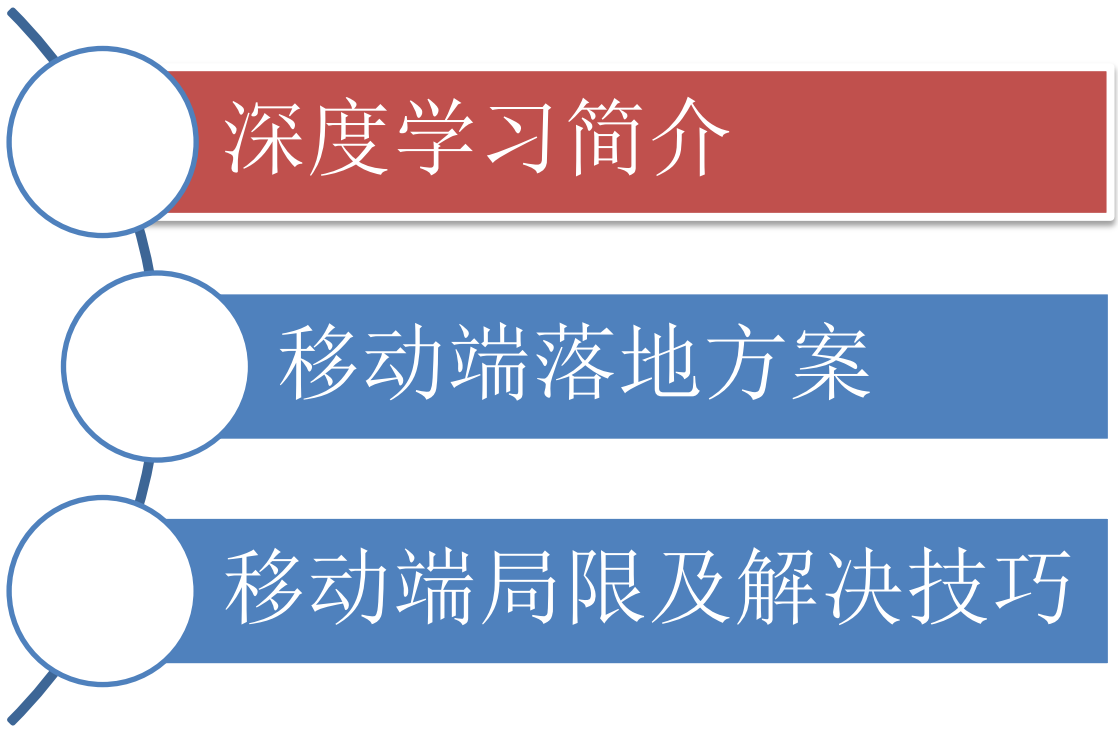
深度学习在移动端的应用

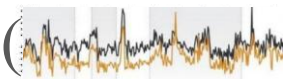
杨延展 百度多模交互搜索部 高级工程师

目录

目录



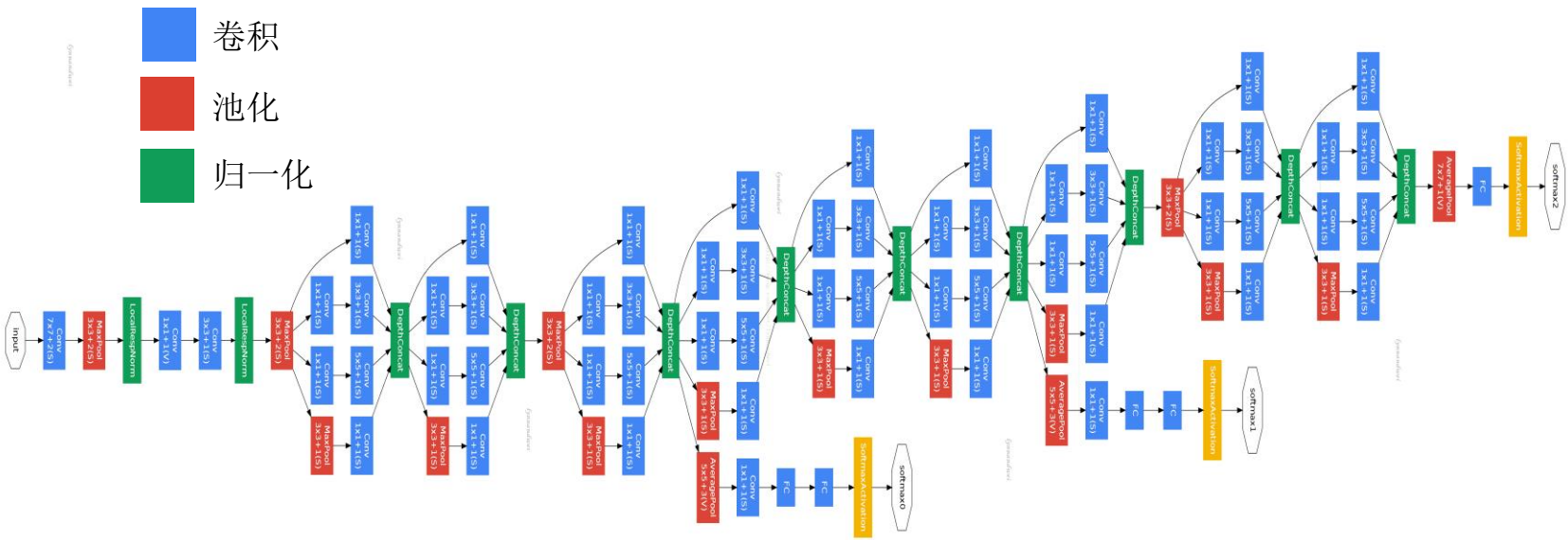
深度学习 - 目标：寻找最佳函数

$f($  $) =$ □你好□

$f($  $) =$ □猫□

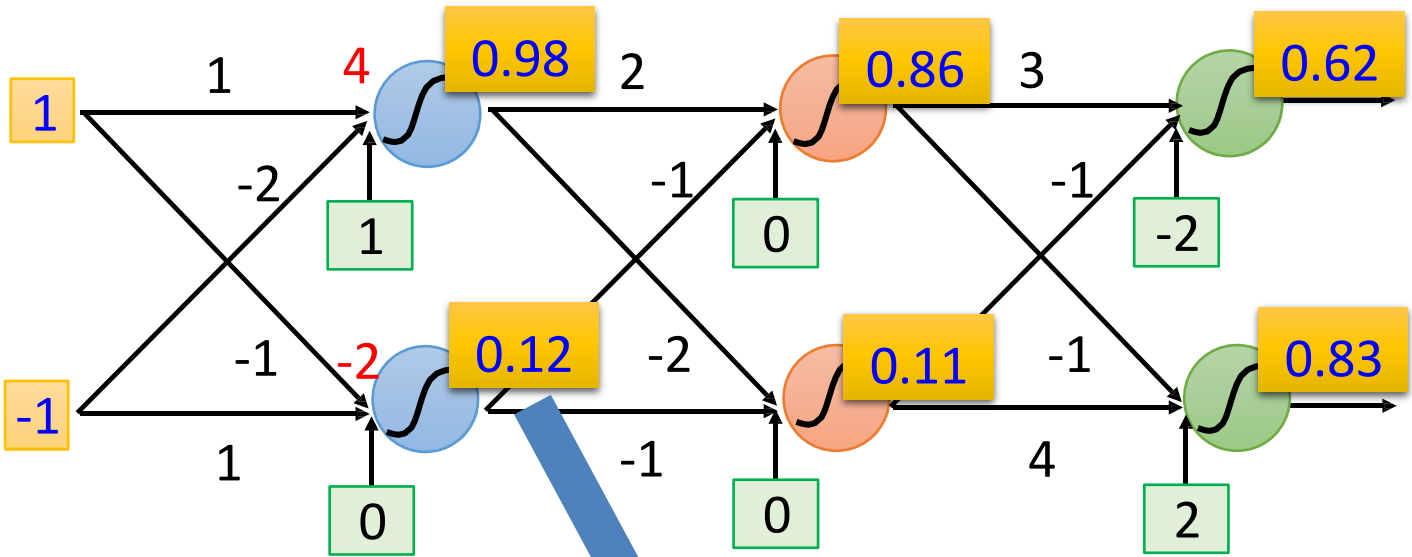
$f($ □嗨！□ $) =$ □你好□

深度学习 - 结构：基于层与层的连接



以GoogLeNet v1举例

深度学习 - 层（举例）：全连接层

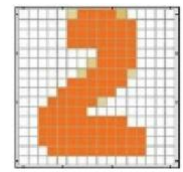


1
-1
0.62
0.83

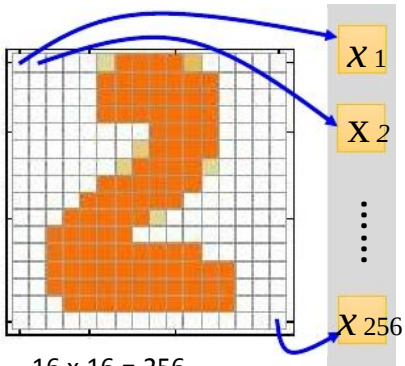
□猫□

$$S(x) = \frac{1}{1 + e^{-x}}$$

深度学习 - 预测过程

$f($  $) = \square 2 \square$

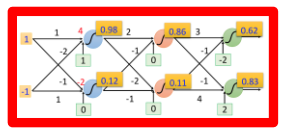
Input



16 x 16 = 256

Ink → 1

No ink → 0

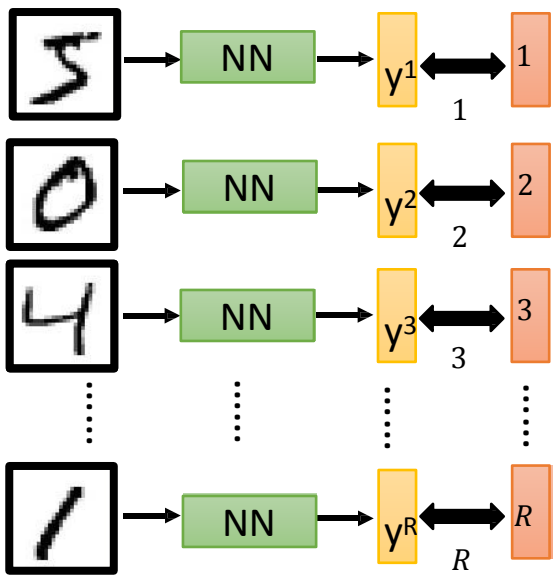


Output

	向量	概率	输出
Y 0	0.62	0.05	is 0
Y 2	0.83	0.8	is 2
⋮			⋮
Y 9	…	0.1	Is 9

The image is “2”

深度学习 - 训练过程



$$f\left(\quad\right) = \square 2 \square$$

交叉熵：计算和目标向量的距离　　图片　　目标向量

$$\frac{1}{N}\sum_{i=1} D(S(WX_i + b), L_i)$$

平均交叉熵　　概率函数

目录

落地方案 - 可能的服务端及移动端分工

- 服务端训练 + 服务端识别
- 移动端训练 + 移动端识别
- 服务端训练 + 移动端识别

落地方案 - 移动端只适合识别过程

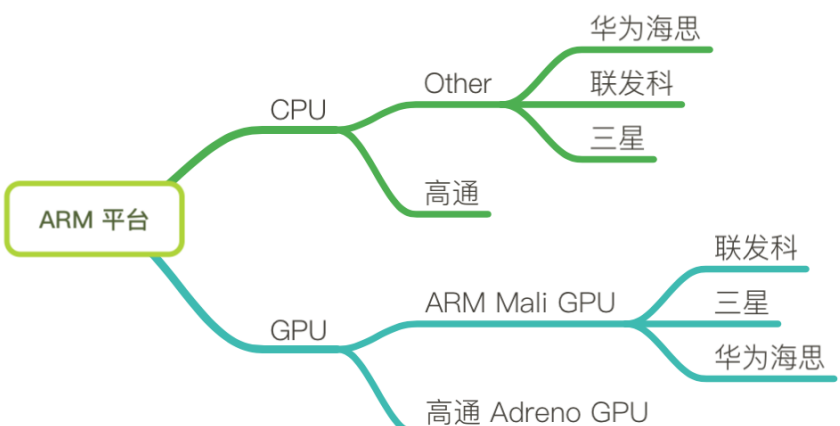
PC Server 训练模型文件

Model File

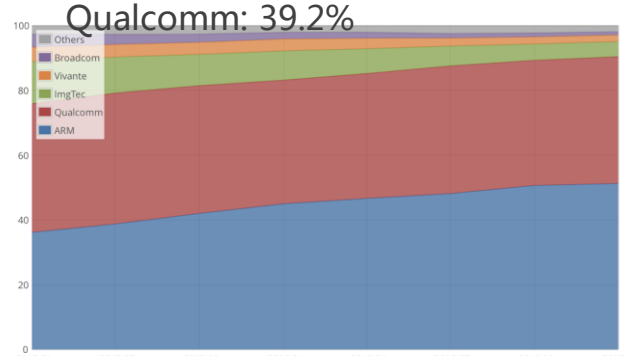
移动端加载模型进行识别

落地方案 - Android硬件现状

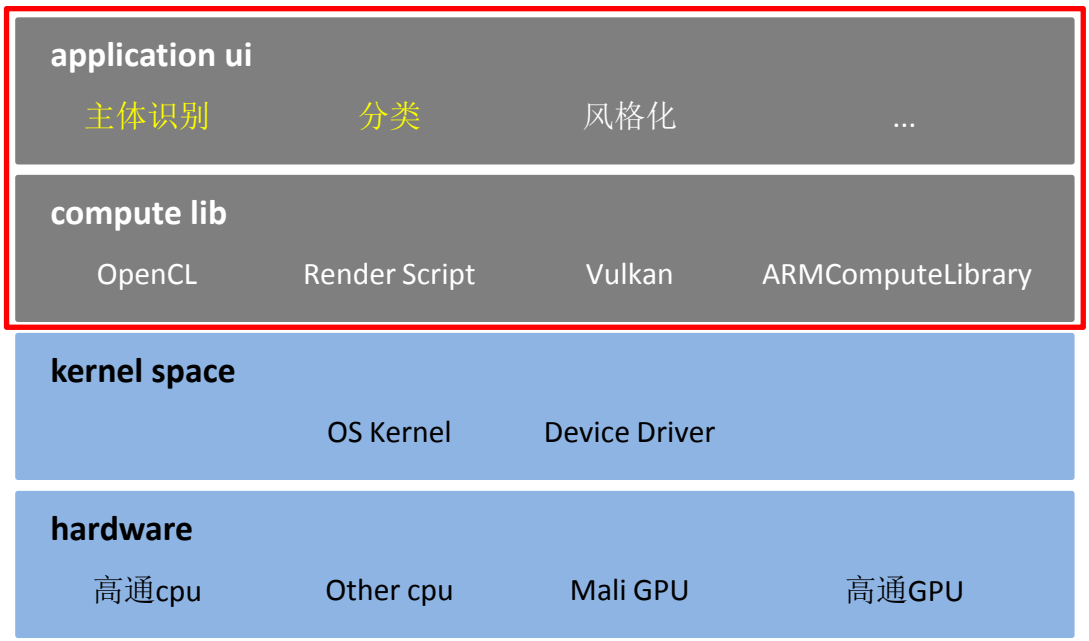
- CPU ：高通 & (三星、联发科、华为)
- GPU ：Mali GPU
- CPU门槛1：骁龙600以上
- CPU门槛2：Mali T820 4核以上



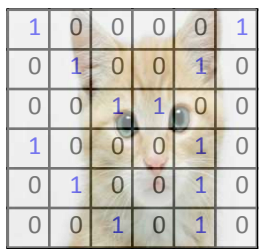
- CPU 98.1% 是ARMv7
- GPU ARM: 51.3%
  Qualcomm: 39.2%



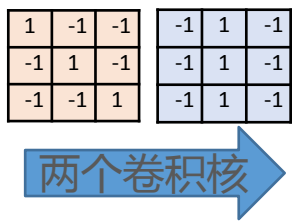
落地方案 - Android深度学习软件现状



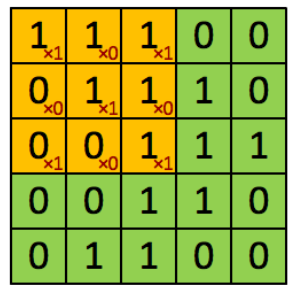
落地方案 - 模型选择 CNN卷积和池化



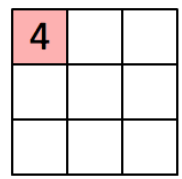
原图



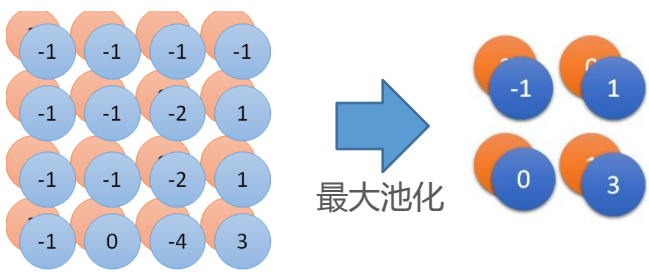
Only 9 x 2 = 18 parameters



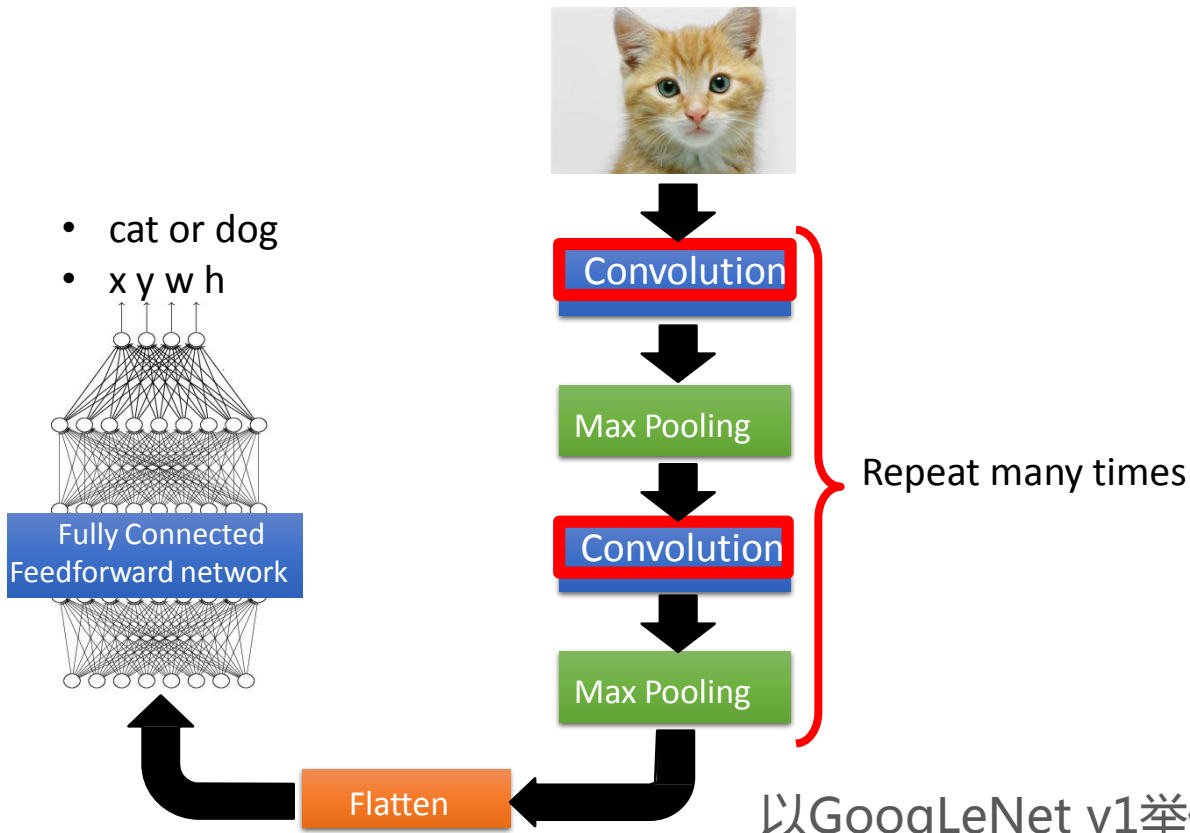
Image



Convolved Feature



落地方案 - 模型选择 CNN常见拓扑



以GoogLeNet v1举例

落地方案 - 框架选择

- 基于Caffe二次开发
  - 可读性
  - 通用性
  - 图像领域应用已久
  - 移植成功案例
- 针对CPU做主要优化
  - GPU的内存拷贝成本与运算效率的综合考量

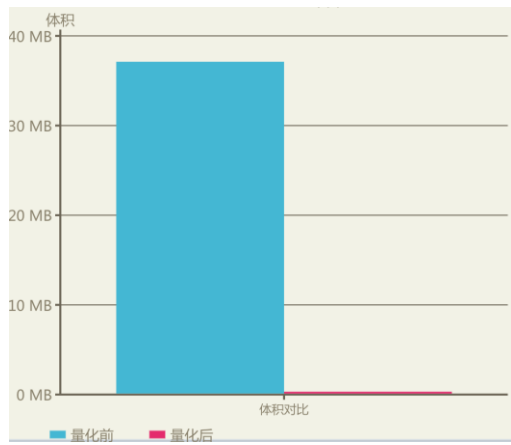
目录

移动端局限 - 安卓落地难点

	服务端	移动端
SO体积	无限制	特定App下严格限制
模型体积	500M+	<10M
加密	无需考量	特定App下严禁泄露
预测速度	类库极其成熟	有待填补
内存限制	无严格限制	内存极其有限
耗电量	无限制	严格限制

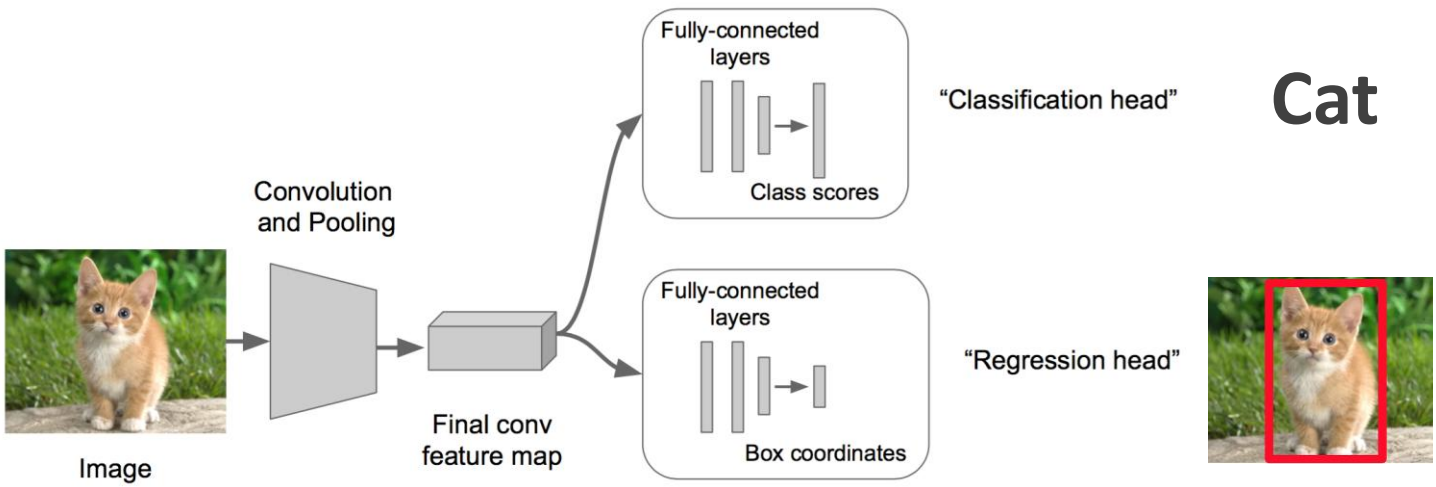
Tricks - SO体积：代码剪枝

Before	After
OpenBlas	手工实现
Glog、Gflag	摘除
Protobuf	手工实现Json解析
后向传播	摘除
层	缩减数量

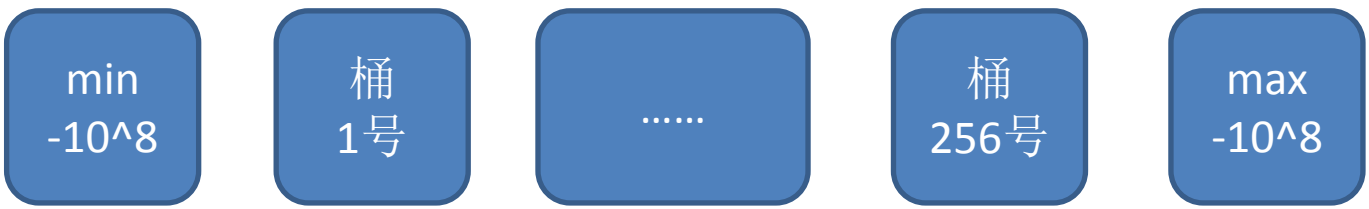


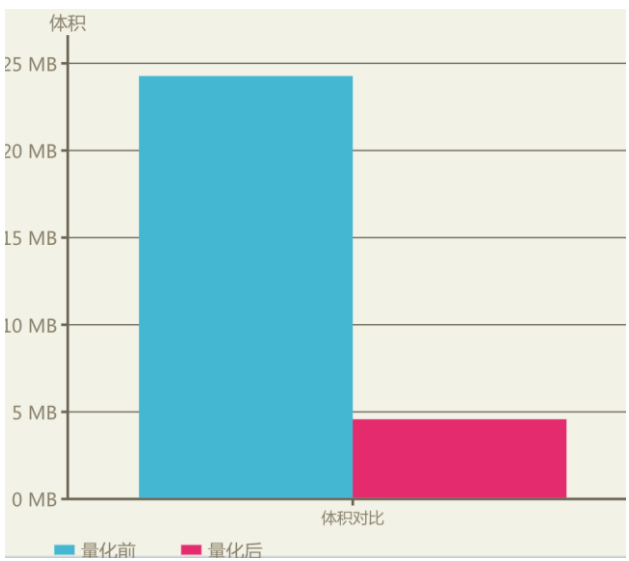
37MB -> 100k

Tricks - 模型体积：权值共享



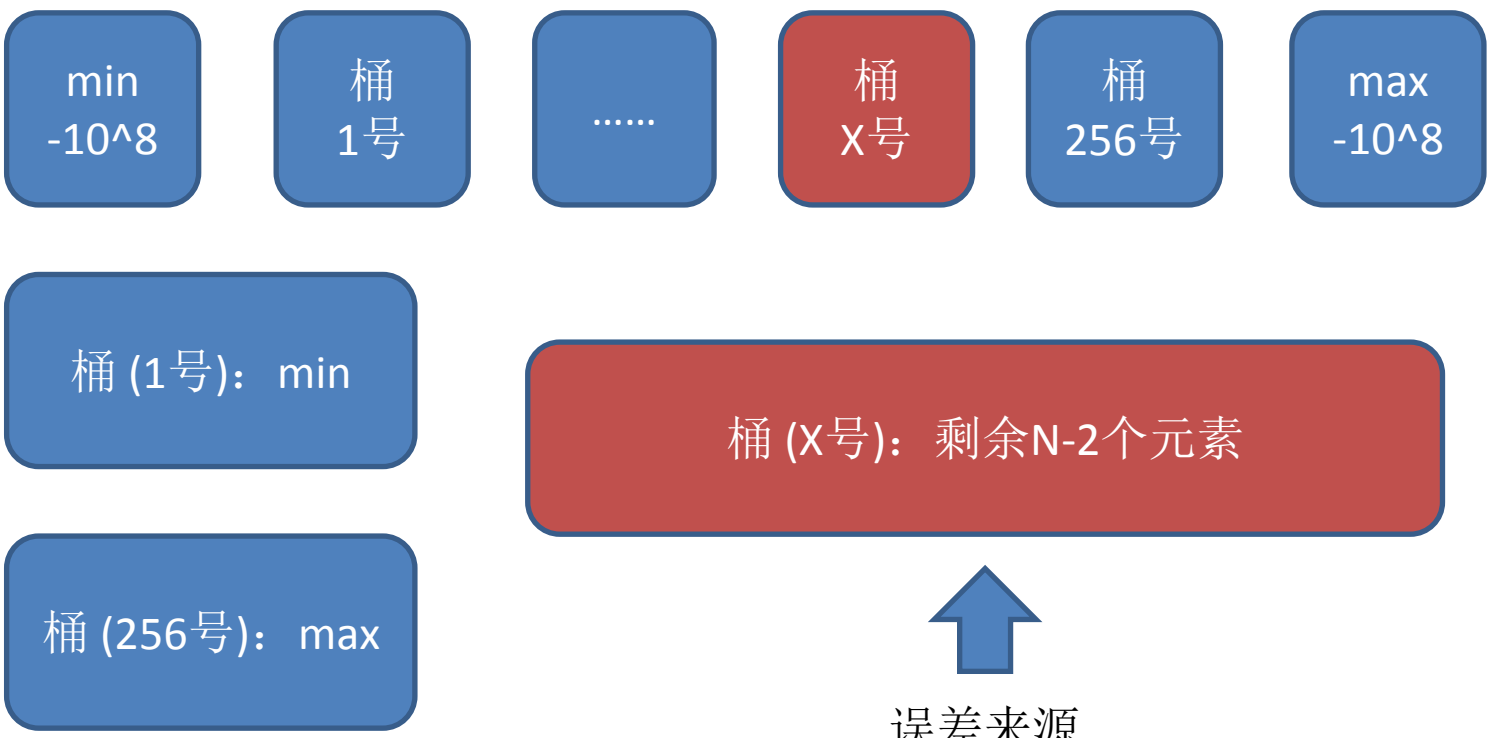
Tricks - 模型体积：量化



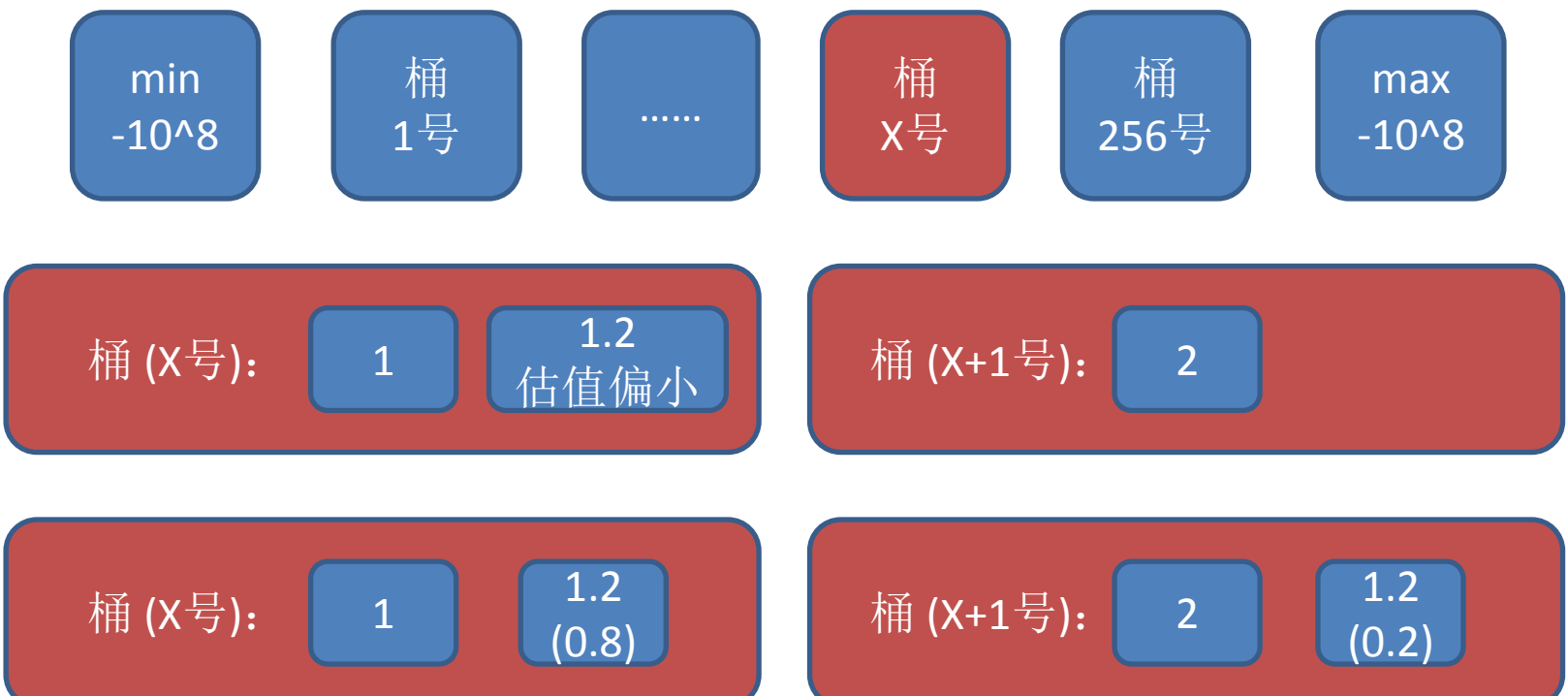


24.2 -> 4.5MB

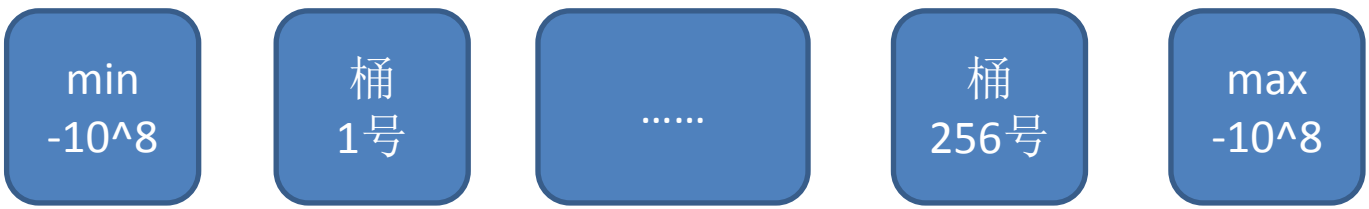
Tricks - 模型体积：纠正量化的误差



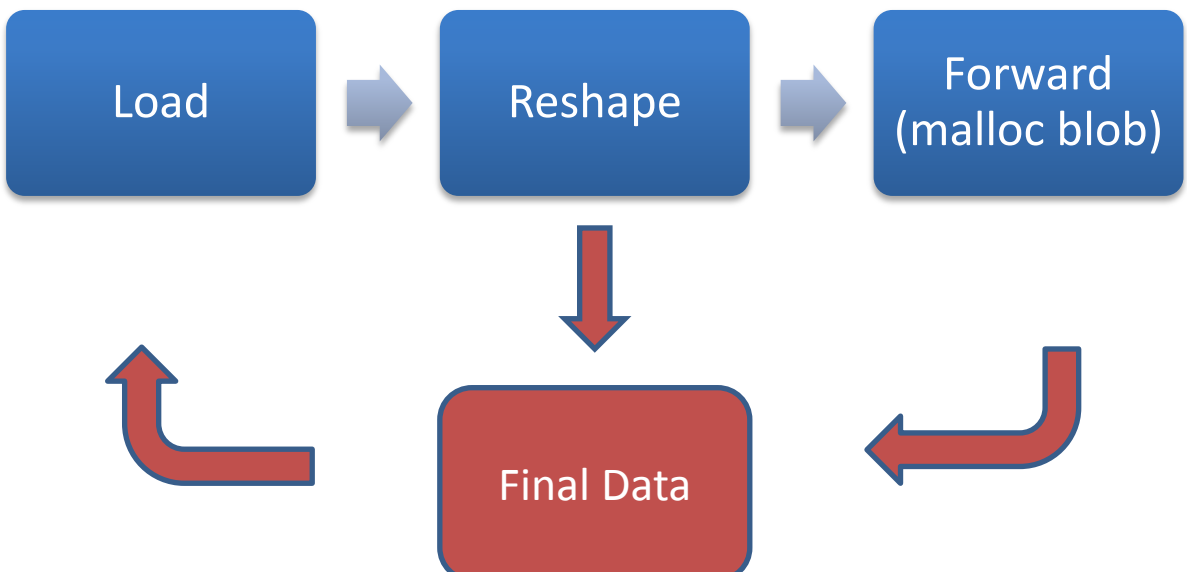
Tricks - 模型体积：纠正量化的偏好



Tricks - 模型体积：加密



Tricks - 速度：常规优化 调整overhead



Tricks - 速度：常规优化 数值计算

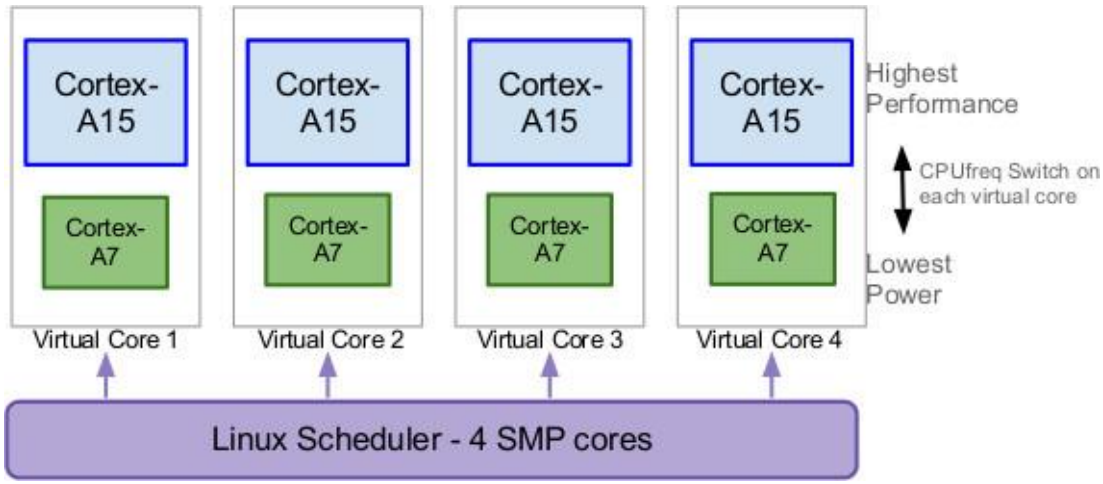
$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots$$

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

记录Taylor展开的系数，用以实现快速的近似计算

Tricks - 速度：CPU Affinity

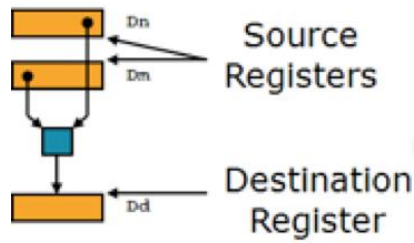


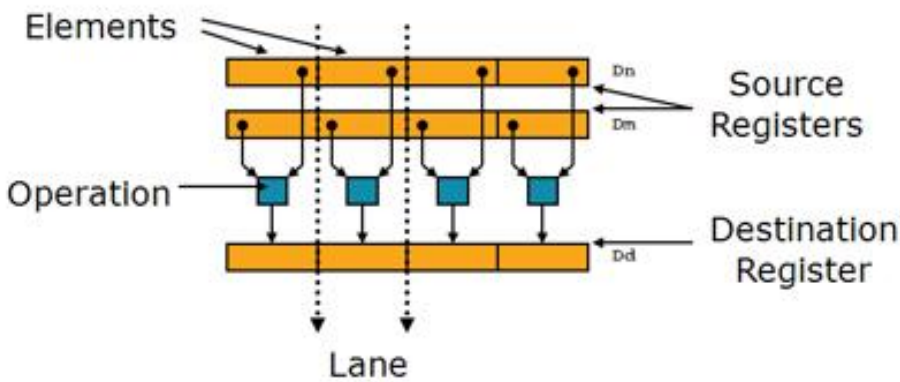
设置亲密度：

1. 减少线程切换。
2. 强制使用Big核心。

Tricks - 速度：NEON intrinsics

- 卷积
- 池化
- LocalRespNorm



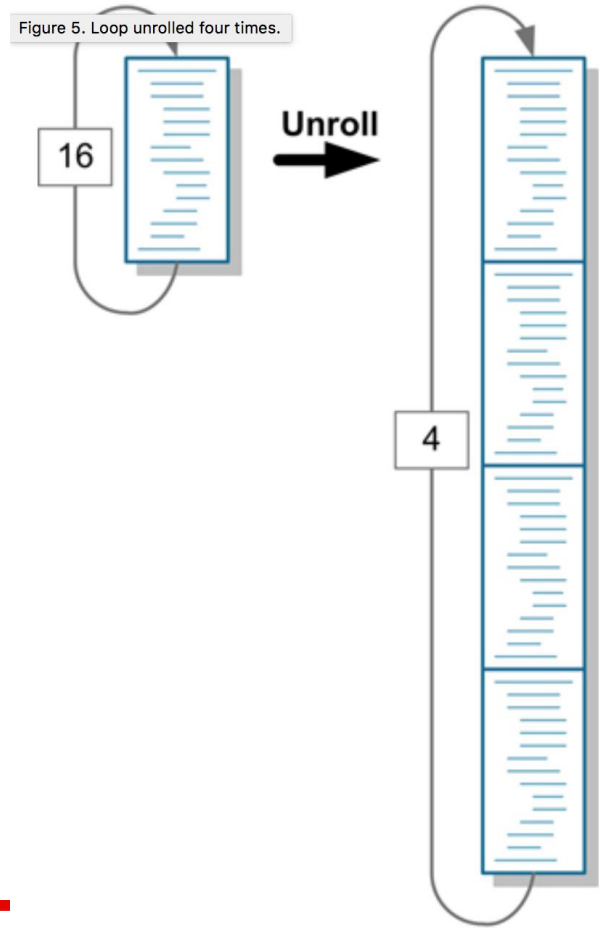


Tricks - 速度：内联汇编

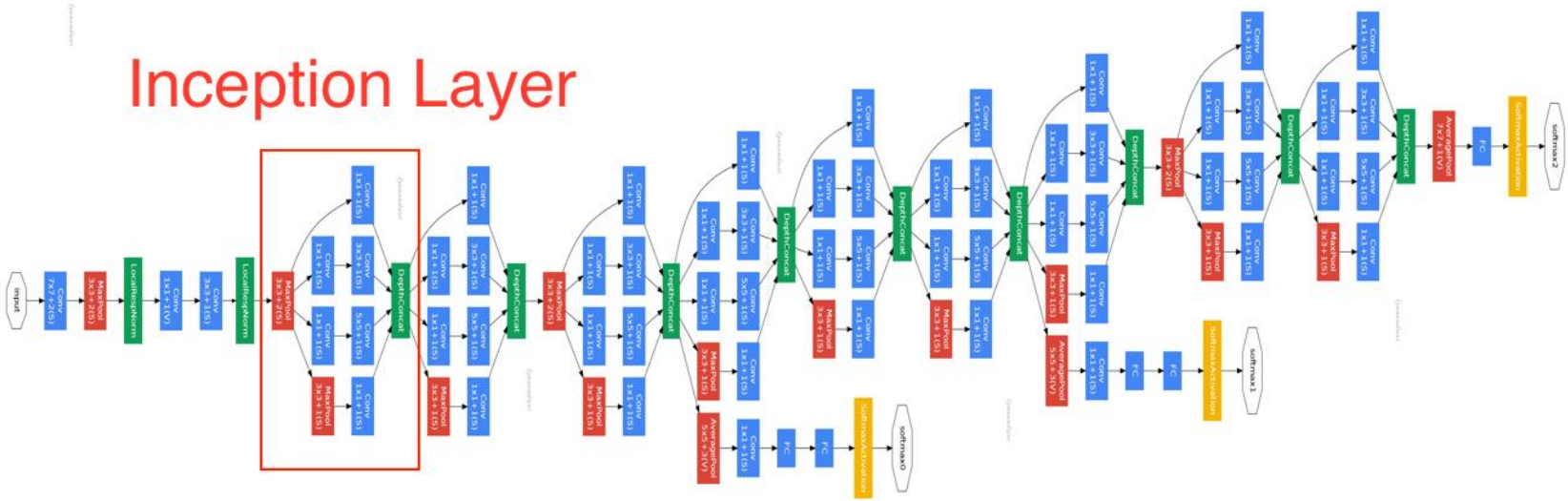
- assembly文件
  纯汇编文件，后缀为”.S”或”.s”。注意对寄存器数据的保存。

- inline assembly内联汇编
  在C/C++代码中嵌入汇编，调用简单，容易调试。

Tricks - 速度：Loop Unrolling



Tricks - 速度：利用拓扑结构执行多线程



利用SplitLayer和ConcatLayer，自动进行多线程分配

Tricks - 速度：改用MobileNet模型

Depthwise Separable Convolution由两部分组成：

- depthwise convolutions
- pointwise convolutions (simple 1×1 convolution)

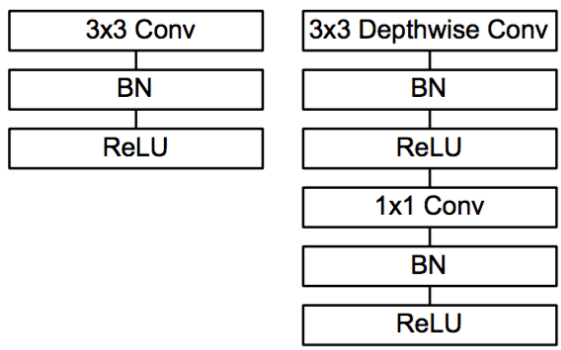
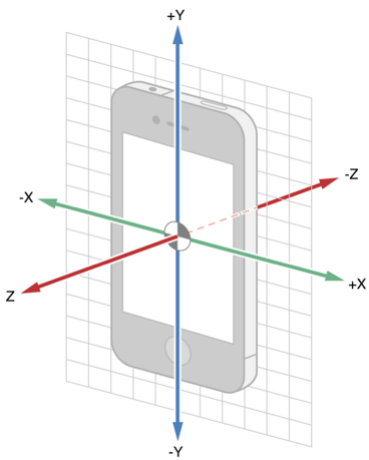


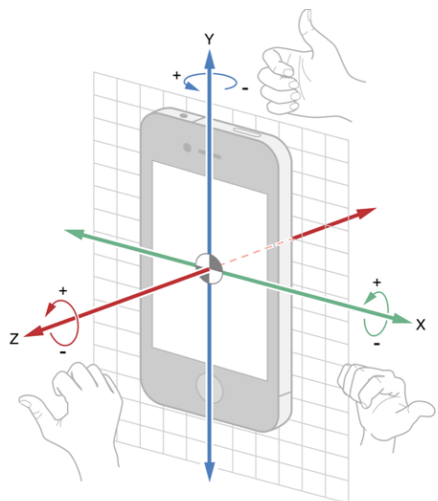
Figure 3. Left: Standard convolutional layer with batchnorm and ReLU. Right: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU.

Tricks - 耗电量：精准确定预测时机

- 在移动客户端运行神经网络耗电量巨大，采用以下策略：
  - 用户手机达到稳定后一段时间开始识别
  - 通过选取合适的识别间隔

陀螺仪&加速计





THANKS!