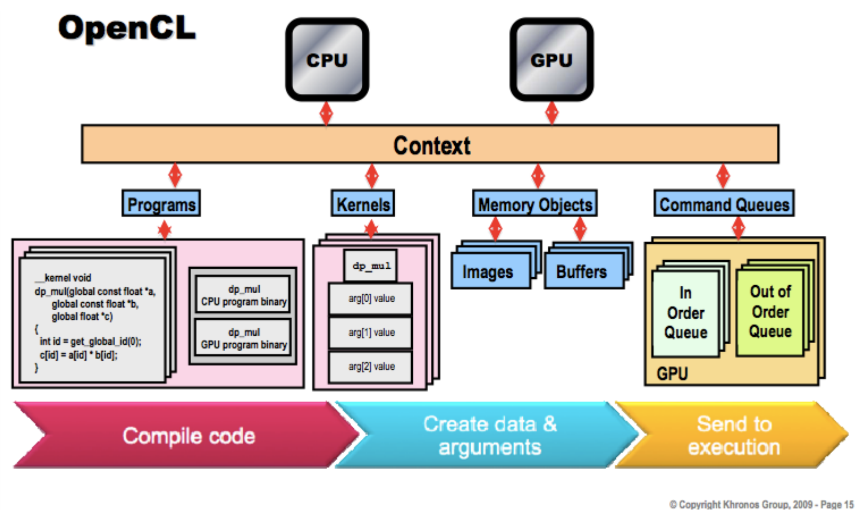
CS133
Parallel & Distributed Computing

Introduction to Convolutional Neural Networks

Instructor: Jason Cong

cong@cs.ucla.edu

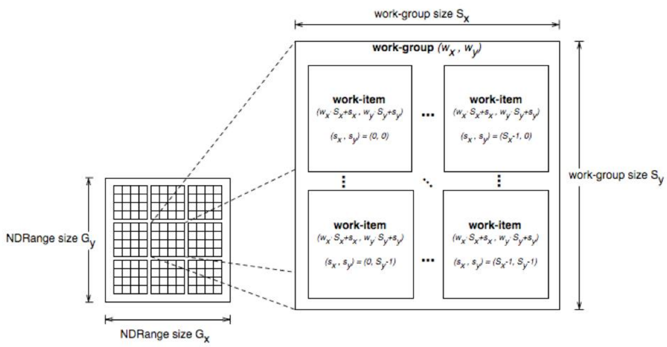
Review of OpenCL Basics



P. Mistry & D. Schaa (Northeastern Univ.), with B. Gaster (AMD), AMD © 2011

Correction from Last Lecture -- Thread Structure

- Work-items can uniquely identify themselves based on:
  - A global id (unique within the index space)
  - A work-group ID and a local ID within the work-group



Impacts of Deep Learning for Many Applications

Unmanned Vehicle

Speech & Audio

Text & Language

WIKIPEDIA The Free Encyclopedia · REUTERS · AP Associated Press

Genomics

Image & Video

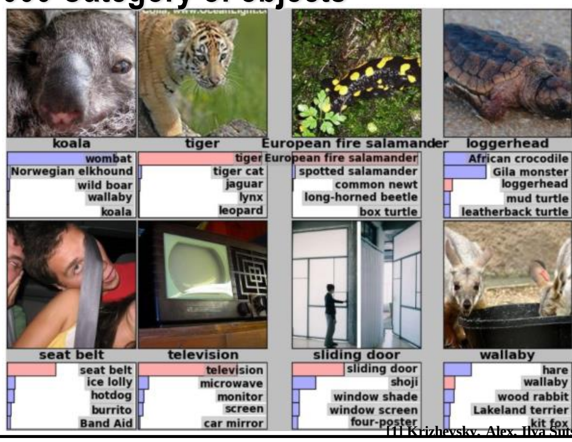
flickr · Google · YouTube

Multi-Media

All images are from internet search

ImageNet Competition

- 1,200,000 Training Images
  - With 50,000 Validation & 100,000 Test Images
- 1000 Category of objects



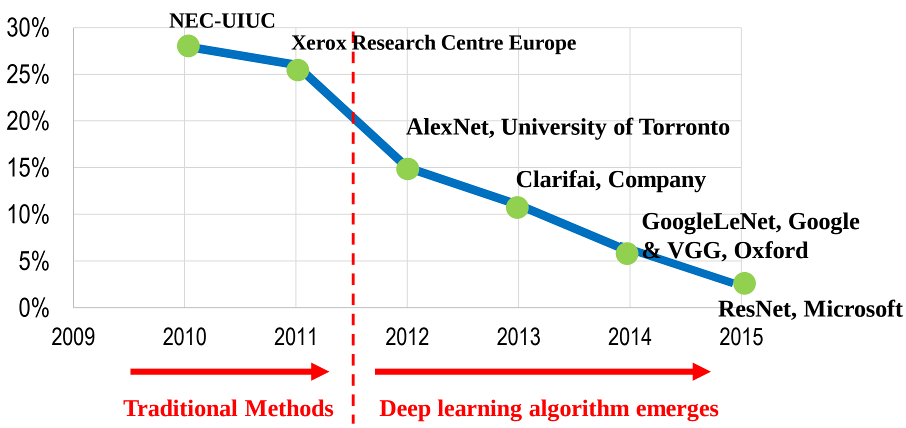
[1]

5

[1] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. NIPS2012

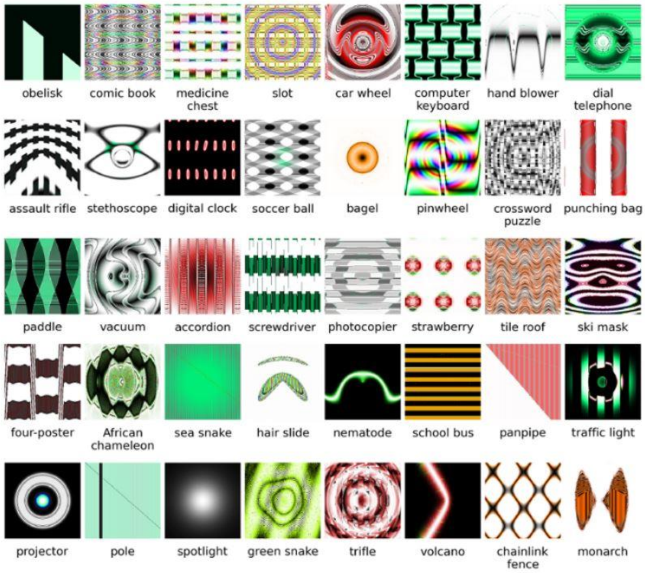
ImageNet Competition Results

Winning % Error



6

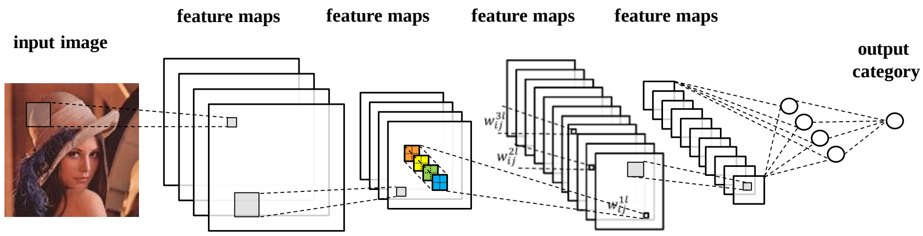
Still not Perfect: Images that Fools CNN/DNN



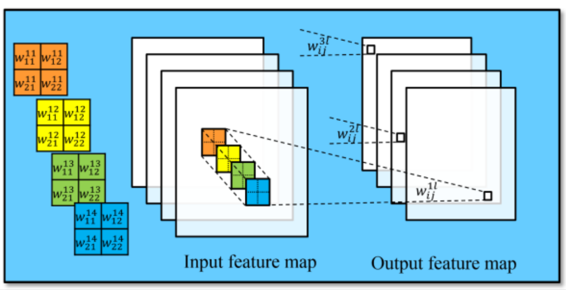
7

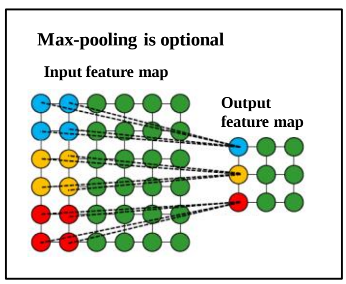
7

Inference: Convolutional Neural Network



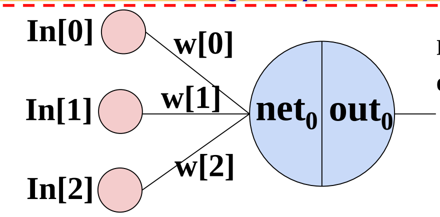
Inference: A feedforward computation





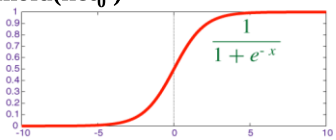
8

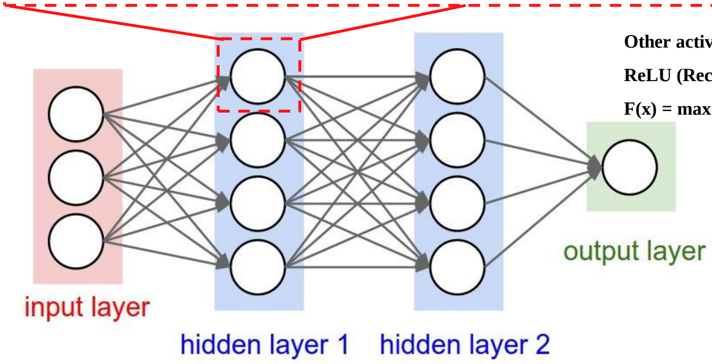
Inference: Deep Neural Network (Fully Connected Layers)



$net_0 = \sum in[i]*w[i], 0 <= i < 3$

out_0 = sigmoid(net_0)





Other activation functions:

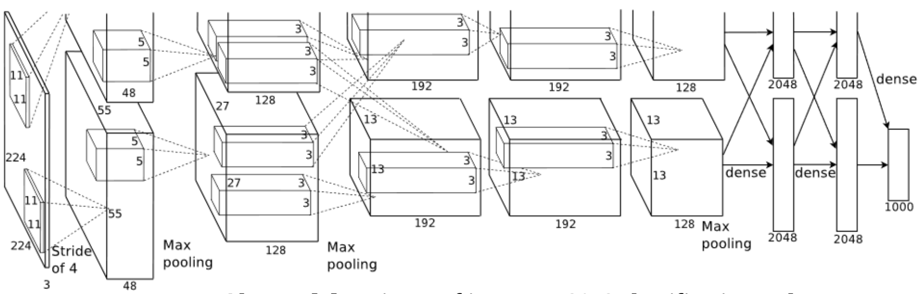
ReLU (Rectified Linear Unit)

F(x) = max (0, x)

9

9

Real-life CNNs: AlexNet, 8 layers



AlexNet [1] : Winner of imagenet 2012 classification task

10

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

10

Computation& Storage Analysis

	CONV	POOL	ReLU	Fully connect
Computation Complexity(10^6)	30600	6.12	13.5	122.7
Percentage	99.5%	0.0%	0.1%	0.4%
Storage Complexity (MB)	113	0	0	471.6
Percentage	19.3%	0.0%	0.0%	80.6%
Time% in pure software	96.3%	0.0%	0.0%	3.7%

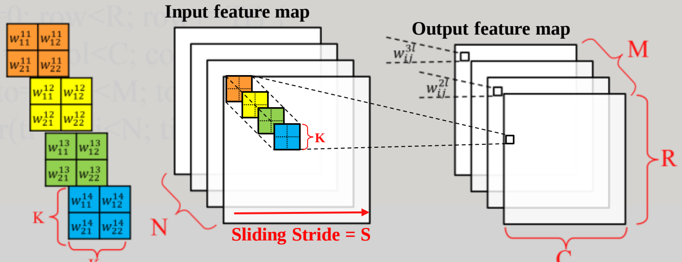
An example of VGG-16 network

A Lot of Interest in Hardware Acceleration !

11

11

Convolution Kernel for Inferencing



$$Y[m][r][c] = \sum_{n=0}^{N-1}\sum_{i=0}^{K-1}\sum_{j=0}^{K-1} W[m][n][i][j] \times X[n][S \times r + i][S \times c + j]$$

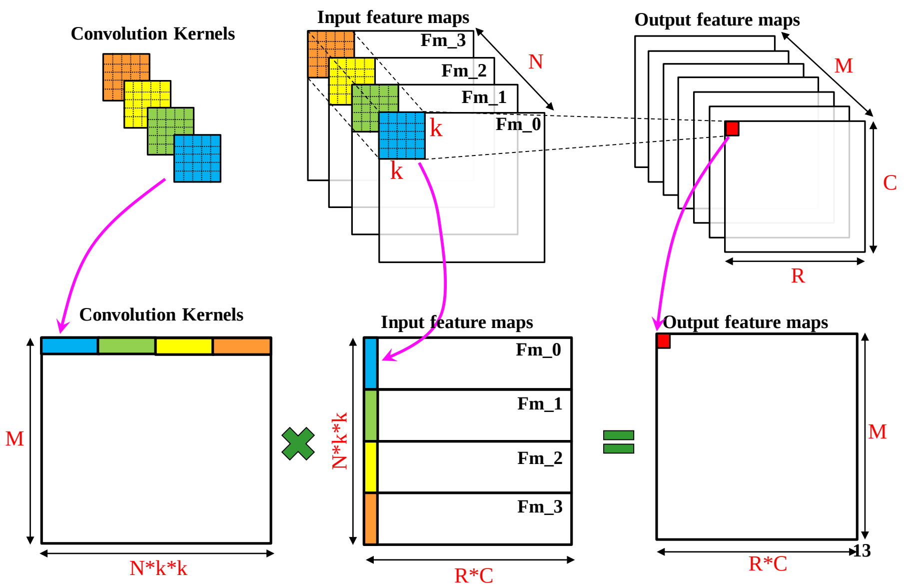
```
for(row=0; row<R; row++) {
  for(col=0; col<C; col++) {
    for(to=0; to<M; to++) {
      for(ti=0; ti<N; ti++) {
        for(i=0; i<K; i++) {
          for(j=0; j<K; j++) {
            output_fm[to][row][col] +=
            weights[to][ti][i][j]*input_fm[ti][S*row+i][S*col+j];
}}}}}}
```

R, C, M, N, K, S are all configuration parameters of the convolutional layer

12

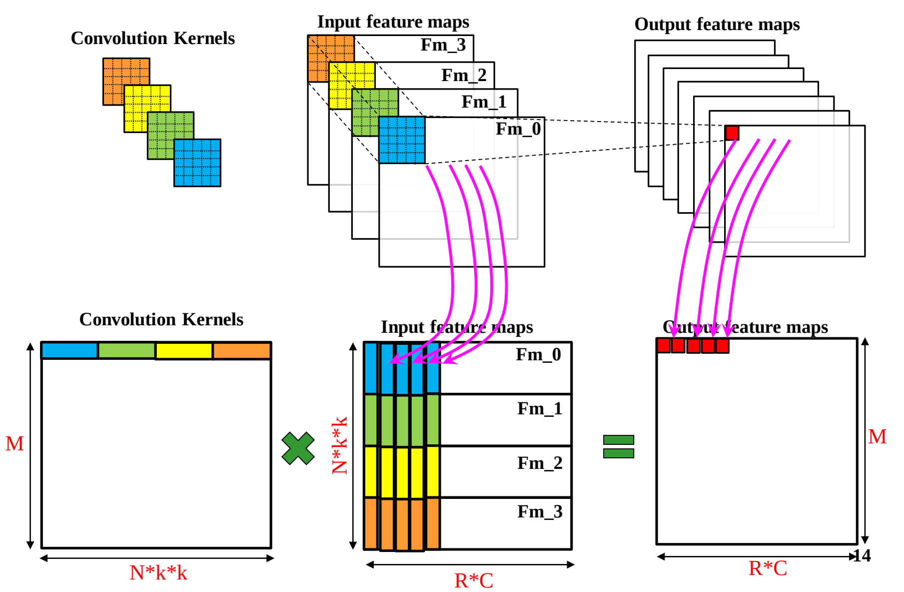
12

Transforming Convolution to MM



13

Transforming Convolution to MM

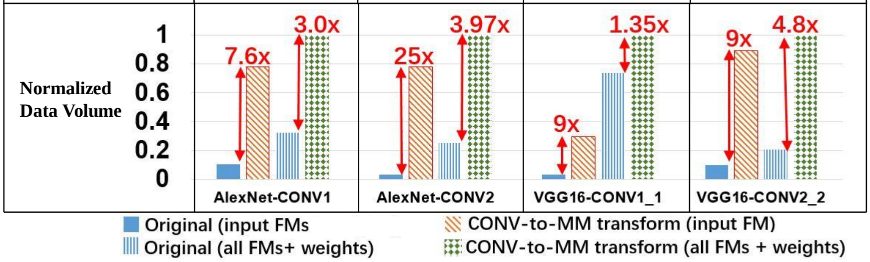


14

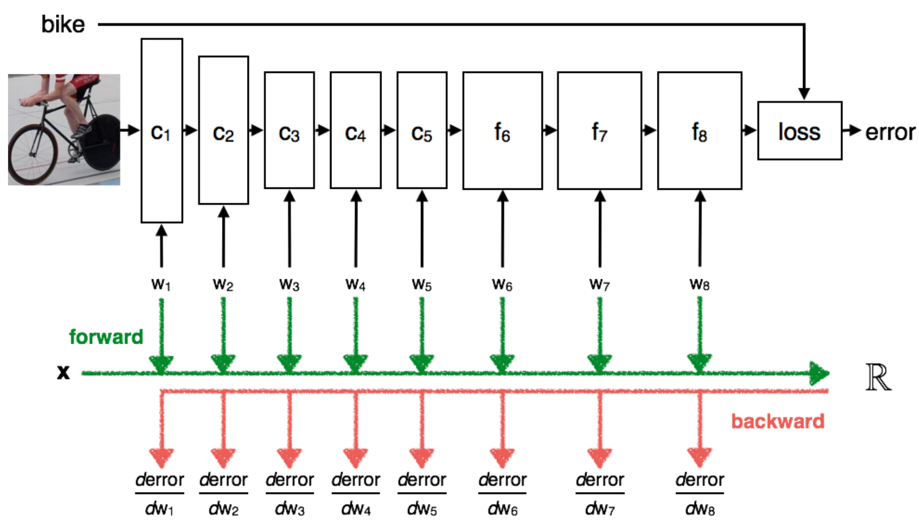
Data Duplications

Data Size	Weight	Input	Output
Convolution	M*N*K*K	~N*(R/S)*(C/S)	M*R*C
Convolution MM	M*N*K*K	N*K*K*R*C	M*R*C
MM/CONV Ratio	1	~(K/S)*(K/S)	1

	AlexNet-CONV1	AlexNet-CONV2	VGG16-CONV1_1	VGG16-CONV2_2
In/out Img # <M, N>	<96, 3>	<256, 48>	<64, 3>	<128, 128>
Image size <R, C>	<55, 55>	<27, 27>	<224, 224>	<112, 112>
Kernel, stride <K, S>	<11, 4>	<5, 1>	<3, 1>	<3, 1>



CNN Training



Feedforward Computation on FPGA

```
for(row=0; row<R; row+=Tr) {                                   (Tile loop)
 for(col=0; col<C; col+=Tc) {                                  (Tile loop)
   for(to=0; to<M; to+=Tm) {                                   (Tile loop)
     for(ti=0; ti<N; ti+=Tn) {                                 (Tile loop)
```

Off-chip Data Transfer: Memory Access Optimization

On-chip Data: Computation Optimization

```
       for(trr=row; trr<min(row+Tr, R); trr++) {              (Point loop)
         for(tcc=col; tcc<min(tcc+Tc, C); tcc++) {            (Point loop)
           for(too=to; too<min(to+Tm, M); too++) {            (Point loop)
             for(tii=ti; tii<(ti+Tn, N); tii++) {             (Point loop)
               for(i=0; i<K; i++) {                           (Point loop)
                 for(j=0; j<K; j++) {                         (Point loop)
                   output_fm[to][row][col] +=
                   weights[to][ti][i][j]*input_fm[ti][S*row+i][S*col+j];
       }}}}}}
}}}}
```


CS 133 Worksheet

- Given that L1 cache is 32KB and L2 cache is 1MB on your machine, please analyze if each of the following set of matrices in your Lab 3 can fit into L1 or L2 cache.
  - All input feature maps
  - All output feature maps
  - All weight matrices

Acknowledgements

- Some slides on CNN are compiled with help of Chen Zhang, a visiting PhD student from Peking Univ. , now at Microsoft Research Asia (MSRA)

19