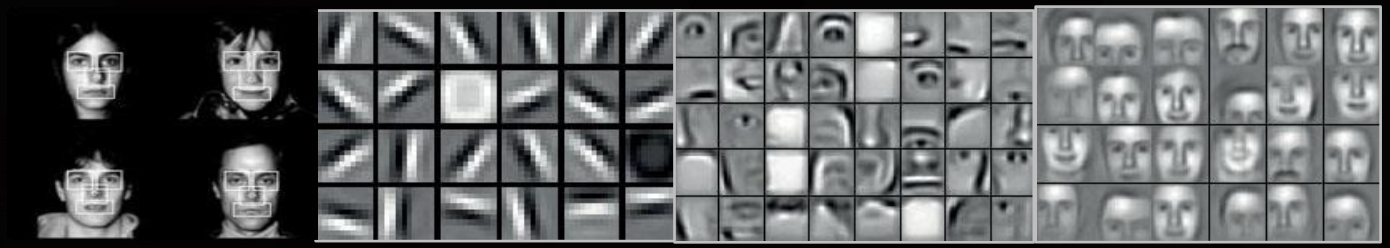
GPU TECHNOLOGY CONFERENCE

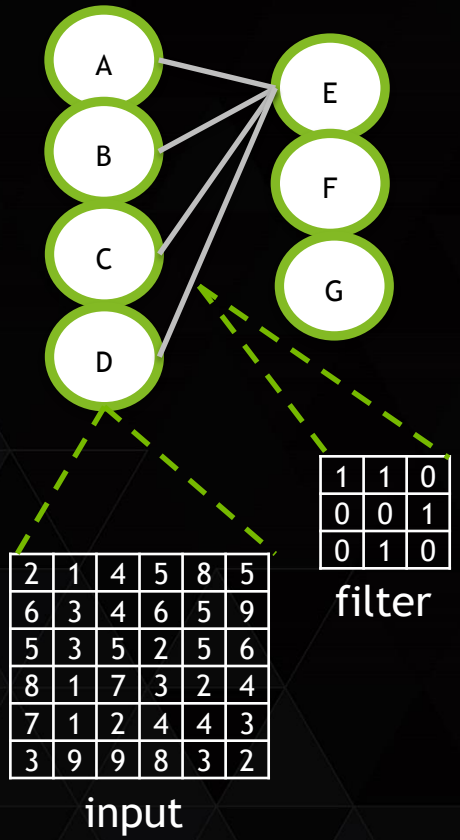
DIRECT CONVOLUTION FOR DEEP NEURAL NETWORK CLASSIFICATION ON TEGRA X1

ALAN WANG, NVIDIA

Convolutional Neural Network



Convolutional Layer



An example:

Parameter	Description	Value in this case
C	#input channels	4
H,W	Input feature map size	6x6
K	#output channels	3
U,V	Output stride	1,1
R,S	Filter size	3x3

4 Input feature maps, and 3 output feature maps, so 12 different filters

E = A convolve with filters[A][E]
+ B convolve with filters[B][E]
+ C convolve with filters[C][E]
+ D convolve with filters[D][E]

Total input pixel = C * H * W = 4 * 15 * 15

Total coefficients = K * C * R * S = 3 * 4 * 3 * 3

Total math = K * C * H * W * R * S = 3 * 4 * 15 * 15 * 3 * 3

Analysis of Overfeat

#1 Analysis of Math/Memory ratio

Total Math = K*C*P*Q*R*S

Total Memory = C*H*W + K*C*R*S + K*P*Q

Overfeat						
Layer	1	2	3	4	5	6
Input Channels	3	96	256	512	512	1024
Output Channels	96	256	512	512	1024	1024
Filter size	7x7	7x7	3x3	3x3	3x3	3x3
Padding size	/	/	1x1	1x1	1x1	1x1
input size	221x221	36x36	15x15	15x15	15x15	15x15
Math/Memory (FFMA/Byte)	**31.8**	**173.8**	**48.5**	**50.6**	**52.1**	**53.3**

Big space to explore on GPU to make it math throughput limited!

#2 Analysis of Layer configuration variety

Overfeat							
Layer	1	2	3	4	5	6	Range
Input Channels	3	96	256	512	512	1024	3 ~ 1024
Output Channels	96	256	512	512	1024	1024	96 ~1024
Filter size	7x7	7x7	3x3	3x3	3x3	3x3	3~7
Padding size	/	/	1x1	1x1	1x1	1x1	0~1
input size	221x221	36x36	15x15	15x15	15x15	15x15	221~15

The implementation should not rely on the assumption of a particular configuration

#3 Analysis of Input/Coefficient ratio

Total inputs = C * H * W

Total coefficients = K * C * R * S

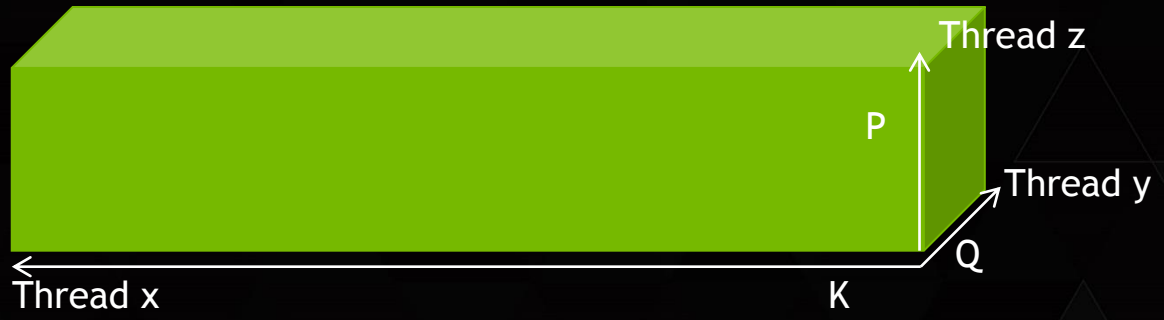
Overfeat						
Layer	1	2	3	4	5	6
Input Channels	3	96	256	512	512	1024
Output Channels	96	256	512	512	1024	1024
Filter size	7x7	7x7	3x3	3x3	3x3	3x3
Padding size	/	/	1x1	1x1	1x1	1x1
input size	221x221	36x36	15x15	15x15	15x15	15x15
Input/Coefficient	**10.383**	**0.103**	**0.063**	**0.063**	**0.031**	**0.027**

Coefficients dominate in most layers except the first few

The direct convolution prototype on Tegra X1

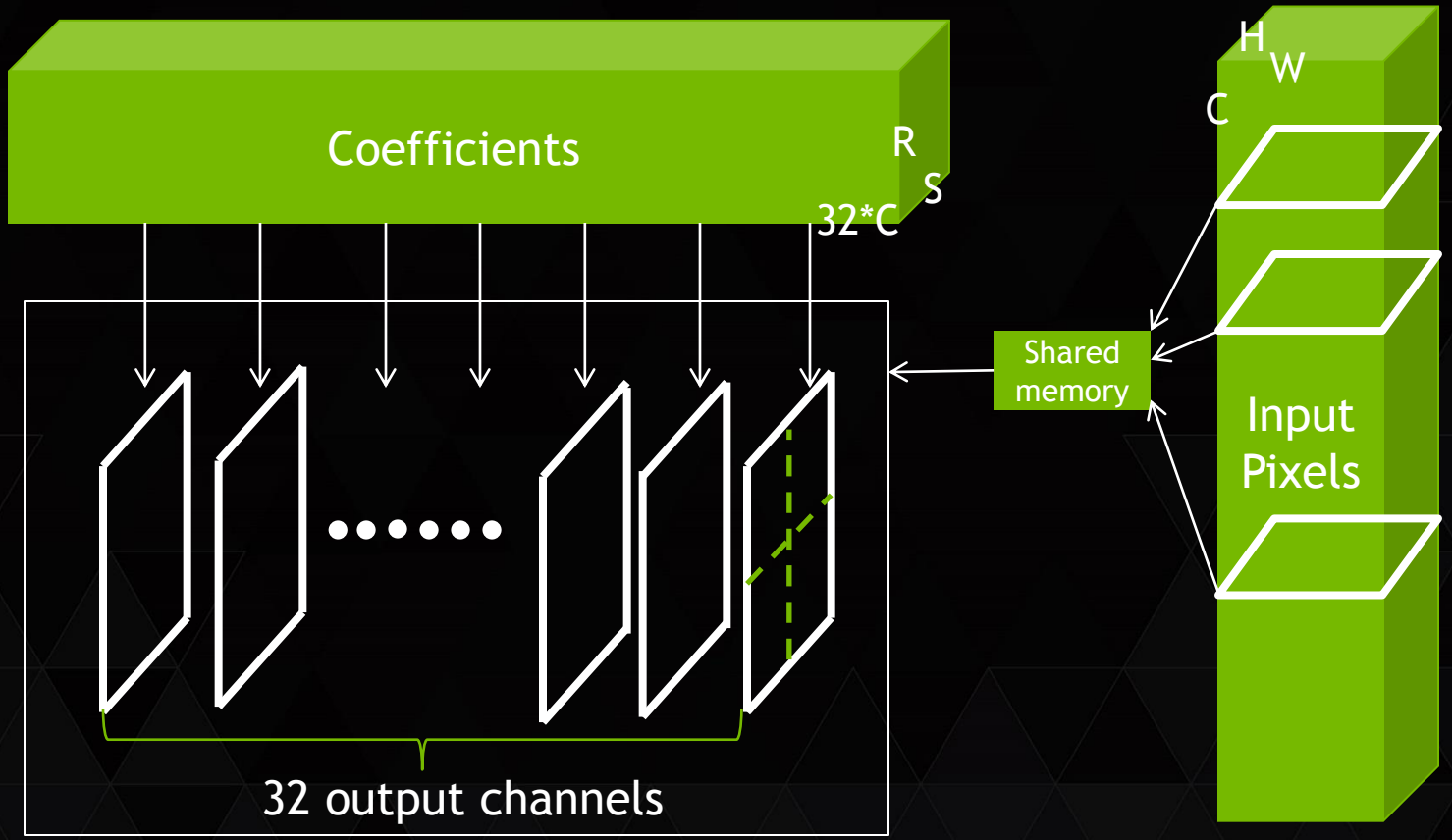
Workload Distribution

Distribute the workload by output space:



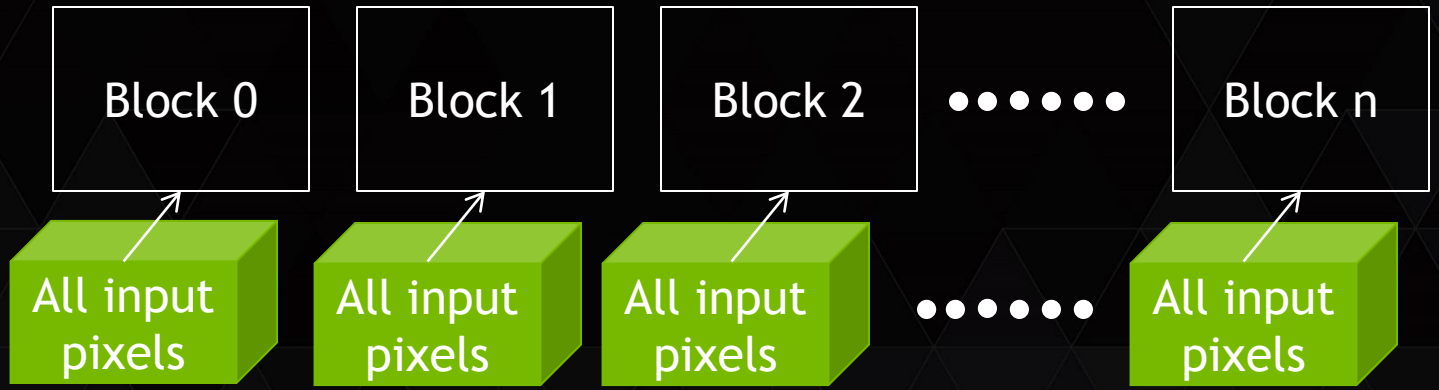
per CUDA block:

Data Reuse



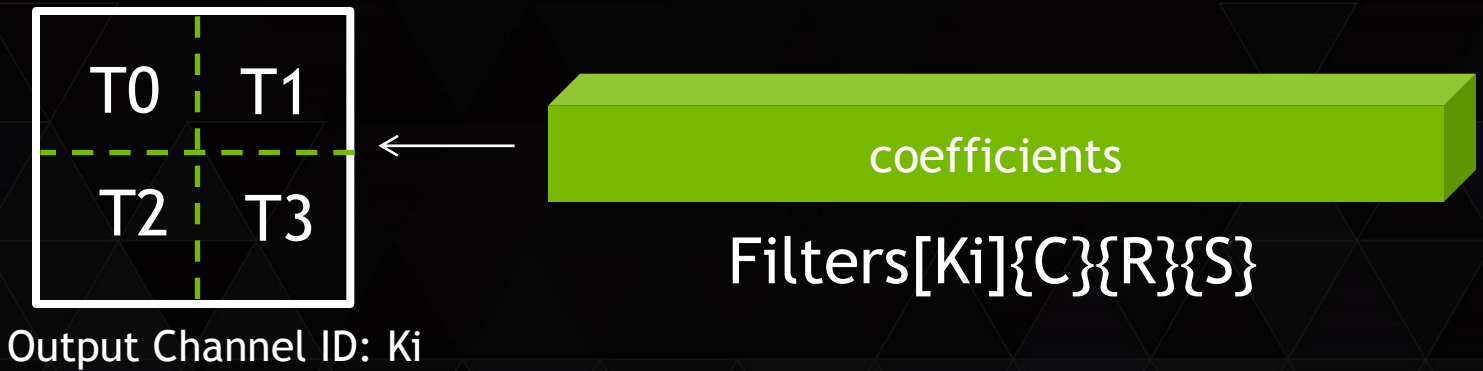
Data Reuse: Input pixels

- Every CUDA block need to fetch the entire input pixel space.
  - Inter-block redundant load handled by cache
  - Inner-block redundant load handled by shared memory
- Total Load ~ K/32 * C * H * W



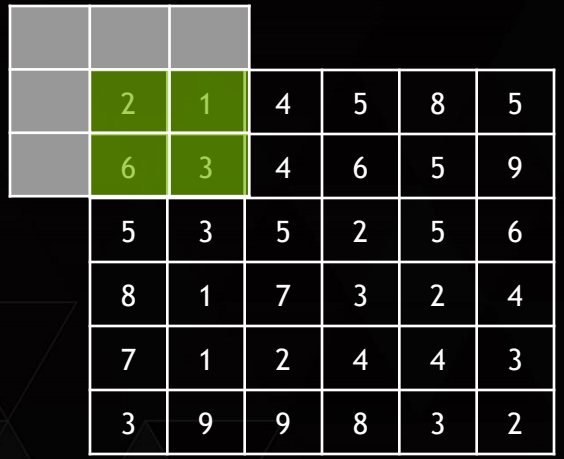
Data Reuse: Coefficients

- No data reuse between blocks
- Inner-block reuse only occurs between the threads coming from the same output channel, handled by cache.



Input Pixels Layout

- Use 3D texture to elegantly handle out-of-bound access in every input channel.



out-of-bound access

inner-bound access

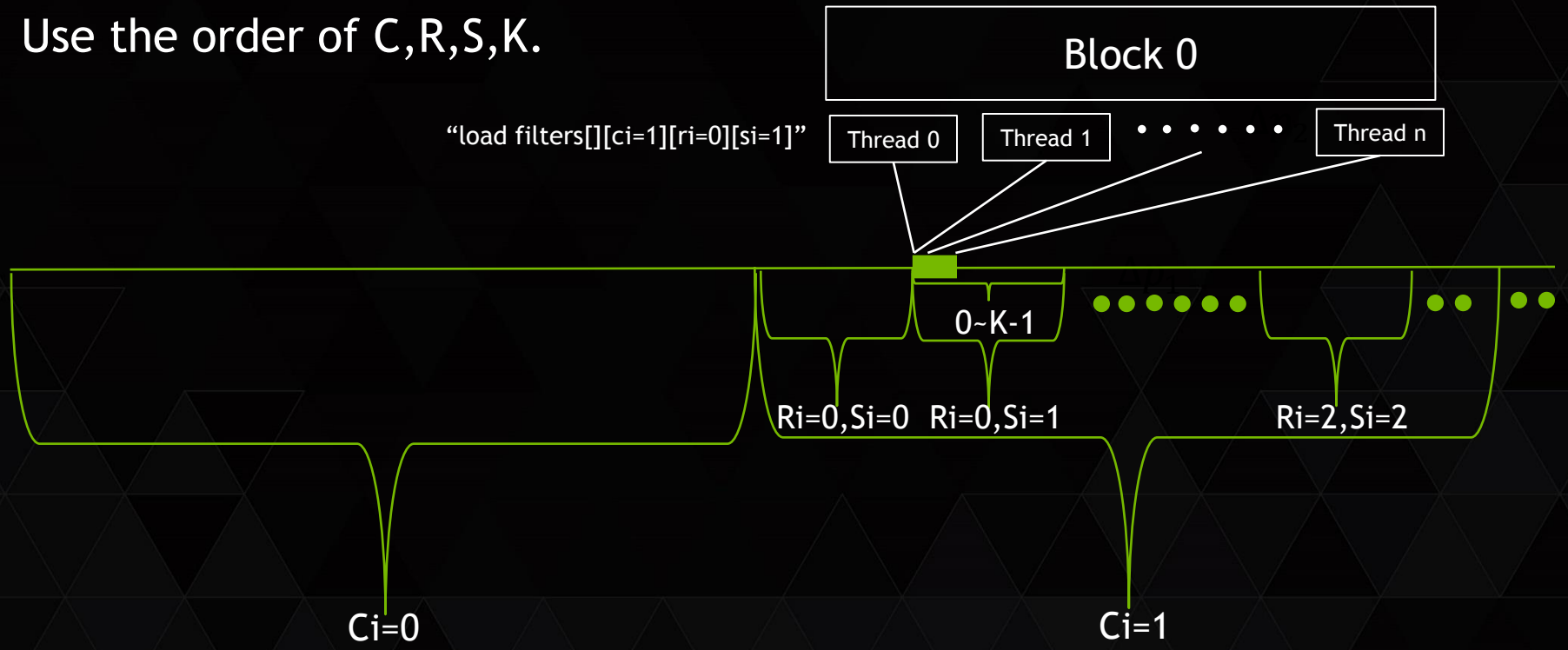
Return data modes when out-of-bound access occurs:

- cudaAddressModeWrap
- cudaAddressModeClamp
- cudaAddressModeMirror
- cudaAddressModeBorder

- Mapping to texture
  - tex3D<float>(textureObject,W,H,C)

Coefficients Layout

- To keep the load request coalesced.
- Use the order of C,R,S,K.



Per-Thread pseudo code

```
for ci
    Load input footprint to shared memory (pixBuffer);
    __syncthreads();
    load 1 coefficient to cbuffer_front;
    for ri
        for si
            load 1 coefficient to cbuffer_back;
                for TILE_X
                    for TILE_Y
                        outputBuffer[] += pixBuffer[] * cbuffer_front;
            switch cbuffer_front <-> cbuffer_back;
Write the outputBuffer to global memory;
```

Performance

- Test layer: Overfeat 6

input channels	output channels	input size	filter size	padding	stride
1024	1024	15x15	3x3	1x1	1x1

- Algorithm configuration

Tile Size	Coefficients Layout	Input Layout	Output Layout
5x8	C,R,S,K	3D texture(W,H,C)	K,P,Q

- Test platform : Tegra X1
- Current performance: GFLOPs Utilization ~ **75%**

Summary

- Proto-type a direct convolution implementation to accelerate the convolutional layer of DNN classification
- Analyze the optimization technique
- Achieve high GFLOPs utilization on Tegra X1, currently 75%, continuing optimization

THANK YOU

alanw@nvidia.com

JOIN THE CONVERSATION

#GTC15