SparseNN:An Energy-Efficient Neural Network Accelerator

Exploiting Input and Output Sparsity

Jingyang Zhu, Jingbo Jiang, Xizi Chen, and Chi-Ying Tsui
Department of Electronic and Computer Engineering , HKUST

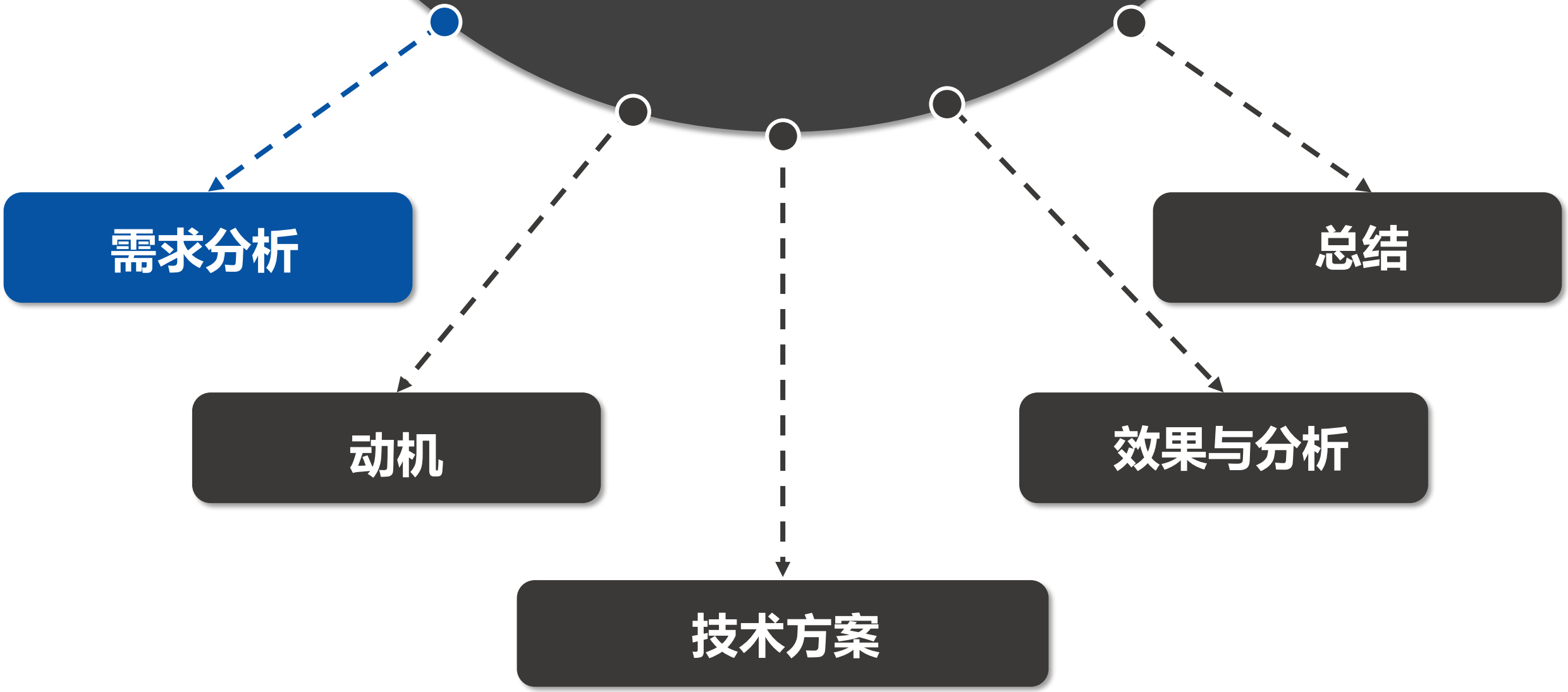
——国防科大2020年高性能评测与优化课程小组讨论

Team 7

何康、朱东、彭雄

指导：龚春叶、甘新标、杨博

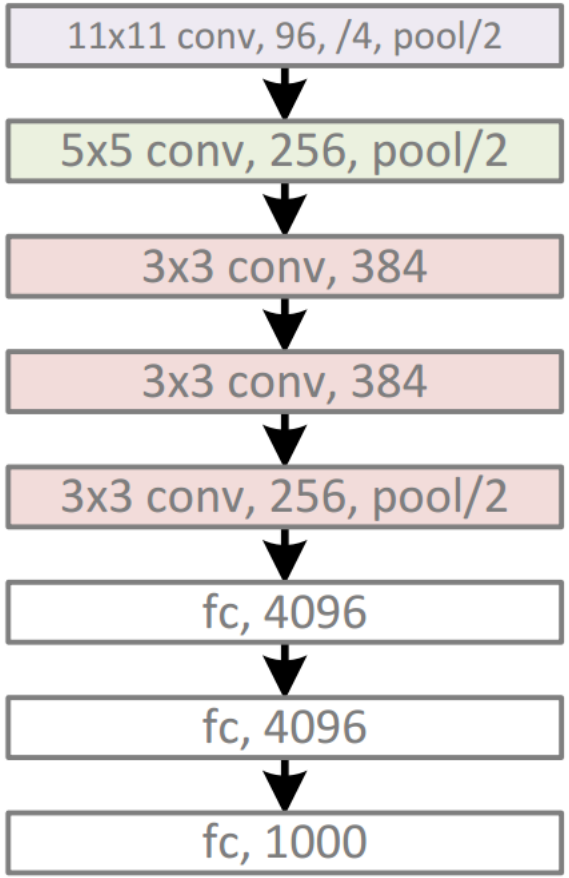
Part One



DNN(Deep Neural Network)庞大的计算和存储需求

AlexNet, 8 layers
(ILSVRC 2012)

Input size:227x227
#FLOPs:0.727G
#Parameter:233M

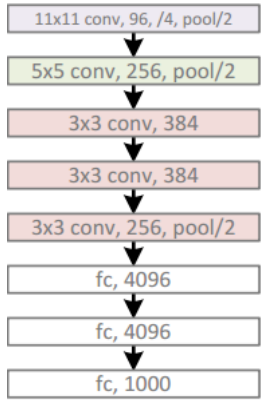


ILSVRC (ImageNet Large Scale Visual Recognition Challenge)

DNN庞大的计算和存储需求

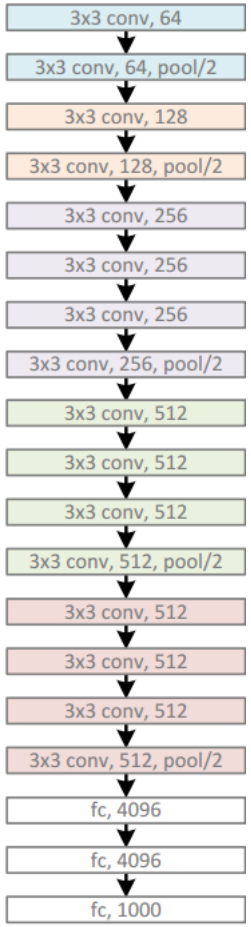
AlexNet, 8 layers
(ILSVRC 2012)

Input size:227x227
#FLOPs:0.727G
#Parameter:233M



VGG, 19 layers
(ILSVRC 2014)

Input size:224x224
#FLOPs:19.6G
#Parameter:548M



ResNet, 152 layers
(ILSVRC 2015)

Input size:224x224
#FLOPs:11G
#Parameter:230M

考虑到嵌入式平台资源的有限性，需要对算法和体系结构进行优化，才能为DNNs提供节能的解决方案。

Part Two

需求分析

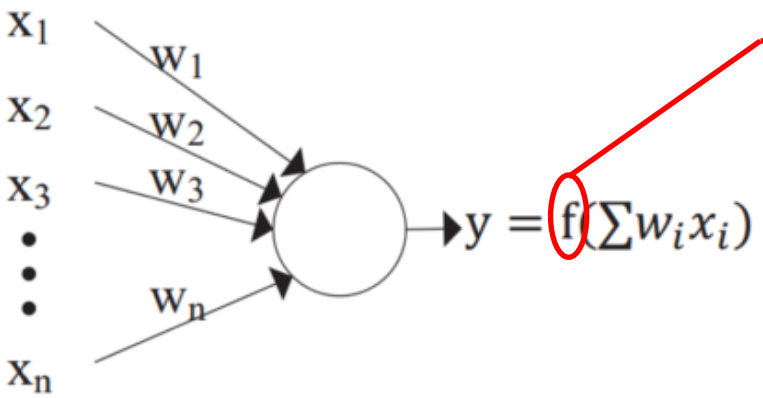
动机

技术方案

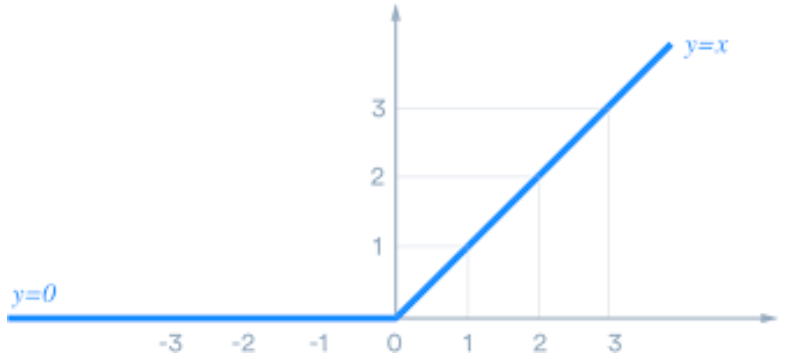
效果与分析

总结

DNN内部的稀疏性



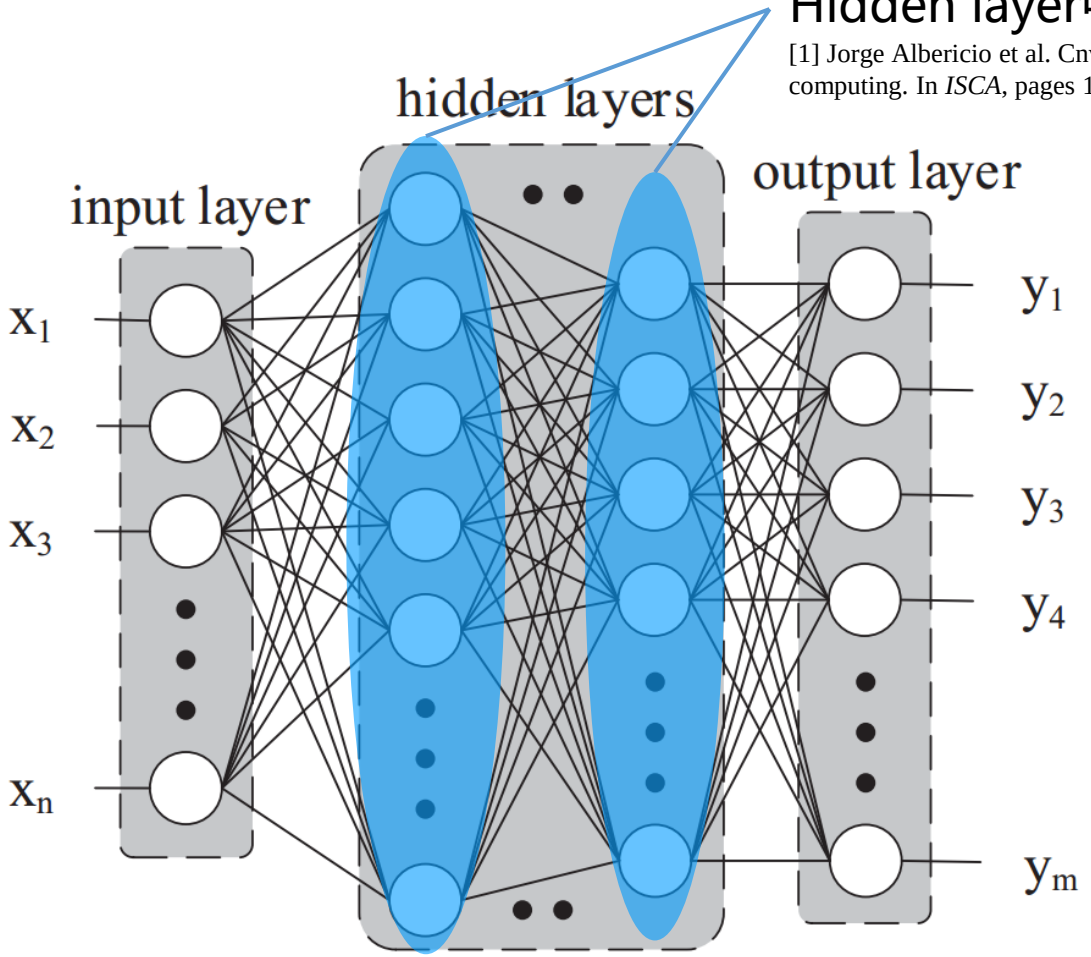
ReLU(Rectified Linear Unit)



DNN内部的稀疏性

Hidden layer中约有50%的稀疏度[1]。

[1] Jorge Albericio et al. Cnvlutin: ineffectual-neuron-free deep neural network computing. In *ISCA*, pages 1–13. IEEE, 2016



- 输入特征图内的零激活值
  - 计算开始时就是知道的
- 输出特征图内的零激活值
  - 在当前层的计算完成之前是未知的

对输出特征图中的零激活值进行预测！！！

Part Three

需求分析

动机

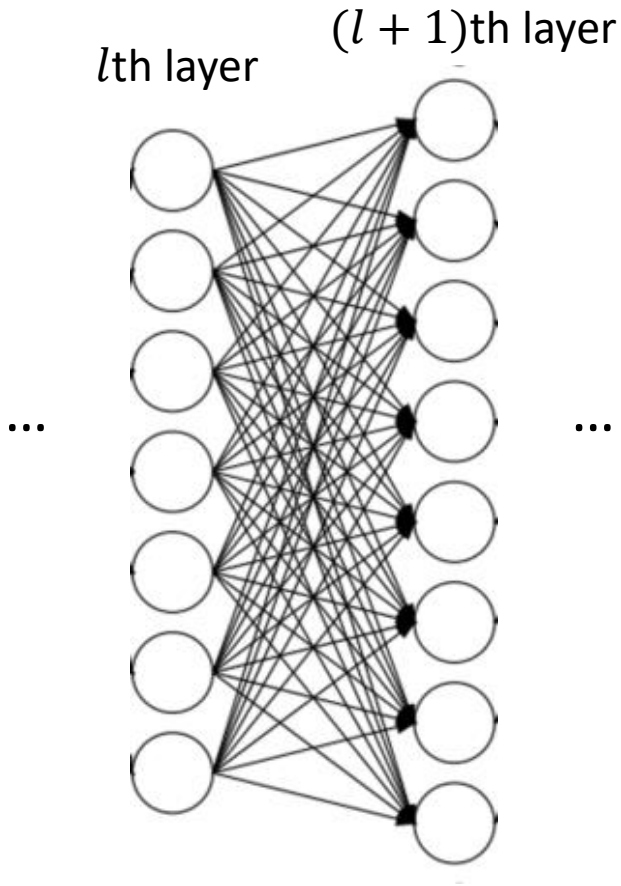
技术方案

效果与分析

总结

基于SVD的预测器[2]

[2] Andrew Davis et al. Low-rank approximations for conditional feedforward computation in deep neural networks. arXiv preprint arXiv:1312.4461, 2013.



$$a^{(l+1)} = f\left(W^{(l)}a^{(l)}\right) --- (1)$$

SVD分解，取前r个奇异值和奇异向量

$$W^{(l)} \approx U^{(l)}V^{(l)}$$

O(r(m+n))

$$p^{(l+1)} = sign\left(U^{(l)}V^{(l)}a^{(l)}\right) --- (2)$$

$$a^{(l+1)} = p^{(l+1)} \circ f\left(W^{(l)}a^{(l)}\right) --- (3)$$

O(mn)

基于SVD的预测器

$$a^{(l+1)} = f\left(W^{(l)}a^{(l)}\right) --- (1)$$

SVD分解，取前r个奇异值

$$W^{(l)} \approx U^{(l)}V^{(l)}$$

$$p^{(l+1)} = sign\left(U^{(l)}V^{(l)}a^{(l)}\right) --- (2)$$

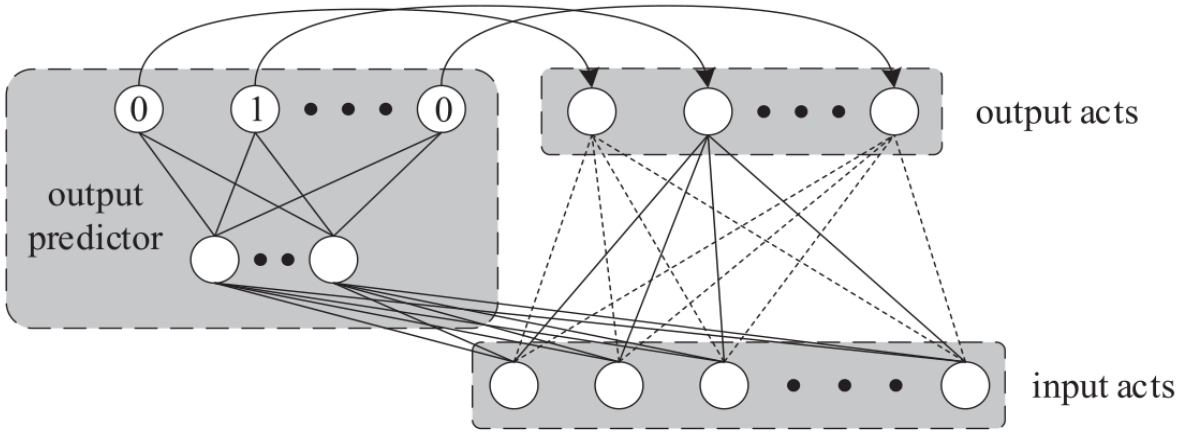
$$a^{(l+1)} = p^{(l+1)} \circ f\left(W^{(l)}a^{(l)}\right) --- (3)$$

存在的问题：

1. 它总是寻找Frobenius范数差最小的解，但它可能不是最优的稀疏预测器。

2. $U^{(l)}$和$V^{(l)}$在训练中每个训练周期更新一次。静态更新规则限制了反向传播的灵活性。

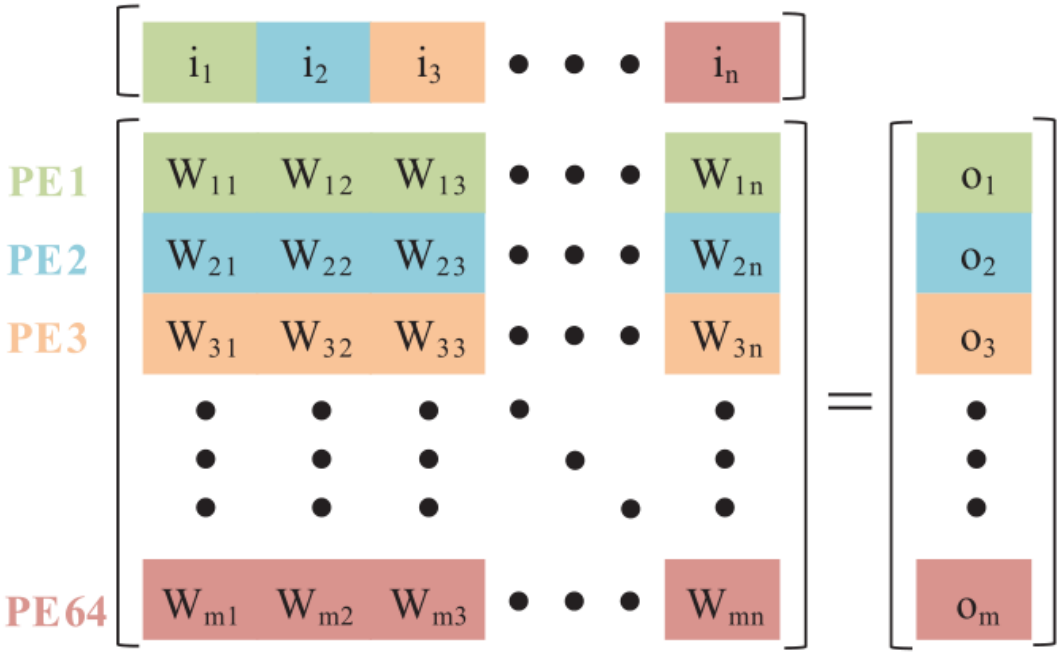
端到端训练的的预测器

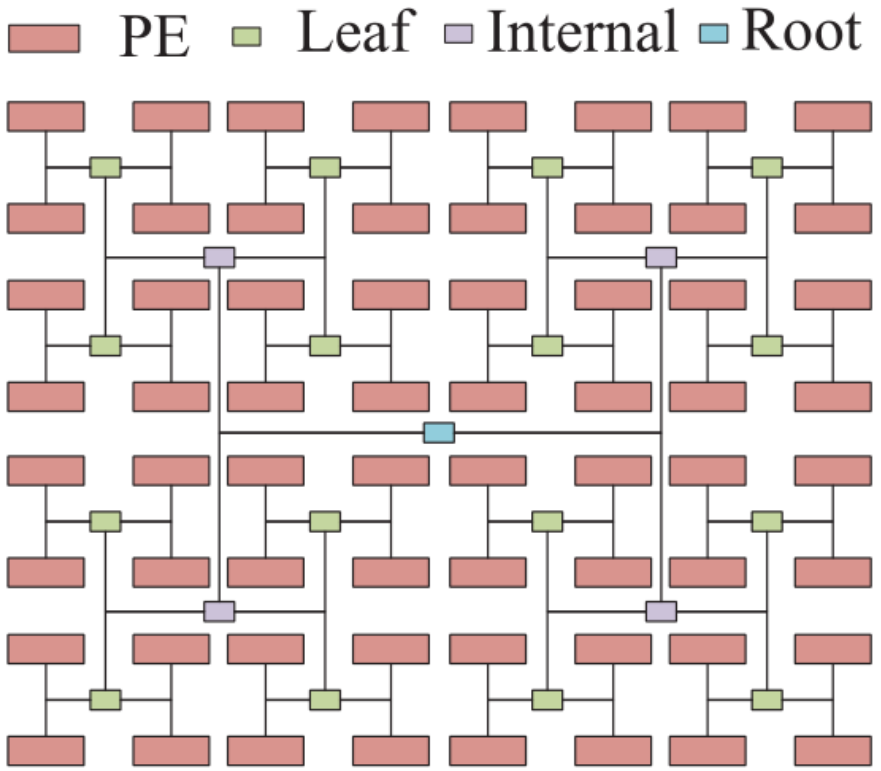


[2] Andrew Davis et al. Low-rank approximations for conditional feedforward computation in deep neural networks. arXiv preprint arXiv:1312.4461, 2013.

预测器的内部结构与之前相同，即它基于一对$U^{(l)}$ 和$V^{(l)}$。然而，得出$U^{(l)}$和$V^{(l)}$的方式是不同的。它们不使用奇异值分解(SVD)，而是从端到端训练阶段得出。

Hierarchical Architecture of SparseNN

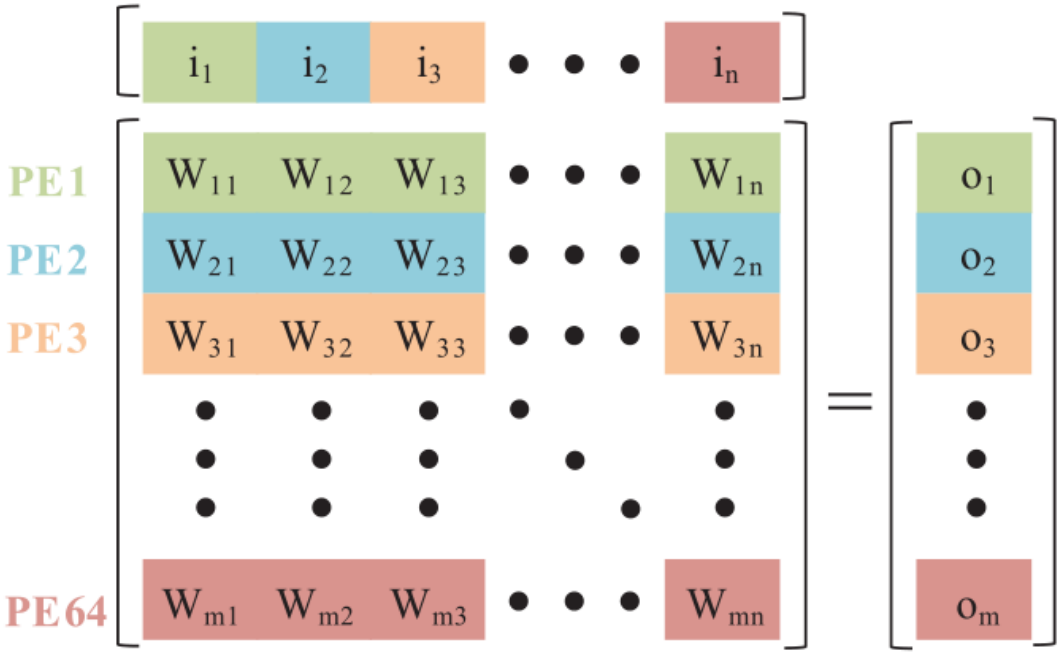


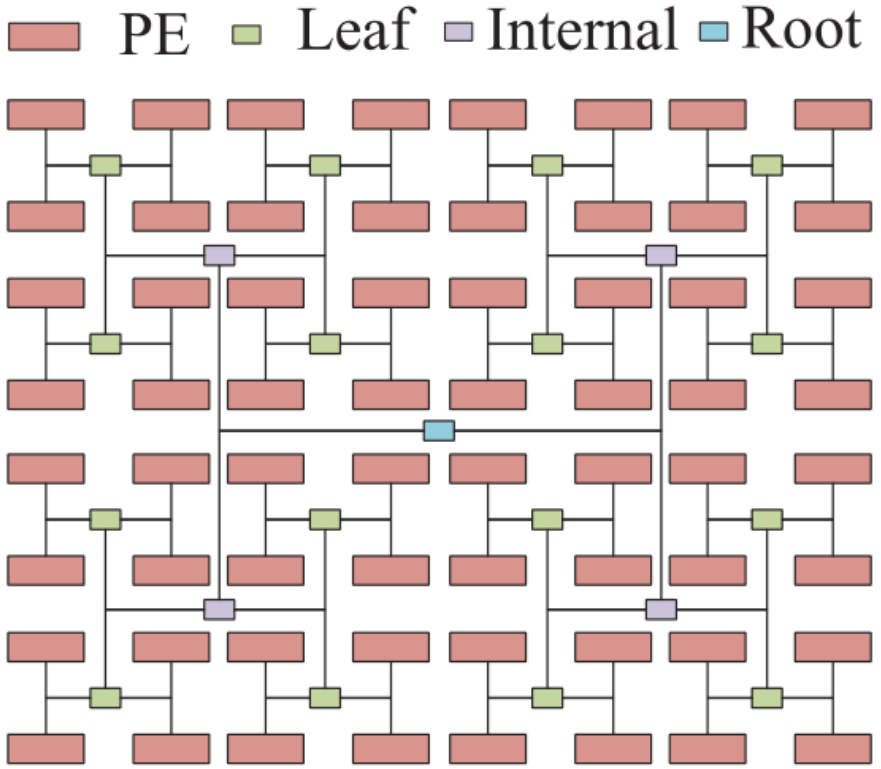


1、64个PE

2、3级片上H-树网络连接，该网络在叶级、内部级和根级具有路由。

Hierarchical Architecture of SparseNN

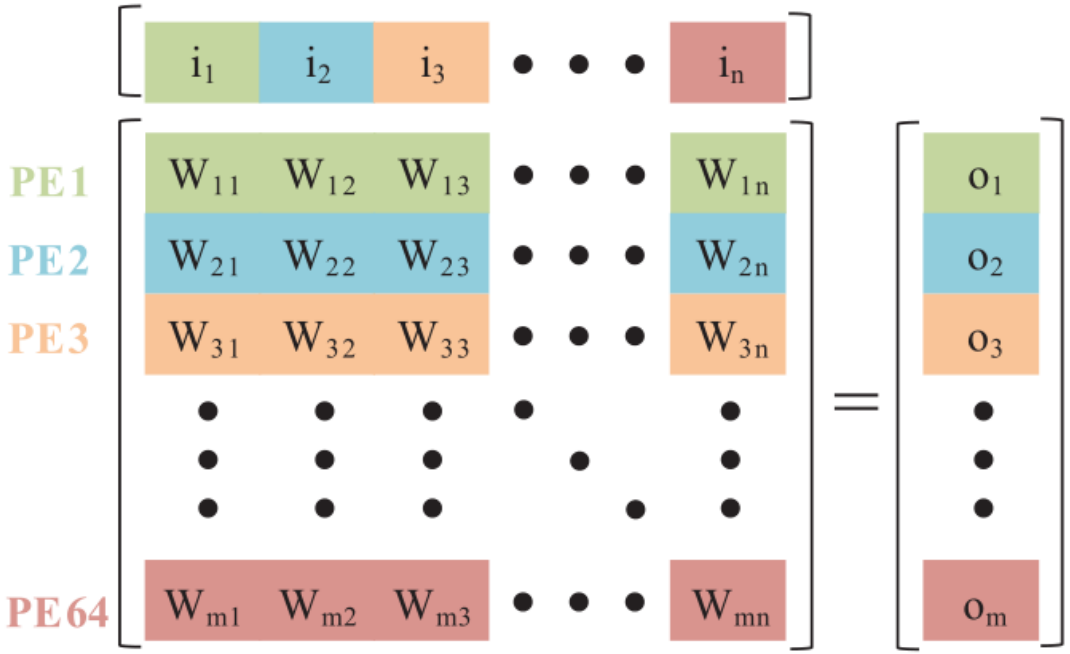




1、64个PE

2、3级片上H-树网络连接，该网络在叶级、内部级和根级具有路由。

Hierarchical Architecture of SparseNN



$$a^{(l+1)} = f\left(W^{(l)}a^{(l)}\right)$$

$$\boldsymbol{o_j = W_{(j,:)} \cdot i_j}$$

$$\boldsymbol{PE_k \leftarrow o_j, k = j \bmod 64}$$

$$p^{(l+1)} = \mathbf{sign}(U^{(l)}V^{(l)}a^{(l)}) \quad (2)$$

$$a^{(l+1)} = p^{(l+1)} \circ f(W^{(l)}a^{(l)}) \quad (3)$$

1、每个PE仅存储输入激活的子集
2、每个PE通过Noc广播其结果

On-chip Network Design

EIE的片上网络

矩阵W的维数通常很大，多行被映射到一个PE，所以每当PE接收到输入激活时，它将需要多个周期来计算与每个映射行的权重的乘法，并且只需要在多个周期之后才需要下一个输入激活。因此，它有足够的时间让下一次广播输入激活到达，以避免空转周期。

对于非平方，胖形矩阵，每个PE只需要几个周期来消耗接收到的输入激活。因此，如果下一次输入激活没有按时到达，将会有空闲周期，并影响整体性能，

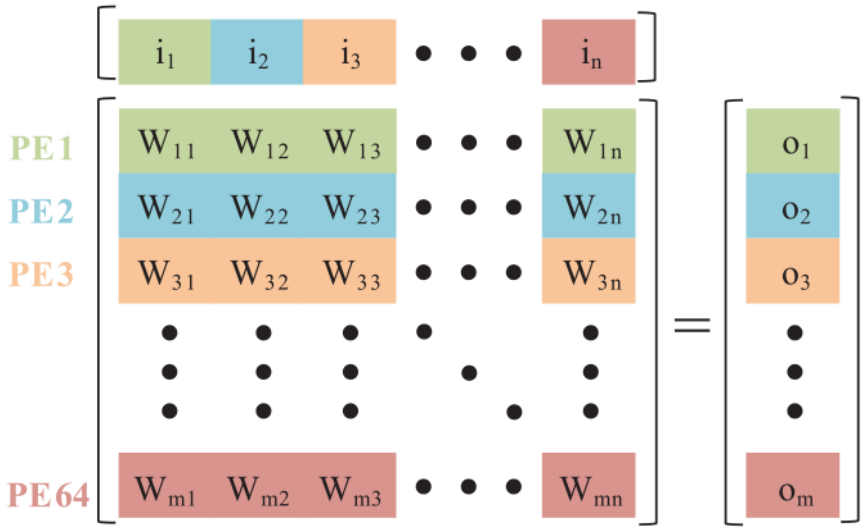
On-chip Network Design

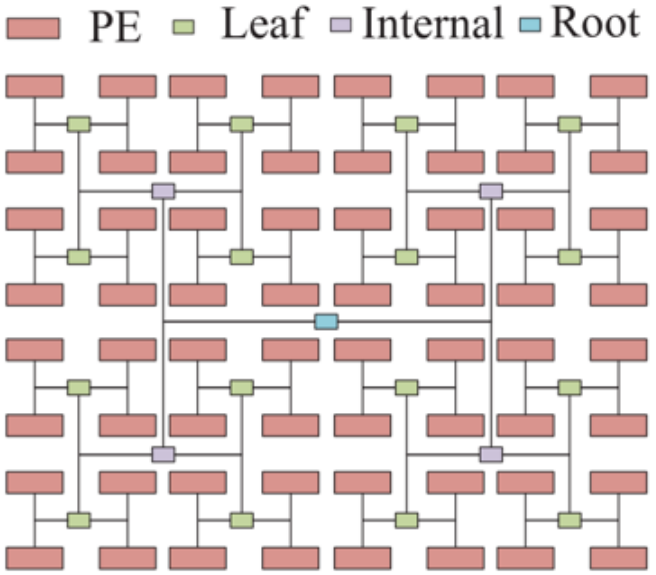
片上网络的通用缓冲流量控制

- 路由节点的每个级仲裁四个非零输入激活。索引最小的激活将被准许进入下一层，而其余的将被存储在当前节点的缓冲器中，等待下一周期的仲裁。
- 到达的输入激活是无序的，这意味着在每个PE处接收的非零输入激活的索引可能不遵循严格的递增顺序
- 无序输入激活不会影响计算结果，因为矩阵-向量乘法是可交换的，并且接收顺序并不重要

Computation Schedule for Sparsity Predictor

基于行的调度

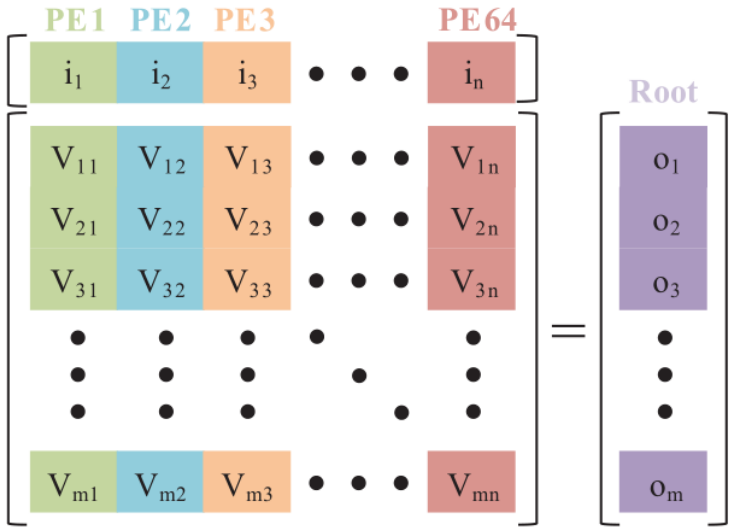


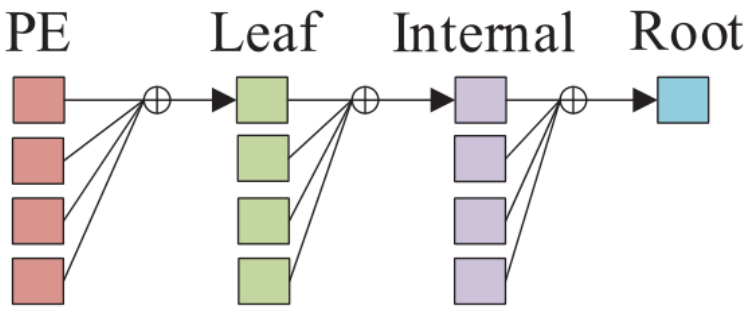


如果权重矩阵W的行数小于64，则在基于行的调度下，不是所有PE都与输出激活映射，硬件利用率低
例如，预测器V的秩通常小于64

Computation Schedule for Sparsity Predictor

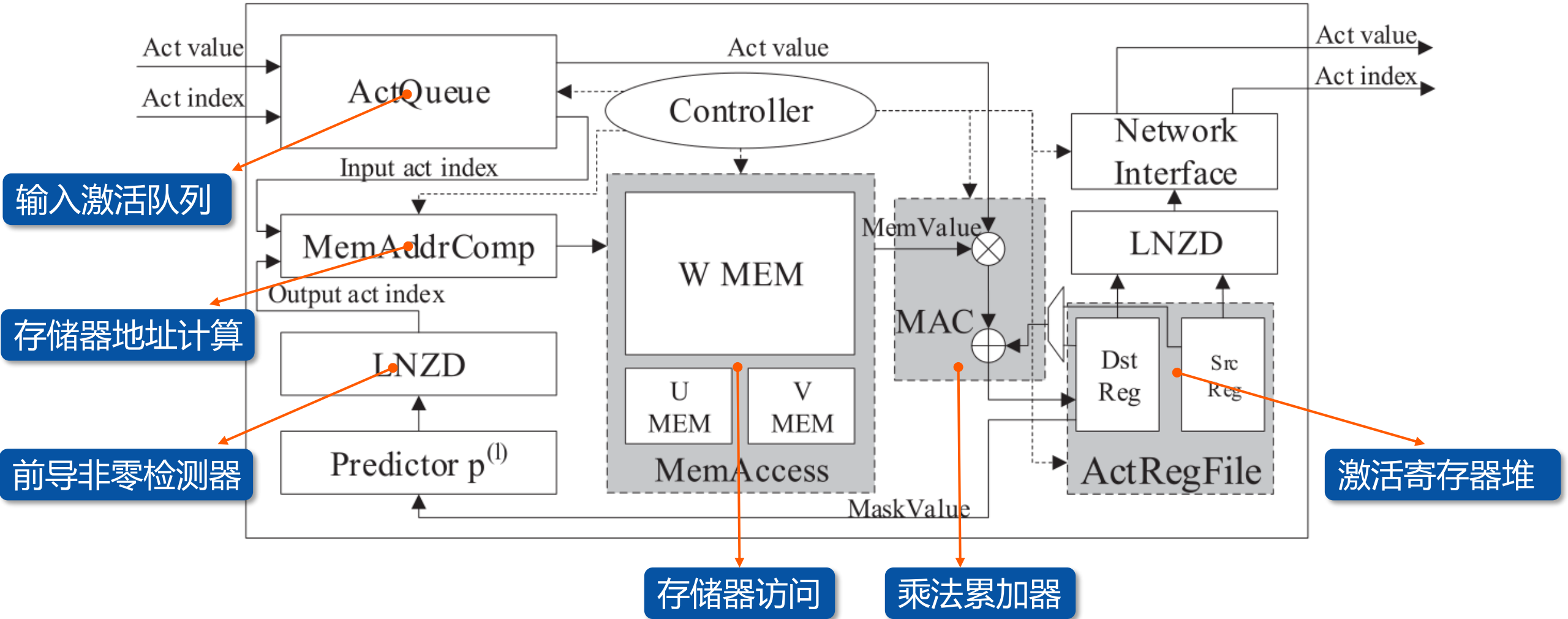
基于列的调度





1、V的列被映射到64个PE，每个PE计算输出激活o的部分和。
2、部分和的累加通过 (b)中的三级H树进行
3、累加运算嵌入到图4(c)所示的4级流水线路由中
4、U的计算使用基于行的调度

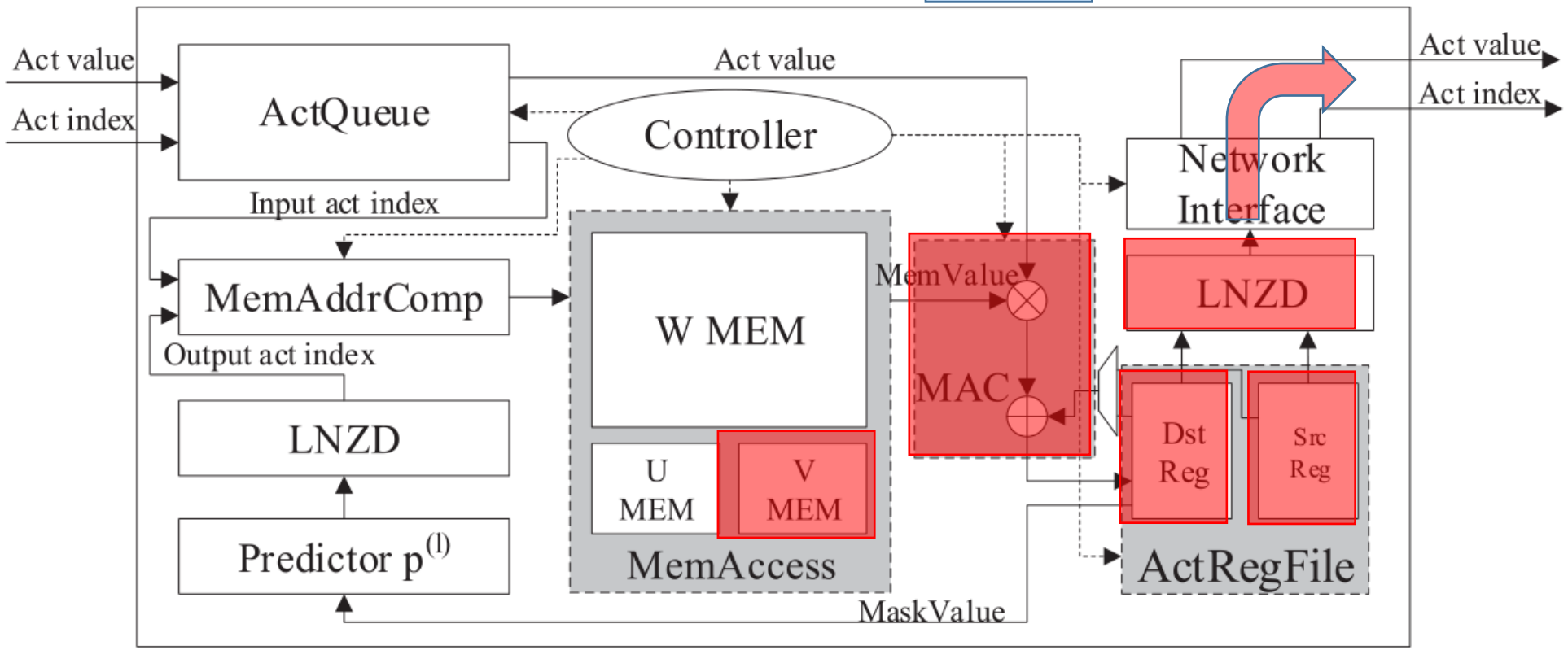
Architecture of PE with the Output Sparsity Bypass



Architecture of PE with the Output Sparsity Bypass

V computation phase

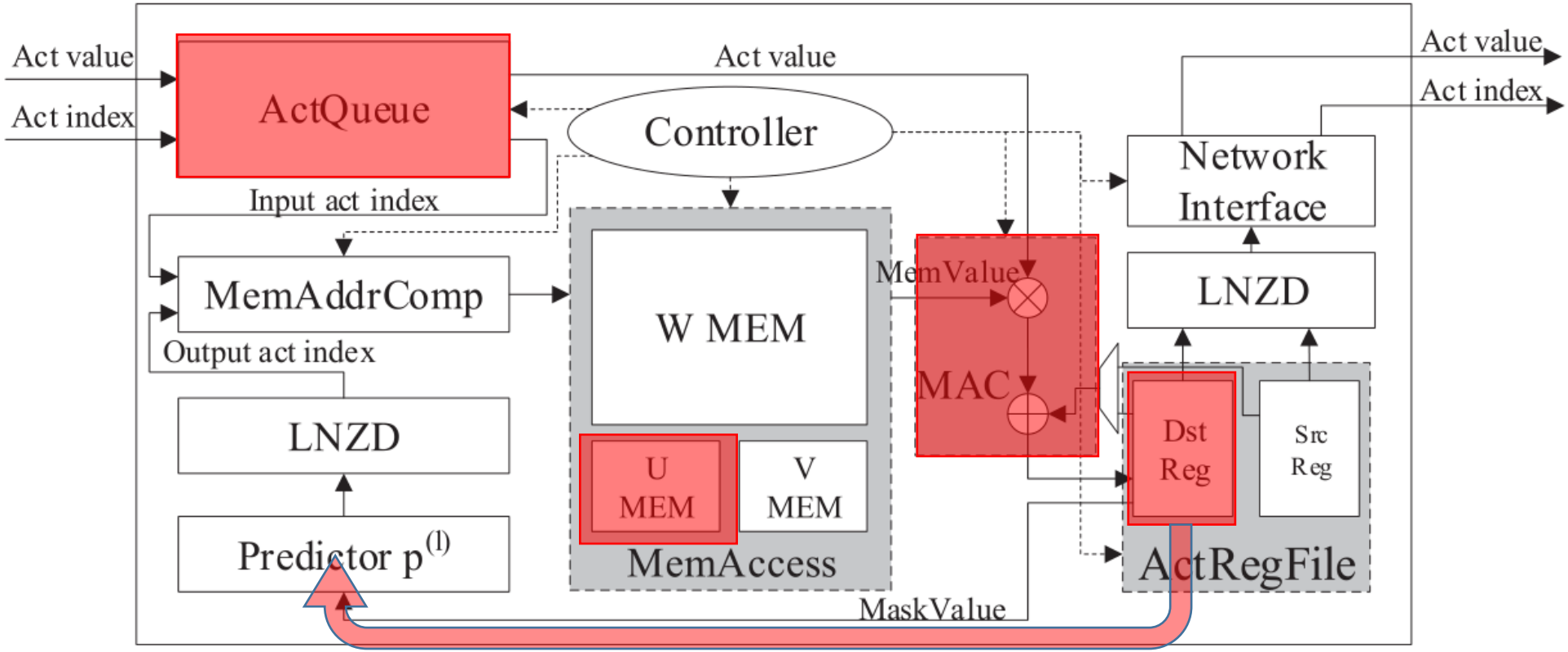
$$p^{(l+1)} = \mathbf{sign}(U^{(l)}V^{(l)}a^{(l)})$$



Architecture of PE with the Output Sparsity Bypass

***U* computation phase**

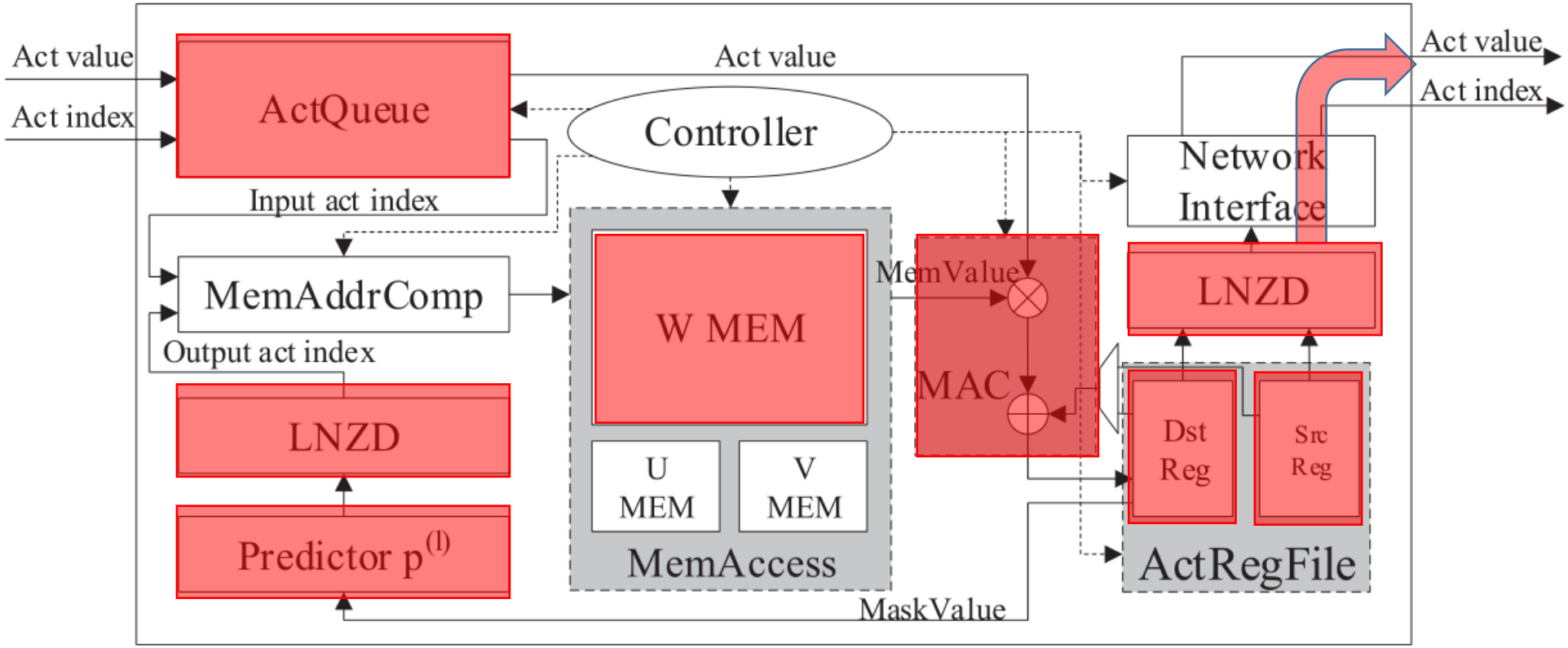
$$p^{(l+1)} = \mathbf{sign}(U^{(l)}V^{(l)}a^{(l)})$$

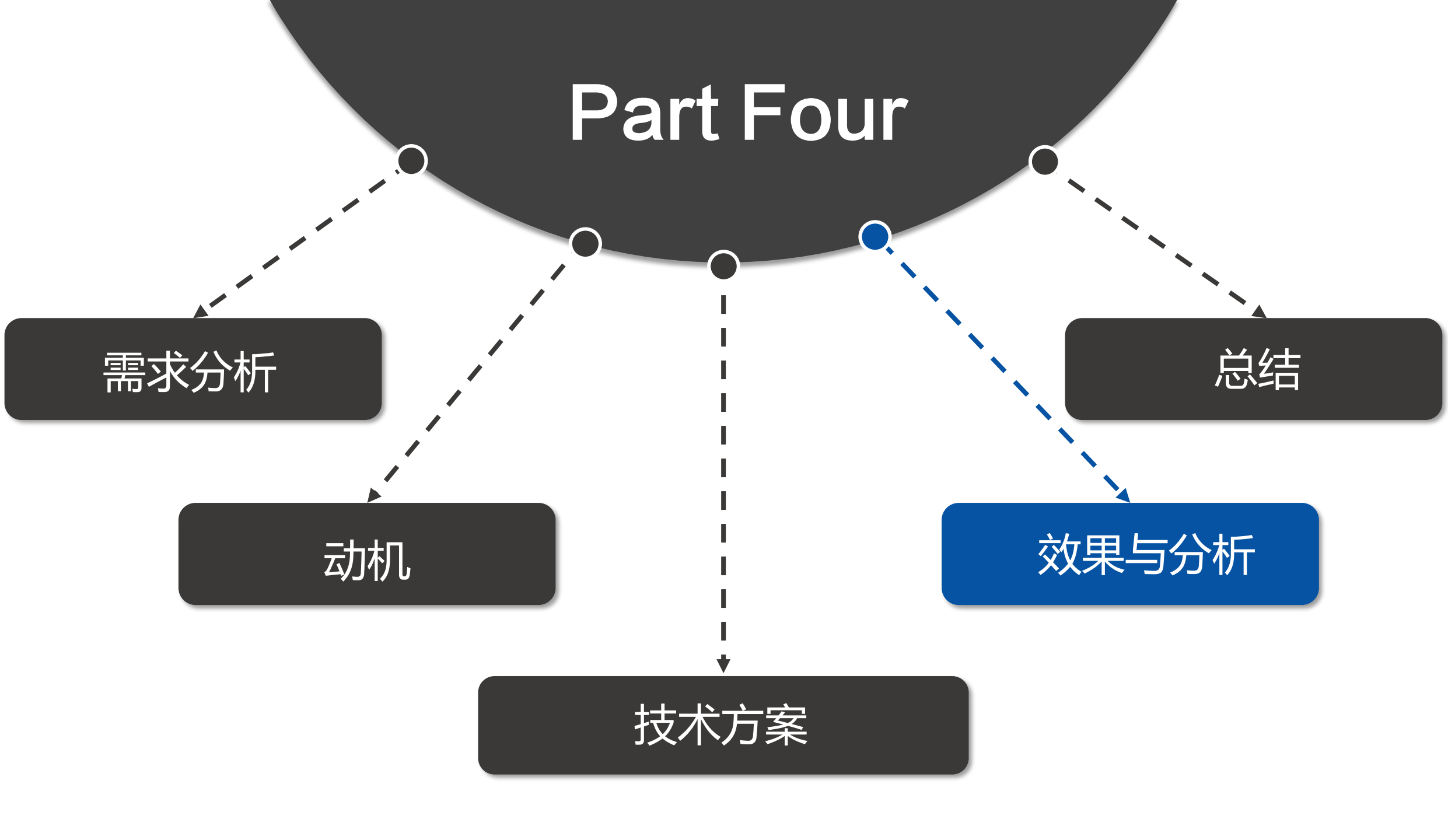


Architecture of PE with the Output Sparsity Bypass

***W* computation phase**

$$a^{(l+1)} = p^{(l+1)} \circ f(W^{(l)} a^{(l)})$$




Part Four
需求分析
动机
技术方案
效果与分析
总结

实验结果

端到端的训练算法

实验配置：
1.采用的数据集：
MNIST-Basic数据集(BASIC)以及两个变体(ROT)和(BG-RAND)

2.两种不同的神经网络结构:3层(1个隐层)和5层(3个隐层)。(每个隐藏层有1000个神经元)

3层神经网络的测试错误率(TER)和预测输出稀疏度如图所示

分析

在单隐层神经网络上，将该端到端训练算法与SVD算法进行了比较。从图中可以看出所提出的端到端训练算法与U-V预测器的秩的大小有很好的比例关系。

性能差异的主要原因是在传统的SVD方案中，U-V更新是静态的，不能调谐，当秩太小时，权重矩阵W的低秩近似是不准确的。

实验结果

1.端到端的训练算法

秩为15的5层神经网络的结果如表所示
(其他等级大小的结果也有相似的趋势)

Dataset	Algorithm	TER(%)	$\rho^{(1)}$	$\rho^{(2)}$	$\rho^{(3)}$
ROT	NO UV	**8.54**	N.A.	N.A.	N.A.
	SVD	10.69	**90.74**	28.12	34.27
	End-to-End	8.8	69.41	**64.13**	**71.07**
BASIC	NO UV	2.738	N.A.	N.A.	N.A.
	SVD	2.728	**62.5**	38.15	39.38
	End-to-End	**2.718**	56.34	**65.89**	**66.7**
BG-RAND	NO UV	10.08	N.A.	N.A.	N.A.
	SVD	10.036	51.61	**51.49**	24.01
	End-to-End	**10.03**	**52.79**	48.23	**41.44**

分析

在表中，比较了用不同算法训练的5层神经网络在每个隐层的TER和输出稀疏性。

端到端训练算法训练的网络保持了与SVD方法相似(甚至更好)的精度，但具有更高的隐含层平均稀疏率

实验结果

2.SparseNN体系结构

实验配置：

1.使用Verilog HDL实现了SparseNN。

2.通过Matlab定点仿真验证了硬件实现的功能仿真。

3.利用Synopsys Design Compiler和TSMC 65 nm LP库，在最坏的PVT条件下合成了SparseNN。

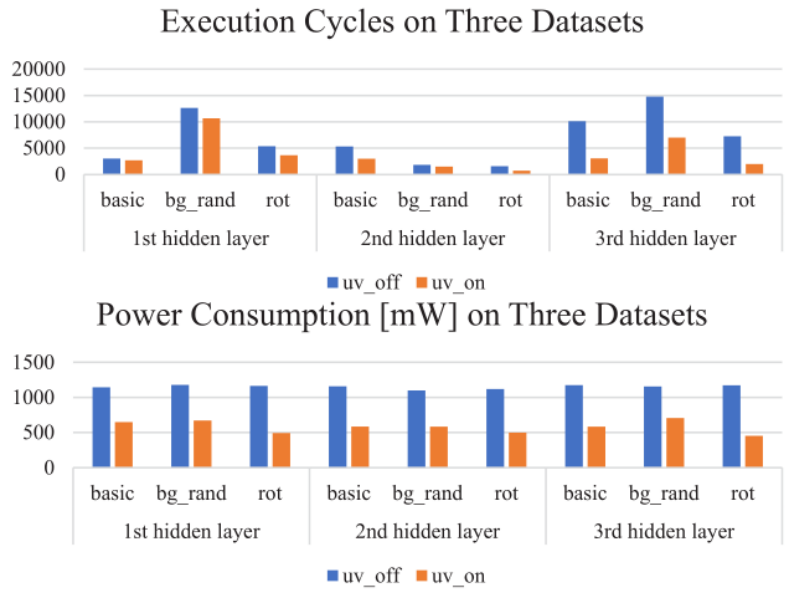
SpraseNN的面积划分

	Area (μm^2)	(%)
Total	78,443,365	(100%)
Combinational	1,716,373	(2.4%)
Buf/Inv	199,038	(0.2%)
Non-combinational	2,068,996	(2.6%)
Macro (Memory)	74,426,310	(94.8%)
Processing element	1,216,457×64	(99.2%)
Routing logics	590,062	(0.8%)

分析

路由节点仅占总面积的一小部分（0.8%，不到1%），主要的面积是PE（99.2%）。因为每个PE中用于W,U,V的SRAM占据了94.8%的面积。

实验结果



SparseNN在三个基准测试上的执行周期和功耗结果

对于剩余的隐藏层，减少的周期可以高达70%。上一层的预测输出稀疏性将增加当前层的输入稀疏性。因此，输入稀疏性和输出稀疏性共同提高了吞吐量。

分析

在不使用UV预测器的情况下，SparseNN与传统的仅利用输入激活稀疏性的EIE架构的功耗相同。
在输出稀疏性的情况下，执行周期的数量的改善在层与层之间是不同的。

对于第一隐层，周期的减少范围为10%~31%。对于启用UV和禁用UV的网络，第一个隐藏层的输入是相同的，因此吞吐量的提高仅来自输出稀疏性。

实验结果

DNNS与现有SIMD硬件平台的比较

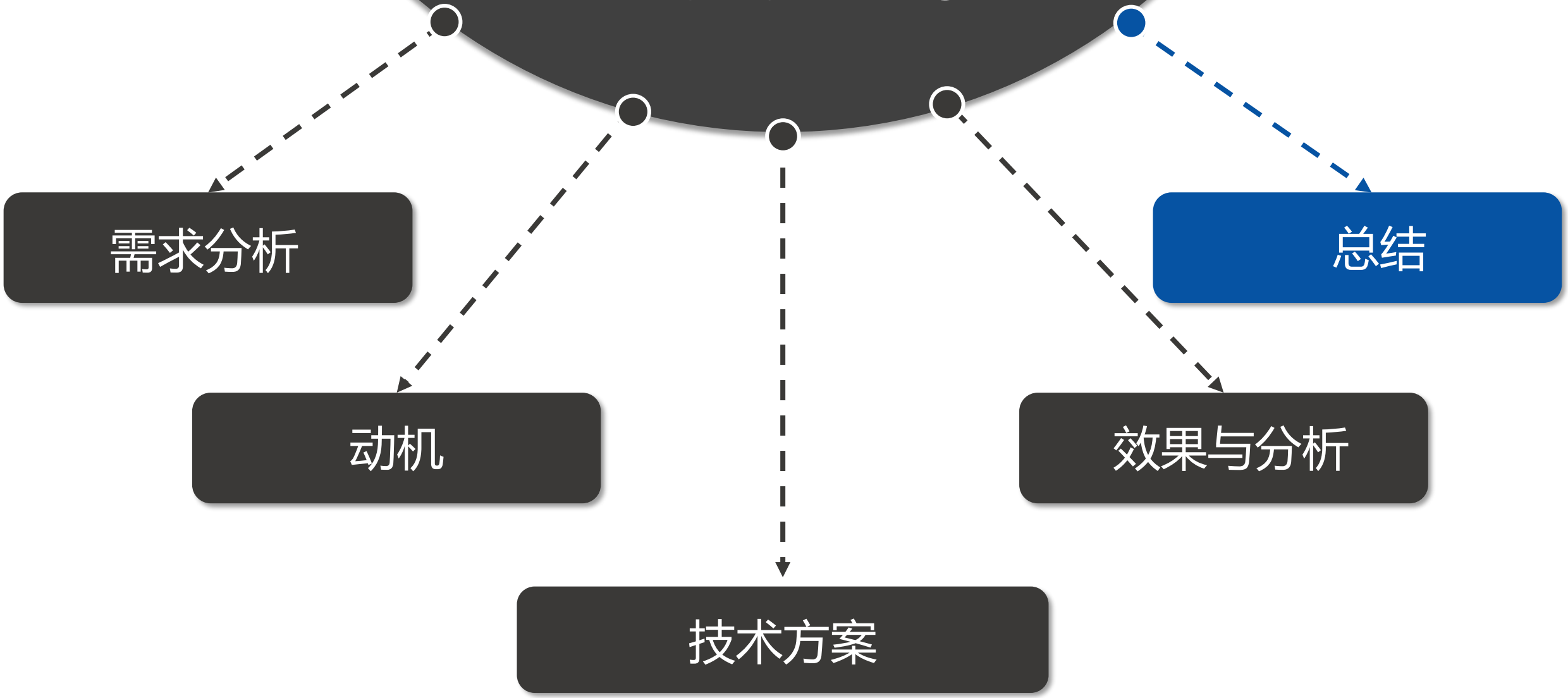
Platform	LRADNN [12]	DNN-Engine [14]	This work
Technology	65nm	28nm	65nm
Peak Perf.	7.08GOPs	19GOPs	64GOPs
W memory	3.5MB	1MB	8MB
Power (mW)	439∼487	63.5	452∼705
Area (mm^2)	51	5.76	78

分析

SIMD架构在并行性（即SIMD窗口）和片上带宽之间存在折中，即LRADNN的工作频率较慢。

从仙人掌内存模型来看，当技术节点从28 nm扩展到65 nm，内存大小从1MB更改到8MB时，每次读访问的能耗大约是11倍。
因此，如果考虑到这种可伸缩性，SparseNN的能效比传统的SIMD架构高4倍。

Part Five



总结

- 提出了一种端到端的训练算法，开发了一种轻量级的运行时预测器，实现了输出激活稀疏性的实时预测
- 端到端的训练算法与传统的截断奇异值分解(SVD)方案相比，具有可扩展性的秩和更好的预测稀疏性
- 提出了一种节能的硬件结构SparseNN，SparseNN的吞吐量可以提高10%到70%，而功耗大约减少一半
- 与最先进的仅利用输入稀疏性的SIMD架构相比，SparseNN显示出更好的可扩展性和更高的能效。

论文创新点

- 利用DNN固有的激活稀疏性来减少执行周期和能量消耗；
- 实现了输出激活稀疏性的实时预测；
- 提出了一种节能的硬件结构SparseNN来同时利用输入和输出的稀疏性

Thank you

欢迎提问