C2 & C3 & C4 \\ C5 & C6 & C7 & C8 \\ C9 & C10 & C11 & C12 \\ C13 & C14 & C15 & C16 \end{pmatrix}$$

应对 - Strassen

$$\begin{pmatrix} a11 & a12 \\ a21 & a22 \end{pmatrix} * \begin{pmatrix} b11 & b12 \\ b21 & b22 \end{pmatrix} = \begin{pmatrix} c11 & c12 \\ c21 & c22 \end{pmatrix}$$

应对 - Strassen

$$\left(\begin{array}{c|c} a11 & a12 \\ \hline a21 & a22 \end{array}\right) * \left(\begin{array}{c|c} b11 & b12 \\ \hline b21 & b22 \end{array}\right) = \left(\begin{array}{c|c} c11 & c12 \\ \hline c21 & c22 \end{array}\right)$$

$$v_1 = (a_{11} + a_{22})(b_{11} + b_{22})$$
$$v_2 = (a_{21} + a_{22})(b_{11})$$
$$v_3 = (a_{11})(b_{12} - b_{22})$$
$$v_4 = (a_{22})(b_{21} - b_{11})$$
$$v_5 = (a_{11} + a_{12})(b_{22})$$
$$v_6 = (a_{21} - a_{11})(b_{11} + b_{12})$$
$$v_7 = (a_{12} - a_{22})(b_{21} + b_{22})$$

$$c_{11} = v_1 + v_4 - v_5 + v_7$$
$$c_{21} = v_2 + v_4$$
$$c_{12} = v_3 + v_5$$
$$c_{22} = v_1 + v_3 - v_2 + v_6$$

使用 10 + 8 次矩阵加减替代1次矩阵乘法（占比1/8）

应对 - Strassen

$$\begin{pmatrix} a11 & a12 \\ a21 & a22 \end{pmatrix} * \begin{pmatrix} b11 & b12 \\ b21 & b22 \end{pmatrix} = \begin{pmatrix} c11 & c12 \\ c21 & c22 \end{pmatrix}$$

$$s_1 = a_{21} + a_{22}$$
$$s_2 = s_1 - a_{11}$$
$$s_3 = a_{11} - a_{21}$$
$$s_4 = a_{12} - s_2$$
$$t_1 = b_{21} - b_{11}$$
$$t_2 = b_{22} - t_1$$
$$t_3 = b_{22} - b_{12}$$
$$t_4 = t_2 - b_{21}$$

$$m_1 = a_{11}b_{11}$$
$$m_2 = a_{12}b_{21}$$
$$m_3 = s_4b_{22}$$
$$m_4 = a_{22}t_4$$
$$m_5 = s_1t_1$$
$$m_6 = s_2t_2$$
$$m_7 = s_3t_3$$

$$u_1 = m_1 + m_2$$
$$u_2 = m_1 + m_6$$
$$u_3 = u_2 + m_7$$
$$u_4 = u_2 + m_5$$
$$u_5 = u_4 + m_3$$
$$u_6 = u_3 - m_4$$
$$u_7 = u_3 + m_5$$

$$c_{11} = u_1$$
$$c_{12} = u_5$$
$$c_{21} = u_6$$
$$c_{22} = u_7$$

使用 4 + 4 + 7 次矩阵加减替代1次矩阵乘法（占比1/8）

应对 - Strassen

$$\left(\begin{array}{c|c} a11 & a12 \\ \hline a21 & a22 \end{array}\right) * \left(\begin{array}{c|c} b11 & b12 \\ \hline b21 & b22 \end{array}\right) = \left(\begin{array}{c|c} c11 & c12 \\ \hline c21 & c22 \end{array}\right)$$

$$s_1 = a_{21} + a_{22}$$
$$s_2 = s_1 - a_{11}$$
$$s_3 = a_{11} - a_{21}$$
$$s_4 = a_{12} - s_2$$
$$t_1 = b_{21} - b_{11}$$
$$t_2 = b_{22} - t_1$$
$$t_3 = b_{22} - b_{12}$$
$$t_4 = t_2 - b_{21}$$

$$m_1 = a_{11}b_{11}$$
$$m_2 = a_{12}b_{21}$$
$$m_3 = s_4b_{22}$$
$$m_4 = a_{22}t_4$$
$$m_5 = s_1t_1$$
$$m_6 = s_2t_2$$
$$m_7 = s_3t_3$$

$$u_1 = m_1 + m_2$$
$$u_2 = m_1 + m_6$$
$$u_3 = u_2 + m_7$$
$$u_4 = u_2 + m_5$$
$$u_5 = u_4 + m_3$$
$$u_6 = u_3 - m_4$$
$$u_7 = u_3 + m_5$$

$$c_{11} = u_1$$
$$c_{12} = u_5$$
$$c_{21} = u_6$$
$$c_{22} = u_7$$

使用 4 + 4 + 7 次矩阵加减替代1次矩阵乘法（占比1/8）

$O(n^3)$

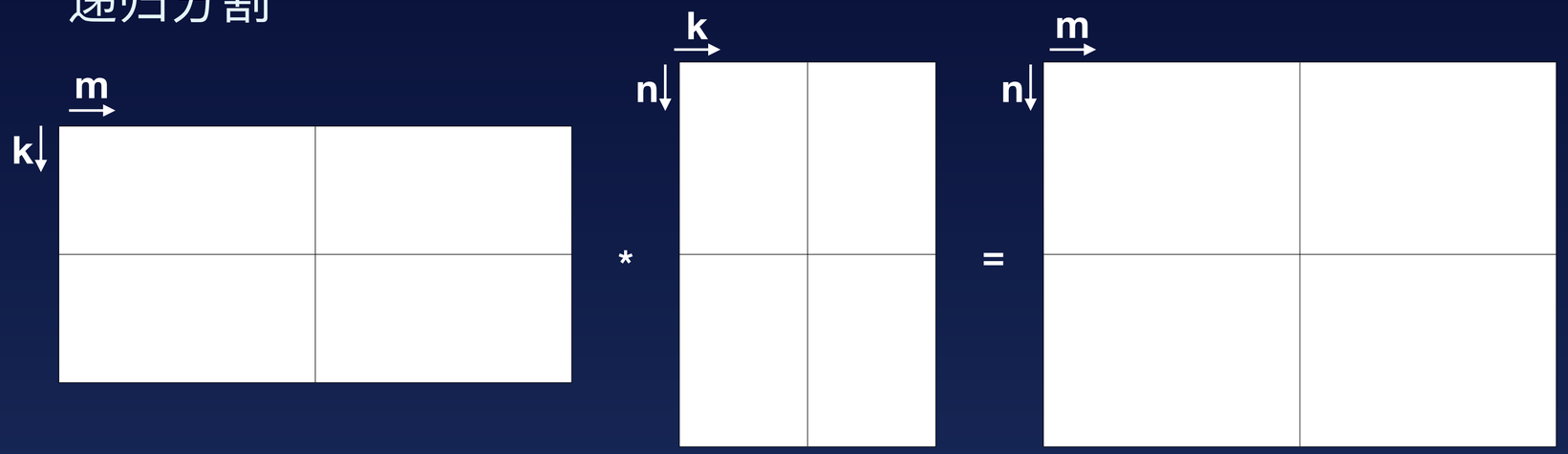
↓

$O(n^{2.81})$

矩阵较大时
矩阵乘法远慢于矩阵加减
收益较为明显

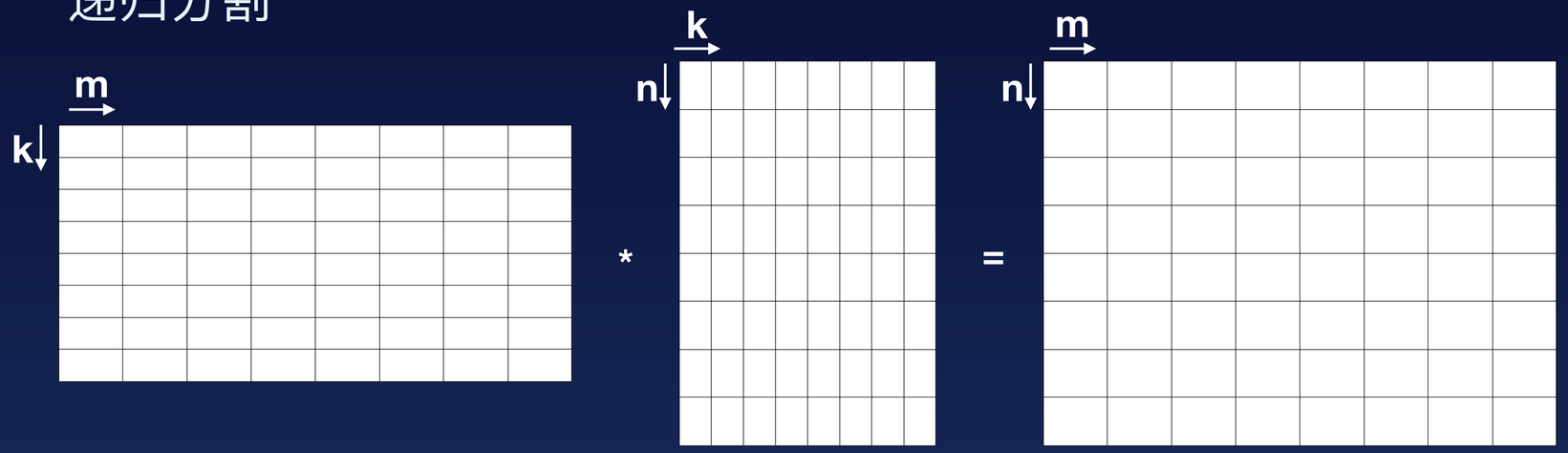
应对 - 递归Strassen

递归分割



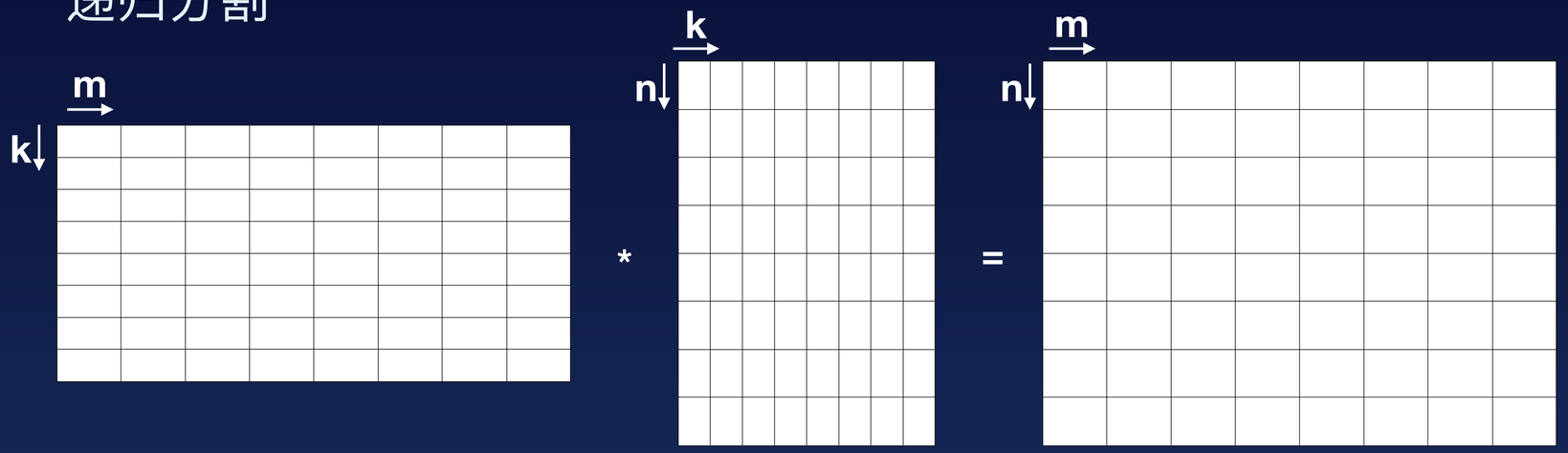
应对 - 递归Strassen

递归分割



应对 - 递归Strassen

递归分割



递归条件

减免矩阵乘法收益 = m * n * k / 8
S矩阵加减代价 = 4 * m / 2 * k / 2 * 3 = 3 * m * k
T矩阵加减代价 = 4 * n / 2 * k / 2 * 3 = 3 * n * k
U矩阵加减代价 = 7 * m / 2 * n / 2 * 3 = 5.25 * m * n
∴ m * n * k / 8 > 3 * m * k + 3 * n * k + 5.25 * m * n

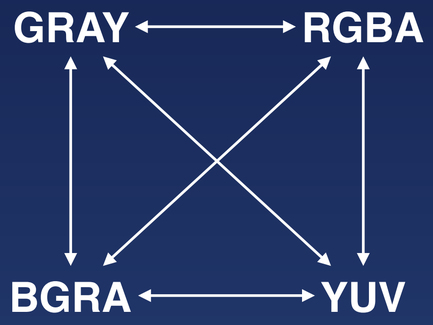
应对 - 链路优化

应对 - 链路优化

应对 - 链路优化

色值变化

色彩空间转换

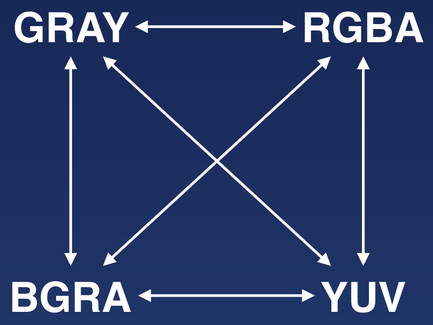


仿射变换

应对 - 链路优化

色值变化

色彩空间转换

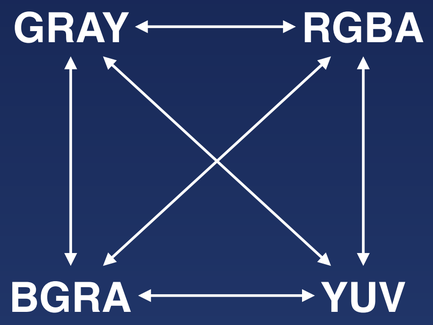


仿射变换

应对 - 链路优化

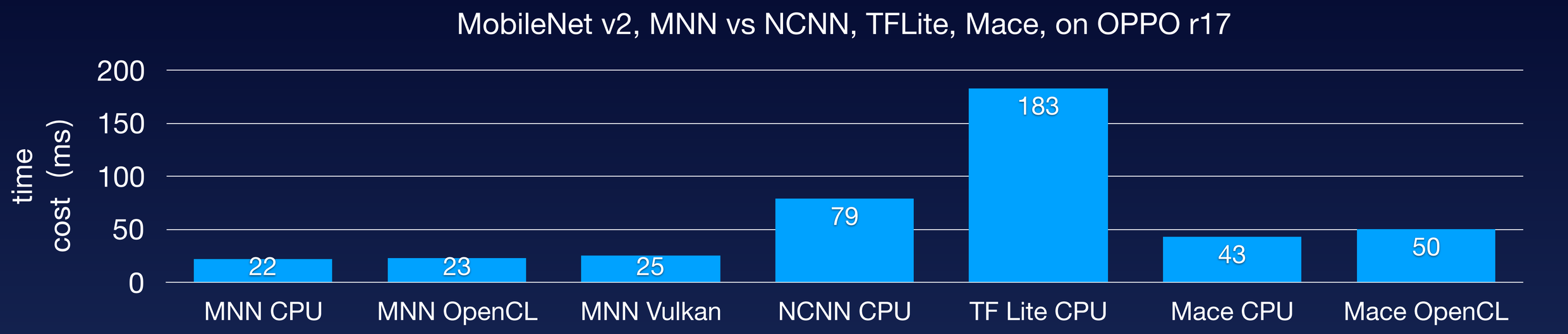
色值变化

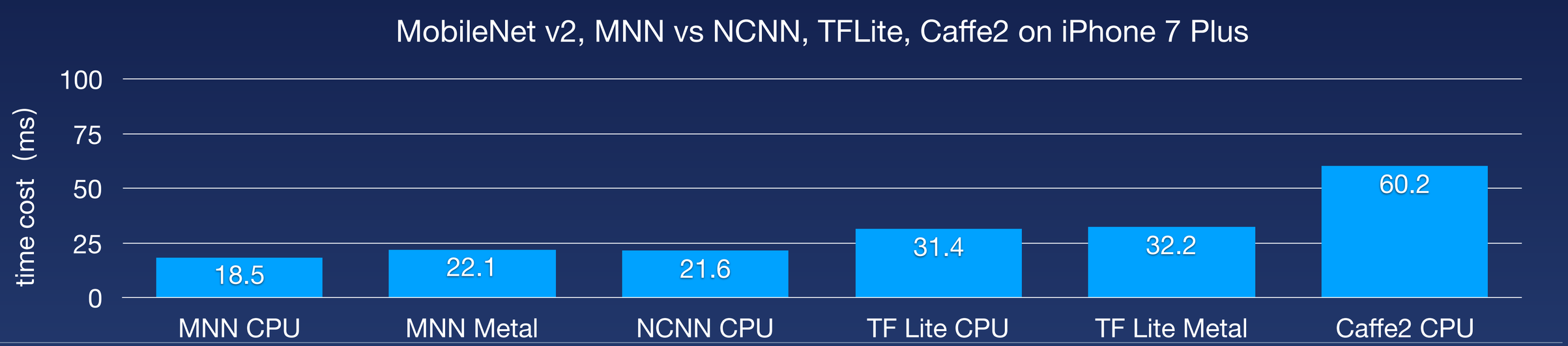
色彩空间转换



仿射变换

性能比较

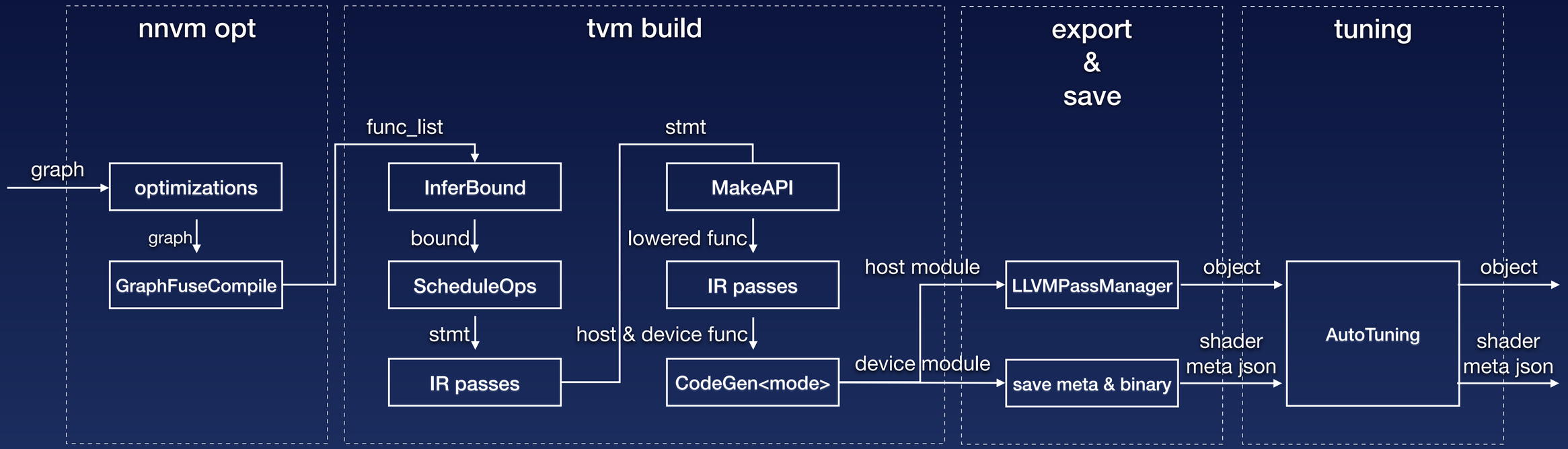

MobileNet v2, MNN vs NCNN, TFLite, Mace, on OPPO r17
time cost（ms）
200
150
100
50
0
22
23
25
79
183
43
50
MNN CPU
MNN OpenCL
MNN Vulkan
NCNN CPU
TF Lite CPU
Mace CPU
Mace OpenCL


MobileNet v2, MNN vs NCNN, TFLite, Caffe2 on iPhone 7 Plus
time cost（ms）
100
75
50
25
0
18.5
22.1
21.6
31.4
32.2
60.2
MNN CPU
MNN Metal
NCNN CPU
TF Lite CPU
TF Lite Metal
Caffe2 CPU

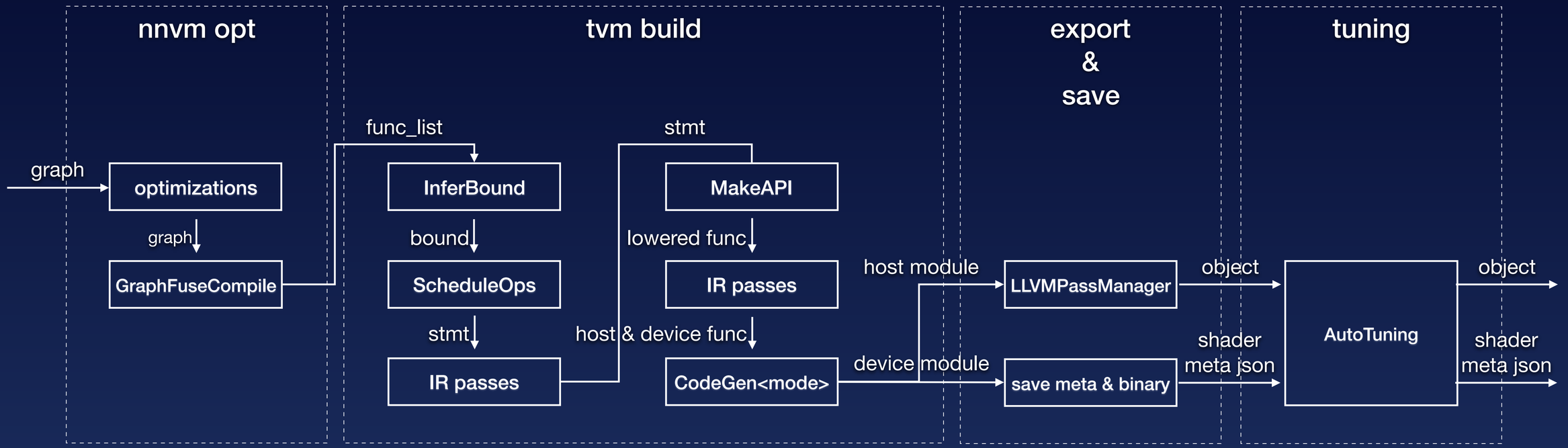
小结

对比	拓展性		缓存	CPU			GPU				CV
	模型	后端		算子	多线程	核心	Metal	OpenGL	OpenCL	Vulkan	
MNN	flatbuffer 自动生成	完全解耦 动态导入	适应输入 内存池	76	GCD/ OpenMP	通用优化 体积较小	55	11	33	35	O
TensorFlow Lite	flatbuffer 自动生成	部分解耦 编译确定	X	93	线程池	BLAS	17	19	X	X	X
Mace	protobuf 自动生成	部分解耦 编译确定	会话级 内存池	61	OpenMP	特定优化 体积较大	X	X	29	X	X
NCNN	自定义 手工编写	框架耦合 编译确定	会话级 内存池	65	OpenMP	特定优化 体积较大	X	X	X	32	O

基于编译器优化的方案

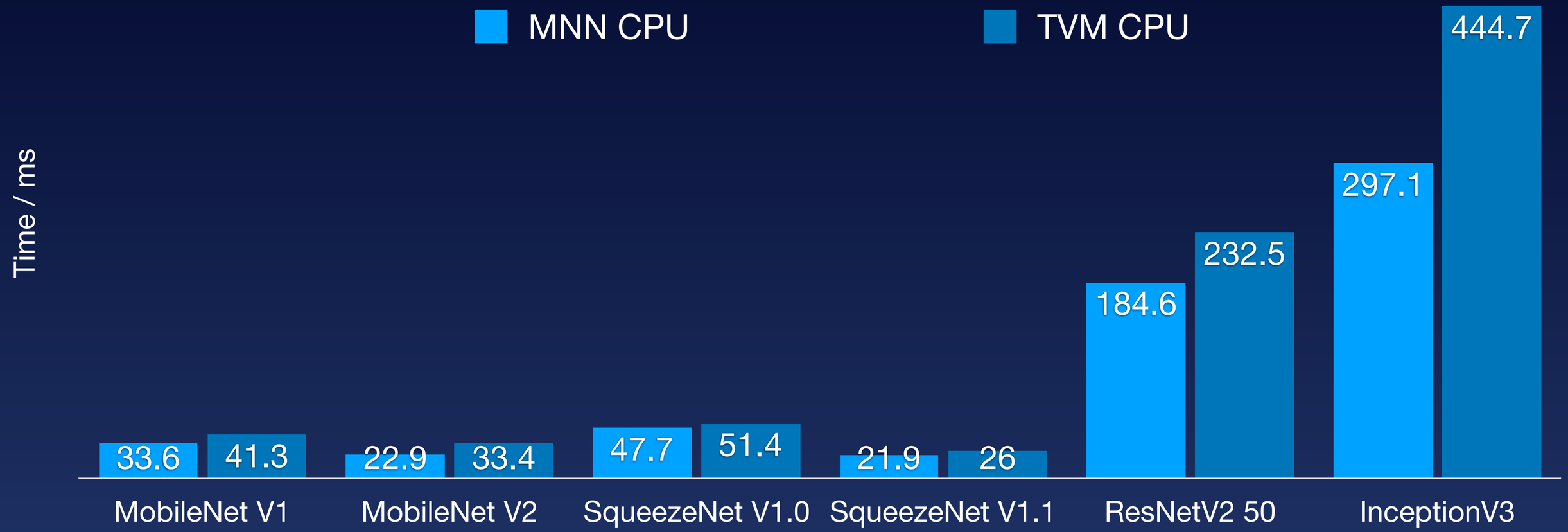

nnvm opt
tvm build
export & save
tuning
graph
optimizations
graph
GraphFuseCompile
func_list
InferBound
bound
ScheduleOps
stmt
IR passes
stmt
MakeAPI
lowered func
IR passes
host & device func
CodeGen<mode>
host module
device module
LLVMPassManager
save meta & binary
object
shader meta json
AutoTuning
object
shader meta json

基于编译器优化的方案



	基本单元	实现	时间	输入变化	模型变化	硬件变化	成本
推理引擎	算子	通用算法 + 特定优化	运行时	有限规则			固定
编译优化	表达式	有限带参模板	编译时	AutoTuning			与设备类型数成正比

性能比较

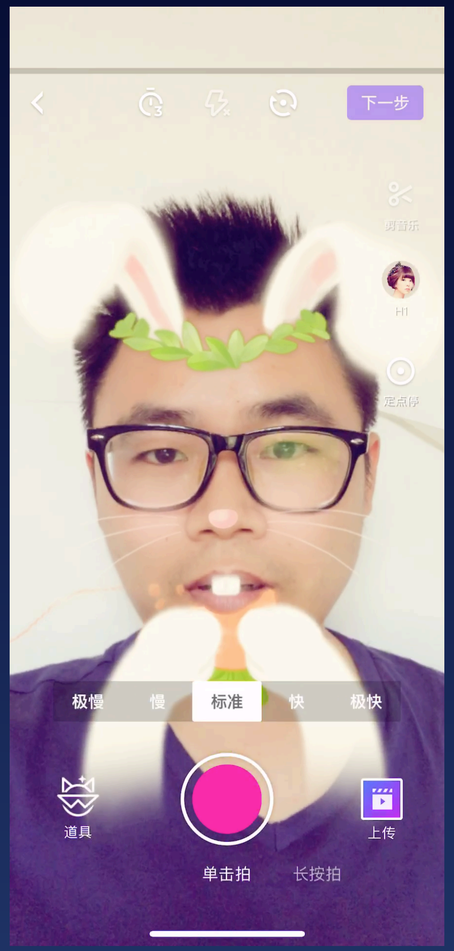


(Soc: HiSilicon Kirin 970) : (4 x Cortex A73 2.36GHz)

TVM数据来源：https://github.com/dmlc/tvm/wiki/Benchmark

应用场景

拍立淘

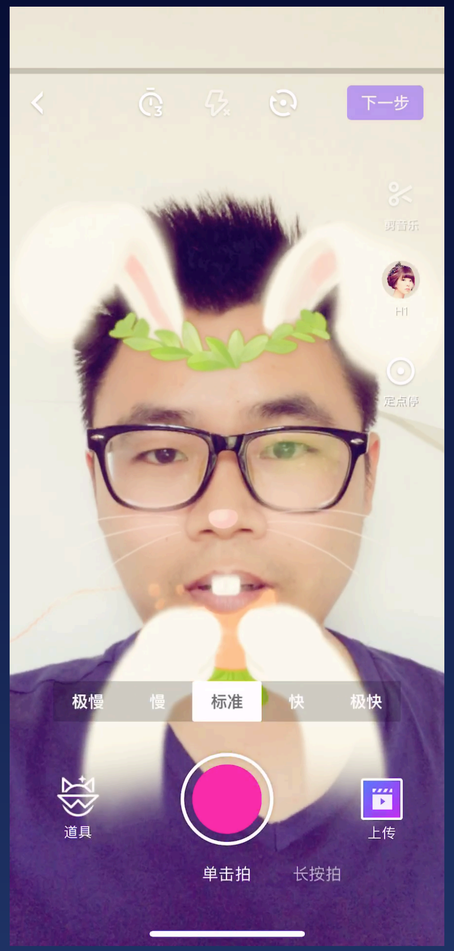


人脸贴纸

石头剪刀布

应用场景

拍立淘



人脸贴纸

石头剪刀布

Github开源

Roadmap

- 转换
  - 追加算子支持
  - 追加图优化匹配模板
  - 模型压缩工具
- 调度
  - 端侧训练
  - 联合学习
  - 设备自动选择
- 执行
  - 持续优化各后端算子实现
  - 优化量化的卷积、矩阵乘算法
  - CV支持GPU
  - 高性能计算库
  - 算法自动选择
- 其他
  - 更多文档、示例

Join Us!

https://github.com/alibaba/mnn

群号：23329087

二维码：

THANKS

GMTC
全球大前端技术大会