Optimal DNN Primitive Selection with Partitioned Boolean Quadratic Programming

Andrew Anderson, David Gregg

ACM CGO Conference

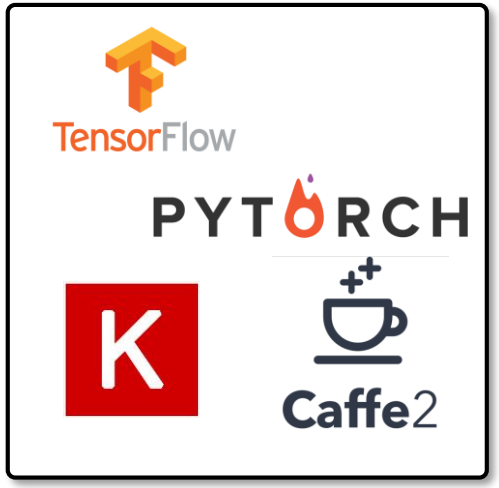
2019-01-29

목차

- **Background**
  - Acceleration primitives, Convolution shapes, Compatibility
- **Key contribution**
  - PBQP based cost selection
- **Performance**
- **Conclusion**

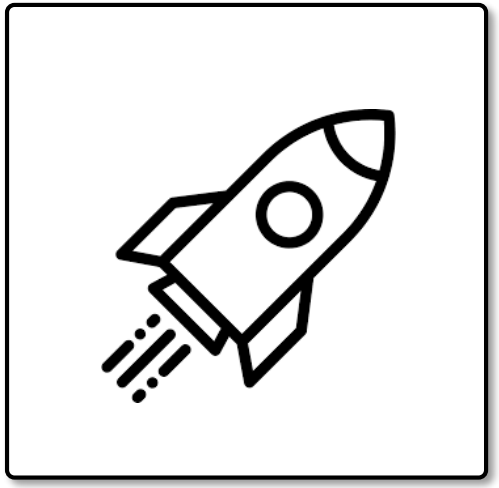
Background

- **딥러닝 가속의 다양한 요소들**
  - Deep learning framework
  - Target device
  - Acceleration primitives



Framework

Target device



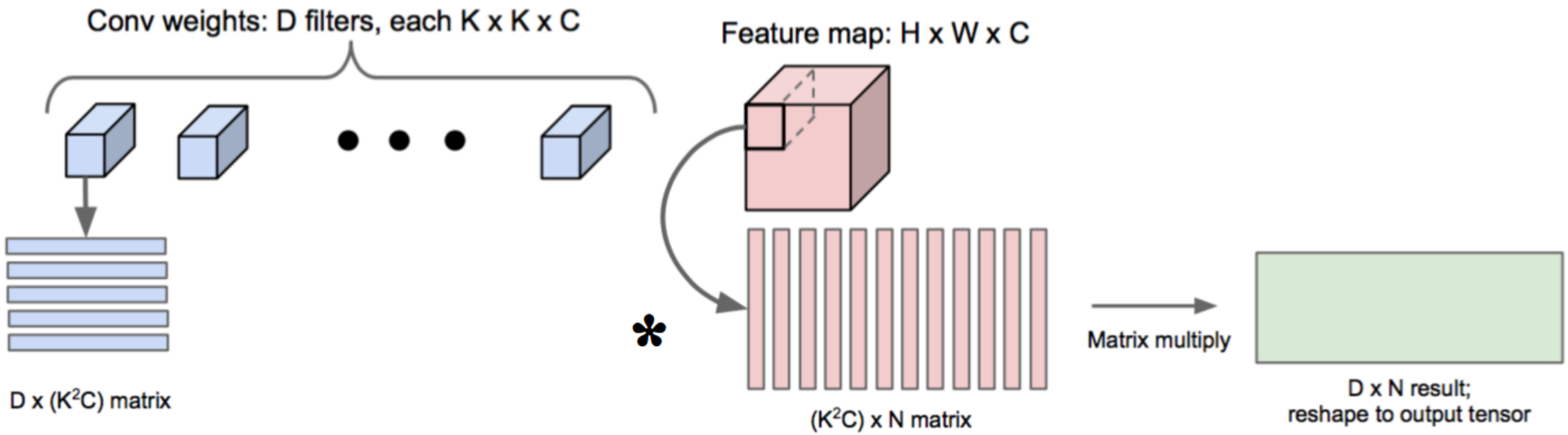
Acceleration primitive

Acceleration primitives

- **Multi channel and Multi kernel (MCMK)**
- **General matrix multiplication (GEMM)**
  - Image to column/row, Kernel to column/row
- **Fast fourier transform (FFT)**
- **Strassen, Winograd**
  - Reduced multiplication, Increased addition

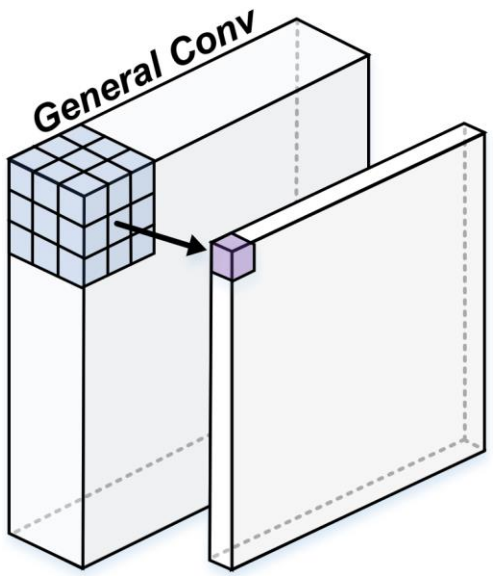
GEMM based primitive

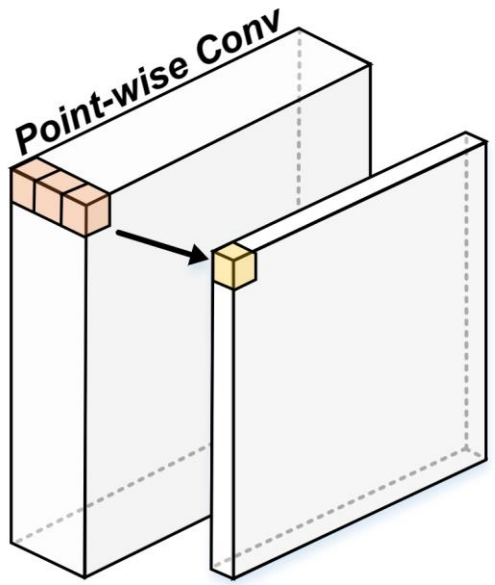
- **Image to column (Im2col)**
  - Transform, GEMM 순으로 동작

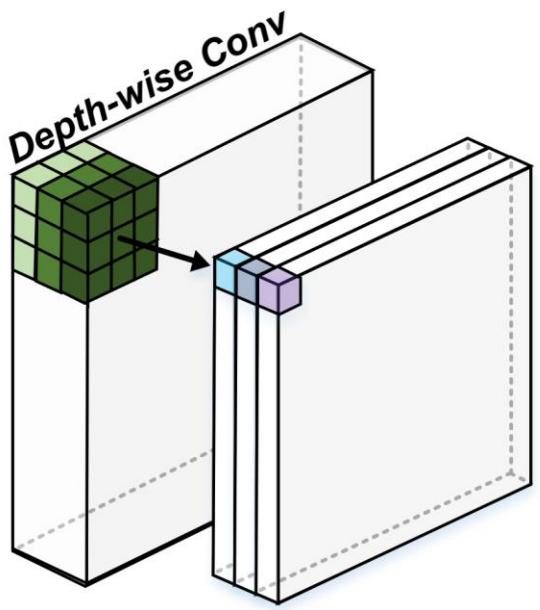


Convolution shapes

- **General**
- **Point, Depth-wise**

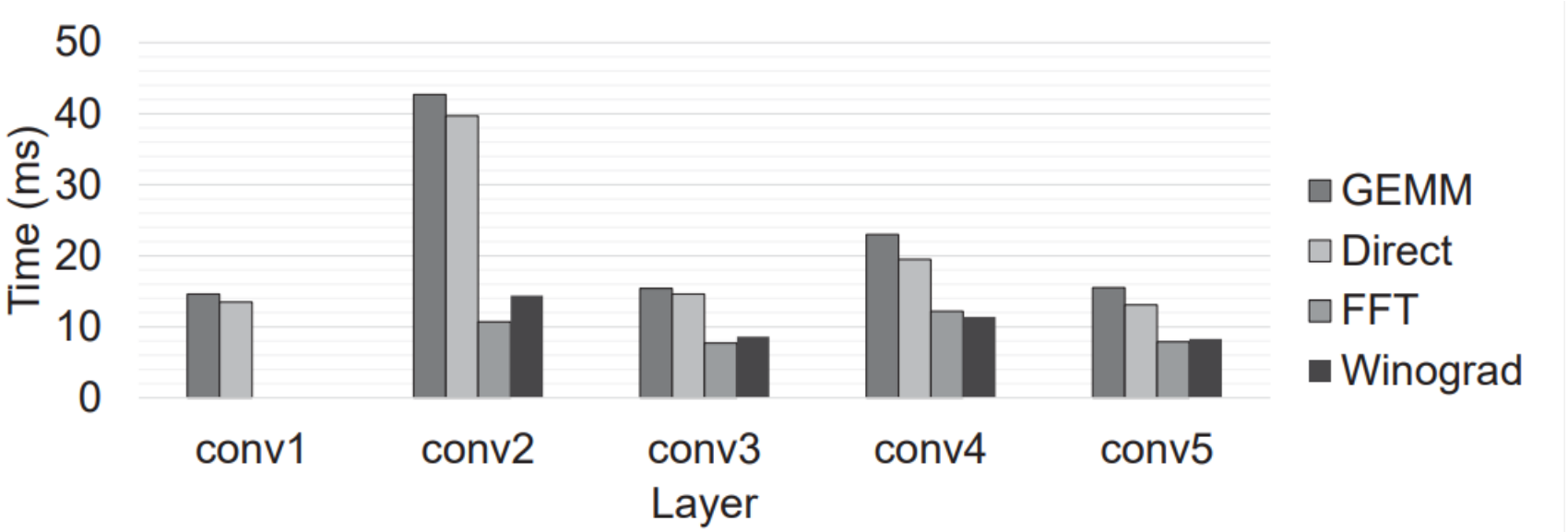






Compatibility between primitives and shapes

- **각 Shape에 적합한 primitive가 다름[1]**
  - 디바이스에 따라서 가장 적합한 primitive가 결정됨
  - cuDNN에는 실행될 때 마다 가장 적합한 primitive를 heuristic하게 결정함

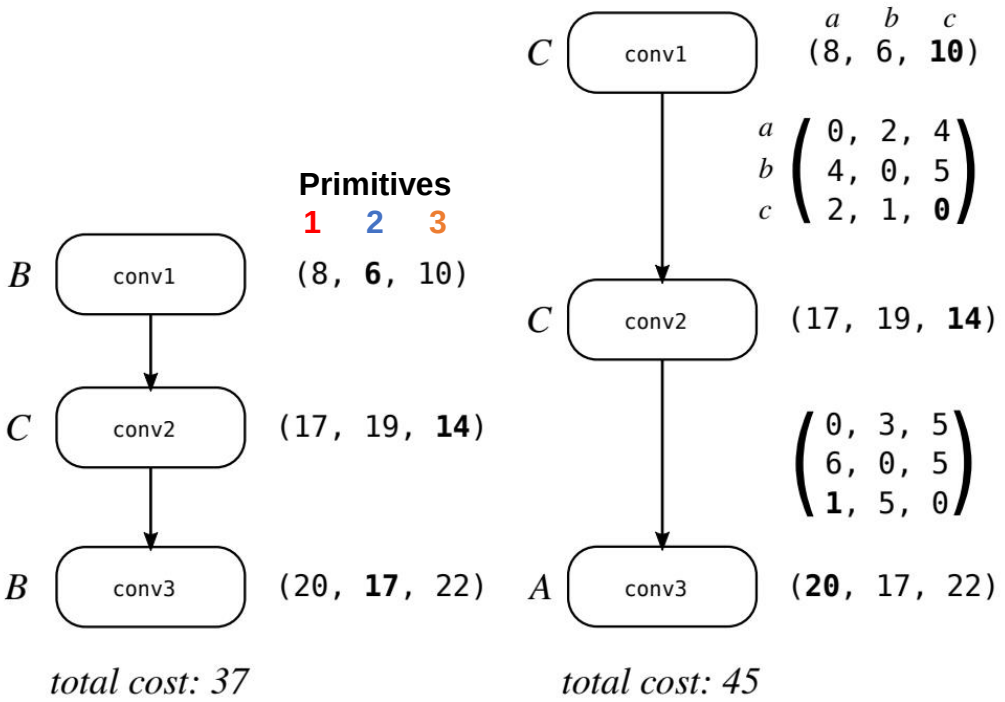


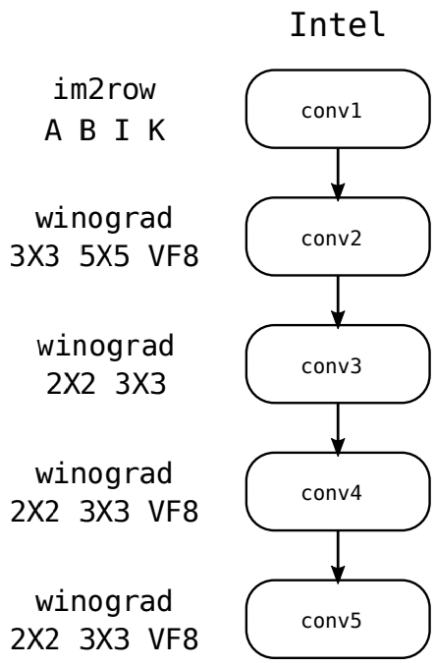
(작을 수록 좋음)

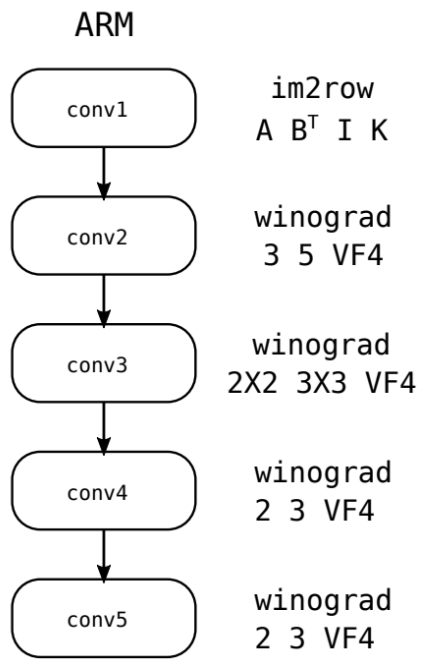
[1] Performance Analysis of CNN Frameworks for GPUs, J. Lee, et.al., *IEEE ISPASS*, 2017

Key contribution

- **가장 cost (실행시간)가 작은 primitive를 선택**
  - 가장 cost가 작은 primitive내에서 GEMM의 방법을 고려함
  - **미리 계산된 cost**를 사용하며, cost는 **C, H, W, δ, K, M**에 영향 받음
  - Partitioned Boolean Quadratic Programming (PBQP) 방법을 사용







Performance

Network	SUM2D	L.OPT	PBQP	CAFFE
(S) AlexNet	711.75	231.75	100	419.565
(S) GoogleNet	1401	465.25	249	1267.07
(M) AlexNet	712.25	186	44.25	286.518
(M) GoogleNet	1400.25	261.5	123.5	919.196

- **Execution time on intel processor**
  - PBQP running on caffe

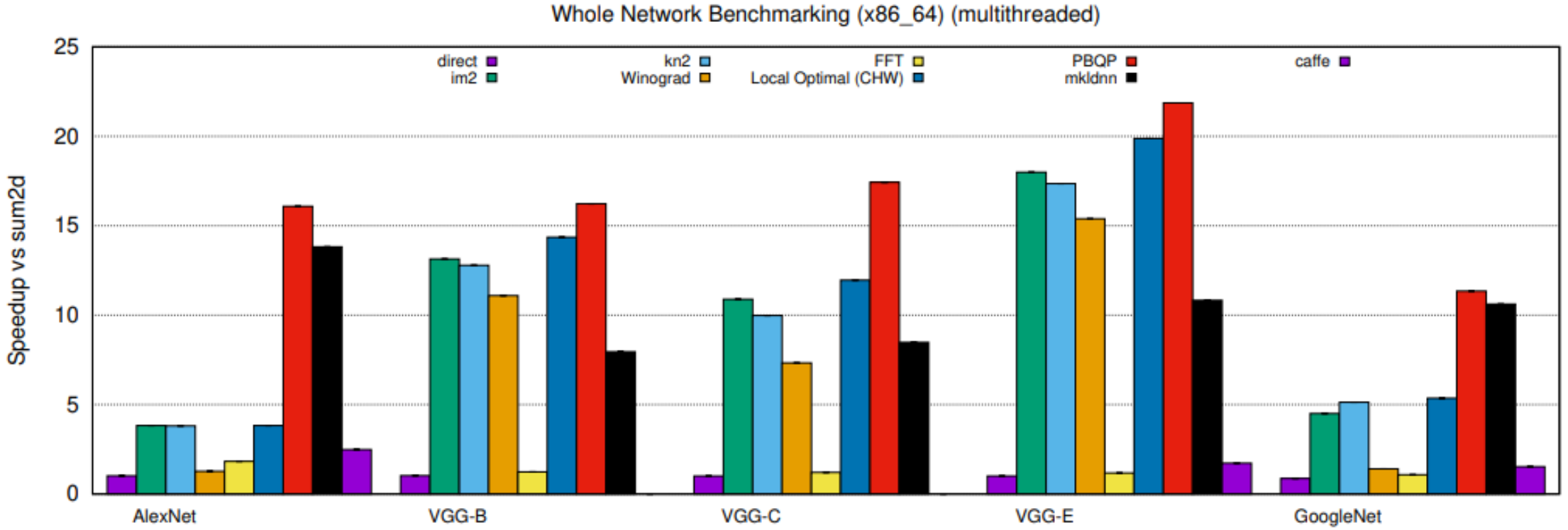
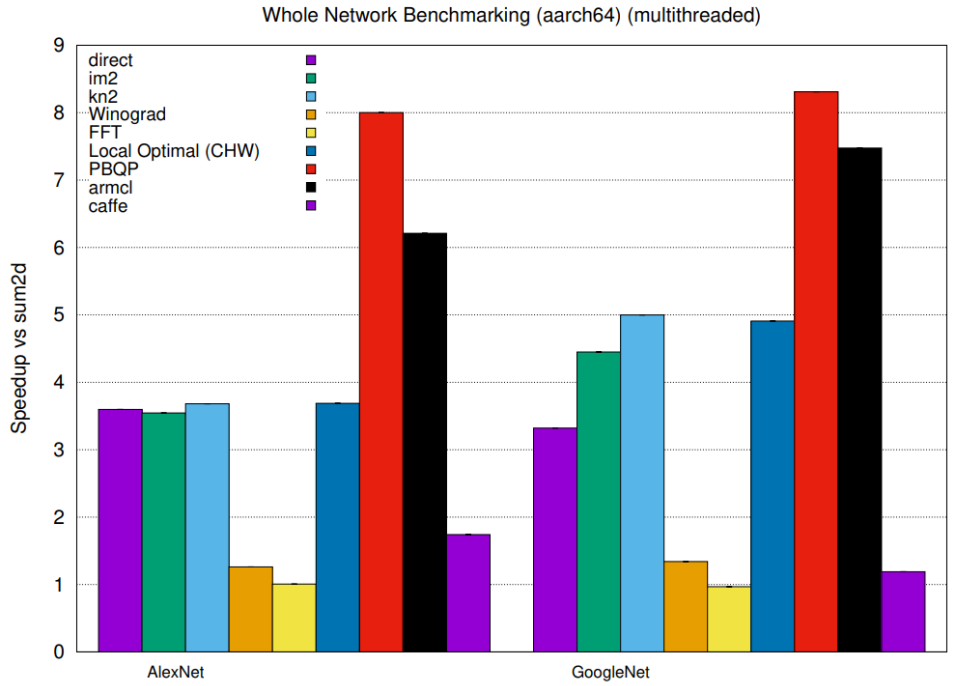


Figure 6. Multi-Threaded Comparison of approaches with PBQP selection on **Intel Haswell**

Performance

- **Execution time on arm processor**
  - PBQP running on caffe

Network	SUM2D	L.OPT	PBQP	CAFFE
(S) AlexNet	2369.5	744.25	461	2341.09
(S) GoogleNet	4544.75	1695.25	1025	5782.4
(M) AlexNet	2432.5	639.25	294	1342.62
(M) GoogleNet	4509.75	919.25	547.5	3707.91



(b) Multi-Threaded Comparison of approaches with PBQP selection on **ARM Cortex-A57**

Conclusion

- **cuDNN을 맹신하지 말라**
  - cuDNN은 컨볼루션이 계산되는 과정에서 가장 최적의 primitive를 찾음
  - 만약에 실행시간이 이상하다면, primitive를 확인해 볼 것 !

4.78. cudnnGetConvolutionBackwardDataAlgorithm

```
cudnnStatus_t cudnnGetConvolutionBackwardDataAlgorithm(
    cudnnHandle_t                       handle,
    const cudnnFilterDescriptor_t       wDesc,
    const cudnnTensorDescriptor_t       dyDesc,
    const cudnnConvolutionDescriptor_t  convDesc,
    const cudnnTensorDescriptor_t       dxDesc,
    cudnnConvolutionBwdDataPreference_t preference,
    size_t                              memoryLimitInByte
    cudnnConvolutionBwdDataAlgo_t      *algo)
```

This function serves as a heuristic for obtaining the best suited algorithm f
given layer specifications. Based on the input preference, this function wi
fastest algorithm within a given memory limit. For an exhaustive search fo
`cudnnFindConvolutionBackwardDataAlgorithm`.

For the following terms, the short-form versions shown in the paranthesis are used in the table below, for brevity:

- CUDNN_CONVOLUTION_FWD_ALGO_IMPLICIT_GEMM (**_IMPLICIT_GEMM**)
- CUDNN_CONVOLUTION_FWD_ALGO_IMPLICIT_PRECOMP_GEMM (**_IMPLICIT_PRECOMP_GEMM**)
- CUDNN_CONVOLUTION_FWD_ALGO_GEMM (**_GEMM**)
- CUDNN_CONVOLUTION_FWD_ALGO_DIRECT (**_DIRECT**)
- CUDNN_CONVOLUTION_FWD_ALGO_FFT (**_FFT**)
- CUDNN_CONVOLUTION_FWD_ALGO_FFT_TILING (**_FFT_TILING**)
- CUDNN_CONVOLUTION_FWD_ALGO_WINOGRAD (**_WINOGRAD**)
- CUDNN_CONVOLUTION_FWD_ALGO_WINOGRAD_NONFUSED (**_WINOGRAD_NONFUSED**)
- CUDNN_TENSOR_NCHW (**_NCHW**)
- CUDNN_TENSOR_NHWC (**_NHWC**)
- CUDNN_TENSOR_NCHW_VECT_C (**_NCHW_VECT_C**)