10、对结果的讨论

10.1、如果引入GPU优化，可行的优化方案有哪些？

将Caffe移植到手机端并使用GPU计算具有可行性，但难度较大：首先，Caffe原版的GPU端并行实现仅支持NVIDIA CUDA通用并行计算技术，也就只能在NVIDIA出品的显卡上运行。但目前广泛使用的高通手机端芯片普遍使用Adreno等非NVIDIA显卡。

这就意味着Caffe的GPU实现无法直接使用，只能令Caffe在CPU模式下运行，同时将开放标准的并行化架构OpenCL实现的OpenBLAS替换进项目依赖库，并指定GPU为OpenCL运行设备，以达到将大部分运算转移到GPU上进行并行操作的效果。

但根据前文所述，Caffe将卷积操作转换为线性代数矩阵相乘的形式，借助BLAS的性能提升运算速度。这种做法显著增多了空间消耗，因为卷积层中，滤波器与特征图局部连接，滤波器权重转写所得的矩阵必然是稀疏的。与矩阵维数相匹配的列向量在原地展开过程中，也引入了大量填零补位。

Caffe因此一直因内存空间消耗巨大为用户诟病。这一劣势在GPU上体现得尤其突出——CPU端执行时，Caffe所需的数据存储在内存中，CPU可以直接访问到，并利用CPU的缓存机制提升性能。但GPU无法直接访问内存，所有滤波器权重和特征图等数据必须经CPU调度，从总线复制到GPU的显存中，才能被GPU访问并计算。桌面端GPU计算能力较强，PCI-E总线带宽虽然是瓶颈，但由于桌面端GPU显存同样较大，可以一次容纳大量数据，数据无需频繁在内存和显存间流动，计算速度优势不至被数据复制的瓶颈抵消；

手机端GPU计算能力很弱，而且GPU显存较小，计算卷积过程中需要频繁地从内存中更新显存中的数据，数据复制所需时间远远盖过GPU端实际进行计算的时间，计算资源无法充分得到利用。

如果希望实现手机GPU端的并行优化，Caffe基于线性代数操作的底层计算模型是不适合的。可行的方案是直接替换掉其底层计算方式，换为直接并行进行卷积，引入Winograd变换以进一步压缩其所需空间【来源请求】，以充分发挥其性能。

10.2、本文优化方式的可能提升空间

本文采用OpenMP进行优化。将特征图展开为列向量的im2col操作因为循环层次太深，且存在复杂的逻辑判断，不便于分析独立性以通过OpenMP并行化，没有进行优化。由前文所述，该操作是Caffe的热点函数之一，其实现仍存在性能提升空间。

如果能重构其代码，通过其他操作代替复杂的逻辑判断，将其转化为简单的多重嵌套循环，那么这段代码各层循环执行的独立性将变高，分析这段代码存在的并行性也将变得容易许多，可以通过OpenMP多线程并行优化该段代码，使其外层循环被并行执行。

另，底层计算库OpenBLAS在本文发表时已经发布了OpenBLAS optimized for deep learning分支版本，引入了针对深度学习的专用优化。如能将本文实现中的OpenBLAS依赖库替换为该版本，对性能应当也有一定的提升。

Armeabi-v7a以及更新架构的指令集引入了浮点指令集与NEON单指令多数据扩展，不再使用性能低下的软模拟方式实现浮点运算。如果能从汇编层面上对OpenBLAS的核心作进一步优化，将非向量化的汇编指令替换为等效的向量化汇编指令，充分发挥单指令多数据汇编的优势，那么Caffe将得到超出OpenMP多线程并行化层次之外的性能提升。

11、结论

本文对业界学界广泛使用的深度学习框架Caffe实现了到ARM架构安卓手机的移植，满足了深度学习开发者在手机上灵活实现神经网络模型的需求，并且使用了OpenMP多线程并行技术对其进行了优化。基于高通820开发板的测试表明，Caffe经本文所述方式优化后，AlexNet总体执行时间减少18%；GoogleNet总体执行时间减少12%。本文提出的优化方式成功使Caffe在AlexNet、GoogleNet等主流卷积神经网络架构上得到了明显的性能提升。