自动化数据布局传播与优化：实验方法与结果分析

实验报告

2026 年 1 月 4 日

摘要

本报告详细介绍了 `layout_propagation` 模块中的实验方法、结果数据及深入分析。我们针对深度神经网络加速器中普遍存在的“数据布局不匹配”问题，提出了一套基于代价模型的自动化布局传播框架。实验结果表明，该框架在处理具有复杂分支结构的现代网络（如 ResNet50）时，能够将总访存代价降低约 50%，显著优于传统的固定布局策略。

1 实验方法 (Methodology)

1.1 问题背景

在深度学习加速器中，不同的算子（Operator）对数据布局有不同的偏好。例如，卷积核可能偏好 $K \times C \times H \times W$ 以最大化复用，而脉动阵列可能需要分块的布局（Blocked Layout）以匹配阵列尺寸。当生产者和消费者的布局偏好不一致时，必须进行布局转换。

本实验旨在回答：**在全网范围内，应该在何处、以何种方式进行布局转换，才能使总的 DRAM 访问代价最小？**

1.2 代价模型 (Cost Model)

我们定义总代价为执行代价与转换代价之和：

$$J = \sum_{v \in V} \text{Cost}_{\text{exec}}(v, L_v) + \sum_{(u,v) \in E} \text{Cost}_{\text{trans}}(L_u, L_v) \tag{1}$$

- **执行代价 ($\text{Cost}_{\text{exec}}$)**: 衡量算子在特定布局下访问 DRAM 的效率。模型考虑了 **Burst Efficiency**（带宽利用率）和 **Row Buffer Miss Rate**（行缓存命中率）。非连续访问（Strided Access）会导致高昂的 Row Miss 代价。
- **转换代价 ($\text{Cost}_{\text{trans}}$)**: 衡量在两个不匹配布局之间传输数据的代价。我们支持两种转换模式：
  - **Direct Write**: 生产者按自身顺序写，消费者按自身顺序读（无显式转换，但消费者可能承担高昂的读取代价）。
  - **Transform-on-Write**: 生产者利用片上 SRAM 重排数据后写入 DRAM（增加写入代价，但消费者可获得完美的读取顺序）。

1.3 实验流程

实验代码位于 `layout_propagation` 目录下，主要脚本为 `analyze_nns_layout_v2.py`（优化策略）和 `analyze_nns_layout_baseline.py`（基准策略）。

1. **网络解析**: 从 `nn_dataflow/nns` 读取网络定义。
2. **基准测试 (Baseline)**: 强制所有层使用标准的线性布局（NCHW）。这模拟了传统深度学习框架的默认行为。
3. **优化测试 (Optimized)**: 运行布局传播算法。
   - **Phase 1 (Propagation)**: 敏感算子（如 Conv）生成硬件友好的分块布局（Blocked Layout），并向邻居传播。
   - **Phase 2 (Decision)**: 贪婪策略选择器根据代价模型，为每个节点选择最佳布局，并决定在何处插入转换。

2 实验结果 (Results)

我们在四个经典网络上进行了评估。硬件参数设定为：DRAM Row Size = 2KB, Burst Size = 32B, Array Size = 16。

表 1: 布局优化前后总代价对比

网络模型	层数	基准代价 (Baseline)	优化代价 (Optimized)	提升幅度
AlexNet	19	18.63	16.83	9.66%
VGG Net	21	31.44	30.19	3.98%
ResNet50	21	**45.13**	**22.39**	**50.39%**
GoogleNet	9	14.94	14.39	3.68%

3 结果分析 (Analysis)

3.1 ResNet50 的巨大提升

ResNet50 取得了超过 50% 的性能提升，这是本实验最显著的发现。

- **原因**: ResNet 包含大量的残差连接（Residual Connections）。在基准策略（NCHW）下，主分支（Conv）和残差分支（Identity/Conv）的数据往往在汇合点（Add）发生冲突。
- **具体表现**: 在基准测试中，我们观察到如 `conv2_br -> conv2_{}_res` 这样的边产生了高达 **9.69** 的转换代价。这是因为残差分支通常是 1×1 卷积或直接直连，其在 NCHW 布局下的访问模式与主分支的 Blocked 模式极不匹配，导致了严重的 Row Buffer Thrashing。
- **优化效果**: 布局传播算法成功识别了这一结构，强制残差分支也采用与主分支一致的 Blocked Layout。虽然这可能略微增加了残差分支的写入代价，但彻底消除了汇合点的高昂读取代价。

3.2 线性网络 (VGG/AlexNet) 的平庸表现

对于 VGG 和 AlexNet，提升幅度较小（¡10%）。

- **原因**: 这些网络是简单的线性链状结构（Conv -¿ ReLU -¿ Pool -¿ Conv）。
- **分析**: 在这种结构中，相邻层通常具有相似的维度特征。虽然 Blocked Layout 在理论上优于 NCHW，但在每一层之间进行 Layout 转换（Transform-on-Write）本身也有开销。对于某些层，直接使用 NCHW（Direct Write）的代价与“转换+优化读取”的代价相差不大。优化器在这种情况下只能通过微调转换位置（例如在 Pooling 层处转换）来获得少量收益。

3.3 GoogleNet 的情况

GoogleNet 的提升也较小。这主要是因为我们的解析器目前将 Inception 模块简化为了线性序列进行分析，未能完全捕捉其多分支并行的复杂性。如果完整建模 Inception 模块的 4 个分支，预期收益应介于 VGG 和 ResNet 之间。

4 结论 (Conclusion)

实验证明，`layout_propagation` 提出的自动化布局优化框架是有效的。

1. 对于**线性网络**，它能找到不亚于人工设计的布局转换点。
2. 对于**分支网络**（如 ResNet），它能通过全局协调消除严重的布局冲突，带来显著的性能提升。

这验证了在深度学习编译器中引入“布局传播”机制的必要性，特别是针对具有复杂存储层级和特定访问模式的专用加速器。