?? ??? Binary ??????

??

1. FPGA的基本单元是 LUT(查找表)，如果把 LUT看成一种逻辑运算单元
    1. 查找表的真值表可以表示静态权重
    2. 查找表的部份输入表示动态权重
2. FPGA的可重构特性，相对于 AI处理器
    1. 可以把部份或者全部的动态权重（操作数）固化成静态的真值表，大大提高 PPA
    2. 不同的模型和参数，可以通过重新配置 FPGA的逻辑单元
    3. 只适合推理，理论上训练的权重不能静态化， PPA优势不大
3. 把 FPGA的 LUT作为 AI芯片的核心算力单元
    1. 有利于硬件的标准化
4. 扩展到二进制神经网络，所有的数据都用二进制表示
    1. 最大的特点是， 所有数据表达范围都是完整的，没有任何稀疏，效率最高
    2. 不仅限于查找表的计算方法，虽然 LUT是很重要的一种二值计算方法
5. binary LUT 通过 bit量化降低数据量
    1. 使用全集的可能权重， 8bit时候需要 256bit表示所有可能的输入
    2. 避免复杂的数字电路计算，以存储替代计算，降低算力和能耗的需求
    3. 结合 CIM扩大存储容量，避免权重搬运
6. 直接整形训练、 Bitnet训练
    1. 传统的大模型训练使用的是浮点数非常注重精度，两个概念共用一个权重，那么这个权重的分辨间隔是非常关键的，必须有足够的冗余来区分这两个概念，量化后会导致冗余度减小，但是也是在可控的范围内。如果本身模型就是 FP8训练的话，那做 int4量化也是合理的。
    2. 所以如果使用整形进行训练的话，训练和推理的晶体管计算效率应该是最高的

??

1. LUT6
    1. 6输入 1输出的标准 LUT，有 64种可能，配置权重有 64bit
    2. xc7k480t包含 74659 slices，每个 slice包含 4 LUT， 8个 flip-flop
    3. 如果所有 **lut**真值表 都用来存储权重， 64bit*74659*4 = 2389088 byte，等效算力 74659*4 / 6 * 2 * 0.5G （ GFlops） = 50 TFlops
    4. 可以组合多个 LUT形成更高位的 LUT单元
2. 向量的映射
    1. 可以表示任意的运算，包括加乘法
    2. 4个 6LUT组成一个 8bit lut， 8个 LUT8 并联输出 8bit，形成一个 8输入 8输出的映射表
    3. 比线性映射拥有更高的表达空间，但是不可微分，不可梯度下降
3. 有没有更原生的适合 LUT的算法？而不是抽象成数值计算
    1. 直接对 LUT组成的网络做梯度下降

1. “多选一 +weight”可以表示一个可微的 LUT
2. 使用任何模型来表示 LUT都是对 LUT
   的表示能力的浪费，应该使用传统方式进行训练，并且约束模型最后的计算范围尽量覆盖
   LUT的所有可能（稠密），才能不浪费 LUT的算力。
   1. 这种情况和稀疏的模型在 GPU上浪费算力一样
   2. 各种精简的乘法电路才能不浪费晶体管，精度越低的乘法器晶体管效率越高
   3. 就算一个传统的 AI模型很稠密，其乘法器等也有大量表达空间浪费，所以低精度（
      FP8）也能达到效果
3. 非常像 存内计算 ，存储容量小，算力非常高
4. 多进多出的 LUT，可以表示任意的 f(x)=y，所有的参数都可以用 LUT表达
   1. EIementWise类、 CNN、GEMM：内置的 weight都可以用 LUT表达
      1. 多个组合的运算也可以用一个 LUT表达
   2. Reduction、normal、softmax：需要进行 Reduce操作
   3. 动态 GEMM（映射向量到指定的空间）：需要外部输入的操作数
5. 动态 LUT
   1. LUT的查找表是根据另外一个操作数的输入，不同于 FPGA是通过配置电路进行操作

?????

????

1. 不应该用传统的 "乘加 ”等计算方法来计算
2. 传统的网络模型的 softmax、normal类计算，是为了弥补 GEMM、CNN等固有的缺陷而加入的
3. 应该是一堆复杂的逻辑运算，根据逻辑的复杂度，可以用 LUT表示，或者直接用逻辑门表示
4. 可微，就是对数据空间分布的一种约束，传统的数学的计算表达方式效率肯定不高，特别是高精度计
   算

??????

1. 计算的信息都是带一定的分布空间约束的，只是逻辑电路（ LUT）可以表示的空间的的一部分
   1. 需要在电路设计上符合数据的约束，减少浪费不必要的晶体管浪费
   2. 传统的硬件按照 ”加减乘除 “
      作为接口，是数学的定义，虽然能天然的约束到数学算法空间，但是有大量的浪费
   3. 为了符合数学上的计算，计算机的数据类型设计可以表达很大的空间，特别是浮点数，但实际
      上很少被全部用到，所以模型的整形量化会有效果
2. 纯计算，无权重
   1. 乘法：表示两个信息相互关联，促进的效果
   2. 加法：表示两个信息积累的效果
      1. 有一个内部的状态统计累加的结果，按照统计规律输出结果
   3. 计算两个信息之间的关联性，相似度
      1. 两个多 bit信息之间，两两 XOR之后的输出，求算不同层级的 AND输出
   4. 可以直接通过简单的逻辑运算（异或、与、或）将两个操作数合并成一个，然后再利用 LUT
      做映射
3. 有权重
   1. 一个信息映射到另外一个信息的维度， linear

    1. 查找表，输入点数　　＝原始信息点数，表个数　　＝输出点数
    2. 使用　“多选一　　+weight”表示一个可微的　　LUT
2. 阈值判断，断言输出　　=> 有权重

AI???????

1. 计算两个数据的相似性
2. 计算两个数据的相互关联，促进的效果
    1. 要确定操作数之间要是对等的，可以直接用来计算的，而不是不同的分布空间
3. 激活、阈值
4. 映射一个数据到另外一个空间（约束信息到另外一个分布）（编码）
5. 累加
6. 累乘
7. 归一化，　　shift和 scale到相同的空间分布

LLM??

1. 所有的基本单元都是　　”多进单出　　“的单　bit单元
    1. 利用统一的基本单元构建复杂的模型网络
2. 原始的状态都是固定的一个复杂的可微的结构
    1. 可以表达所有的空间：所有的输入都能影响到输出
    2. 训练完成后可以被映射到几种固定的简化的　　”处理单元　　“
    3. 通过在每个单元内部的特殊计算结构，结合单元外部的梯度下降，实现基于　　统计 的训练方法
3. 基本单元采用　　8bit的全连接结构
    1. 通过复杂的组合来实现更多　　bit的关联性
    2. 8bit 需要　256个　weight
    3. weight的调整采用统计方式，和历史的激励有关联，和其他　　bit的状态也有关联
        1. 仿真生物的行为：过敏、脱敏、连续放大、贪心
    4. 优化的目标
        1. 尽量能把　　LUT训练成固定的，简单的逻辑代码，减少生物和计算机的能量消耗
        2. 把 LUT优化成更少的　　bit的 LUT
        3. 自洽
4. 一个具有非常多输入的单输出节点
    1. 优化　/训练成（或量化成）简单处理逻辑，忽略大部份不激活的节点
    2. 优化成（或量化成）简单的输出之间的处理算法：　　“累加　”，“逻辑与　”等等
    3. 要求
        1. 全连接，输入的节点完全对称
        2. 基于统计、贪心、类脑的训练
        3. 能用梯度下降进行优化?
5. 模型
    1. x -> x
        1. 类似　attention的方式，生成　　qkv进行　bit映射
            1. QK分为动态参数（额外的一个数据参与生成　　qk）和静态参数（只和　　input有关）
    2. 通过有限的　　LUT组合进行　　reduce
6. 构建一种稀疏的**LUT**模型
    1. 支持类脑训练

??

1. 替代传统的网络 LLM/CNN
    1. 更好、更容易的量化
    2. 更好的 PPA
    3. 适合 FPGA和专用电路
2. 通过类脑模型实现智能

???

1. 生物脑的典型的神经元结构能实现上述所有的计算需求
2. AI模型理论实际上是在理论数学的方法模拟大脑的工作，作为一个模拟器有大量的浪费和低效
3. 神经元的功能相对固定
4. 神经元之间的初始拓扑关系是 DNA决定的
5. 神经元之间的细节连接 可以通过学习（训练）而改变
6. 生物神经元的模型和 LUT非常的像
    1. 神经元对时序也敏感，统计
    2. 都是多输入单输出
    3. 多个输入会进行精细的统计、不同的阈值、不同的 scale、不同的时序后决定是否激活输出

??????

1. BitNet b1.58 https://mp.weixin.qq.com/s/G9ZbMnBVbeH1m45HY2JIKA
2. 量化本质上还是遵循数学的计算定义：加减乘除组合形成的数据变换

??

1. 效率
2. 精度问题

??

1. https://arxiv.org/html/2502.19008v1
2. 二值神经网络（ Binary Neural Networks）最新综述 https://zhuanlan.zhihu.com/p/117285043
3. https://github.com/ryuz/BinaryBrain
4. バイナリニューラルネットとハードウェアの関係 https://www.slideshare.net/kentotajiri/ss-77136469
5. BinaryConnect: Training Deep Neural Networks with binary weights during propagations https://arxiv.org/pdf/1511.00363.pdf
6. Binarized Neural Networks https://arxiv.org/abs/1602.02505
7. Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1 https://arxiv.org/abs/1602.02830
8. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks https://arxiv.org/abs/1603.05279

9. Xilinx UltraScale Architecture Configurable Logic Block User Guide
   https://japan.xilinx.com/support/documentation/user_guides/ug574-ultrascale-clb.pdf
10. 1-bit AI Infra: Part 1.1, Fast and Lossless BitNet b1.58 Inference on CPUs
    https://arxiv.org/abs/2410.16144v2
11. Bitnet.cpp: Efficient Edge Inference for Ternary LLMshttps://arxiv.org/abs/2502.11880v1
12. Continual Quantization-Aware Pre-Training: When to transition from 16-bit to 1.58-bit pre-training for BitNet language models?https://arxiv.org/abs/2502.11895v1
13. (NEW!) BitNet v2: Native 4-bit Activations with Hadamard Transformation for 1-bit LLMs
    https://arxiv.org/abs/2504.18415
14. (NEW!) BitVLA: 1-bit Vision-Language-Action Models for Robotics Manipulation
    https://arxiv.org/abs/2506.07530

Revision #71
Created 2025-04-12 08:03:12 UTC by Colin
Updated 2026-04-29 07:33:44 UTC by Colin