ICLR 2023 — More Convnets in the 2020s: Scaling up Kernels Beyond 51x51 Using Sparsity

摘要

大卷积核经由ConvNeXts重新引入并将基础general-CNN推高到新的baseline之后，一系列工作沿着将卷积核的尺寸继续增加来推高此基准，其中在 Sparse Large Kernel Network (SLaK)中卷积核的尺寸达到了51*51。然而，如slak中所用到的大卷积核、稀疏权重矩阵都不是硬件友好算子。并且即便考虑了稀疏权重矩阵，因为卷积感受野的原因，大卷积核重新被引入的始作俑者transformer的所具备的特性之一稀疏注意力也没能在大卷积核中有所体现(翻译重点在后边这句)。基于此，我们提出了shift-wise的算子——使用普通小卷积核和平移操作可达到此类大卷积核的目的并且可以实现远距离的稀疏依赖效果。这样，使用shift-wise不仅是软硬件友好的，而且使得CNN在稀疏注意力方面有了更好的体现。我们提出的shift-wise不仅在精度上超越了state-of-art的SLaK方法，而且我们还在稀疏依赖的角度将普通CNN的baseline推高到更高，并大幅降低计算量。综上，我们对SLaK的改进算子shift-wise可以将更基础的CNN提升到新的基准，同时为CNN的基础结构贡献了新的成员。部分代码已开源:https://github.com/lidc54/shift-wiseConv。

关键词：CNN；Large Convolutional Kernel；shift-wise

1 引言

最近以大卷积核为主要特征的改进大幅提升CNN网络能力的背景下，相关研究迅速瞄准了这一个对CNN的提升的新的研究方向。对在此相关研究进行梳理后，我们发现SLaK-net [28]是一种将卷积核尺寸推到新高度且表现较为优秀的研究，它的实现兼顾了效率与实现难度。因此，我们选择SLaK作为我们研究的基础。基于此，我们探究是否能能够继续推高CNN的基础能力甚至形成新的算子，最终形成有价值的研究。

视觉识别在2020年井喷式发展，始于Vision transformer的引入，它很快取代了卷积神经网络(ConvNets)成为最先进的图像分类模型。ConvNeXts [29]通过分析transformer的设计空间，对Resnet进行重新设计，率先重启了大卷积核范式的应用。相比于其他设计要素，卷积核的尺寸和表现是其中较为反直觉的存在。针对卷积核尺寸可以增加大多大并带来收益这一问题的研究，RepLK-net [7]通过特征重参数化将卷积核进一步提升到了31x31。SLaK-net [28]则进一步通过特征分解和权重矩阵稀疏训练将卷积核增大到了51*51，并在此过程中让所提结构有持续的收益。然而，大卷积核的使用却不是最近才出现的。实际上，对大卷积核的使用由来已久，早在Alexnet [22]就大量使用11x11和5x5的卷积，这也是一篇在深度学习中有着非常重要地位的研究，为后续的研究奠定了非常好的研究基础。后续出现很多CNN中经典的主干

框架，比如广泛使用并不断得到优化的 Resnet [12]其早期版本stem layer中也沿用了使用大卷积核的方式。同样在基础网络结构中具有重要地位的GoogleNet提出的Inception V1 [35]使用了多个并行的不同大小的卷积核进行信息的融合的结构，如图1(a)。同一时期的VGG-net [34]探索并论证了感受野的概念，即堆叠小卷积核可以在使用更少的参数量和计算量时即可理论上达到大卷积核相同的效果，反而借此可以引入多层激活层增加了网络的非线性表达能力。后期limu研究团队 [13]对resnet训练中所用到的tricks进行汇总，提出新的resnet结构，基本上将resnet中的stem-layer的大卷积核的使用也打入历史。然而，大卷积核的使用并未完全消失，在一些领域还在使用。大卷积核的特点之一是增大了感受野，在扩大大感受野的相关研究领域，就一直有一些相关要素存在。比如虽然Resnet及其变种因其良好的效果和合理的推理速度占据了CNN相当比重的研究，但是在多尺度特征融合这条路上，后续不断借鉴GoogleNet提出的Inception V1架构的相关研究却非常多。比如Kaiming he [18]使用的spatial pyramid pooling(SPP)，如图 1(b)模块，而在pspnet中 [41]使用的多尺度pooling层和普通卷积组成事实上的大卷积核的pyramid pooling moudle结构，如图1(d)所示，进一步将这种特征融合增强的DeepLab [2]中ASPP (Atrous Spatial Pyramid Pooling)模块和Trident Networks [25]中三叉戟结构则使用了感受野不断增大的空洞卷积来实现类似功能，如图1(c)所示。对于空洞卷积类似大卷积核的用法，在FCN [30]引入到目标分割后也有广泛的类似应用。同样是增加多尺度特征信息融合为目的，类似于inception结构的SegNeXt [10]则并行地使用了尺寸为1*7、1*11和1*21的卷积做目标分割。而堆叠的1*k和k*1的大卷积核的操作，在2017年提出的Global Convolutional Network(GCN) [32]也有类似的模块设计。在多尺度信息融合和大感受野的使用这些相关研究方向上，大卷积核的影子延续已久。

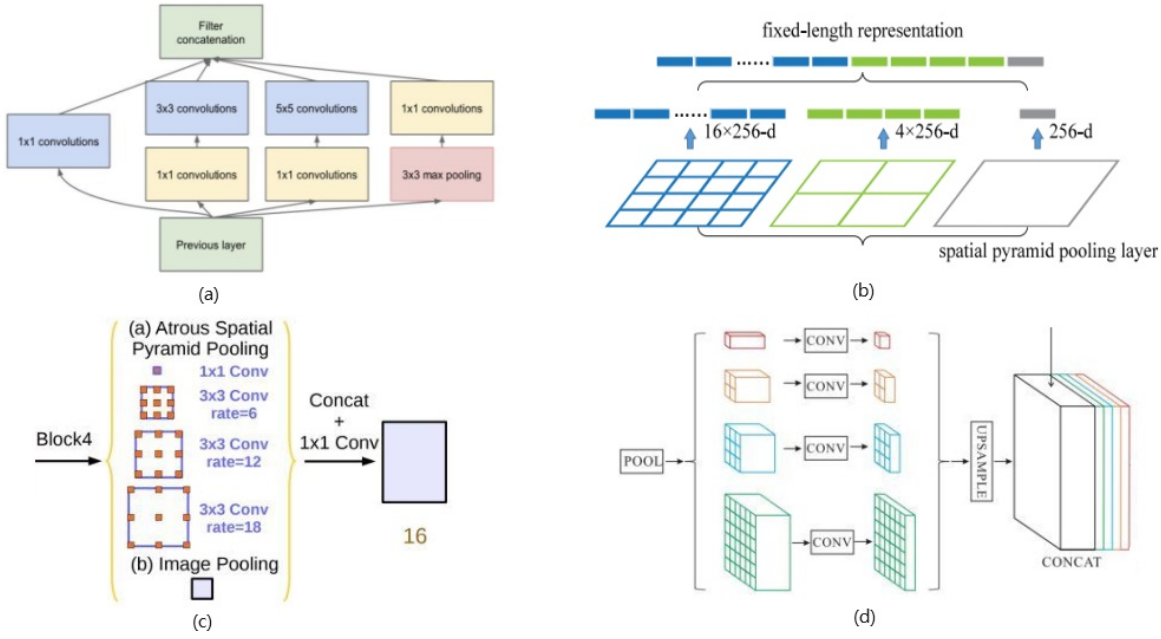

图 1. 多尺度特征融合. (a)GoogleNet提出的Inception V1;(b)spatial pyramid pooling(SPP) 模块;(c)deeplabv3+中ASPP模块代码;(d)pspnet中的pyramid pooling moudle结构.

此外，需要用到注意力机制的地方，也往往以较远距离的特征依赖来构建相应模块。比如被设计成特征选择的注意力机制，在双线性网络 [27]有所体现，其将两个孪生网络的输出相乘用于提取细粒度特征，Gated-SCNN [36]同样使用这种特征相乘的方式来处理细粒度

特征任务。类似的特征相乘的方式在DANet [9]中做成了多重注意力来完成语义分割。而对于自注意力机制有一个广泛使用且简约的使用是non-local [38]结构，其在理论上和Non-Local Means非局部均值去燥滤波有点相似，是在更大的范围上进行加权。这些操作虽然没有使用大卷积核，但是通过全局池化和1x1卷积的配合使用，达到了大卷积核的效果。考虑到空洞卷积相对于大卷积核的意义，可变形卷积DCN [6]在自适应空域聚合能力上表现独树一帜，某种程度上可看作特殊的空洞卷积。DCN家族所特有的可解释性和特征自组织机制，也不断得到壮大。在PointRend [21]进行对输出边界点位置进行学习，可精细调整实例分割边缘。Dynamic Snake Convolution(DSConv) [33]在视网膜血管分割中，通过添加约束使得可变形卷积参数能附着在血管变化上。 DCN家族以较少的参数量达到了较大的感受野，并可以自适应的对特征进行聚合。虽然其计算成本较高和计算对齐造成的延迟较高，其天然的带来了可解释性。纵观这些大卷积核或与之相关的影子操作，我们发现在长距离特征依赖关系的构建和特征空间信息聚合方面，大卷积核的使用是有其独特价值的。这和ConvNeXts、RepLKnet和SLaK的实验结果也是一致的，即通过增加卷积核尺寸扩大感受野来建立远距离特征依赖，最终提升了模型的表现能力。在UniRepLKNet [8]强调了真实感受野和理论感受野的关系——通过堆叠深度所达到的感受野与真实感受野存在开方而非线性累积的关系，在理论层面进一步显示了真实感受野存在是非常必要的。虽然如此，我们也看到SLaK中的实验显示卷积核尺寸超过51*51之后模型精度是下降的，而在ParCNetV2 [40]中将卷积核大小扩展到输入特征图两倍的程度，来实现隐式的位置编码并建立特征的超远程依赖关系，以杜绝模型表现能力受到感受野的影响，模型反而可以在参数更少时达到不错的效果。这不禁让人深思，大卷积尺寸的增加虽然将CNN结构的精度提升到和ViT相当的程度，但是大卷积核被小卷积核逐步取代的过程是否还是必然？除了进一步提升卷积尺寸乃至超越特征图的方向，是否有其他维度的信息可以被探究？尤其考虑到Transformer结构在构建特征间稀疏依赖关系具有很强的建模能力 [26]，CNN是否也可以具备类似的构造呢？

基于此思考，我们提出了shift-wise构造。我们首先论证了分解的大卷积核可以通过使用shift-wise用多个小卷积完全等价。然后，我们基于此对小卷积组进行成员稀疏训练，形成非稠密的事实大卷积核，以此来让CNN逼近Transformer中的稀疏特征依赖关系。

结果显示，使用shift-wise的结构，可以更灵活的引入一些重参数化结构，让模型的表现能力进一步的提升，同时，所使用的参数量也更少。

综上，我们所提出的shift-wise结构，可以让CNN网络结构使用常用的软硬件友好的算子即可以达到较好的表现能力；在此实现过程中，我们也探讨了CNN的稀疏依赖的实现形式，并形成一个基础的算子。

2 相关工作

2.1 最近对大卷积核的研究

相比于CNN， transformer具有其局限性，比如，其计算复杂度高，训练推理成本都更高，需要更多训练数据，偏移不变性差等等。鉴于transformer和cnn本质上的差异，这形成了对神经网络所形成的一些理论认知的盲区，即CNN的落后是因为某些特性还是因为本质的缺陷。ConvNeXts [29]通过分析transformer的设计空间，成功地对Resnet进行了重新设计，率先重启了大卷积核范式的应用。随后Visual Attention Network [11]在ConvNeXts基础上，着重强调大

感受野的所带来的区域互联的意义，通过组合了空洞卷积、depth-wise、point-wise对权重张量形式分解，利用这三个维度上的分解增强了空间感知范围。RepLK-net [7]则在ConvNeXts稍后提出了针对卷积核可以做到多大的探究，将卷积核进一步提升到了31x31。SLaK-net [28]继续这一思路，将卷积核增大到了51*51，并且会给分类和下游任务带来显著收益。与前者着力研究大卷积核不同，InternImage [37]是可变形卷积调整之后应用到transformer架构而形成了新一版本的DCN，并且特征的长距离依赖和空间特征自组织聚合两个概念上和大卷积核概念进行比较分析收益。UniRepLKNet [8]则思考大卷积核的必要性，强调了真实有效感受野大小和堆叠深度存在开方的关系的结论，进而提出通过并行的多个空洞卷积进行重参数化并使用Squeeze-and-Excitation block [14]结构实现了使用较少堆叠实现更接近理论的有效感受野的目的。

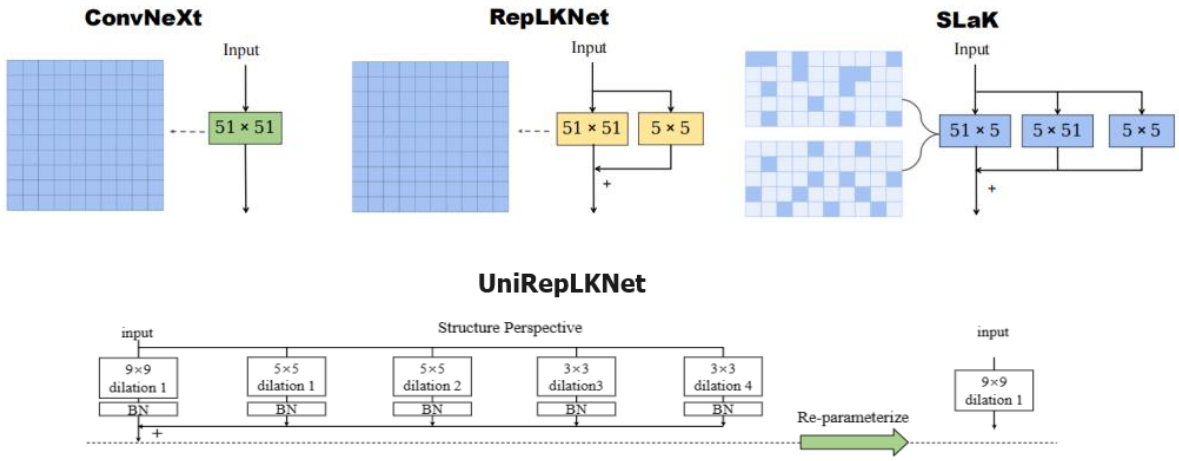

图 2. ConvNeXts、RepLK、SLaK、UniRepLKNet

随后渐渐出现一些对于大卷积核的应用拓展，其中有一些改进的沿用惯例表现出一个有趣现象。Large Separable Kernel Attention [24]在Visual Attention Network [11]基础上，对VAN所使用的三种卷积算子进一步分解,可以用更低的成本获得类似的效果。 oriented 1D convolutions [20]将传统的k*1、1*k矩阵的形式分解添加了两个对角的形式分解，并且将k扩展到较5、31来提升模型的表现能力。 Deformable Large Kernel Attention [1]则使用可变形卷积得到注意力权重对原有特征进行加权，在医学分割数据集上达到了很好的效果。 Dilated Convolution with Learnable Spacings (DCLS) [19]对空洞卷积各参数的空洞率转变成可学习的参数，但限定在空洞卷积kernel-size内进行参数位置移动，以此来增加卷积核的表达能力，又避免了同为异构卷积的方法DCN在推理时会有复杂计算的问题。 ParCNetV2 [40]将卷积核大小扩展到输入特征图两倍的程度，来实现隐式的位置编码并建立特征的超远程依赖关系。 Huang [16]探究了大卷积核的网络是否能将特征提取能力和对感受野的影响的能力转移给普通小卷积核的网络，并用实验证明了大卷积核网络在这方面确实有优势，从侧面证明了大卷积核的特征提取和组织能力可以让数据拥有更符合期望的数据流形。

虽然大卷积核在最近持续推升CNN的表现能力方面不可或缺，但大卷积核本身的问题也同样不可忽视：

1. 大卷积核的kernel-size过于的大，突破常规算子对其的优化，在一系列软硬件平台上的支持受限，导致其推广使用受到阻碍。

2. 特别是RepLK-net、SLak接连对大卷积核尺寸上限的研究，显示出大卷积对CNN重新达到有竞争力的基准是非常必要的。但是增大卷积核尺寸和建立远距离依赖的关系并非持续线性(ConvNeXts、RepLK-net和SLaK都发现持续提升卷积核尺寸会存在效果的峰值)，进一步对二者复杂关系的研究还未出现。**同时，transformer具有稀疏注意力的能力而不必关注所有的输入token，这一特性需要也赋予CNN**。(SLaK虽然权重也是稀疏的，但是信息的局部相关性的原因导致其构造依然是高相互关联的)。

3. 虽然对大卷积核重新引入，将基准CNN拔高到新的高度，但是算力开销也构成推广的障碍。即便如空洞卷积、DCN等少参数量的异构卷积也在探索CNN的这个新的基准，但是这些算子本身对于硬件并不算友好。这需要探究有没有基于常规CNN算子的框架结构可以重新定义此基准。

2.2 shift操作

Shift操作可以看做是Depthwise Convolution的一种特殊情况，在2017年由wu [39]提出，可以由较低的成本实现卷积输出的多样性，如图3(a)所示。 Active Shift [17]将shift参数化，提升shift的自适应能力，如图3(b)所示。chen则在2019年进一步扩展了shift操作提出了Sparse Shift Layer [3]，发现active shift位移位置是高斯分布的，大部分可学习shift参数都没有意义，少量的操作即可以有不错的效果，如图3(c)所示。 ACmix [31]在分析transformer的自注意力时，提出可以将普通卷积看作1*1的卷积和shift的连续操作，而前者可以直接用于transformer的自注意力计算，这样总体加上shift操作后，相比原有的自注意力机制可以用较低的成本获得卷积的信息加成。 Shift Graph Convolutional Network [5]使用特征移动的方法在图神经网络上表示出全局和局部的空间信息以及时空错位信息，进行信息的融合。 xvolution [4]也使用类似的全局和局部的思想，具体的是在卷积神经网络上使用偏移特征图得到类空洞卷积空洞率的特征组合，来近似模拟全局相关信息做特征注意力机制。

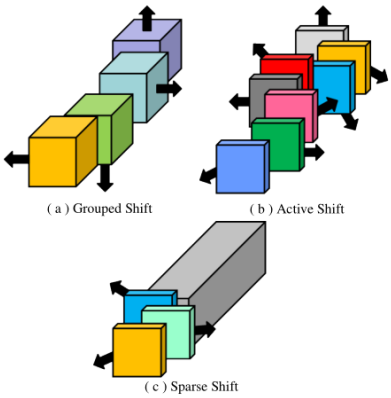

图 3. shift操作. (a)分组shift操作;(b)depth-wise shift操作及对位移的参数化;(c)部分shift操作.

shift-wise和这些shift操作有以下的不同：

1. 虽然shift操作在最初被wu [39]提出以及后续的改进，都是对特征图整体的操作，并且兼具了调整感受野的功能，甚至在shift GCN [5]和xvolution [4]都考虑了使用shift来近似全局特征，我们所提的shift-wise是在一个维度上进行以尽可能扩展真实感受野为目标的移

动。相比之下，shift早期操作是在局部进行特征移动以提供近似特征的变化，xvolution则是将特征平移均匀的放到了在所定义的局部的邻域上来近似全局特征。考虑到UniRepLKNet [8]中所论述的真实感受野大小和堆叠深度的理论感受野存在开方的关系，这些shift操作的真实感受野的扩展是有限的。

2. shift-wise在扩展感受野的同时，还将“特征的局部点具备稀疏依赖”的探索添加进来形成原子操作，相比于xvolution中所定义的原子(atomic)算子操作的算力需求是大大降低的，因为后者是一个特征对应多个平移后的克隆样本，增加了很多计算复杂度。

我们限定移位操作是一个维度上进行，并且对齐到固定格网大小让所移位特征的真实感受野尽可能的大，再去除部分位置达到数据驱动的多样性局部依赖。

2.3 剪枝操作

神经网络粗粒度剪枝是通过让网络连接或权重失活的方式，达到降低网络参数量的目的。然而，在遇到resnet分支网络合并时剪枝常常会比较麻烦，另外，前后层的输入输出数量的统一和通道的对应也常常现在粗粒度剪枝的应用。与传统粗粒度特征剪枝的方式相比，保留了神经网络的整体结构不变，在resnet多分支合并的网络结构中这一点具有特别的意义。同时，我们所提的shift-wise算子，在利用Rep特征重融合的同时，可以调整优化特征的依赖关系。据我们所知，这是一种少有的可以让数据流形在训练过程中持续发生变化并且同时保持网络结构整体不变、不退化的的神经网络。

2.4 创新点总结

在方法层面，我们的创新点总结如下：

- RepLK-net [7]虽然引入了自定义的算子，来加速实现31*31卷积核的运行效率，SLaK验证了卷积核大到61*5时，上述卷积算子运行效率也是非常高的。然而，我们所使用的推理单元可能不止一种平台、一种框架，新引入的优化算子总会在一些情况下造成适用的障碍。

- 同时，大卷积核虽然让CNN达到了“文艺复兴”的效果，相关研究也推动了CNN达到了Transformer相匹敌的能力，但是大卷积核的使用和研究却远没有达到此定位该有的预期。shift-wise可以让大卷积核的使用成本更低。

- 相比与早已成熟应用广泛的depth-wise、point-wise甚至其他的一些其他形式的空间展开，shift-wise的引入无疑是在新的维度上对卷积核的组织形式进行了扩展。

3 本文方法

3.1 本文方法概述

SLaK的基本框架如图2中所示，其特点是使用分解的稀疏的大卷积权重矩阵。我们对其有如下的改进：

进行改进的模块结构如图 4(a)所示，具体地，我们将大卷积核转换成一组普通小卷积核，然后对各卷积结果使用shift操作。我们在本文把这个方法称之为shift-wise算子。

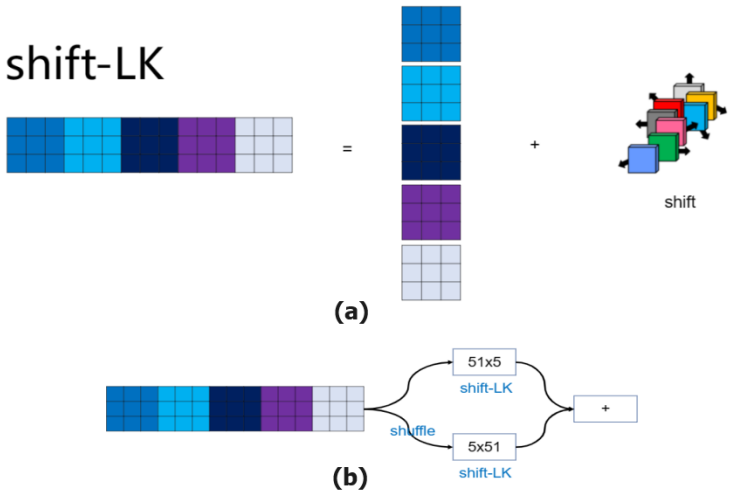

图 4. 方法示意图.(a)将M*N的卷积核分解成k个N*N的卷积核，并使用shift操作完成大卷积核的等价操作(对于SLaK部分stage，M=51,N=5);(b)整体模块结构.

很明显，这样的操作可以使用常规小卷积来做到最新引入的大卷积核的效果，以此达到不使用大卷积而具有相同网络能力的目的。如此，最近对resnet的改进可以有更广泛地应用，也更能发挥CNN相比transformer的优势。并且，我们在结构中加入剪枝的操作，可以进一步降低参数量，达到仅仅使用少量的卷积，即可具备相似的精度。

4 复现细节

4.1 与已有开源代码对比

Sparse Large Kernel Network - SLaK提供了官方pytorch实现，地址为https://github.com/VITA-Group/SLaK。我们在其基础上新增了shift-wise完全等价推理SLaK的模型和相应的辅助代码，包括模型权重加载和重新分配。同时，我们进一步提供了稀疏训练的代码。更多的代码将在后续根据论文进展情况放出。在SLaK基础上，我们主要根据本文所提方法章节编写了相关的代码，具体如下：

- 我们提供了使用常规小卷积来做到最新引入的大卷积核的效果的代码。对应代码为$models/SLaK_reg.py$，可以直接使用在SLaK的checkpoint，并且在Imagenet-1K上的精度是完全一致。
- 我们提供了taichi [15]来实现shift-wise算子。关于shift的操作，Lart [23] 总结过可行的操作，具体的包括: slicing index, torch.roll, deformable convolution, depth-wise convolution, 以及 grid sampling F.grid_sample. 这个算子对应于$models/taichi_kernel.py$。
- 我们还在$models/SLaK_reg.py$基础上做了很多其他的实验，比如双分支参数共享的方法对应于$models/SLaK_pw_wt.py$。具体的可以参考github仓库。

- 我们还实现了相比SLaK参数和计算量都少，但是精度更高的代码。相应代码将在后续陆续放出。

这些代码不是一下子就超越了SLaK的，我们在一步步的验证所提方法的可行性、调优所用到的超参数。我们在精度上超越了SLaK的框架，在推理速度在理论上我们的也具有更好的状态，实际推理速度还在进行中，不过我们也会稍后在结果展示环节展示部分实验结果。这些让我们所提的方法无论在理论上还是在结果上完全与SLaK脱离开来。

4.2 实验环境搭建

我们使用的环境是CUDA 11.7, cudnn 8.2.0, PyTorch 1.10.0，pytorch环境使用conda来维护管理，具体的环境创建过程可以参考github仓库。

4.3 创新点

- RepLK-net [7]虽然引入了自定义的算子，来加速实现31*31卷积核的运行效率，SLaK验证了卷积核大到61*5时，上述卷积算子运行效率也是非常高的。然而，我们所使用的推理单元可能不止一种平台、一种框架，新引入的优化算子总会在一些情况下造成适用的障碍。

- 同时，大卷积核虽然让CNN达到了“文艺复兴”的效果，相关研究也推动了CNN达到了Transformer相匹敌的能力，但是大卷积核的使用和研究却远没有达到此定位该有的预期。shift-wise可以让大卷积核的使用成本更低。

- 相比与早已成熟应用广泛的depth-wise、point-wise甚至其他的一些其他形式的空间展开，shift-wise的引入无疑是在新的维度上对卷积核的组织形式进行了扩展。

- 在CNN网络能力的继续发展上，除了增大卷积核尺寸，如RepLK-net、SLaK所做的一样，我们还提供了一个新的研究方向，并验证了可行性。

5 实验结果分析

5.1 理论计算量降比分析

实验设定如下：SLaK使用$m*n$的depth-wise卷积，两个$m*n$的分支加一个$n*n$的identity同为depth-wise卷积分支；shift-wise conv则使用$n*n$的卷积，再加上相应的shift操作，输入通道为C输出通道为kC的group卷积,$k = ceil(m/n)$。假设输入特征大小为$BCHW$，分别是batch、channel、高和宽，relu都在两个操作的后续层，所以只需考虑卷积即可。

SLaK的depth-wise计算量：

$$m*n*H*W*C*B*2 + Rep(n*n*H*W*C*B + C_{BN})$$

$$C_{slak} = (2m + \tau_n)*n*H*W*C*B \tag{1}$$

加号“+”前后分别为大卷积核和小卷积核部分。因为有merge-bn的存在，可以不用考虑bn层的参数量和计算量。同时，小卷积核部分原则上是可以通过Rep方法在推理时合并进入两个大分支，所以τ_n也可以忽略不记。

SLaK的depth-wise参数量：

$$P_{slak} = m * n * C * 2 + Rep(n * n * C) \tag{2}$$

Shift-wise是1对k的 group conv，其计算量：

$$n * n * \acute{H} * \acute{W} * k * C * B + add(H * W * k * C * 2)$$

$$C_{shift} = (k * n) * n * H * W * C * B + \delta \tag{3}$$

其中，$k = ceil(m/n)$, $\acute{H}$ 和 $\acute{W}$ 是加了padding之后的计算量，根据padding的不同，会比原尺寸大$p - n//2(p \subset [n//2, n-1])$,identity分支是从里面直接拿random的C个feature，另一个大卷积核也是直接拿全部重排后的feature，在计算feature这一步第二个大卷积核分支是没有计算量的。加号后面是所需要的$shift$和add操作，需要选定的k个group通过$shift$之后进行特征相加，所有的C组特征经过相同的操作。考虑到add计算量很低，在统计时可作为忽略不计，暂记作δ。

Shift-wise的参数量：

$$P_{shift} = n * n * k * C \tag{4}$$

这部分只有一组depth-wise卷积，k作为后续shift-add时的组进行操作。如上，我们计算SLaK的计算量C_{slak}，如公式1，和shift-wise的计算量C_{shift}，如公式3，之间的比值,我们暂时忽略掉δ，并且忽略 $\acute{H}$ 和 $\acute{W}$与H 和W的差异，则其**计算量比值**为

$$r_C = C_{slak}/C_{shift} = (2m + {}_n)/(k * n) \approx 2 \tag{5}$$

如我们使用了5个stage的$kernelsize = [51, 49, 45, 13, 5]$，带入$m$和$k$分别计算可得比值[1.85, 1.96, 2.0, 1.73, 2.0]，考虑到δ、$\acute{H}$ 和 $\acute{W}$，该值会有所降低，但是影响较小。

我们计算SLaK的参数量P_{slak}，如公式2，和shift-wise的参数量P_{shift}，如公式4，之间的**参数量比值**比值

$$r_P = P_{slak}/P_{shift} = (2m)/(k * n) \approx 2 \tag{6}$$

参数量和计算量都同比例的降低，约为SLaK的一半，如公式6。

5.2 模块的耗时情况分析

针对SLaK所提的Sparse Large Kernel Block如图2，本文提出了shift-wise的算子进行替换。而shift操作在Sparse Shift Layer [3]报告了耗时较高，如图5所示，其中图(a)和(b)分别展示了Shift操作在CPU上占3.6%的运行时间，在GPU上占28.1%。对于shift操作可行的实现，Lart [23]为shift操作总结了五种可行操作，分别是切片索引、torch.roll 、可变形卷积、 depth-wise卷积和网格采样 F.grid_sample。这些操作计算成本都比较高，我们使用taichi [15]重写了对应的算子。

其次，为了实现模块的替换，我们使用了如图 4(b)所示结构进行实现，这样可以进一步降低计算量。

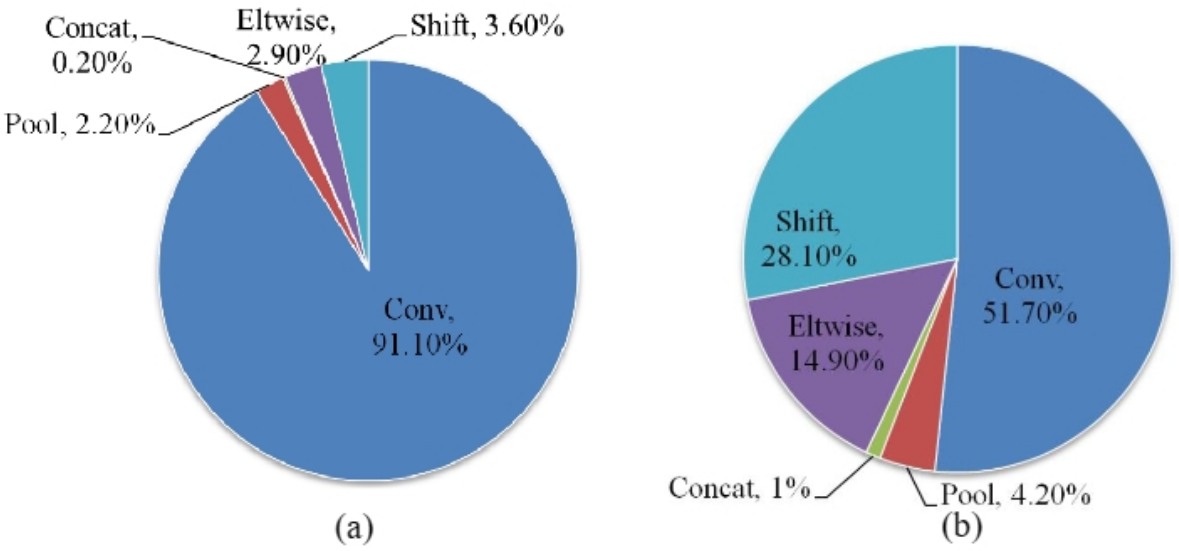

图 5. Sparse Shift Layer测试的运行时分析。为了进行清楚的比较，BN层和ReLU层都被忽略了，因为它们可以合并到卷积层中进行推理。这里也没有考虑数据加载和预处理时间。结果是在Caffe下用batchsize=32获得的，跑100次取平均值。(a)在 CPU上的ShiftNet-A [39]耗时(Intel Xeon E5-2650，atlas)。(b) GPU上的ShiftNet-A (TITAN X Pascal, CUDA8 and cuDNN5) 。

SLaK沿用了ConvNeXt的优化后的ResNet结构，对于4个stage使用的block数量在不同网络中是不同的， SLaK-T 结构的block数量是[3, 3, 9, 3]， 而SLaK-S/B是 [3, 3, 27, 3] 。对于每个Resnet模块，BN层直接跟在卷积后。同时，SLaK中还大量使用了大卷积核，卷积核的短边是5，长边在各个stage分别是[51 49 47 13]。stem模块使用了4x4的卷积，步长是4。输入图形是224*224，经过stem layer后4倍下采样特征大小变为56*56。第一个stage的channel数为128。为了我们假设batch-size为90，对于一个90*128*56*56的输入，我们分别对比如下模组的耗时情况， DepthWiseConv2dImplicitGEMMs是RepLK-net [7]引入的大卷积核优化后的算子，这里用来处理51*5的卷积；depthwise是用pytorch的去实现卷积核是51*5的情况；后续的实现都需要先用pytorch的group卷积，所以我们也统计group卷积的耗时情况，注意此项的输出是中间变量。 ConvGroupShift是group卷积基础上，再加上后续的padding完成的shift操作； ConvGroupShiftTaichi是group卷积基础，使用taichi完成shift操作。各个算子的时间消耗情况如表1所示，时间统计由torch.autograd.profiler.profile所完成，DepthWiseConv2dImplicitGEMMs可能因为统计时间方式而略有偏差。总体来看，group conv耗时25ms，taichi版本的shift操作在7ms左右。pytorch depthwise 的时间消耗约为ConvGroupShiftTaichi的一半。而考虑到SLaK需要两个分支，其时间消耗和当前ConvGroupShiftTaichi将**基本持平** 。

5.3 稀疏分析

为了提高精度降低计算量，我们还加入了稀疏训练，可以对最终精度有略微提升。虽然SLaK中，也使用了稀疏矩阵的训练，但是他们是细粒度稀疏，不能降低推理时的计算量。我们使用粗粒度的剪枝方法，可以达到更少计算量即可具有相似精度的效果。

表 1. 各算子的耗时情况。 DepthWiseConv2dImplicitGEMMs为cuda所实现； group conv的ConvGroupShift、ConvGroupShiftTaichi的中间模块，不作为独立的算子

name	forward(ms)	backward(ms)
DepthWiseConv2dImplicitGEMMs	20.727ms	40.635ms
pytorch depthwise	15.449ms	24.791ms
group conv	25.028ms	34.408ms
ConvGroupShift	40.281ms	49.108ms
ConvGroupShiftTaichi	31.931ms	46.810ms

表 2. 使用一定数量的常规卷积，可达到SLaK中所使用的depthwise卷积相同的精度效果，此数字对应到groups一列；通过稀疏训练后，可以使用更少的卷积达到相同的效果，降低数量后的卷积数量分布如group后0-11所示，即代表了使用的卷积数量情况。

layer	density	group	0	1	2	3	4	5	6	7	8	9	10	11	channels
stages.0.0	0.2339	11	13	29	20	22	25	7	7	1	0	0	0	0	124
stages.1.0	0.6012	10	0	12	22	19	16	26	25	44	44	30	11		249
stages.2.0	0.5925	9	2	5	15	56	72	99	126	76	45	3			499
stages.3.0	0.9225	3	0	20	192	786									998

具体地，如表格2所示，SLaK中使用了depth-wise卷积，我们使用小的常规卷积核时，需要使用非常多的数量，才能在精度上达到等同的效果，各个stage有所差别，具体如表格中groups一列所示。但是，当我们使用稀疏训练之后，我们发现，不必让每一个depth-wise使用如此多数量的卷积也可以。同时，为了保持特征的多样性，使用的卷积的数量也并没有统一到一个具体的数值，而是多种组合形态。我们发现使用少量的卷积即可达到一样精度的效果。

为了更好的表示这种组合形态，我们将stages.0.0单独拿出来查看它的组合数量的分布情况。结果如图6所示。我们可以直观的从图中看到，本应用11个常规卷积进行等价的卷积操作，稀疏训练后，大部分只需要少量的卷积即可。

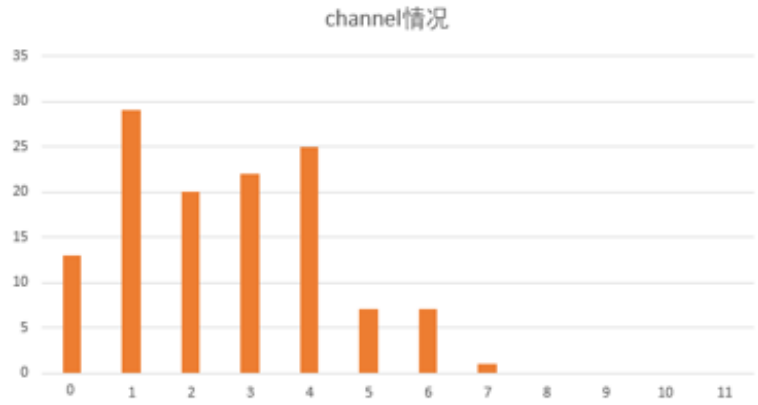

图 6. 稀疏训练后stages.0.0达到原有精度，需要多种数量的卷积进行shift操作，据此展示具体的数量分布情况的直方图

5.4 模块的精度分析

SLaK-T在imagenet-1k上的训练120epoches精度为**81.6**，如果使用使用稀疏同时两个分支共享基础卷积的方式来处理，120epoches的精度是**82.05**。进一步增加训练批次，我们期待会有比SLaK更大的提升。因为重参数化并没有增加推理时的时间消耗，是CNN研究的一个重要的技术，我们甚至也期望这项技术可以让我们有相比SLaK更大的提升空间。对于SLaK等单纯的提升卷积的尺寸的方法，我们的方法**从稀疏依赖的角度继续提升了CNN的精度**。
这为CNN的baseline的提升，贡献了新的研究角度。

5.5 模块的推理速度实验

表 3. 不同算子的耗时比较在相同大小输入特征时。depth-wise 卷积是大卷积核的一个实现方式，所提出的shift-wise算子实现借用了多个group卷积作为中间过程，其中group卷积的个数由density所决定。group卷积是模仿没有粗粒度稀疏训练的效果。其中density是模仿网络的stage0的稀疏度进行设置的。(ci,co,ks,g)分别对应输入通道数，输出通道数，卷积核尺寸，卷积分组情况。HW代表了输入特征的尺寸

name	weight: (ci,co,ks,g)	input: HW	density	CUDA time
depth-wise conv	(124,124,51*5,124	(128,128)	0.25	11.850ms
group-conv	(124, 124*11, 5*5,124)	(128,128)	0.25	20.974ms
shift-wise	(124,124,5*5,124)	(128,128)	0.25	**7.018ms**

我们也测试了shift-wise模块在单分支时的时间消耗，如表格3所示。结果显示所提算子可以在一定稀疏度的情况下降低时间消耗。考虑到SLaK使用了双分支网络，而shift-wise模组使用了相比SLaK一半的卷积，最终实现对推理时间消耗的降低会更加明显。

6 总结与展望

所提出的算子已经在计算效率提高的情况下，超越了SLaK的精度。我们所提算子虽然基于SLaK这种提升卷积核尺寸的研究，但是无论受制于算力易用性，还是卷积特征的尺寸的限制，我们认为这种研究方向都是不可持续的，我们认真探究了稀疏依赖，并将这一transformer的特征成功赋给了CNN。虽然如此，我们还有一些事情需要突破：1）进一步的实验，达到与SLaK、UniRepLKNet等全面对比的地步；2）继续扩展taichi的代码，让相应的模块优化程度进一步的提升。目前我们所提的方法还有一些实验等待着验证，需要进一步的评估其效果。特别是在下游任务的泛化性上，我们期待这可以为CNN基础研究贡献一些力量。

参考文献

[1] Reza Azad, Leon Niggemeier, Michael Huttemann, Amirhossein Kazerouni, Ehsan Khodapanah Aghdam, Yury Velichko, Ulas Bagci, and Dorit Merhof. Beyond self-attention: Deformable large kernel attention for medical image segmentation, 2023.

[2] L. C. Chen, G Papandreou, I Kokkinos, K Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2018.

[3] Weijie Chen, Di Xie, Yuan Zhang, and Shiliang Pu. All you need is a few shifts: Designing efficient convolutional neural networks for image classification. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7234–7243, 2019.

[4] Xuanhong Chen, Hang Wang, and Bingbing Ni. X-volution: On the unification of convolution and self-attention, 2021.

[5] Ke Cheng, Yifan Zhang, Xiangyu He, Weihan Chen, Jian Cheng, and Hanqing Lu. Skeleton-based action recognition with shift graph convolutional network. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 180–189, 2020.

[6] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks, 2017.

[7] Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11963–11975, 2022.

[8] Xiaohan Ding, Yiyuan Zhang, Yixiao Ge, Sijie Zhao, Lin Song, Xiangyu Yue, and Ying Shan. Unireplknet: A universal perception large-kernel convnet for audio, video, point cloud, time-series and image recognition, 2023.

[9] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation, 2019.

[10] Meng-Hao Guo, Cheng-Ze Lu, Qibin Hou, Zhengning Liu, Ming-Ming Cheng, and Shi-Min Hu. Segnext: Rethinking convolutional attention design for semantic segmentation, 2022.

[11] Meng-Hao Guo, Cheng-Ze Lu, Zheng-Ning Liu, Ming-Ming Cheng, and Shi-Min Hu. Visual attention network. *Computational Visual Media*, 9(4):733–752, 2023.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[13] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks, 2018.

[14] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks, 2019.

[15] Yuanming Hu, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Transactions on Graphics (TOG)*, 38(6):201, 2019.

[16] Tianjin Huang, Lu Yin, Zhenyu Zhang, Li Shen, Meng Fang, Mykola Pechenizkiy, Zhangyang Wang, and Shiwei Liu. Are large kernels better teachers than transformers for convnets? *arXiv preprint arXiv:2305.19412*, 2023.

[17] Yunho Jeon and Junmo Kim. Constructing fast network through deconstruction of convolution, 2018.

[18] Kaiming, He, Xiangyu, Zhang, Shaoqing, Ren, Jian, and Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.

[19] Ismail Khalfaoui-Hassani, Thomas Pellegrini, and Timothée Masquelier. Dilated convolution with learnable spacings. *arXiv preprint arXiv:2112.03740*, 2021.

[20] Alexandre Kirchmeyer and Jia Deng. Convolutional networks with oriented 1d kernels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6222–6232, 2023.

[21] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering, 2020.

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[23] Lart. Spatial-shift-operation的5种实现策略. https://www.yuque.com/lart/ugkv9f/nnor5p. 2022-05-18 12:34.

[24] Kin Wai Lau, Lai-Man Po, and Yasar Abbas Ur Rehman. Large separable kernel attention: Rethinking the large kernel attention design in cnn. *Expert Systems with Applications*, 236:121352, 2024.

[25] Yanghao Li, Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. Scale-aware trident networks for object detection. *CoRR*, abs/1901.01892, 2019.

[26] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers, 2021.

[27] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnns for fine-grained visual recognition, 2017.

[28] Shiwei Liu, Tianlong Chen, Xiaohan Chen, Xuxi Chen, Qiao Xiao, Boqian Wu, Tommi Kärkkäinen, Mykola Pechenizkiy, Decebal Mocanu, and Zhangyang Wang. More convnets in the 2020s: Scaling up kernels beyond 51x51 using sparsity. *arXiv preprint arXiv:2207.03620*, 2022.

[29] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022.

[30] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation, 2015.

[31] Xuran Pan, Chunjiang Ge, Rui Lu, Shiji Song, Guanfu Chen, Zeyi Huang, and Gao Huang. On the integration of self-attention and convolution, 2022.

[32] Chao Peng, Xiangyu Zhang, Gang Yu, Guiming Luo, and Jian Sun. Large kernel matters–improve semantic segmentation by global convolutional network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4353–4361, 2017.

[33] Yaolei Qi, Yuting He, Xiaoming Qi, Yuan Zhang, and Guanyu Yang. Dynamic snake convolution based on topological geometric constraints for tubular structure segmentation, 2023.

[34] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[35] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[36] Towaki Takikawa, David Acuna, Varun Jampani, and Sanja Fidler. Gated-scnn: Gated shape cnns for semantic segmentation, 2019.

[37] Wenhai Wang, Jifeng Dai, Zhe Chen, Zhenhang Huang, Zhiqi Li, Xizhou Zhu, Xiaowei Hu, Tong Lu, Lewei Lu, Hongsheng Li, Xiaogang Wang, and Yu Qiao. Internimage: Exploring large-scale vision foundation models with deformable convolutions, 2023.

[38] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks, 2018.

[39] Bichen Wu, Alvin Wan, Xiangyu Yue, Peter Jin, Sicheng Zhao, Noah Golmant, Amir Gholaminejad, Joseph Gonzalez, and Kurt Keutzer. Shift: A zero flop, zero parameter alternative to spatial convolutions, 2017.

[40] Ruihan Xu, Haokui Zhang, Wenze Hu, Shiliang Zhang, and Xiaoyu Wang. Parcnetv2: Oversized kernel with enhanced attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5752–5762, 2023.

[41] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. *IEEE Computer Society*, 2016.