Preliminary Studies in Optimizing Deep Learning for Computer Vision: Memory Traffic Analysis Across CNN Layers

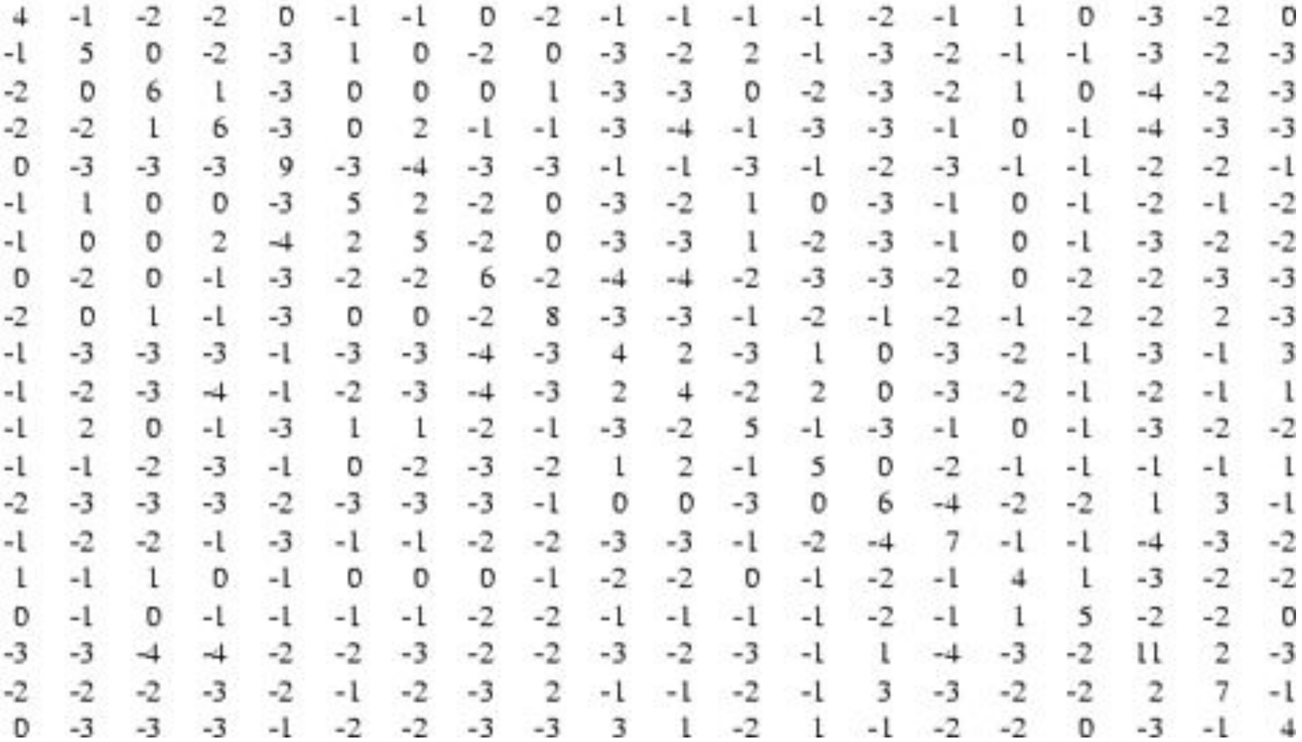
Presenter: Shih-wei Liao
Pin-Wei Liao, Wei-Chung Hsu
National Taiwan University
liao@csie.ntu.edu.tw

Outline

- Human vision vs. Computer vision
- How human sees vs. How computer sees
- Deep learning
- CNN
- Motivation for memory traffic reudction
- Related work
- Liao’s algorithm
- Results
- Conclusion

Human Vision vs.
Computer Vision (which is to teach computer to “see”)

Computer Vision



4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0
-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3
-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3
-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3
0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1
-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2
-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2
0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3
-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3
-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3
-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1
-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2
-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1
-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1
-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2
1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2
0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0
-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3
-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1
0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

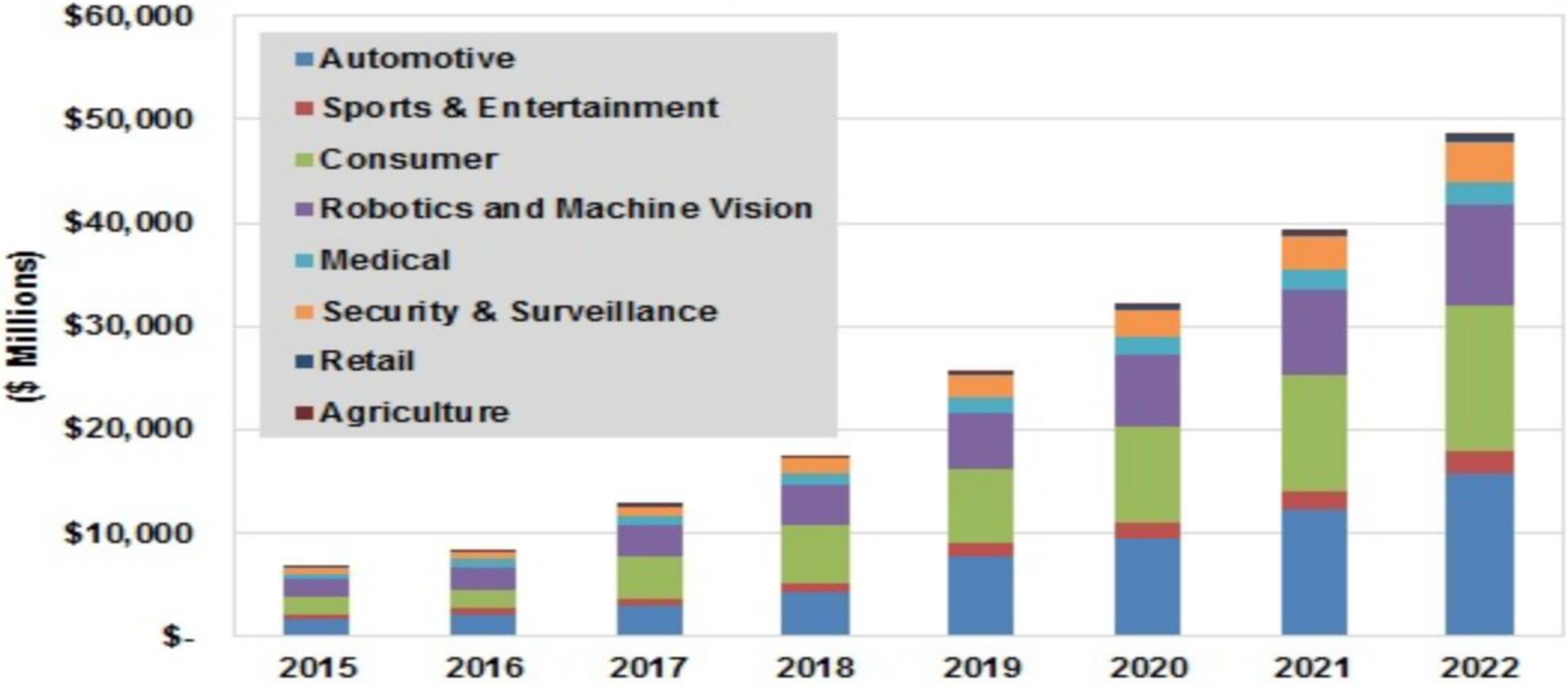
Computer Vision

Computer vision is an interdisciplinary scientific field that deals with how computers can be made to gain high-level understanding from digital images or videos. From the perspective of engineering, it seeks to automate tasks that the human visual system can do.

Computer vision tasks include methods for acquiring, processing, analyzing and understanding digital images, and extraction of high-dimensional data from the real world in order to produce numerical or symbolic information, *e.g.*, in the forms of decisions. (Wikipedia)

Tractica

Computer Vision Revenue by Application Market, World Markets: 2015-2022

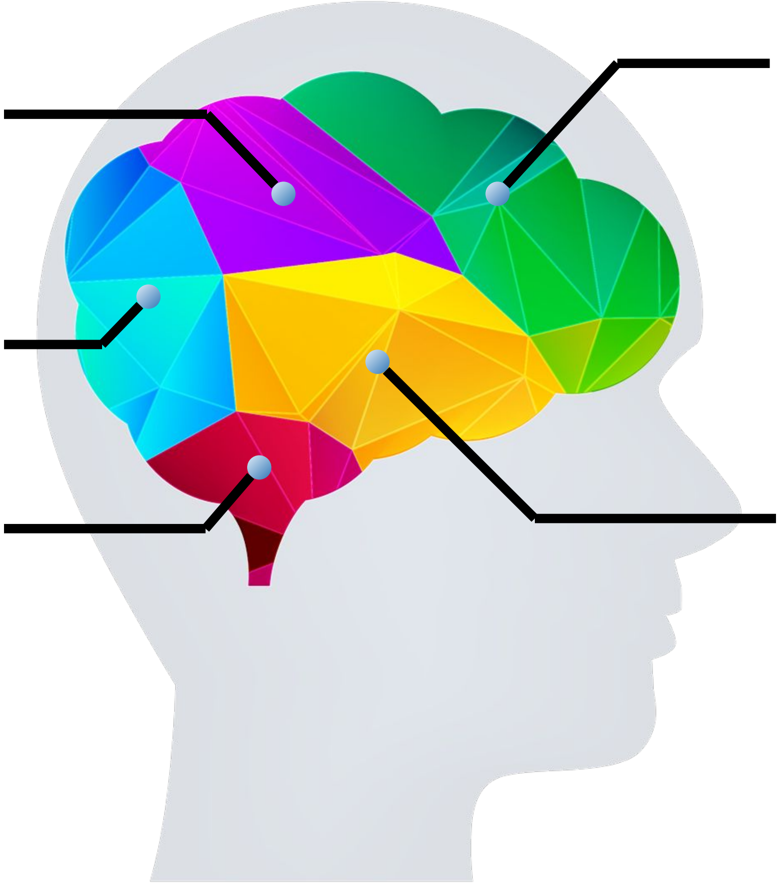


Source: Tractica

Outline

- Human vision vs. Computer vision
- How human sees vs. How computer sees
- Deep learning
- CNN
- Motivation for memory traffic reudction
- Related work
- Liao’s algorithm
- Results
- Conclusion

Human Vision: Occipital & Parietal Lobe



Parietal Lobe
(Checker)

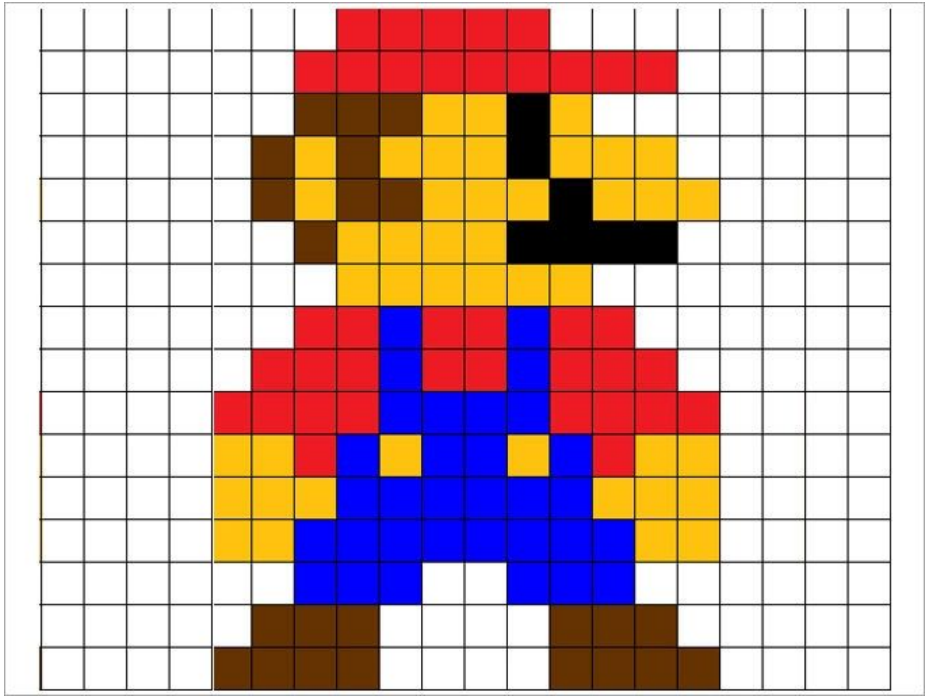
Occipital Lobe (Town Crier)
Real world data ingestion

Brainstem
Basic function maintenance

Frontal Lobe

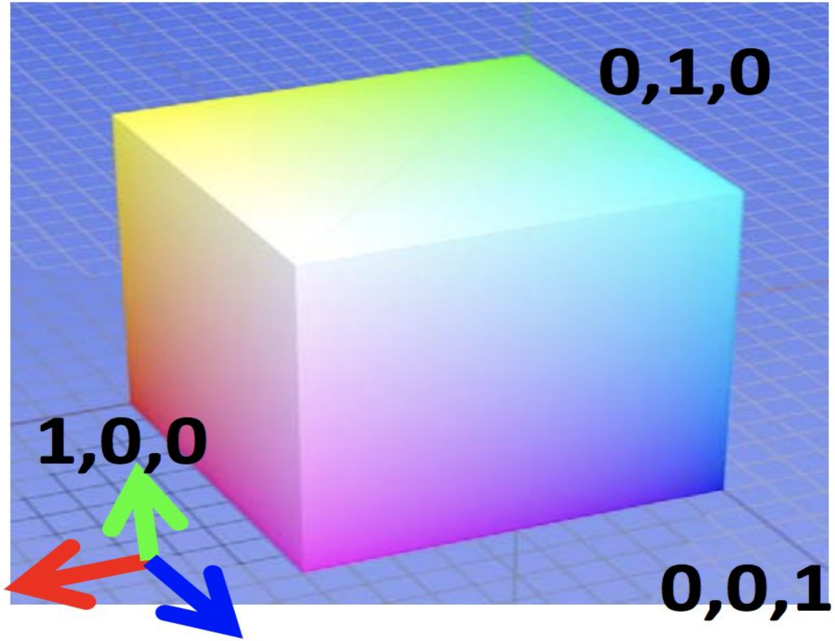
Temporal Lobe
Interaction

How computer sees: Basic: Pixel



Example Color Space: RGB

- Default Color Space

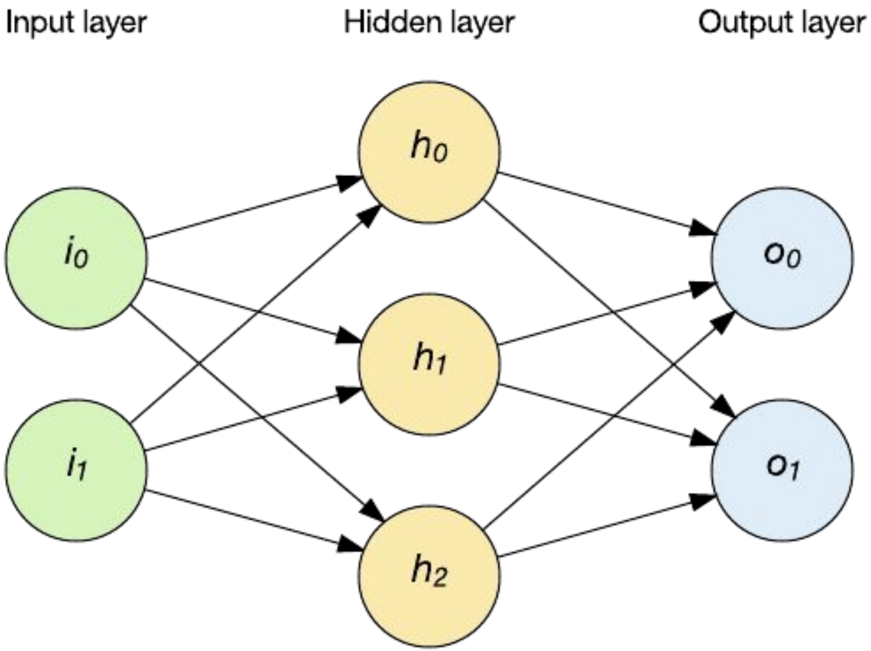


R
(G=0,B=0)

G
(R=0,B=0)

B
(R=0,G=0)

How computer sees: Neural Network



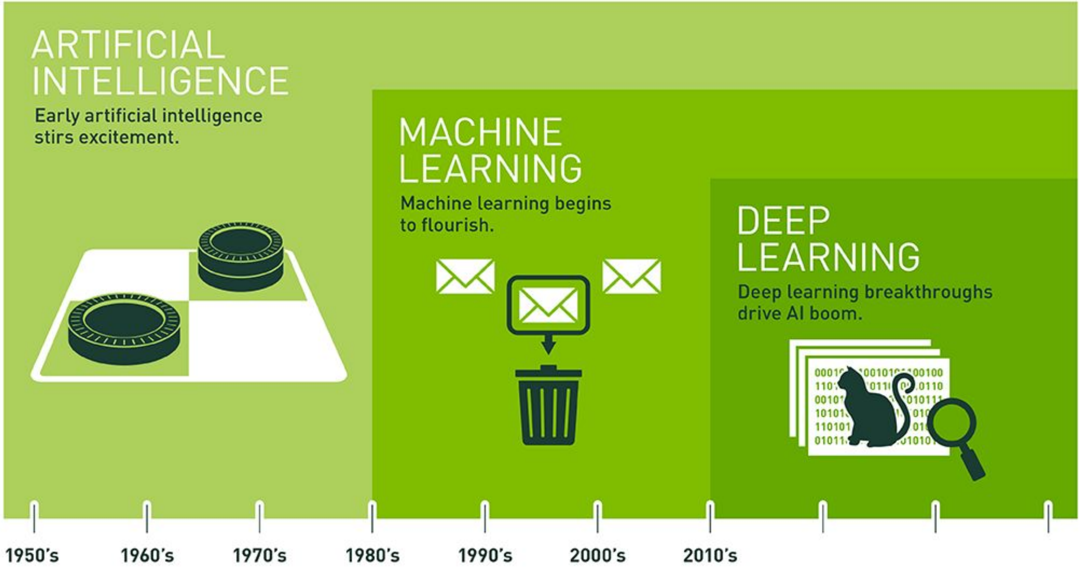
Neural Network: Main architecture for Deep Learning. After input, hidden layer performs matrix operations and then output.

Can be many Hidden layers, hence the name ”Deep.”

Outline

- Human vision vs. Computer vision
- How human sees vs. How computer sees
- Deep learning
- CNN
- Motivation for memory traffic reudction
- Related work
- Liao’s algorithm
- Results
- Conclusion

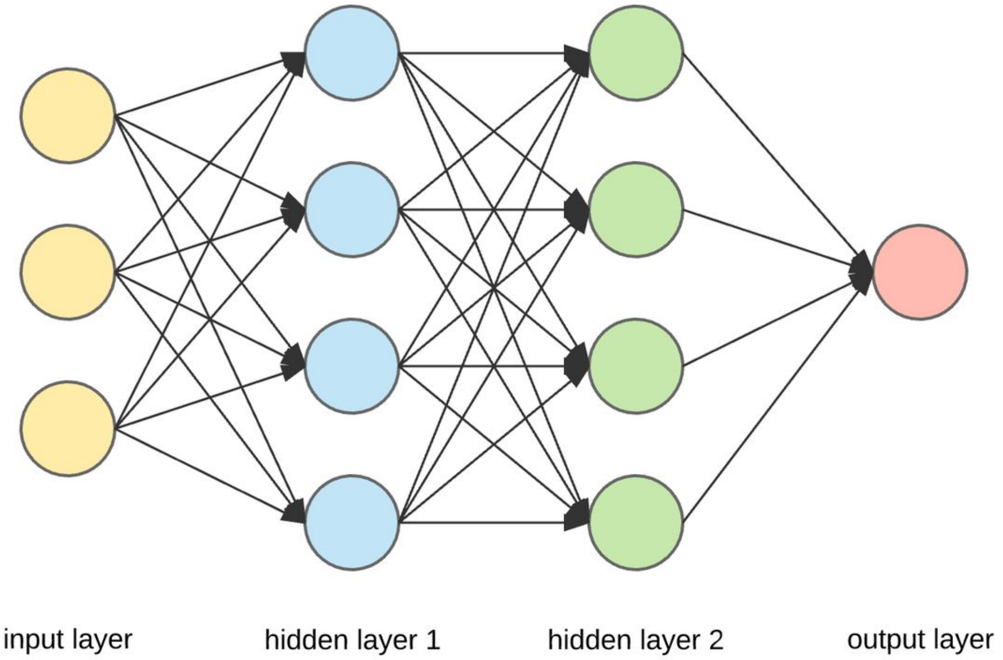
AI, Machine Learning, and Deep Learning


ARTIFICIAL INTELLIGENCE
Early artificial intelligence stirs excitement.
MACHINE LEARNING
Machine learning begins to flourish.
DEEP LEARNING
Deep learning breakthroughs drive AI boom.
1950's
1960's
1970's
1980's
1990's
2000's
2010's

AI, machine learning, deep learning: Deep learning is ML's subset. And machine learning is the subset of AI.

They are from different eras.

Deep Learning Neural Network

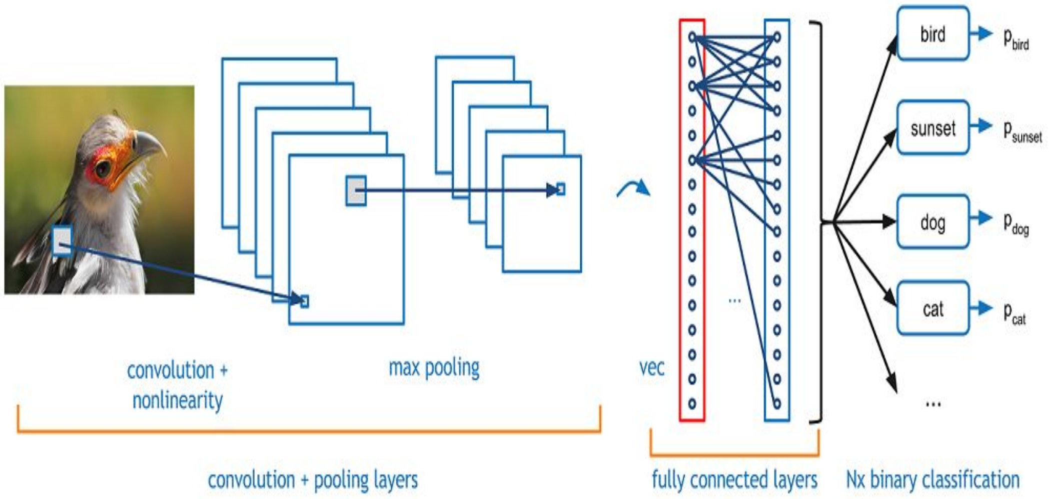


More than 1 Hidden layers, hence the name ”Deep.”

Outline

- Human vision vs. Computer vision
- How human sees vs. How computer sees
- Deep learning
- CNN
- Motivation for memory traffic reudction
- Related work
- Liao’s algorithm
- Results
- Conclusion

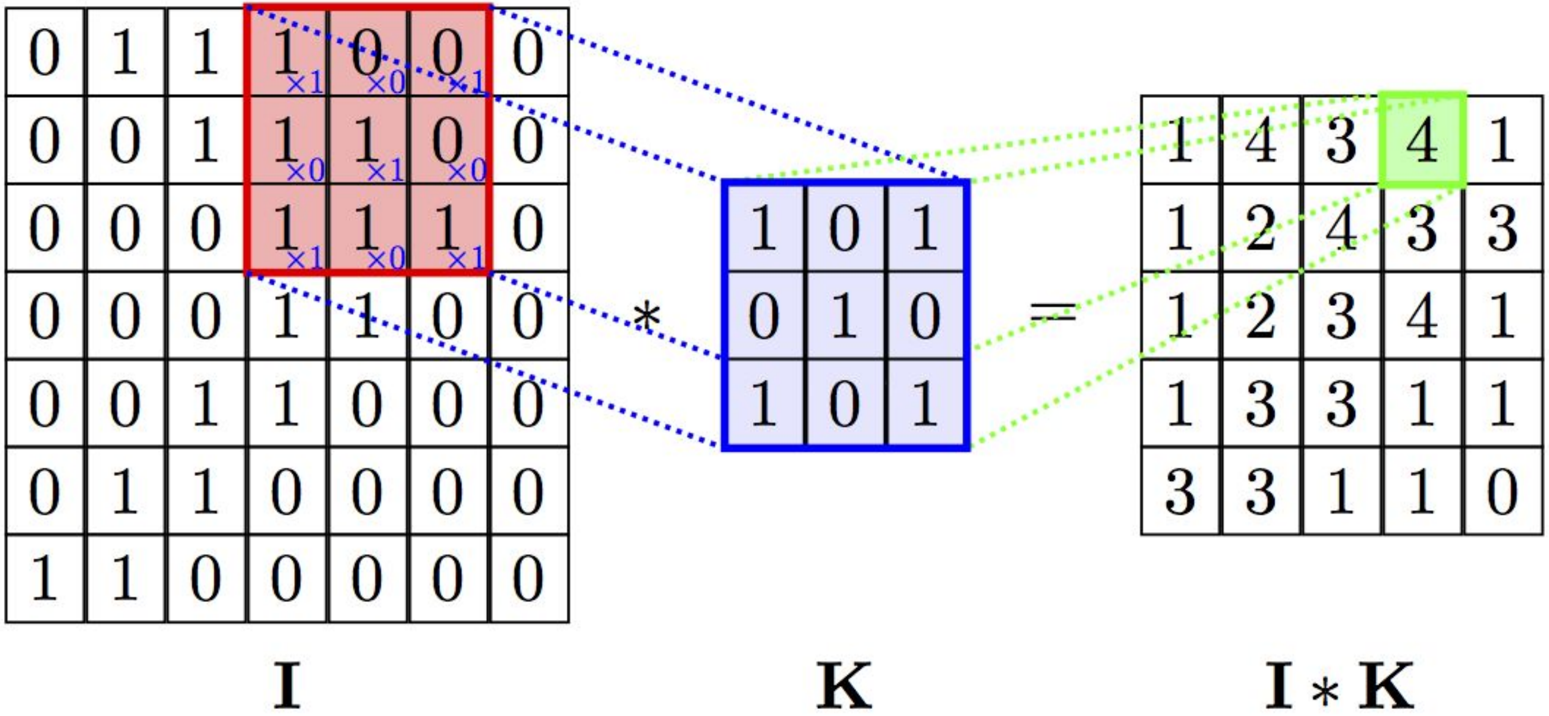
Convolutional Neural Network (CNN)



Convolutional Neural Network derives from Neural Network. CNN for images and videos → Breakthrough!

Key: Use ”Sliding window” over picture to perform ”convolve“, the extracted feature is passed to the next layer. Use the extracted feature vector to classify.

Convolution



0	1	1	1	0	0	0
0	0	1	1	1	0	0
0	0	0	1	1	1	0
0	0	0	1	1	0	0
0	0	1	1	0	0	0
0	1	1	0	0	0	0
1	1	0	0	0	0	0

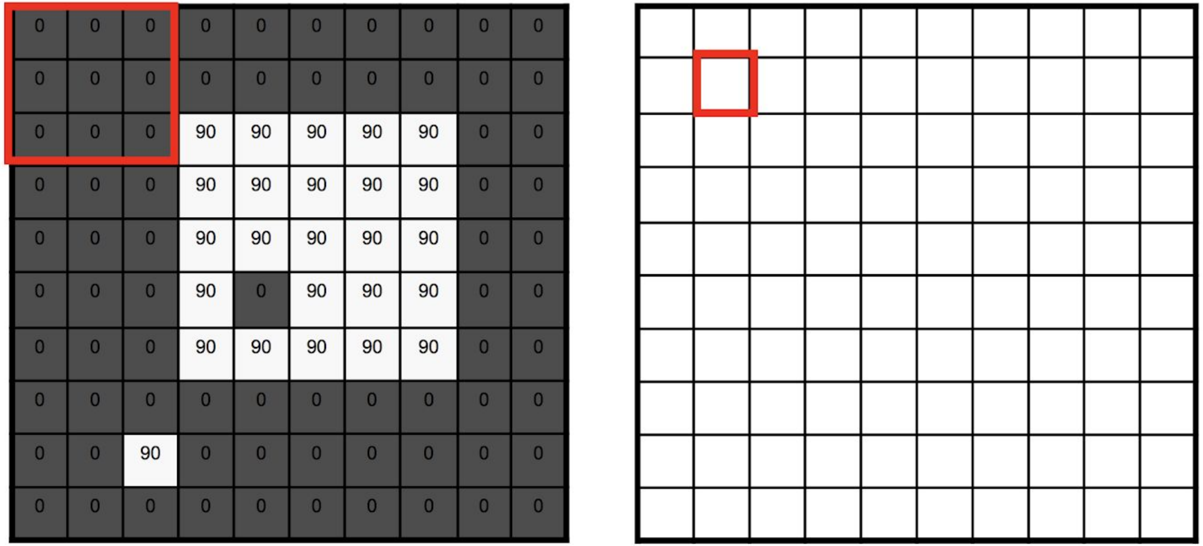
1	0	1
0	1	0
1	0	1

1	4	3	4	1
1	2	4	3	3
1	2	3	4	1
1	3	3	1	1
3	3	1	1	0

$$g[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

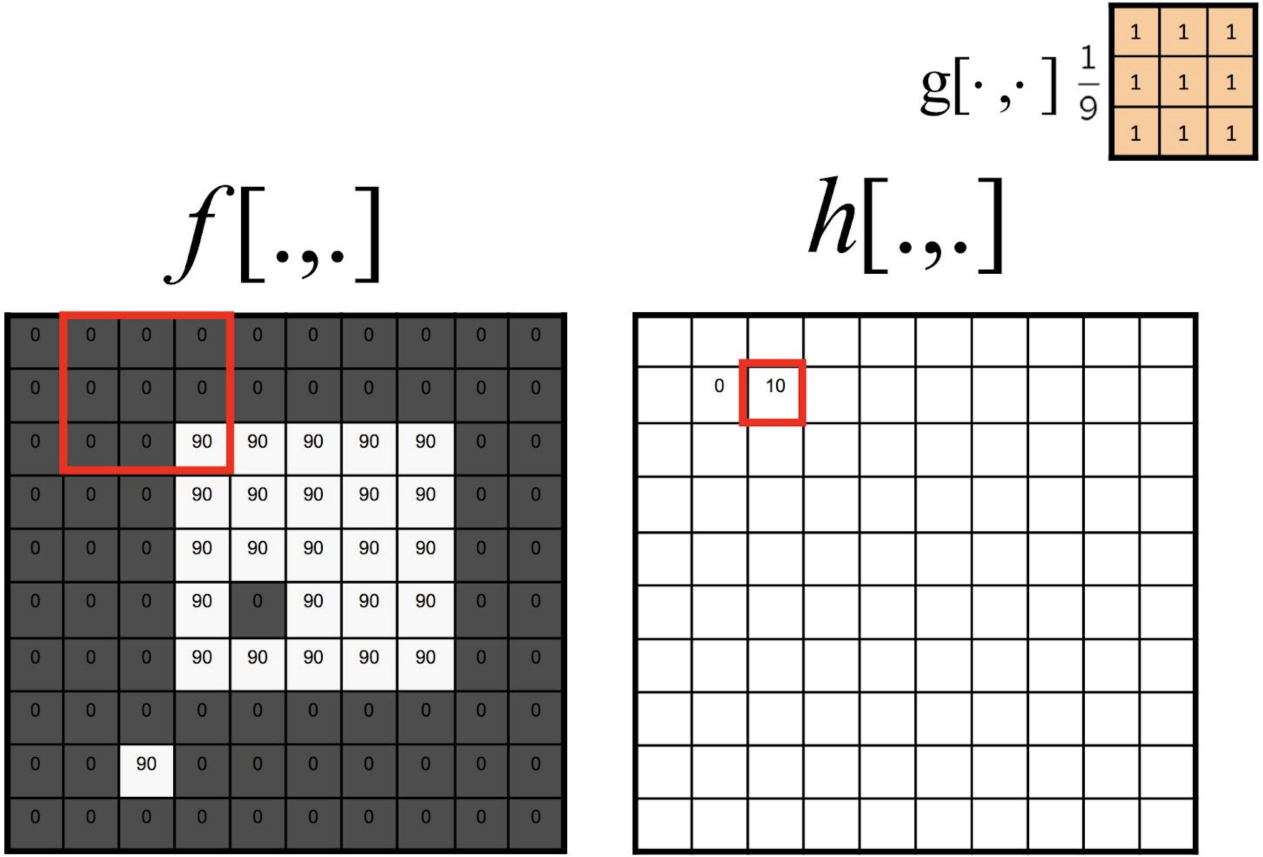
$f[.,.]$

$h[.,.]$



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$



$g[\cdot\,,\cdot\,]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$

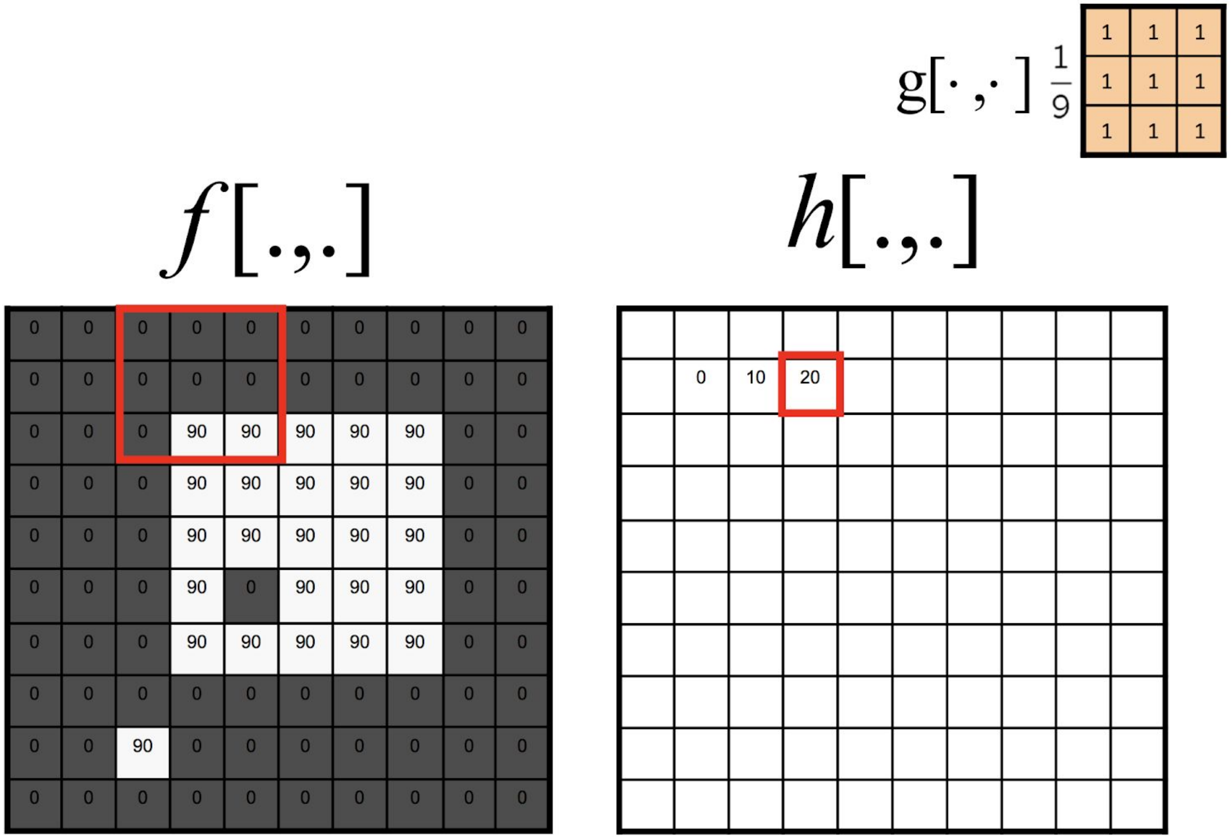
$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10							

$$h[m,n]=\sum_{k,l} g[k,l]\,f[m+k,n+l]$$



$g[\cdot,\cdot]\ \frac{1}{9}$

1	1	1
1	1	1
1	1	1

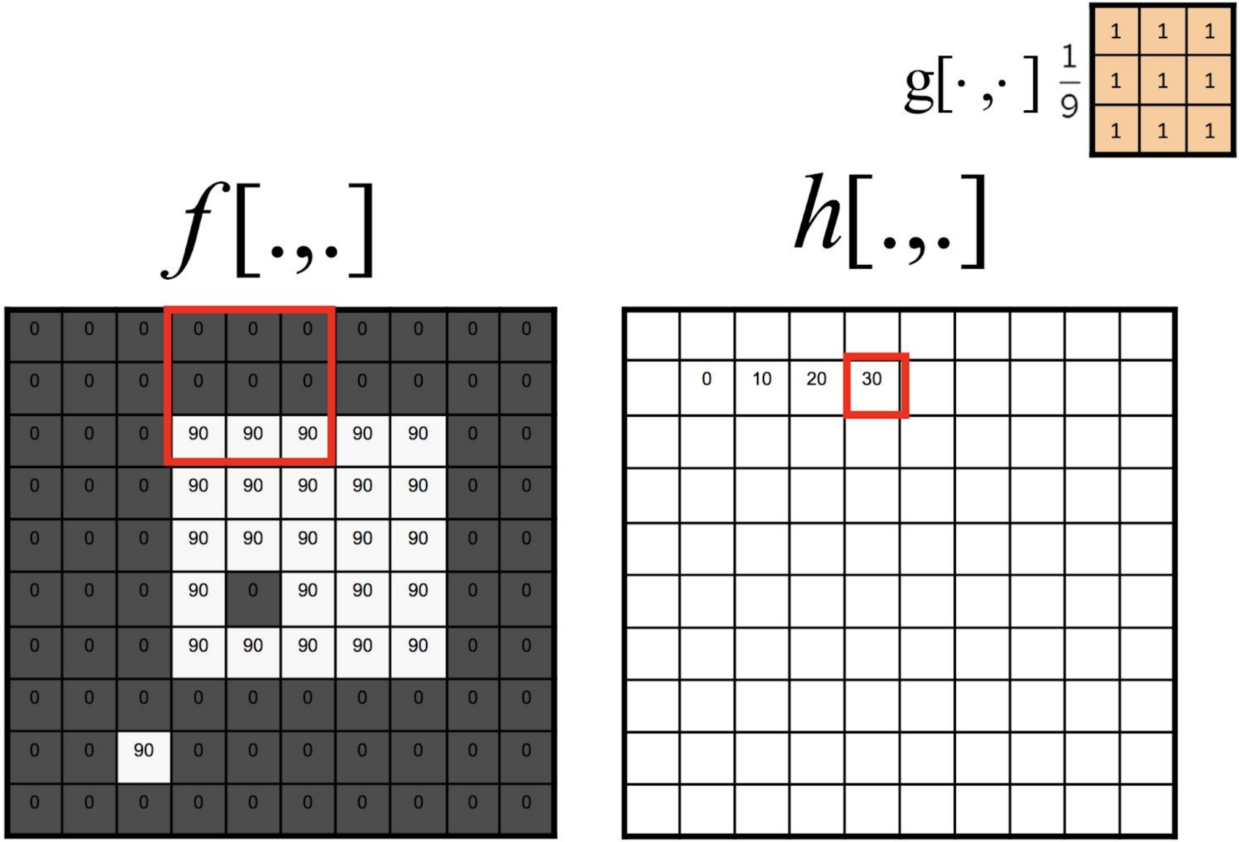
$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20						

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$$



$g[\cdot\,,\cdot\,]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$

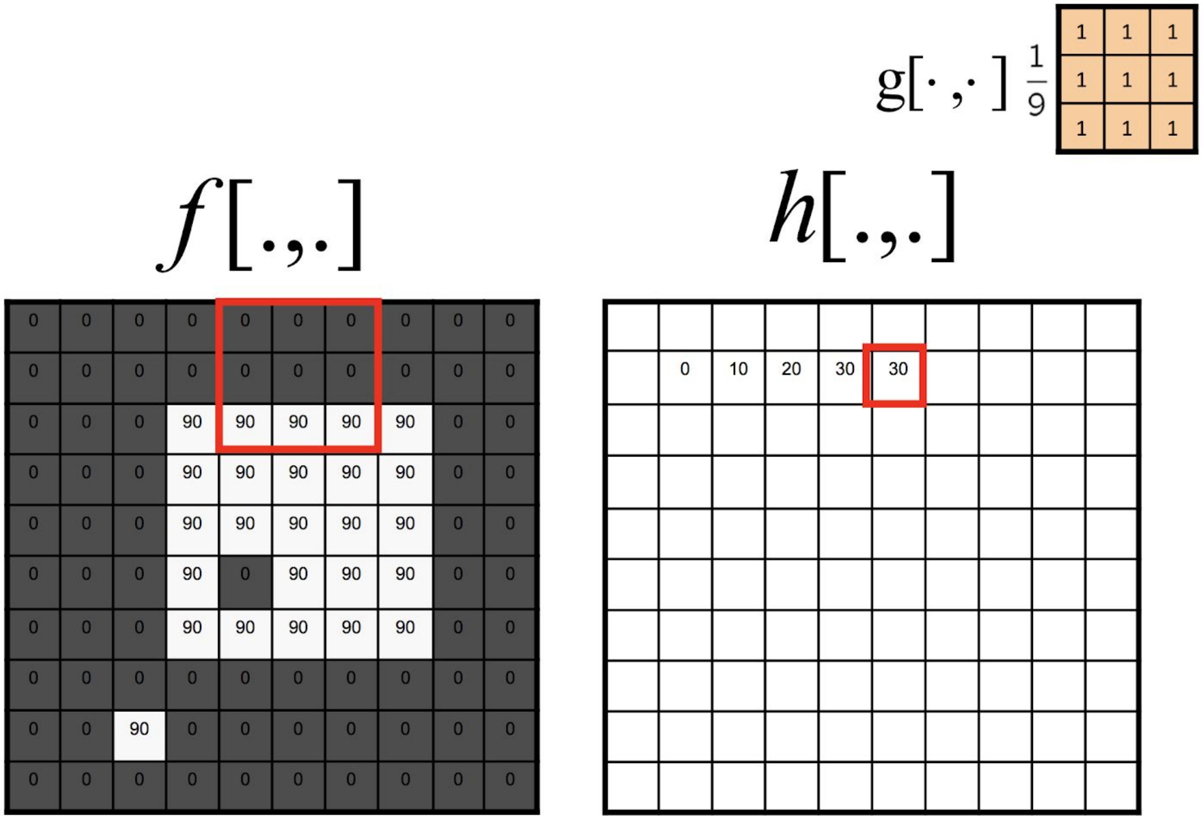
$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30					

$$h[m,n]=\sum_{k,l} g[k,l]\, f[m+k,n+l]$$



$g[\cdot,\cdot]\ \frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

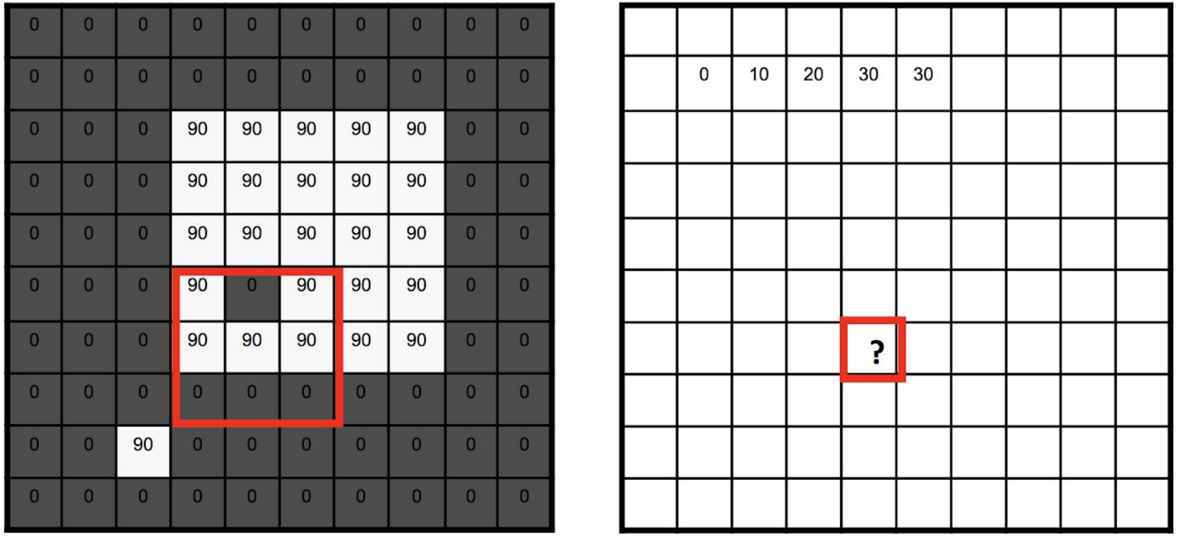
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30	30				

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

$$g[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$



$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30	30				
				?					

$$h[m,n]=\sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Outline

- Human vision vs. Computer vision
- How human sees vs. How computer sees
- Deep learning
- CNN
- Motivation for memory traffic reudction
- Related work
- Liao’s algorithm
- Results
- Conclusion

Why Memory Traffic Reduction

- CNN (Convolutional Neural Network) operations have a large number of parameters.
- However, such parameters are too many to fit in the limited on-chip SRAM.
- Therefore, we need to make careful management to the most effective use of the limited on-chip SRAM.

Related work on Memory Traffic Reduction

- “A data locality optimizing algorithm” by Wolf and Lam in 1991 proposed an algorithm to improve cache performance for loop nests by using interchange, reversal, skewing and tiling.
- “SUIF Explorer” by Liao and Lam in 1999 added reduction and contraction.
- Many reduces CNN memory traffic via data compression:
    - Eyeriss from MIT proposed quantization [Vivek Sze, ISCA@2016].
    - Weight pruning [Song Han, J. Pool, J. Tran, William Dally from Stanford, NIPS@2015, Song Han, H. Mao, William Dally, ICLR@2016]
    - Weight pruning: Dynamic network surgey [Y. Guo, A. Yao, Yurong Chen from Intel, NIPS @2016].

We focus on scope of 1-CNN-layer vs. 2-CNN-layer

- Most prior studies optimize the use of on-chip SRAM for a single convolution layer, they tend to ignore the opportunity of inter-layer data reuse.
- We have proposed an algorithm to schedule 2 adjacent layers of CNN operations.
- Our goal is to reduce traffic between DRAM and local memory more than allocating the buffer to only a single layer.

Other multi-layer work:

- 1-layer: Arthur Stoutchinin, F. Conti, Luca Benini 2019 proposed an analytical memory bandwidth model for loop nest optimization targeting architectures with application-managed buffers.
    - They tiled the loop nest once to represent two-level memory hierarchy.
- dMazeRunner@Arizona State University: Dave et al. proposed dMazeRunner framework, which is a model to spatio-temporally execute a perfectly nested loop on dataflow accelerators [ICASSP 2020]
- At Stony Brook University, Alwani et al. proposed "Fused-layer CNN accelerators" at Micro Conference 2016. They design their own hardware. Can choose which layers to fuse, and then tell the hardware how much on-chip storage to add to use their schedule.

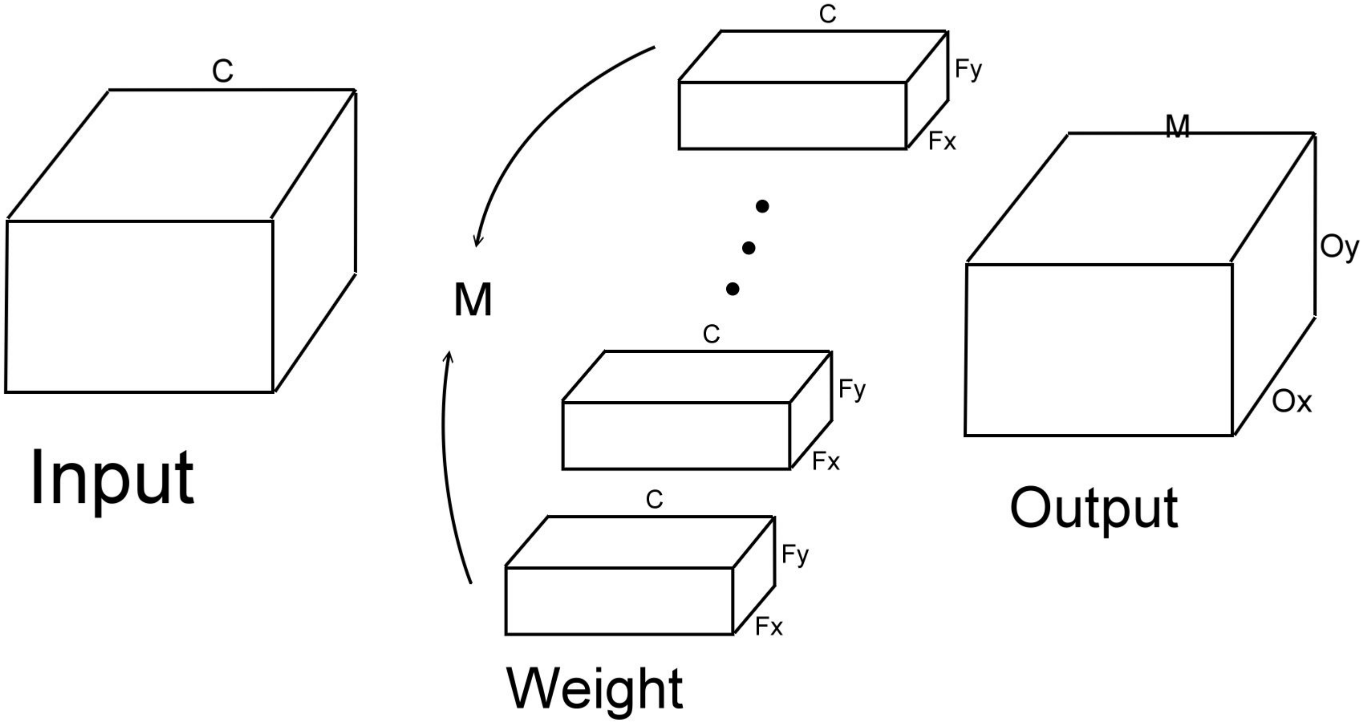
Outline

- Human vision vs. Computer vision
- How human sees vs. How computer sees
- Deep learning
- CNN
- Motivation for memory traffic reudction
- Related work
- Liao’s algorithm
- Results
- Conclusion

Architecture

- Assume a 2-layer memory hierarchy:
  - DRAM
  - On-chip SRAM
- On-chip SRAM is an application-managed buffer
- Memory traffic refers to the number of bytes transferred between DRAM and on-chip SRAM

Convolution



Convolution loop-nest

```
LOF: for (int m = 0; m < M; m++) // filters
  LIF: for (int c = 0; c < C; c++) // channels
    LSY: for (int oy = 0; oy < Oy; oy++) // ofmap rows
      LSX: for (int ox = 0; ox < Ox; ox++) // ofmap cols
        LFY: for (int fy = 0; fy < Fy; fy++) // filter height
          LFX: for (int fx = 0; fx < Fx; fx++) // filter width
            O[m][oy][ox] += I[c][oy+fy-1][ox+fx-1] *
              W[m][c][fy][fx];
```

Note: In ResNet-34, adjacent layers may have same C and M. For 7x7x512 case:

- Convolution matrix, aka sliding window = 3x3. Fx=Fy=3. Being small, the inner 2 loops are not tiled. W contains the sliding window.
- For input and output: Ox=Oy=7. These are dimensions of feature maps.
- M = C = 512. For ResNet-34, it can be 512 filters here, each generating one out of 512 channels.

Liao’s memory traffic reduction algorithm has 3 parts:

- Case 1. Single-layer convolution scheduling
- Case 2. Cross-layer convolution scheduling
- Case selection algorithm

Case 1. Single-layer convolution scheduling (1/4)

We propose a method to find optimal scheduling for a single convolution by changing loop ordering, tiling size and buffering level.

- Loop ordering
  - We can reorder m, c, oy, ox, fy, fx. There are 6! permutations.
- Tiling size
  - We tile four loop LOF, LIF, LSY, LSX. The four tile sizes can be different.
- Buffering level
  - We can choose buffering level for each array reference [Stoutchinin, Conti, Benini @ 2019]
  - E.g., if buffering_level(W) = L2, that means the array W is buffered in L0, L1 and L2. Note that L0 is the innermost loop. L1 is its outer loop and so on and so forth.

Convolution loop nest after tiling (2/4)

```
for (int mm = 0; mm < M; mm += MB)
  for (int cc = 0; cc < C; cc += CB)
    for(int oxx = 1; oxx <= Ox; oxx += OxB)
      for (int oyy = 1; oyy <= Oy; oyy += OyB)
         // buffering O
         for (int c = cc; c < min(cc + CB, C); c++)
         // buffering I
         for (int fy = 0; fy < Fy); fy++)
           for (int fx = 0; fx < Fx; fx++)
           // buffering W
           for (int oy = oyy; oy <= min(oyy + OyB, Oy); oy++)
             for (int ox = oxx; ox <= min(oxx + OxB, Ox); ox++)
               for (int m = mm; m < min(mm + MB, M); m++)
                    O[m][oy][ox] += I[c][oy+fy-1][ox+fx-1] * W[m][c][fy][fx];
```

Function “memory_traffic” (3/4)

- Calculate the memory traffic of O, I, W arrays separately.
- The footprint of an array inside a loop is the portion of array it will access while iterating the loop space.
- We call T_x the memory traffic for array X. Assume $L_0 ... L_{N-1}$ are the loops in the current schedule, ordered from the innermost to the outermost one.
  - The number of memory accesses to X = The footprint of X with respect to Li multiplied by the total number of times that loop L_i is executed.
- The total traffic is the sum of the traffic of O, I, W three arrays
- Memory_traffic T = T_I+ T_W+T_O*2

Dataflow schedule selection (4/4)

- Given a buffer size, Goal: Find a schedule to minimize the off-chip memory traffic.
    - Enumerate different loop orders, loop tile sizes and buffering levels
        - Check if the footprint of the three arrays added up is less than the on-chip buffer size.
    - Choose the combination which yields the minimum off-chip memory traffic

Case 2. Cross-layer (1/5)

- CNN usually has many *connected* convolution layers.
  - The output of each layer will be the input of the next layer.
- We can leave the output feature maps in the buffer.
  - The input feature maps of the following layer can be read from the buffer.
- We consider scheduling 2 adjacent convolution layers.
  - See ResNet examples

Cross-layer memory traffic (2/5)

- Assuming that the output feature maps can all be put into the buffer.
- We leave the output feature maps in the buffer, so we save the traffic of reading and writing the output feature maps from DRAM.
    - The following layer can read the input feature maps from the buffer, so can save the traffic of reading the input from DRAM.
- T_1=Memory_traffic_layer1()
    - $T_1=T_I+T_W$
    - No T_O because output fits into the buffer.
- T_2=Memory_traffic_layer2()
    - $T_2=T_W+T_O*2$
    - *2: Because you need to read back the half-way results and write back.

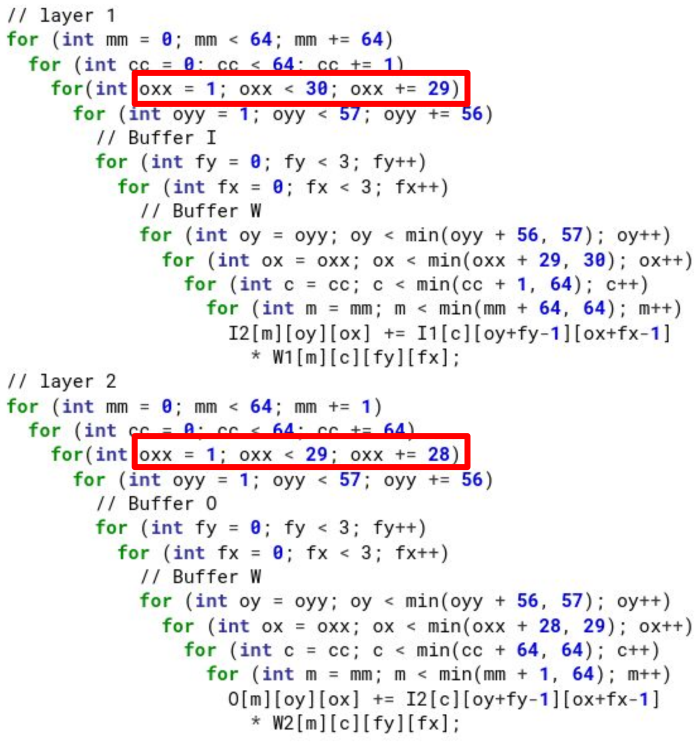
Cross-layer memory traffic (3/5)

- We use a similar method to single-layer to find the schedule of the first and second layer.
  - We try different loop order, tiling size, buffering level, but the formula for calculating memory traffic is somewhat different.
  - If the output feature maps cannot be put into the buffer completely, we cut the input into several parts from X and Y directions. Only calculate part of one layer at a time.

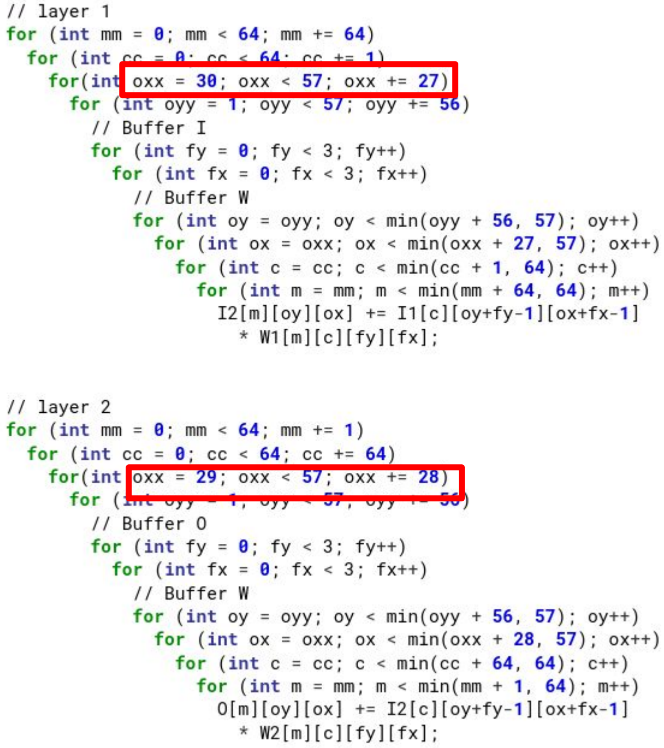
Cross-layer loop transform example (4/5)

Input 56*56*64

Buffer: 512KB



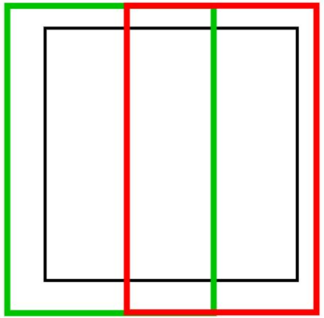
```
// layer 1
for (int mm = 0; mm < 64; mm += 64)
  for (int cc = 0; cc < 64; cc += 1)
    for(int oxx = 1; oxx < 30; oxx += 29)
      for (int oyy = 1; oyy < 57; oyy += 56)
        // Buffer I
        for (int fy = 0; fy < 3; fy++)
          for (int fx = 0; fx < 3; fx++)
            // Buffer W
            for (int oy = oyy; oy < min(oyy + 56, 57); oy++)
              for (int ox = oxx; ox < min(oxx + 29, 30); ox++)
                for (int c = cc; c < min(cc + 1, 64); c++)
                  for (int m = mm; m < min(mm + 64, 64); m++)
                    I2[m][oy][ox] += I1[c][oy+fy-1][ox+fx-1]
                      * W1[m][c][fy][fx];
// layer 2
for (int mm = 0; mm < 64; mm += 1)
  for (int cc = 0; cc < 64; cc += 64)
    for(int oxx = 1; oxx < 29; oxx += 28)
      for (int oyy = 1; oyy < 57; oyy += 56)
        // Buffer O
        for (int fy = 0; fy < 3; fy++)
          for (int fx = 0; fx < 3; fx++)
            // Buffer W
            for (int oy = oyy; oy < min(oyy + 56, 57); oy++)
              for (int ox = oxx; ox < min(oxx + 28, 29); ox++)
                for (int c = cc; c < min(cc + 64, 64); c++)
                  for (int m = mm; m < min(mm + 1, 64); m++)
                    O[m][oy][ox] += I2[c][oy+fy-1][ox+fx-1]
                      * W2[m][c][fy][fx];
```



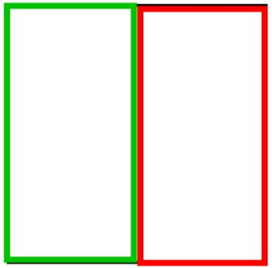
```
// layer 1
for (int mm = 0; mm < 64; mm += 64)
  for (int cc = 0; cc < 64; cc += 1)
    for(int oxx = 30; oxx < 57; oxx += 27)
      for (int oyy = 1; oyy < 57; oyy += 56)
        // Buffer I
        for (int fy = 0; fy < 3; fy++)
          for (int fx = 0; fx < 3; fx++)
            // Buffer W
            for (int oy = oyy; oy < min(oyy + 56, 57); oy++)
              for (int ox = oxx; ox < min(oxx + 27, 57); ox++)
                for (int c = cc; c < min(cc + 1, 64); c++)
                  for (int m = mm; m < min(mm + 64, 64); m++)
                    I2[m][oy][ox] += I1[c][oy+fy-1][ox+fx-1]
                      * W1[m][c][fy][fx];

// layer 2
for (int mm = 0; mm < 64; mm += 1)
  for (int cc = 0; cc < 64; cc += 64)
    for(int oxx = 29; oxx < 57; oxx += 28)
      for (int oyy = 1; oyy < 57; oyy += 56)
        // Buffer O
        for (int fy = 0; fy < 3; fy++)
          for (int fx = 0; fx < 3; fx++)
            // Buffer W
            for (int oy = oyy; oy < min(oyy + 56, 57); oy++)
              for (int ox = oxx; ox < min(oxx + 28, 57); ox++)
                for (int c = cc; c < min(cc + 64, 64); c++)
                  for (int m = mm; m < min(mm + 1, 64); m++)
                    O[m][oy][ox] += I2[c][oy+fy-1][ox+fx-1]
                      * W2[m][c][fy][fx];
```

Cross-layer loop transform example (5/5)

- The second loop-nest will use the result calculated by the first loop-nest.
- The fourth loop-nest will use the result of the third loop-nest and a small part of the result of the first loop-nest.
- We choose to store the parts that still need to be used in DRAM after the first loop-nest is calculated and read this part of the result before the fourth loop-nest.



1st layer output



2nd layer output