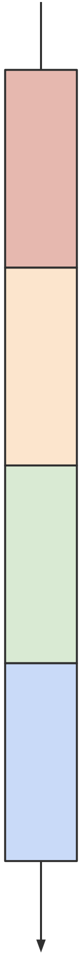
出门问问

Accelerating Deep Learning Inference

Approaches Overview: Concepts and Examples

Shen Li 2018.07.26

Deep Learning Dev Process



Task Modeling -

- Input and Output Representation
- Network Architecture Design

Model Training -

- Knowledge Distillation
- Transfer/Multi-task Learning

Model Conversion -

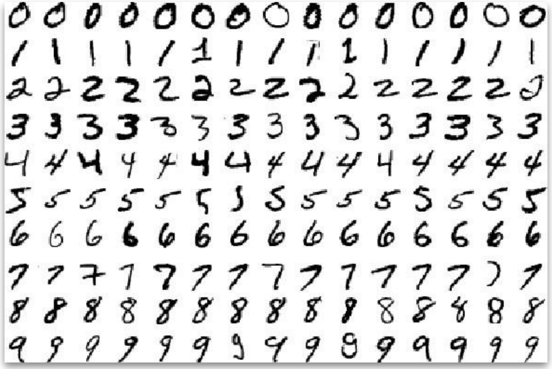
- Model Pruning and Quantization
- Computation Graph Optimization

Model Deployment -

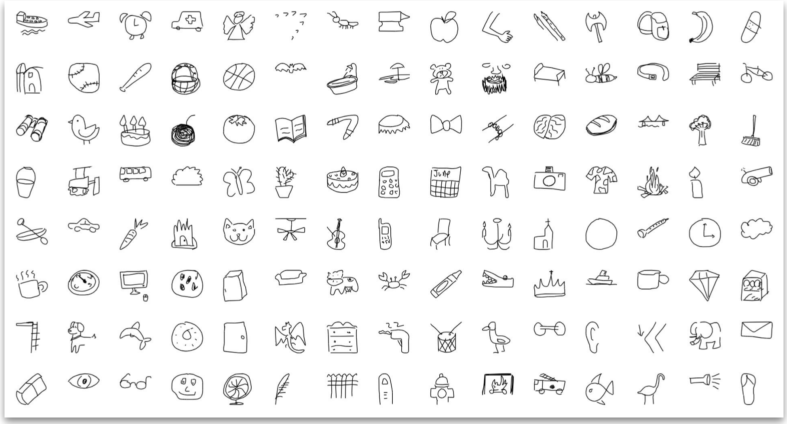
- Optimized Kernel Implementation
- Hardware Acceleration

Task Modeling

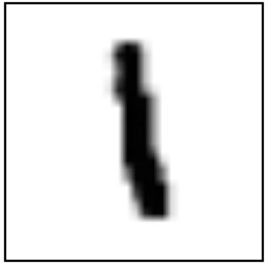
Efficient Representation for Inputs and Outputs



MNIST Handwritten Digits Dataset



Quick,Draw! Drawing Dataset

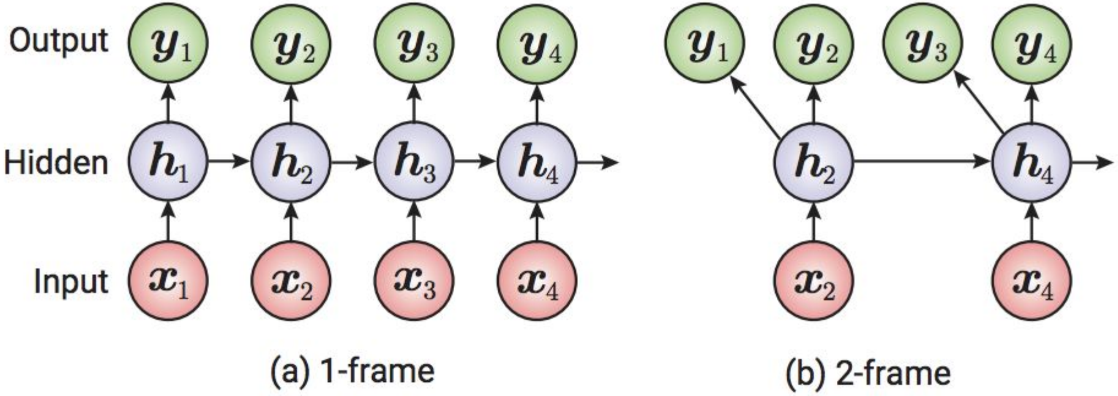


$$\simeq \begin{bmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&.6&.8&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\
0&0&0&0&0&0&.5&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.7&0&0&0&0&0\\
0&0&0&0&0&0&0&1&1&0&0&0&0&0\\
0&0&0&0&0&0&0&.9&1&.1&0&0&0&0\\
0&0&0&0&0&0&0&.3&1&.1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}$$

$$\begin{bmatrix}
0 & 0.2 & -0.3 & \dots & -0.7 & \dots \\
0 & 0.5 & 0.4 & \dots & 0.9 & \dots \\
1 & 0 & 0 & \dots & 1 & \dots
\end{bmatrix}$$

Task Modeling

Efficient Representation for Inputs and Outputs

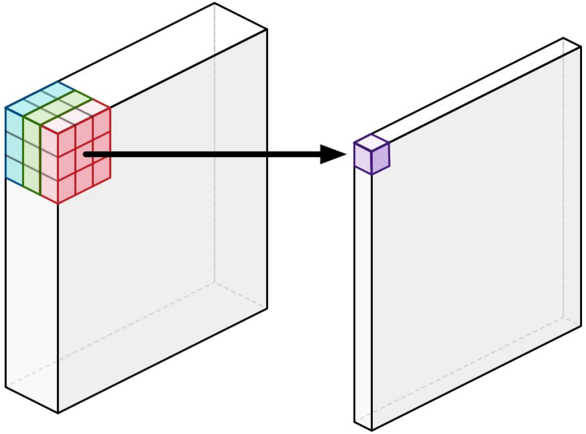


Multi-frame bundled inference for RNN

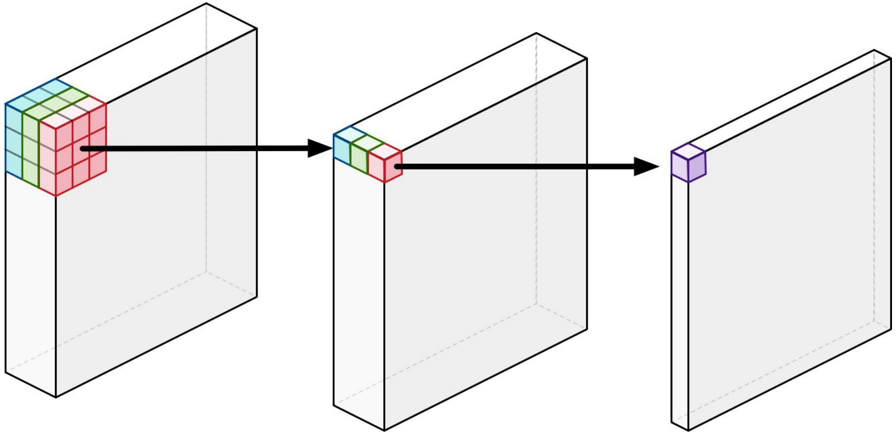
Zen, Heiga, et al. "Fast, compact, and high quality LSTM-RNN based statistical parametric speech synthesizers for mobile devices." *arXiv preprint arXiv:1606.06061* (2016).

Neural Network Design

Factorized Neural Network



Regular Convolution

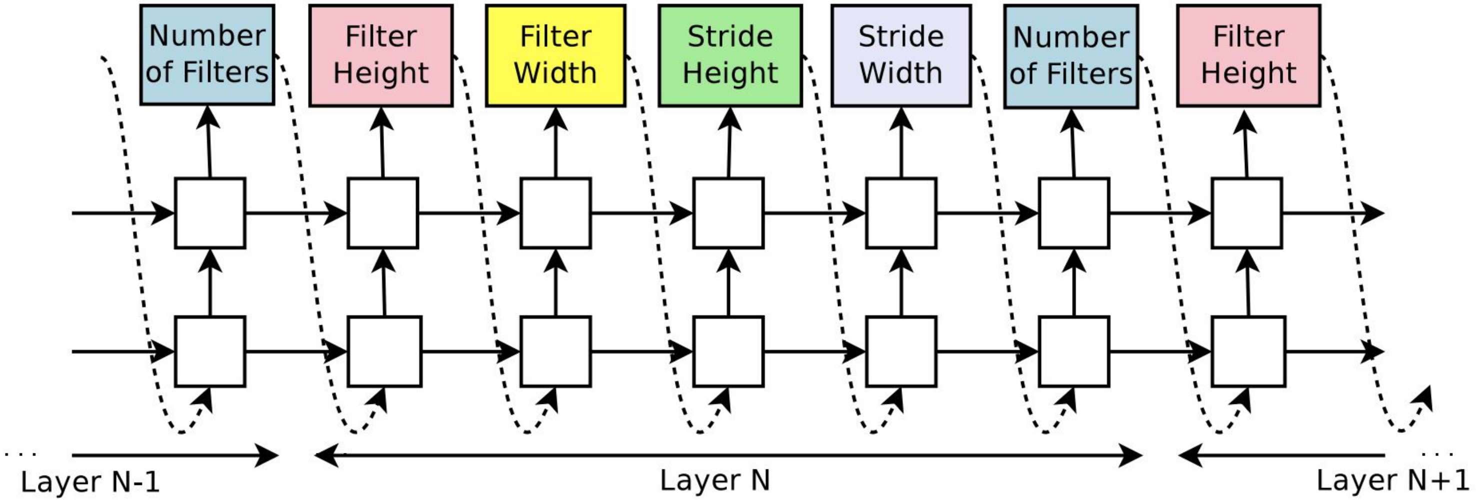


Depthwise Separable Convolution

http://machinethink.net/blog/googles-mobile-net-architecture-on-iphone/

Neural Network Design

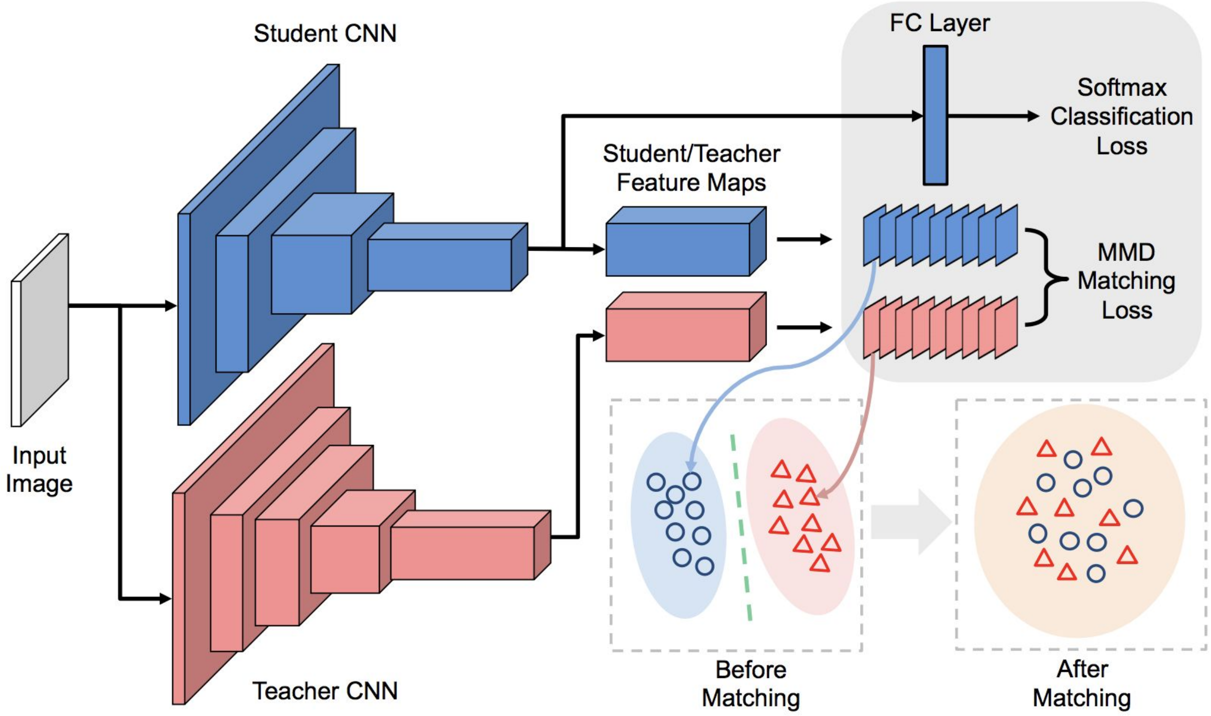
Neural Architecture Search



Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." arXiv preprint arXiv:1611.01578 (2016).

Knowledge Distillation

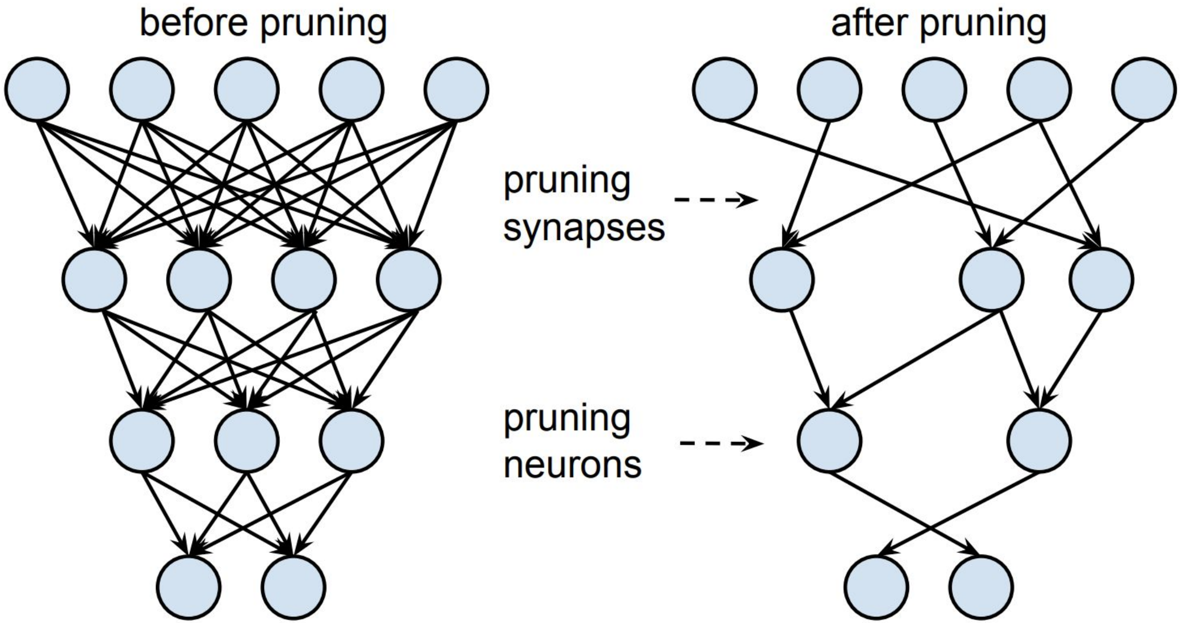
Guided Training Student Model from Teacher Model



Huang, Zehao, and Naiyan Wang. "Like what you like: Knowledge distill via neuron selectivity transfer." arXiv preprint arXiv:1707.01219 (2017).

Model Compression

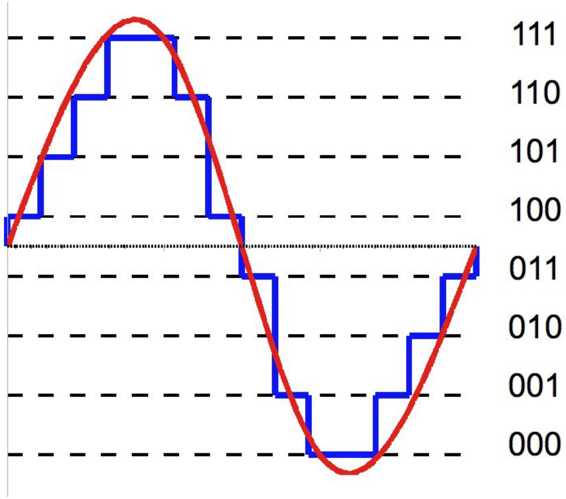
Neural Network Model Pruning

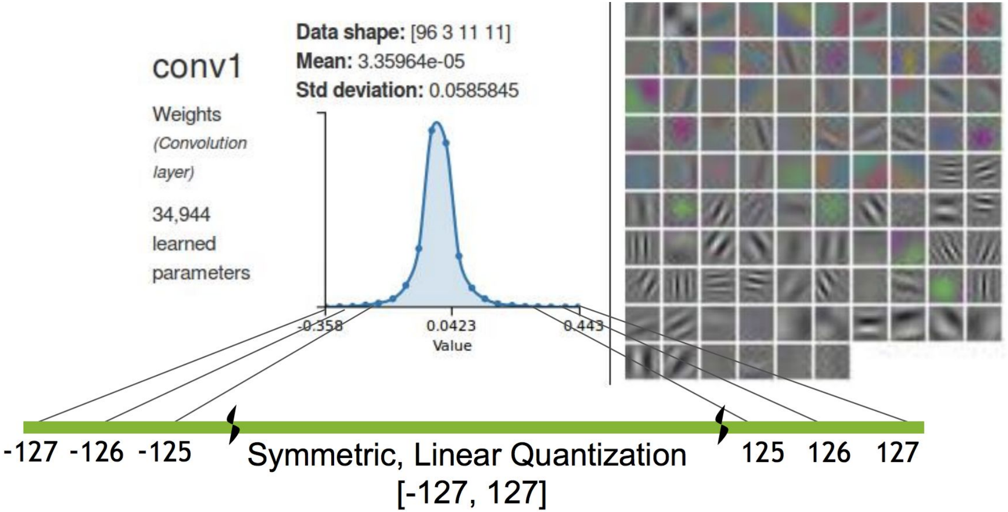


Han, Song, et al. "Learning both weights and connections for efficient neural network." Advances in neural information processing systems. 2015.

Model Compression

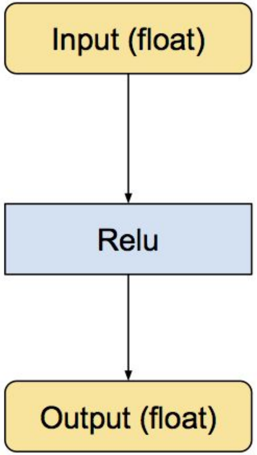
Model Quantization

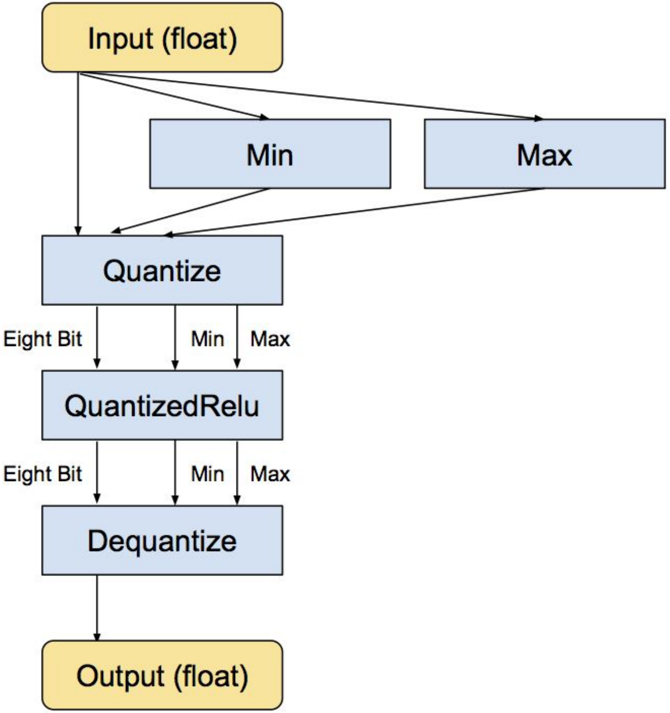




Model Compression

Model Quantization

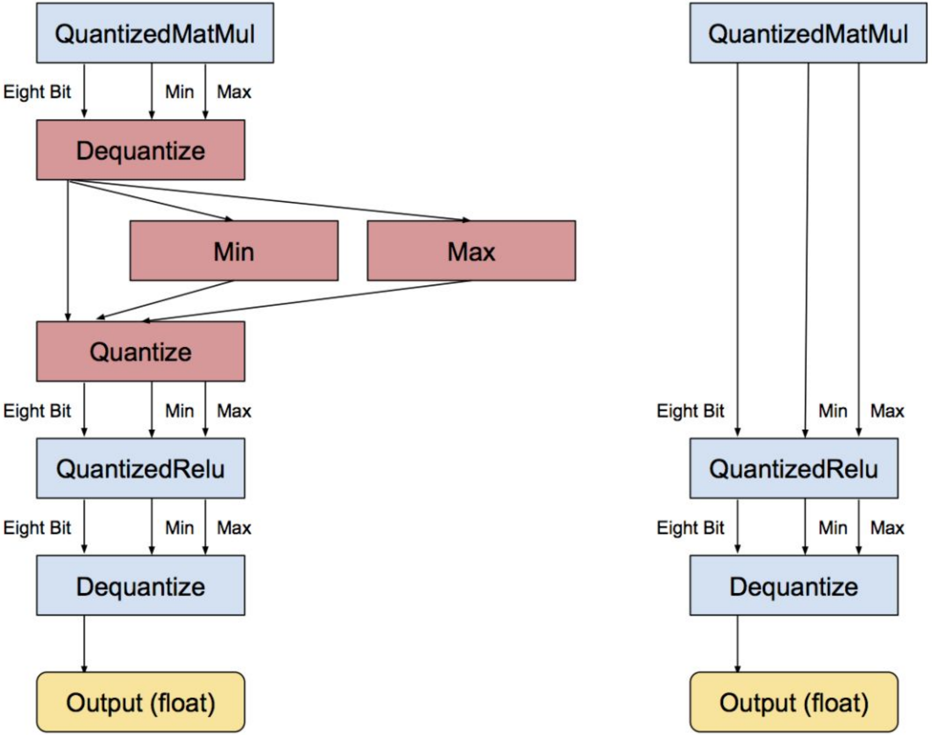




https://petewarden.com/2016/05/03/how-to-quantize-neural-networks-with-tensorflow/

Model Compression

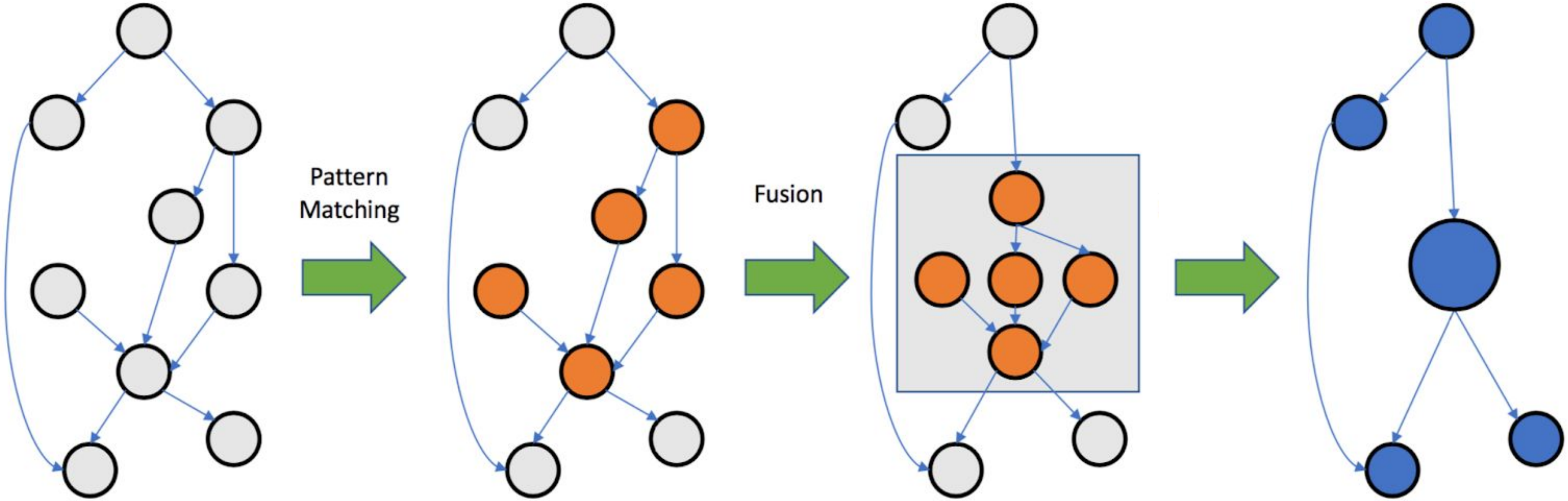
Model Quantization



https://petewarden.com/2016/05/03/how-to-quantize-neural-networks-with-tensorflow/

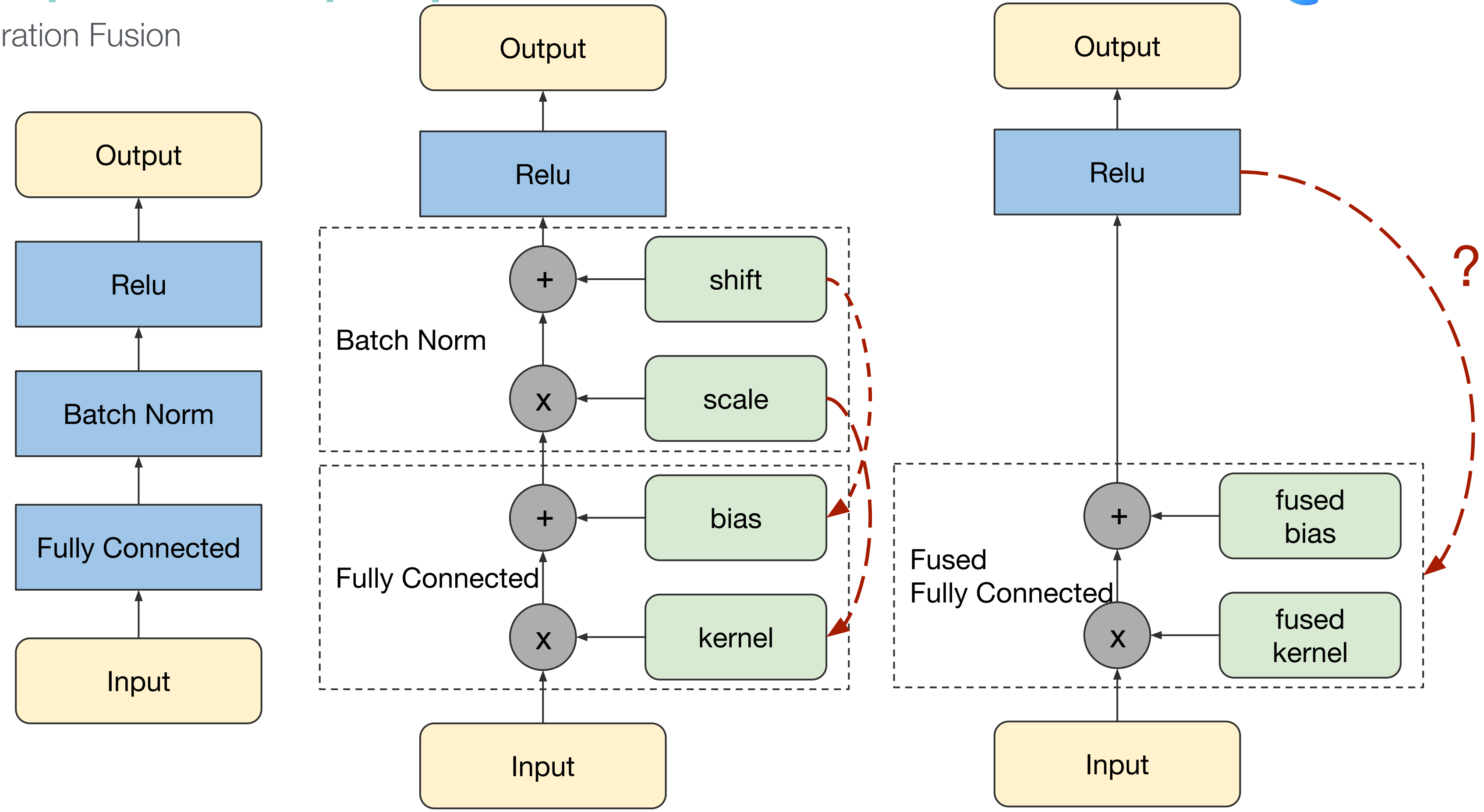
Computation Graph Optimization

Operation Fusion



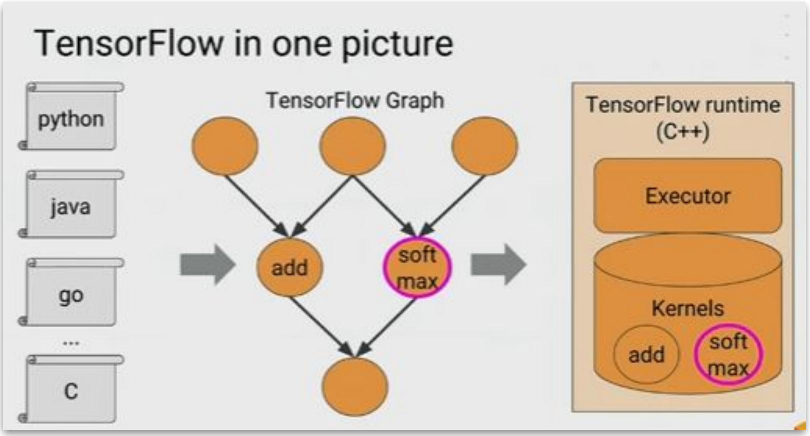
Computation Graph Optimization

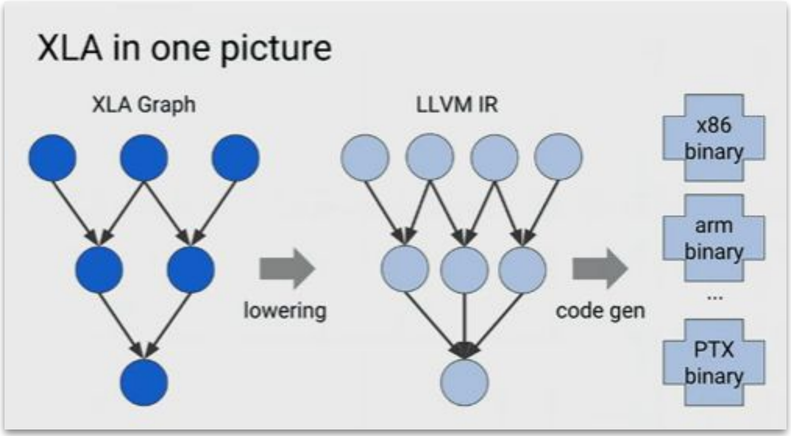
Operation Fusion



Computation Graph Optimization

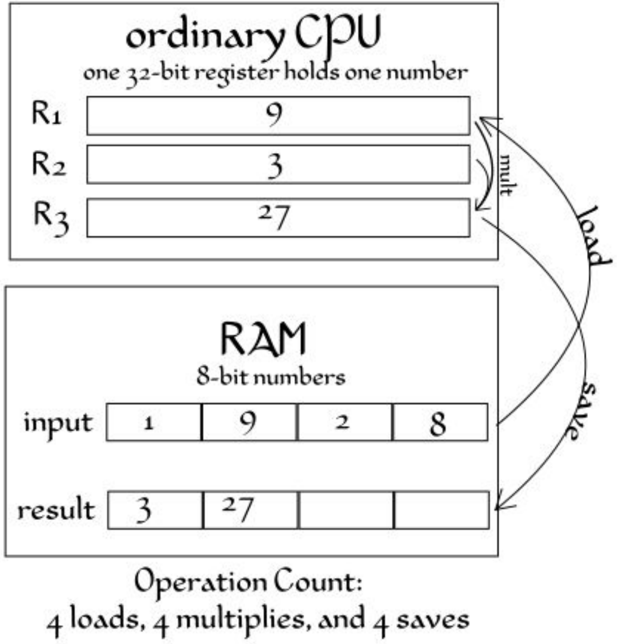
AOT & JIT Compilation



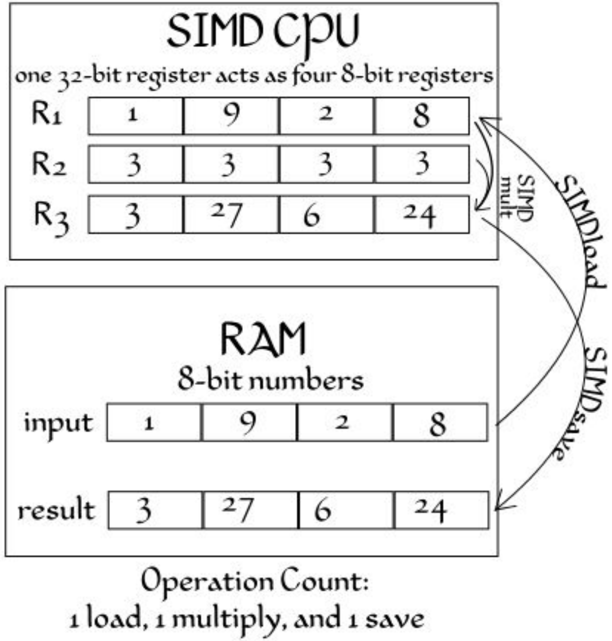


Optimized Implementation

Parallelism: Multi-thread, SIMD



Operation Count:
4 loads, 4 multiplies, and 4 saves

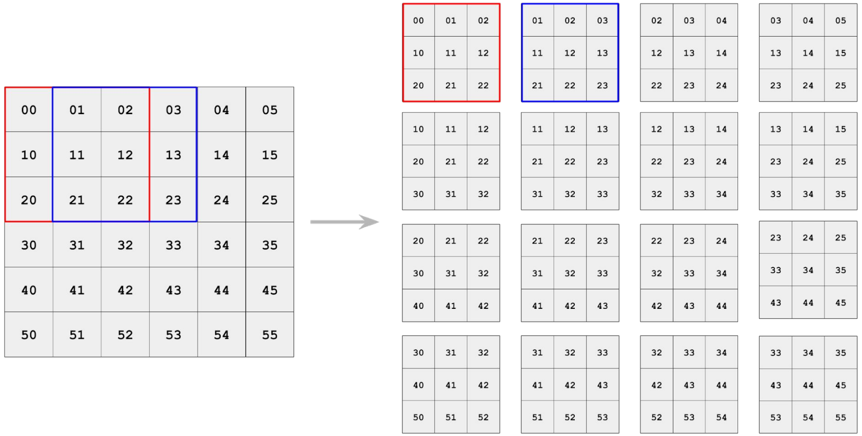


Operation Count:
1 load, 1 multiply, and 1 save

https://en.wikipedia.org/wiki/SIMD

Optimized Implementation

Data Locality



00	01	02	03	04	05
10	11	12	13	14	15
20	21	22	23	24	25
30	31	32	33	34	35
40	41	42	43	44	45
50	51	52	53	54	55

00	01	02
10	11	12
20	21	22

01	02	03
11	12	13
21	22	23

02	03	04
12	13	14
22	23	24

03	04	05
13	14	15
23	24	25

10	11	12
20	21	22
30	31	32

11	12	13
21	22	23
31	32	33

12	13	14
22	23	24
32	33	34

13	14	15
23	24	25
33	34	35

20	21	22
30	31	32
40	41	42

21	22	23
31	32	33
41	42	43

22	23	24
32	33	34
42	43	44

23	24	25
33	34	35
43	44	45

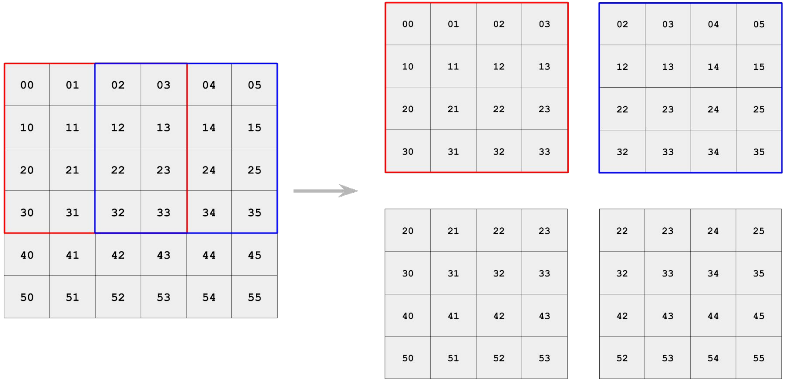
30	31	32
40	41	42
50	51	52

31	32	33
41	42	43
51	52	53

32	33	34
42	43	44
52	53	54

33	34	35
43	44	45
53	54	55

No Tiling



00	01	02	03	04	05
10	11	12	13	14	15
20	21	22	23	24	25
30	31	32	33	34	35
40	41	42	43	44	45
50	51	52	53	54	55

00	01	02	03
10	11	12	13
20	21	22	23
30	31	32	33

02	03	04	05
12	13	14	15
22	23	24	25
32	33	34	35

20	21	22	23
30	31	32	33
40	41	42	43
50	51	52	53

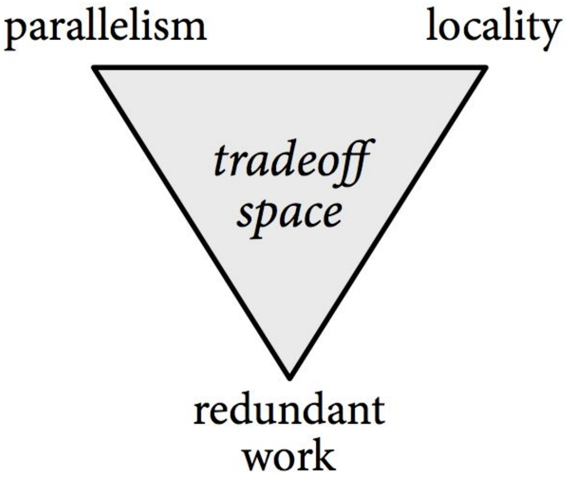
22	23	24	25
32	33	34	35
42	43	44	45
52	53	54	55

Tiling

https://tvm.ai/2017/08/22/Optimize-Deep-Learning-GPU-Operators-with-TVM-A-Depthwise-Convolution-Example.html

Optimized Implementation

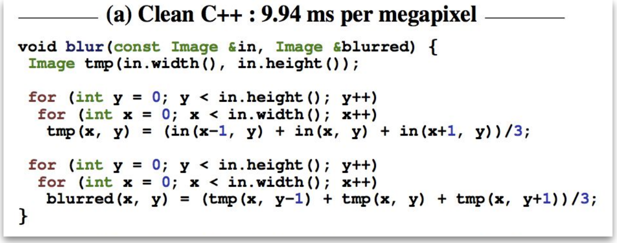
Tradeoffs between parallelism, locality, and redundant computation



Ragan-Kelley J M. Decoupling algorithms from the organization of computation for high performance image processing[D]. Massachusetts Institute of Technology, 2014.

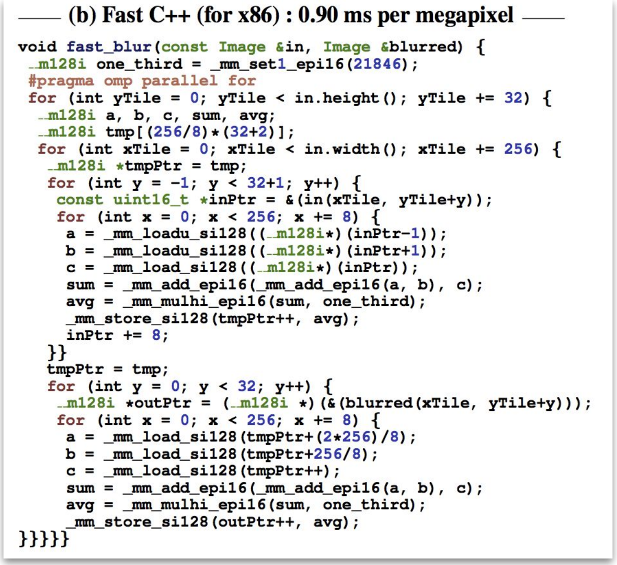
Optimized Implementation

Tools to optimize computation scheduling: Halide, TVM



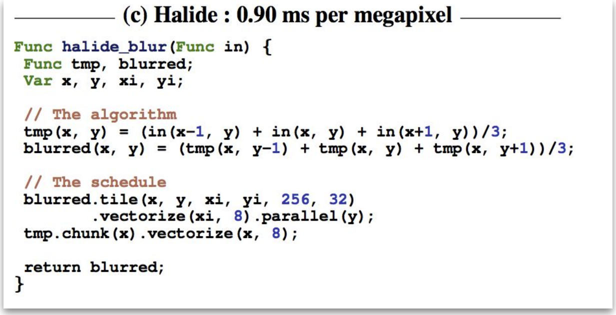
(a) Clean C++ : 9.94 ms per megapixel

```
void blur(const Image &in, Image &blurred) {
 Image tmp(in.width(), in.height());

 for (int y = 0; y < in.height(); y++)
  for (int x = 0; x < in.width(); x++)
   tmp(x, y) = (in(x-1, y) + in(x, y) + in(x+1, y))/3;

 for (int y = 0; y < in.height(); y++)
  for (int x = 0; x < in.width(); x++)
   blurred(x, y) = (tmp(x, y-1) + tmp(x, y) + tmp(x, y+1))/3;
}
```



(b) Fast C++ (for x86) : 0.90 ms per megapixel

```
void fast_blur(const Image &in, Image &blurred) {
 __m128i one_third = _mm_set1_epi16(21846);
 #pragma omp parallel for
 for (int yTile = 0; yTile < in.height(); yTile += 32) {
  __m128i a, b, c, sum, avg;
  __m128i tmp[(256/8)*(32+2)];
  for (int xTile = 0; xTile < in.width(); xTile += 256) {
   __m128i *tmpPtr = tmp;
   for (int y = -1; y < 32+1; y++) {
    const uint16_t *inPtr = &(in(xTile, yTile+y));
    for (int x = 0; x < 256; x += 8) {
     a = _mm_loadu_si128((__m128i*)(inPtr-1));
     b = _mm_loadu_si128((__m128i*)(inPtr+1));
     c = _mm_load_si128((__m128i*)(inPtr));
     sum = _mm_add_epi16(_mm_add_epi16(a, b), c);
     avg = _mm_mulhi_epi16(sum, one_third);
     _mm_store_si128(tmpPtr++, avg);
     inPtr += 8;
   }}
   tmpPtr = tmp;
   for (int y = 0; y < 32; y++) {
    __m128i *outPtr = (__m128i *)(&(blurred(xTile, yTile+y)));
    for (int x = 0; x < 256; x += 8) {
     a = _mm_load_si128(tmpPtr+(2*256)/8);
     b = _mm_load_si128(tmpPtr+256/8);
     c = _mm_load_si128(tmpPtr++);
     sum = _mm_add_epi16(_mm_add_epi16(a, b), c);
     avg = _mm_mulhi_epi16(sum, one_third);
     _mm_store_si128(outPtr++, avg);
}}}}}
```



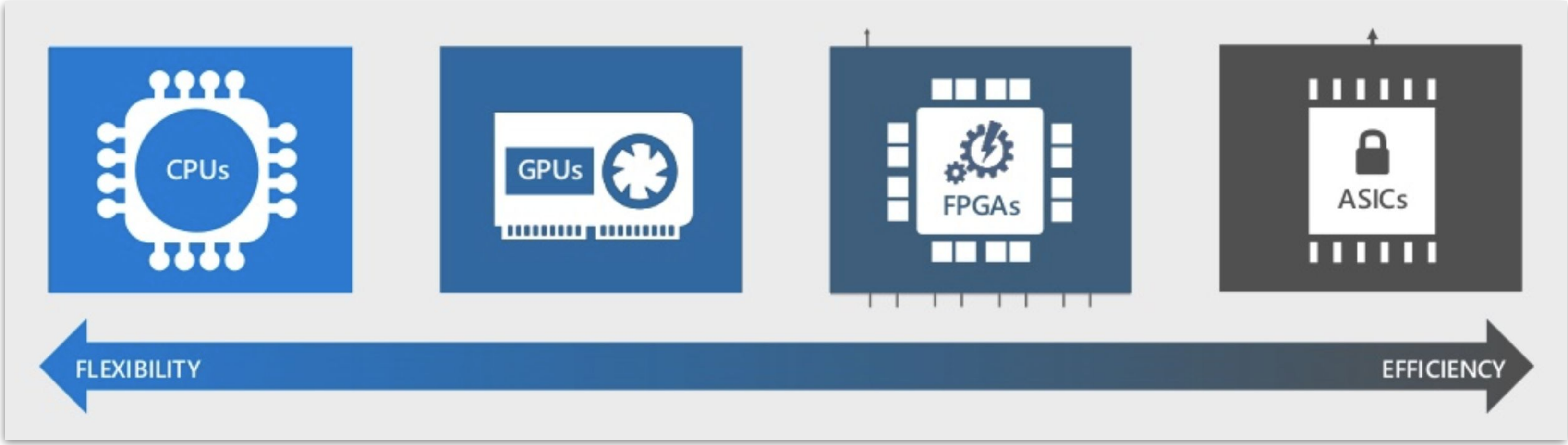
(c) Halide : 0.90 ms per megapixel

```
Func halide_blur(Func in) {
 Func tmp, blurred;
 Var x, y, xi, yi;

 // The algorithm
 tmp(x, y) = (in(x-1, y) + in(x, y) + in(x+1, y))/3;
 blurred(x, y) = (tmp(x, y-1) + tmp(x, y) + tmp(x, y+1))/3;

 // The schedule
 blurred.tile(x, y, xi, yi, 256, 32)
        .vectorize(xi, 8).parallel(y);
 tmp.chunk(x).vectorize(x, 8);

 return blurred;
}
```

Ragan-Kelley J, Adams A, Paris S, et al. Decoupling algorithms from schedules for easy optimization of image processing pipelines[J]. 2012.

Hardware Acceleration

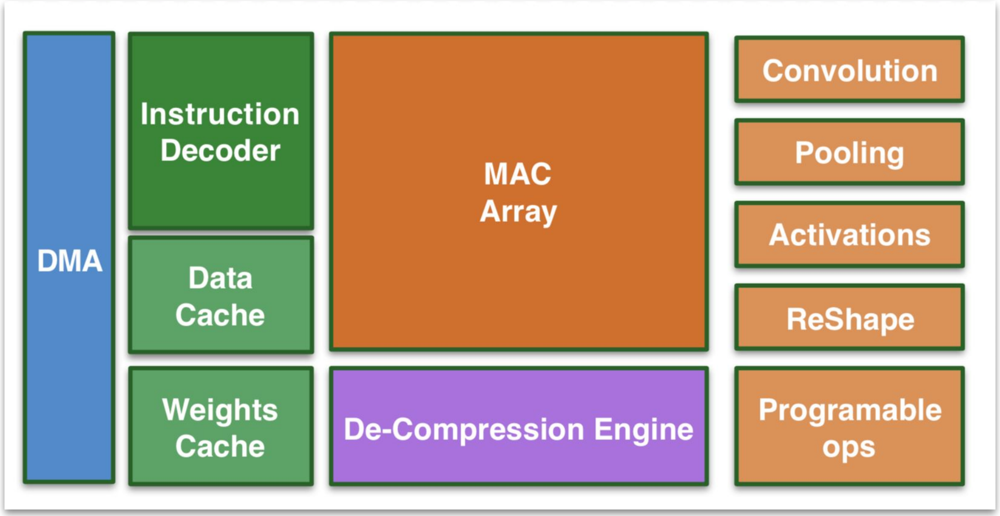
GPU, FPGA, ASIC



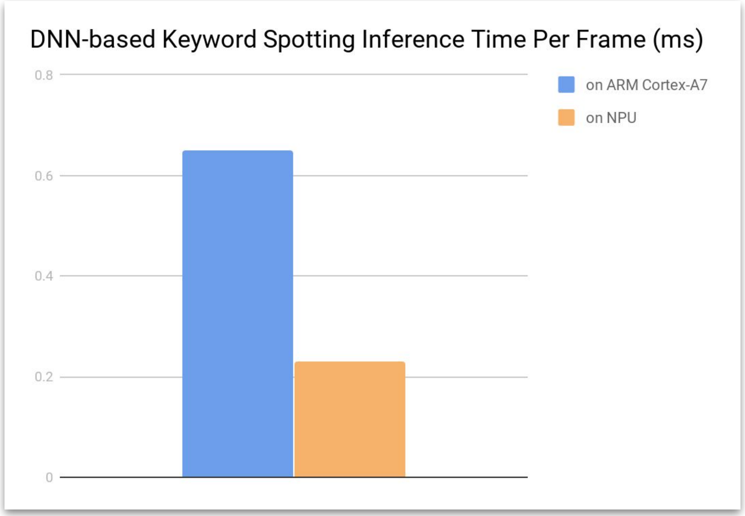
Mark Russinovich, “Inside Microsoft's FPGA-Based Configurable Cloud”, Microsoft Build 2017

Hardware Acceleration

Neural Processor Unit in Mobvoi A1



Neural Processor Unit (by NationalChip)



Inference on NPU achieve ~3x speed up compared with ARM Cortex-A7 Dual 1.2GHz CPU.

Innovation Mindset

Think 10x