Compilation and Hardware Support for Approximate Acceleration

Thierry Moreau, Adrian Sampson, Andre Baixo,
Mark Wyse, Ben Ransford, Jacob Nelson,
Hadi Esmaeilzadeh (Georgia Tech), Luis Ceze and Mark Oskin
University of Washington
moreau@uw.edu

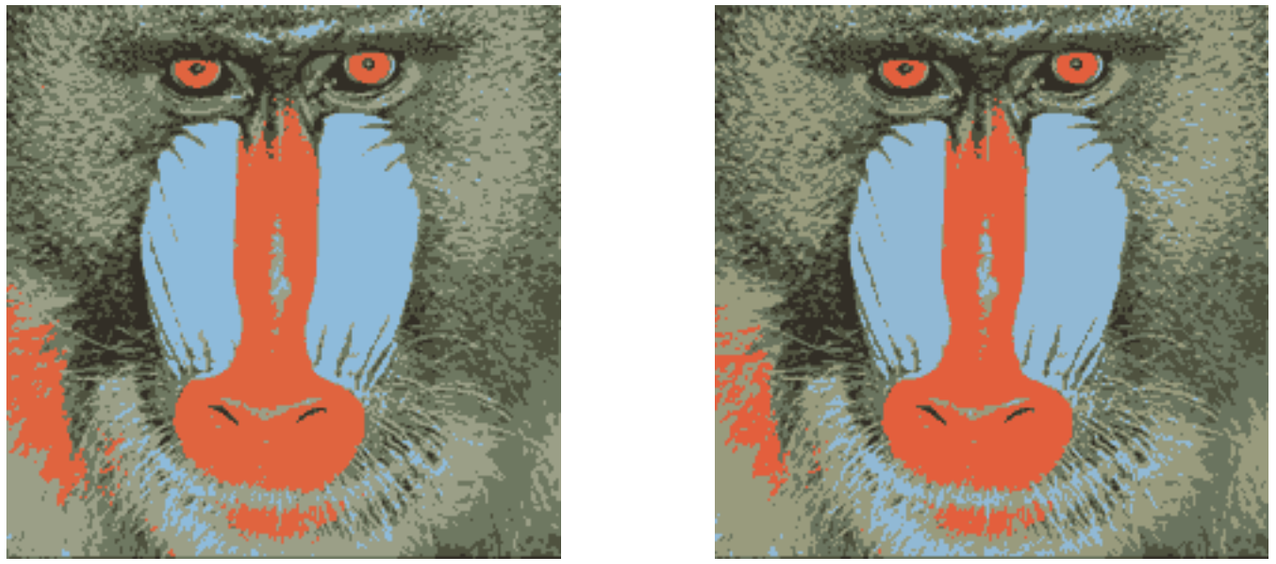
Semiconductor Research Corporation
STARnet

Theme: 2384.004

Approximate Computing

Aims to exploit **application resilience** to **trade-off quality** for **efficiency**

Approximate Computing



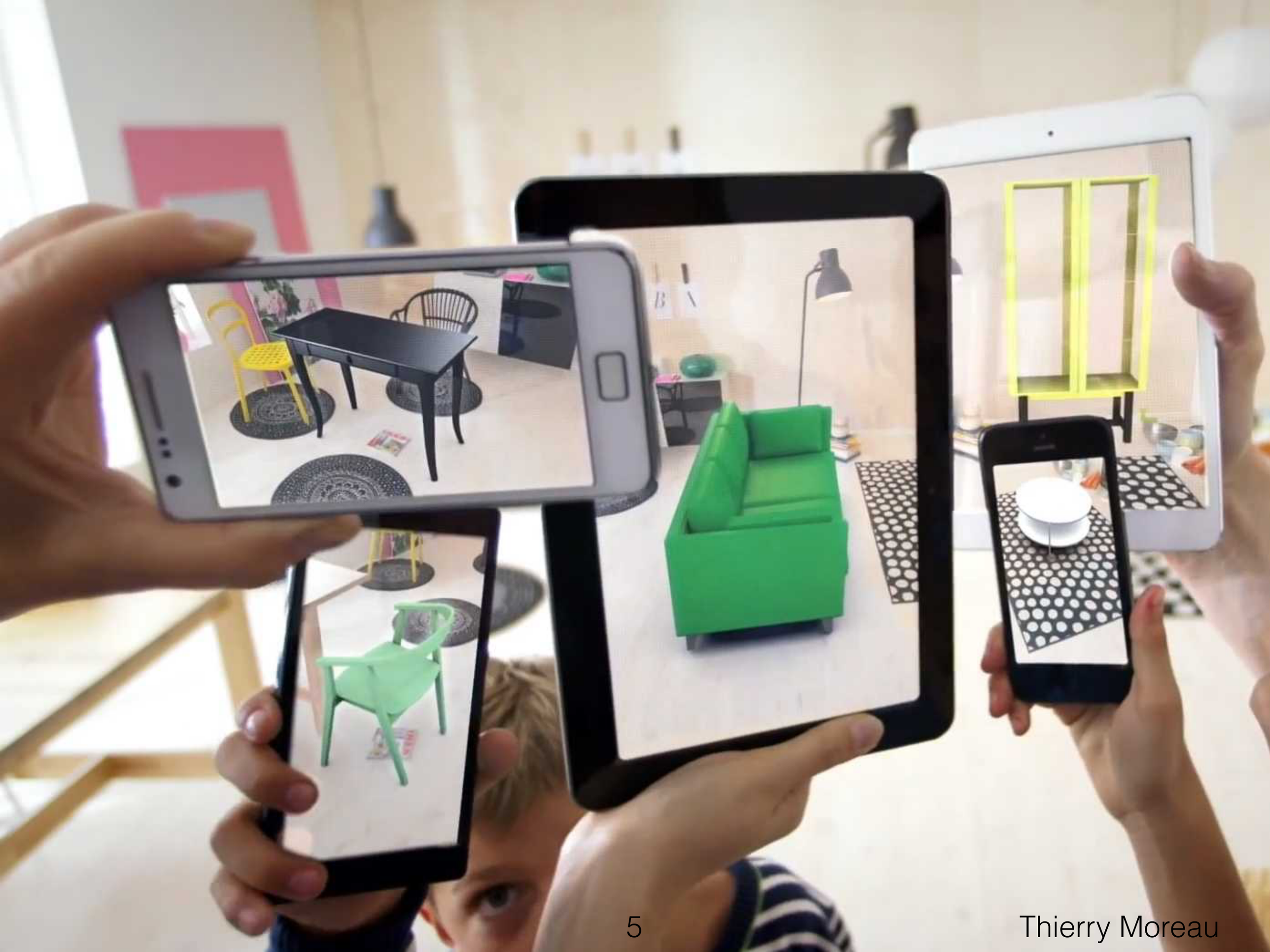
Approximate Computing

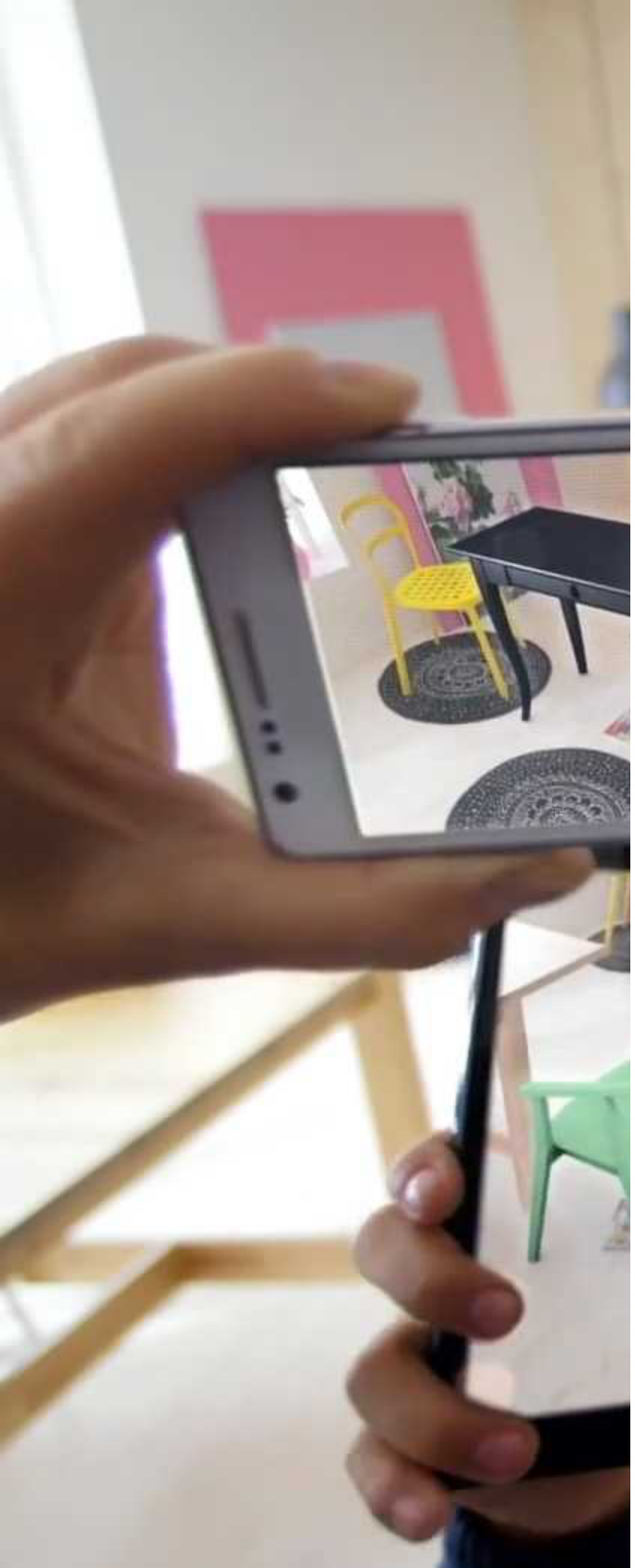
✅ Accurate

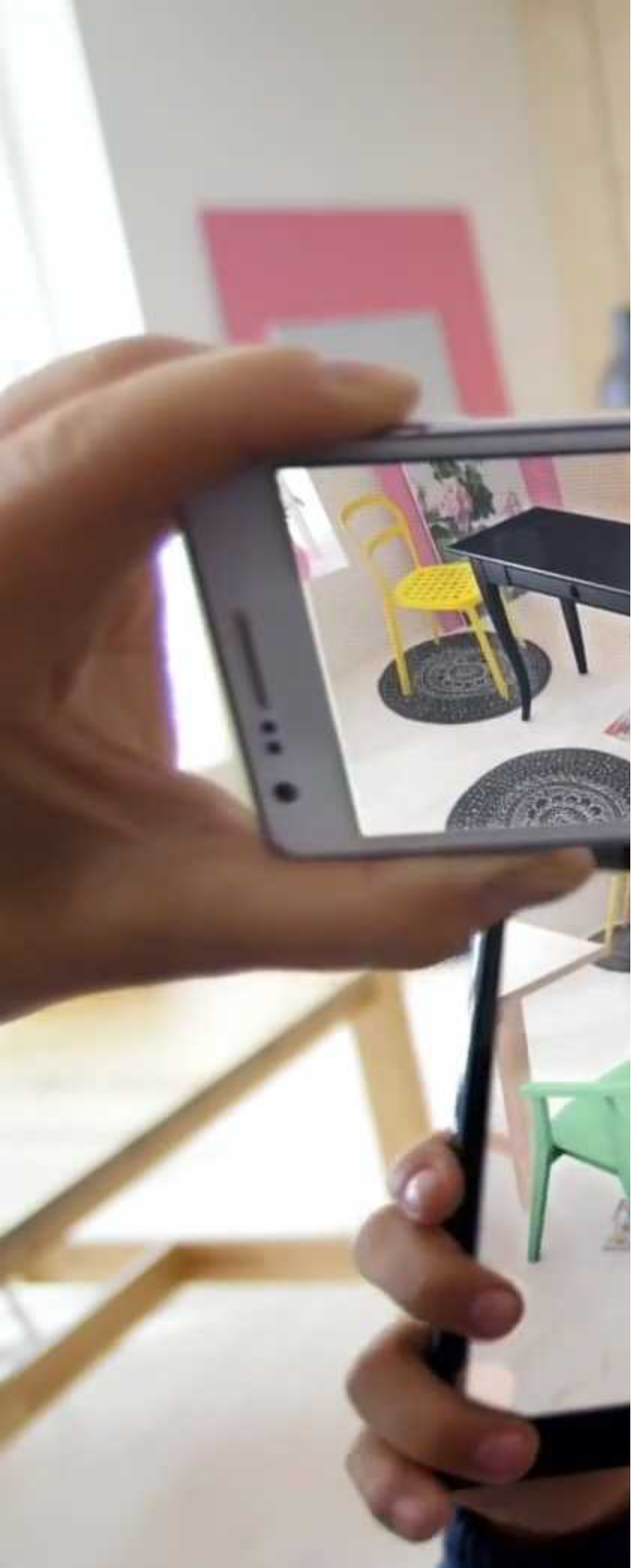
❌ Expensive

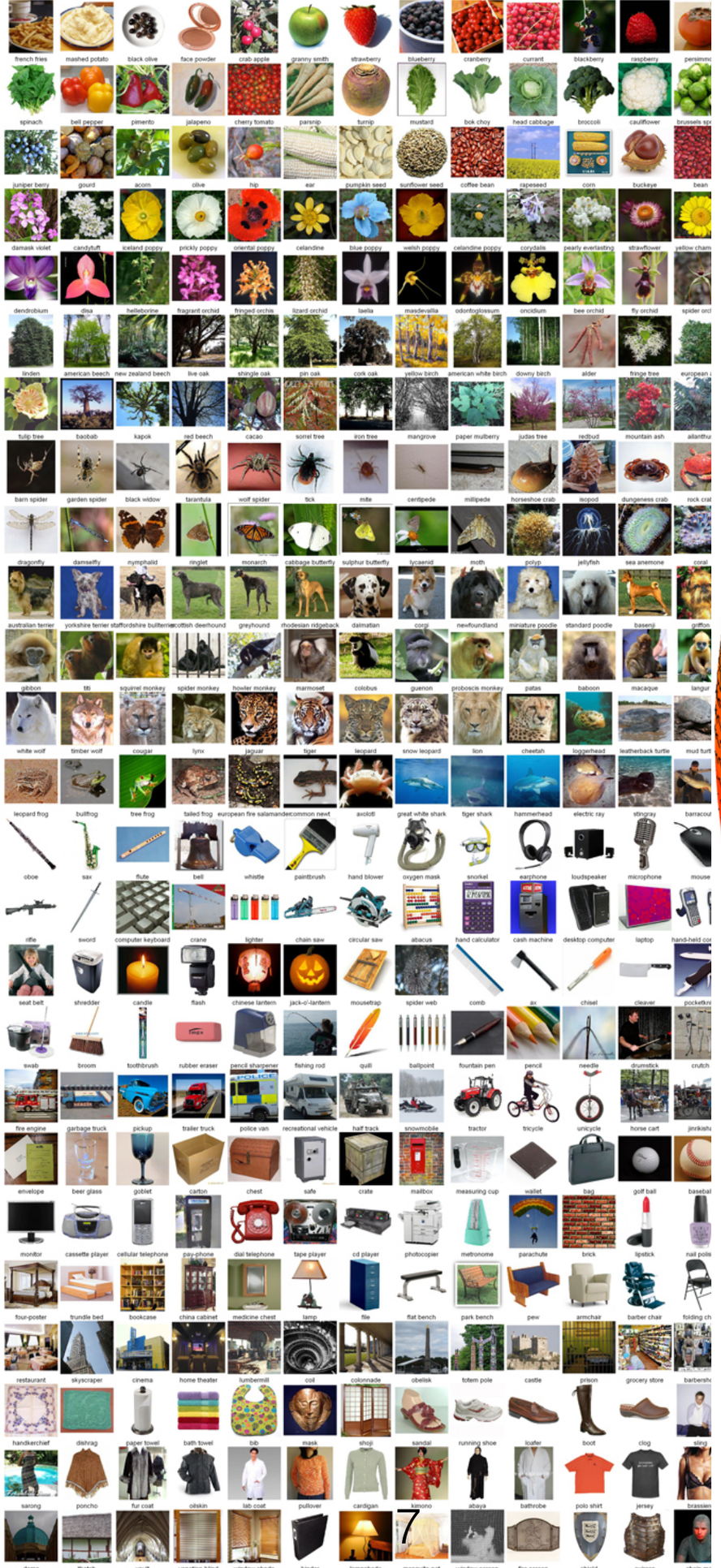
✅ Approximate

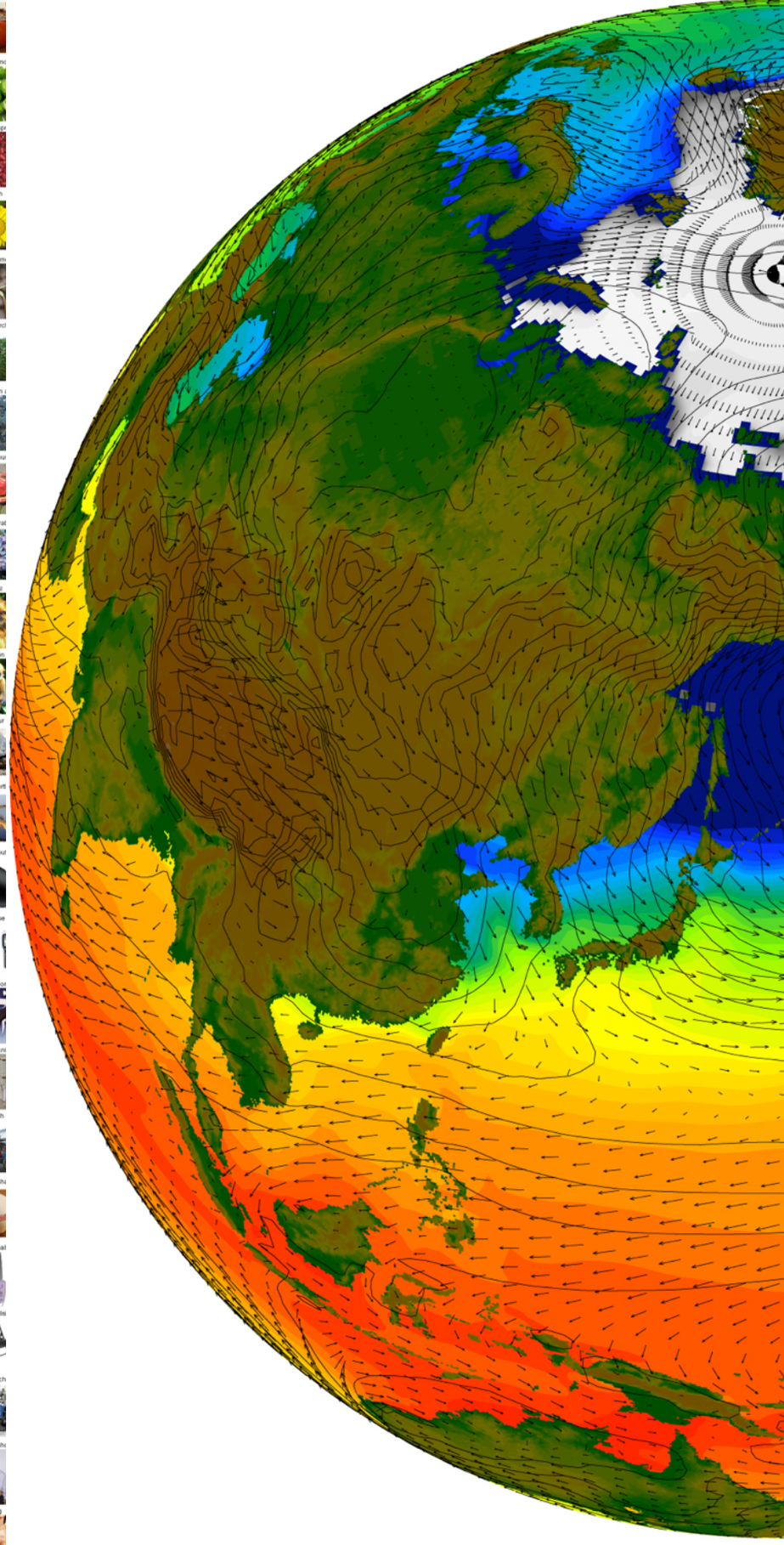
✅ Cheap



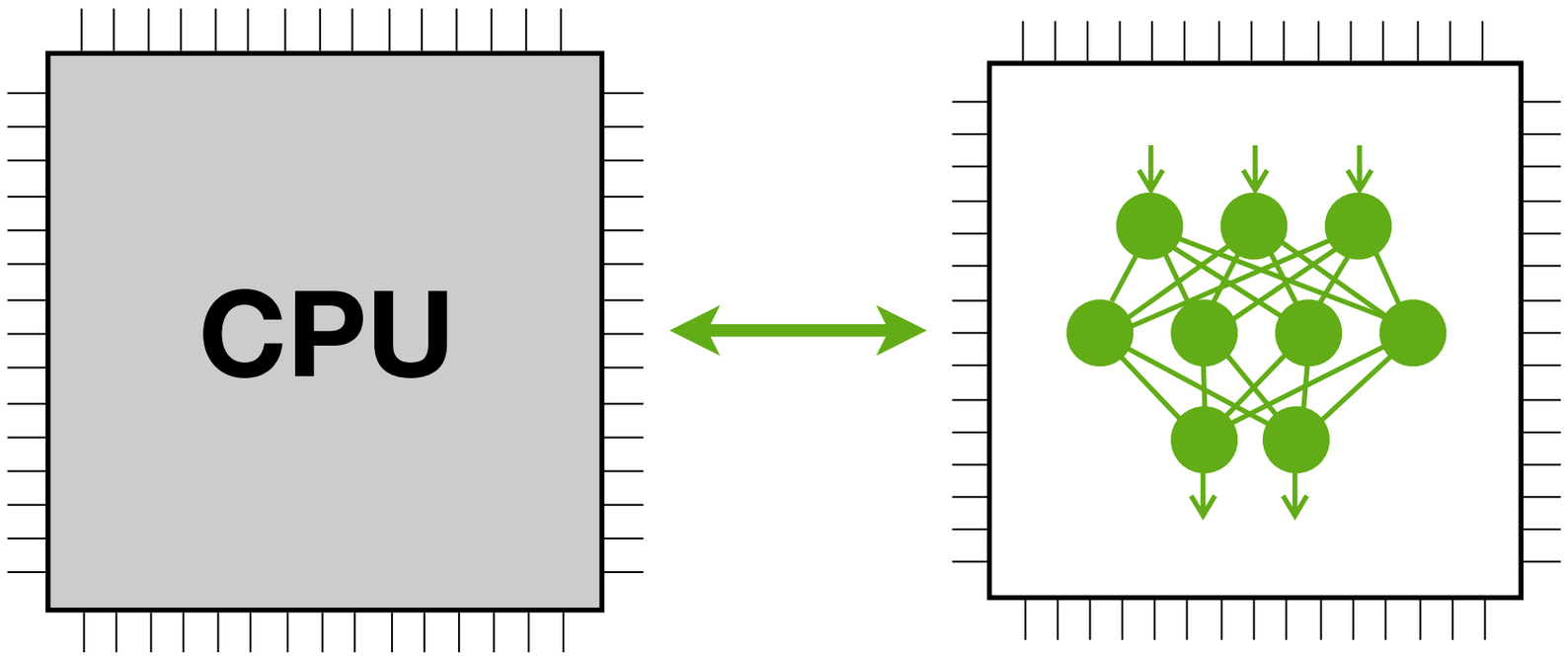








Neural Networks as Approximate Accelerators



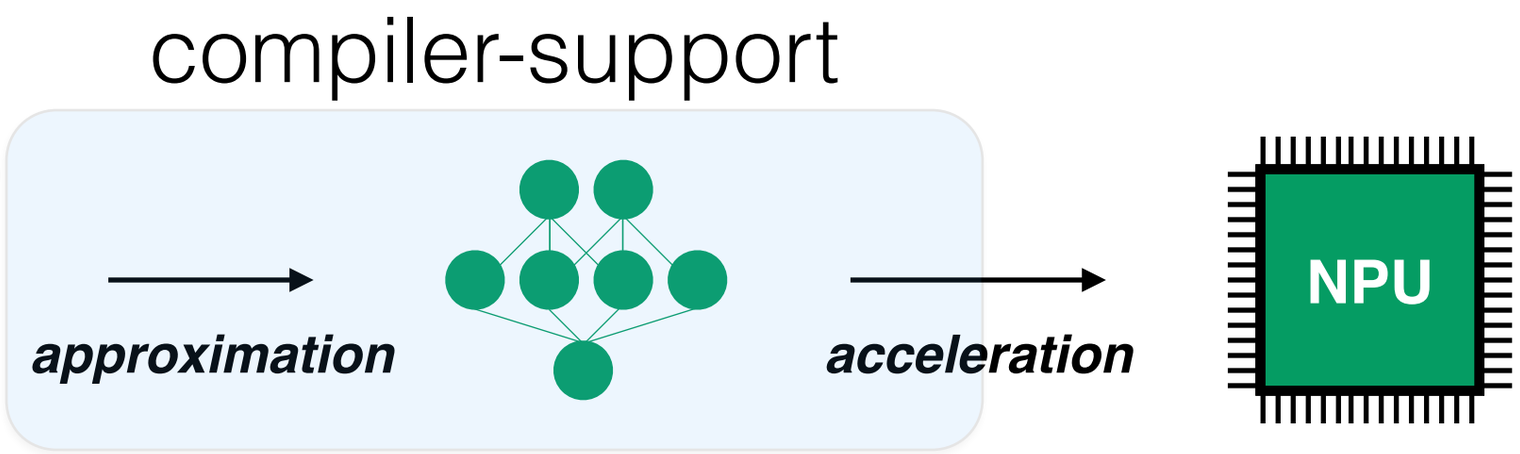
Esmaeilzadeh et al.
[MICRO 2012]

Neural Acceleration

```
float foo (float a, float b)
{
    …
    return val;
}
```

Neural Acceleration

```
float foo (float a, float b)
{
    ...
    return val;
}
```

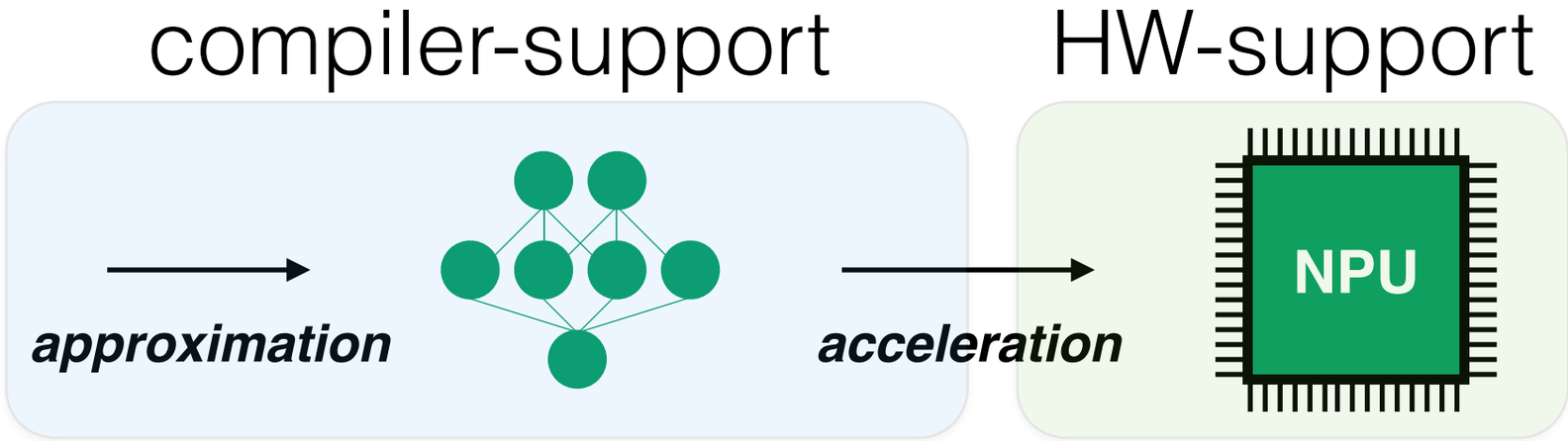


ACCEPT*

*Sampson et. al [UW-TR]

Neural Acceleration

```
float foo (float a, float b)
{
    ...
    return val;
}
```



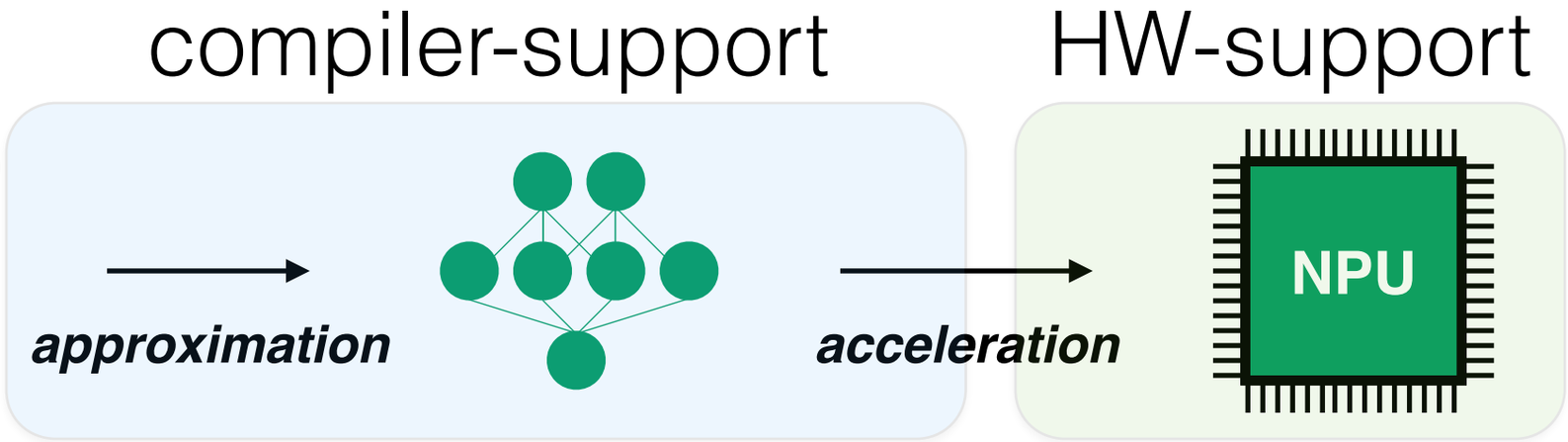
ACCEPT

SNNAP*

*Moreau et. al [HPCA2015]

Neural Acceleration

```
float foo (float a, float b)
{
    ...
    return val;
}
```



3.8x speedup and 2.8x efficiency - 10% error

Talk Outline

Introduction

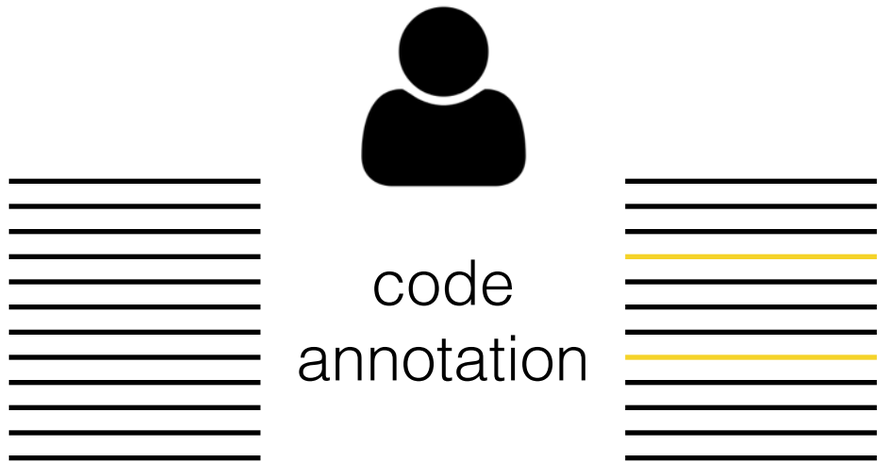
Compiler Support with ACCEPT

SNNAP Accelerator design

Evaluation & Comparison with HLS

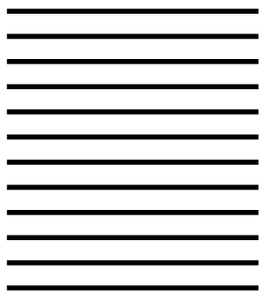
Compilation Overview

1. Region detection

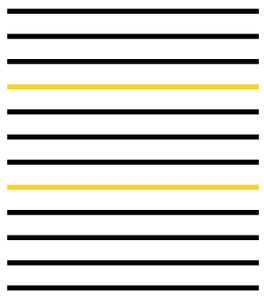


Compilation Overview

1. Region detection

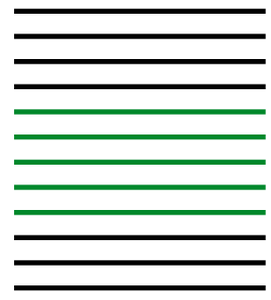


code annotation



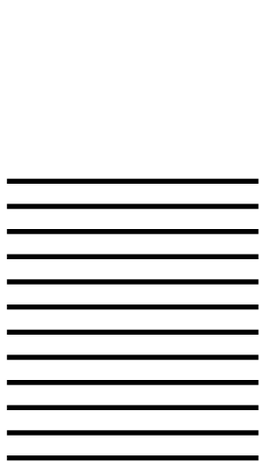
ACCEPT

region detection & program instrumentation

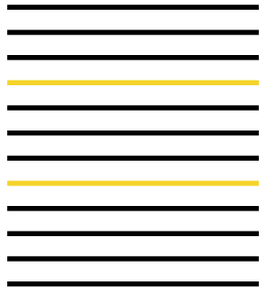


Compilation Overview

1. Region detection

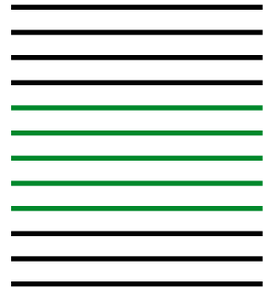


code annotation



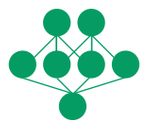
ACCEPT

region detection & program instrumentation



2. ANN Training

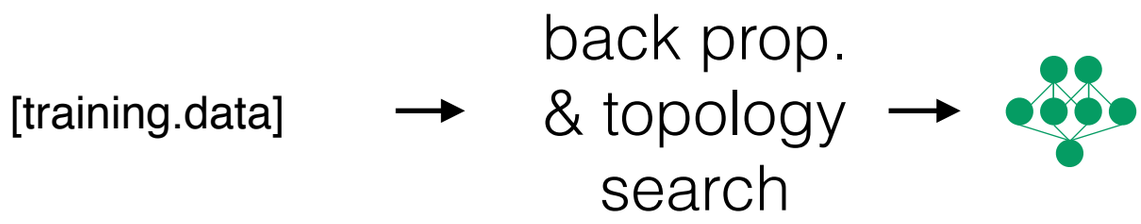
[training.data] → back prop. & topology search →



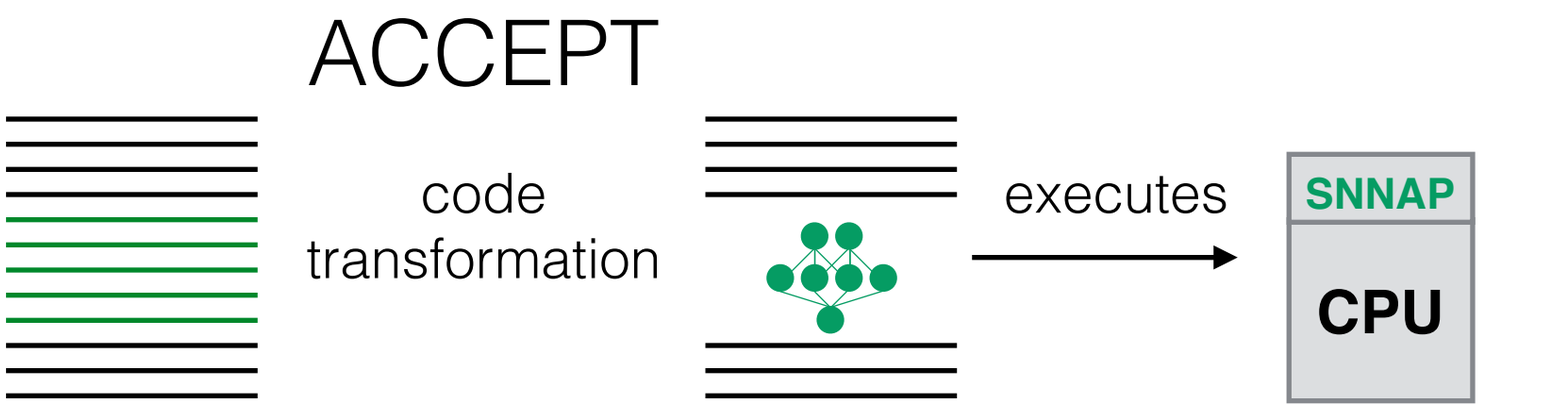
Compilation Overview

1. Region detection

2. ANN Training



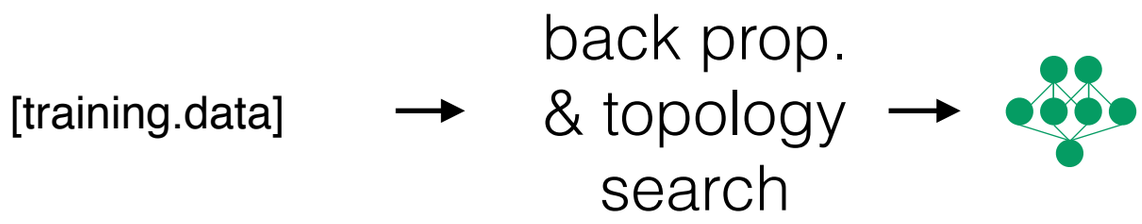
3. Code Generation



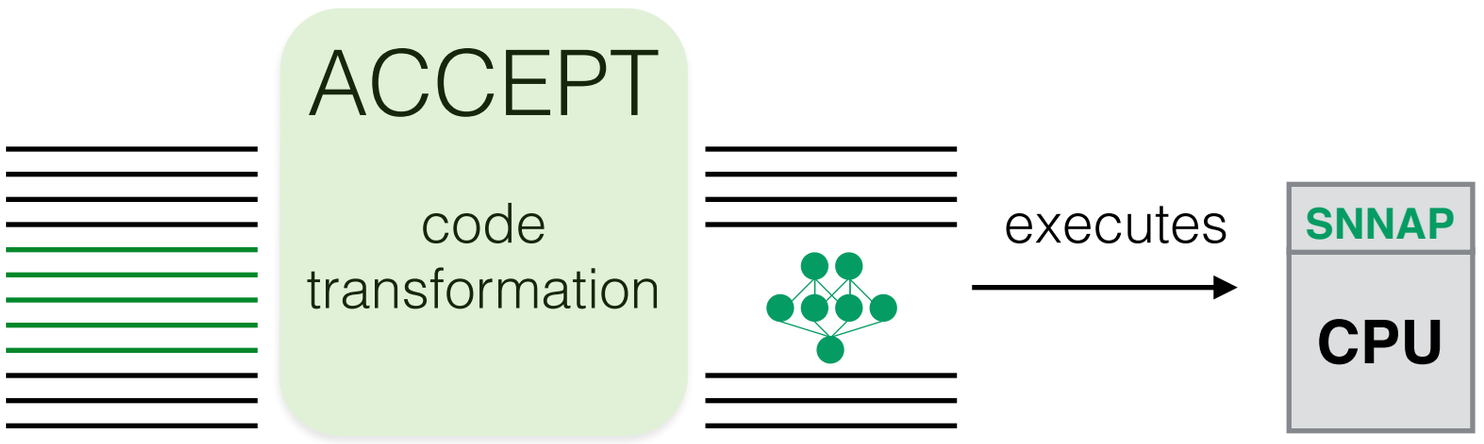
Compilation Overview

1. Region detection

2. ANN Training



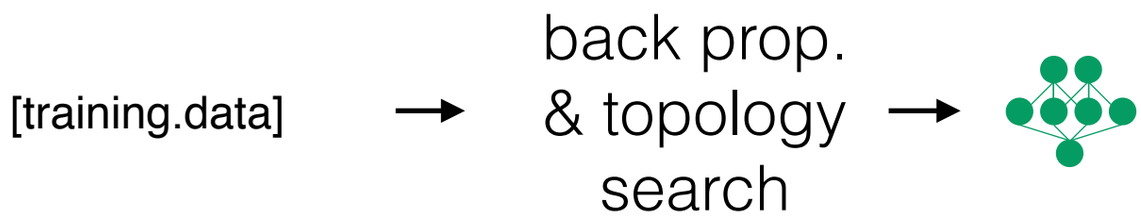
3. Code Generation



Compilation Overview

1. Region detection

2. ANN Training



3. Code Generation

Programming Model

sobel

```
float sobel (float* p);
. . .

float** src;
float** dst;

while (true) {
   src = read_from_camera();
   for (y=0; y < h; ++y) {
      for (x=0; x < w; ++x) {
         dst[y][x] = sobel(& src[y][x]);
      }
   }
   display(dst);
}
```

Programming Model

sobel

```
APPROX float sobel (APPROX float* p);
. . .

APPROX float** src;
APPROX float** dst;

while (true) {
   src = read_from_camera();
   for (y=0; y < h; ++y) {
      for (x=0; x < w; ++x) {
         dst[y][x] = sobel(& src[y][x]);
      }
   }
   display(ENDORSE(dst));
}
```

Programming Model

sobel ↓

```
APPROX float sobel (APPROX float* p);
. . .

APPROX float** src;
APPROX float** dst;

while (true) {
  src = read_from_camera();
  for (y=0; y < h; ++y) {
    for (x=0; x < w; ++x) {
      dst[y][x] = sobel(& src[y][x]);
    }
  }
  display(ENDORSE(dst));
}
```

✅ no side effects

✅ executes often

Checking for Quality

annotated
program

```
sobel.c
```

Checking for Quality

annotated program

`sobel.c`

quality metric

$d(y, y')$

Checking for Quality

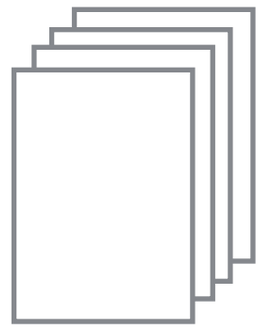
annotated program

`sobel.c`

quality metric

$d(y, y')$

input data



Checking for Quality

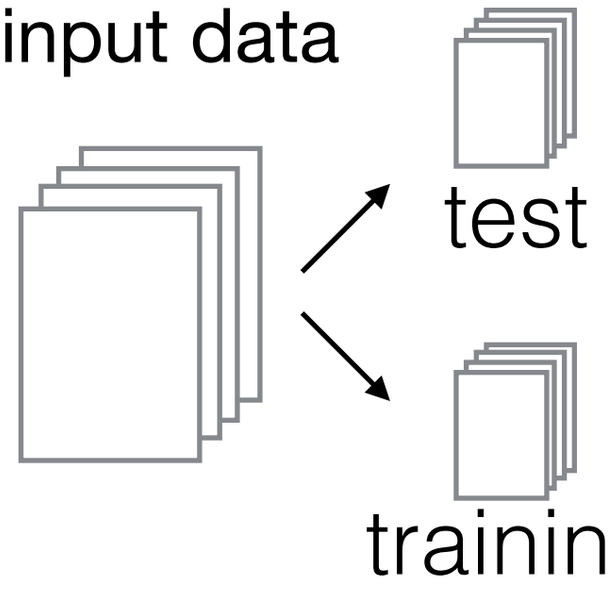
annotated program

```
sobel.c
```

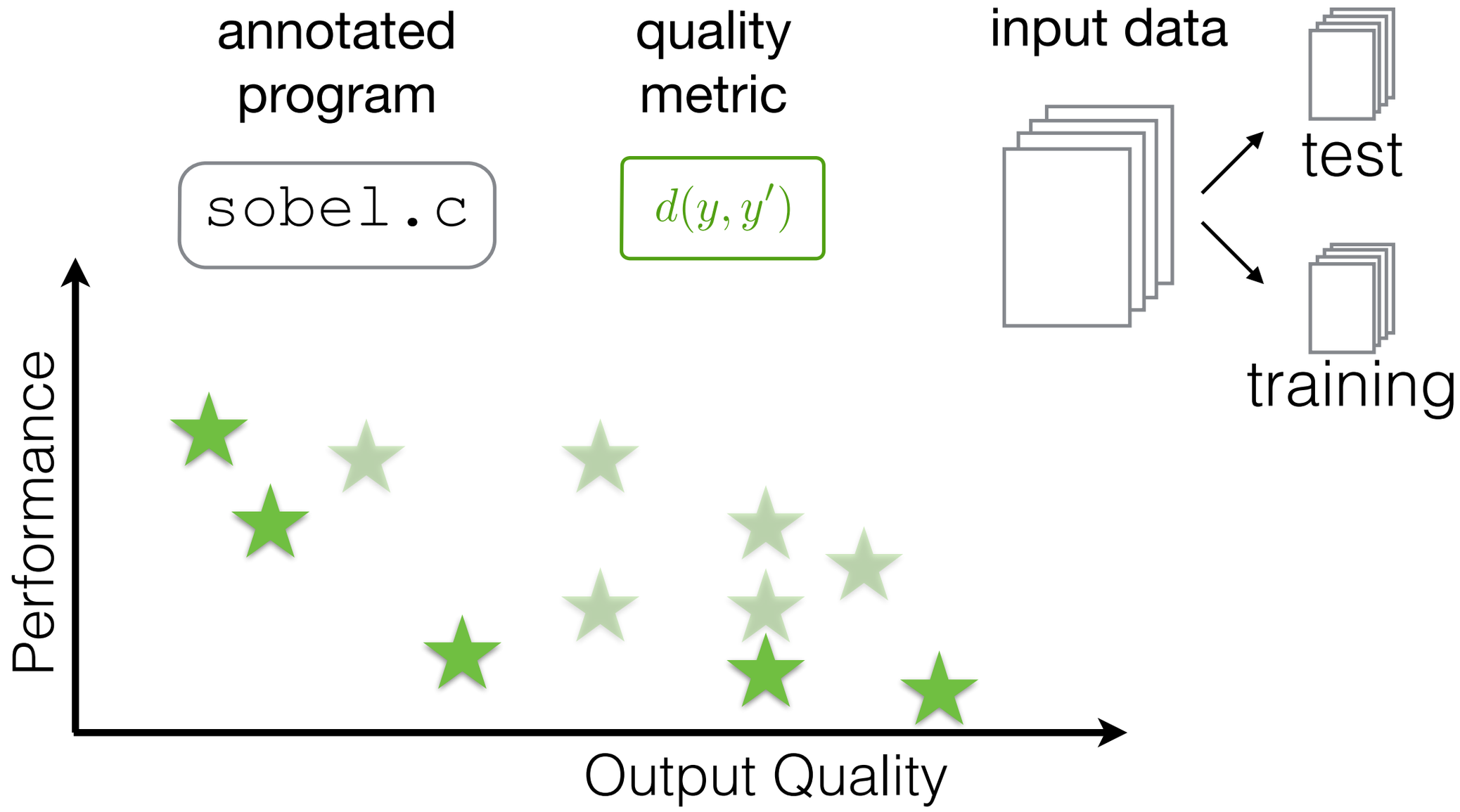
quality metric

$d(y, y')$

input data



Checking for Quality



Talk Outline

Introduction

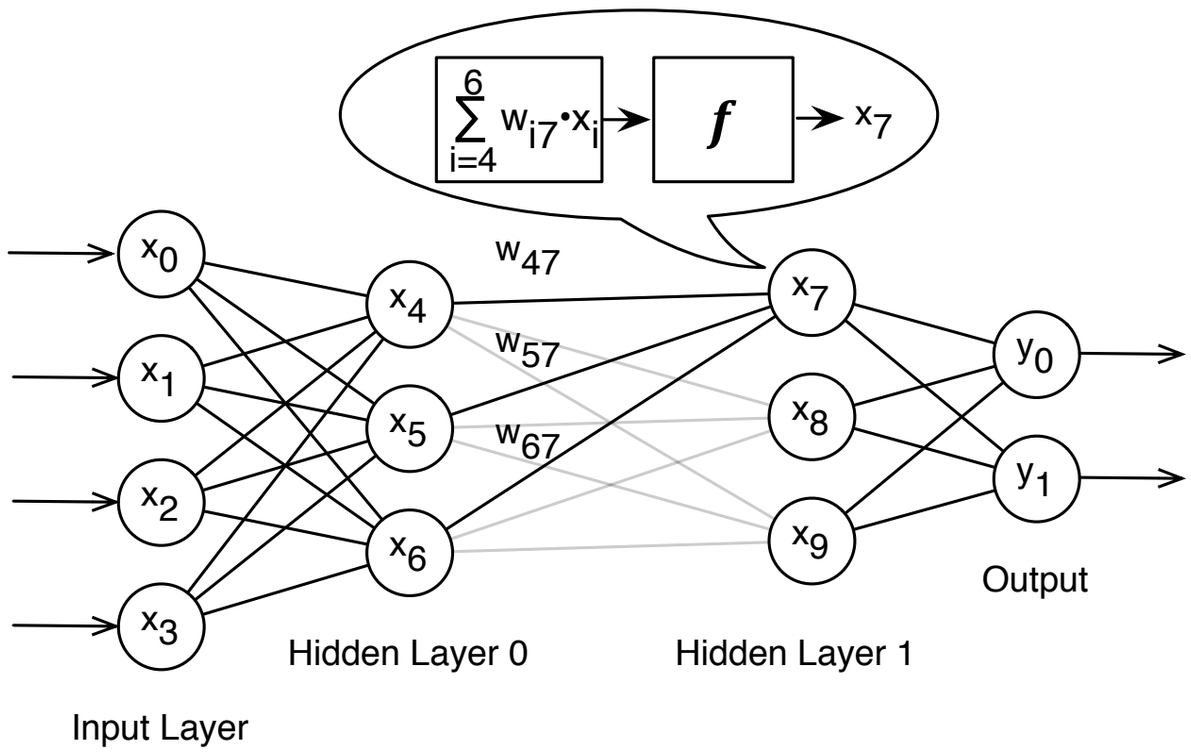
Compiler Support with ACCEPT

SNNAP Accelerator design

Evaluation & Comparison with HLS

Background: Multi-Layer Perceptrons

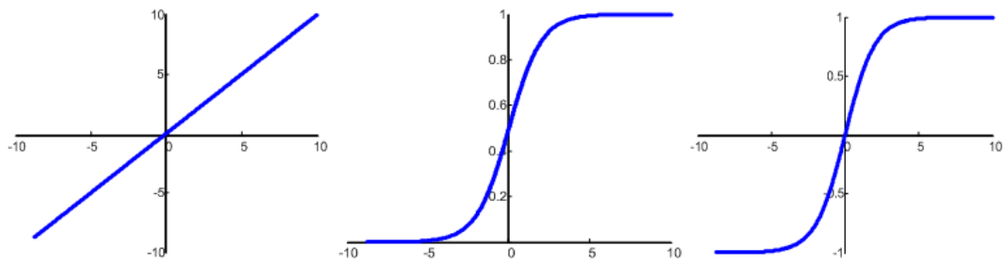
neural network



computing a single layer

$$\begin{bmatrix} x_7 \\ x_8 \\ x_9 \end{bmatrix} = f\left(\begin{bmatrix} w_{67} & w_{57} & w_{47} \\ w_{68} & w_{58} & w_{48} \\ w_{69} & w_{59} & w_{49} \end{bmatrix} \begin{bmatrix} x_6 \\ x_5 \\ x_4 \end{bmatrix} \right)$$

activation function ***f***

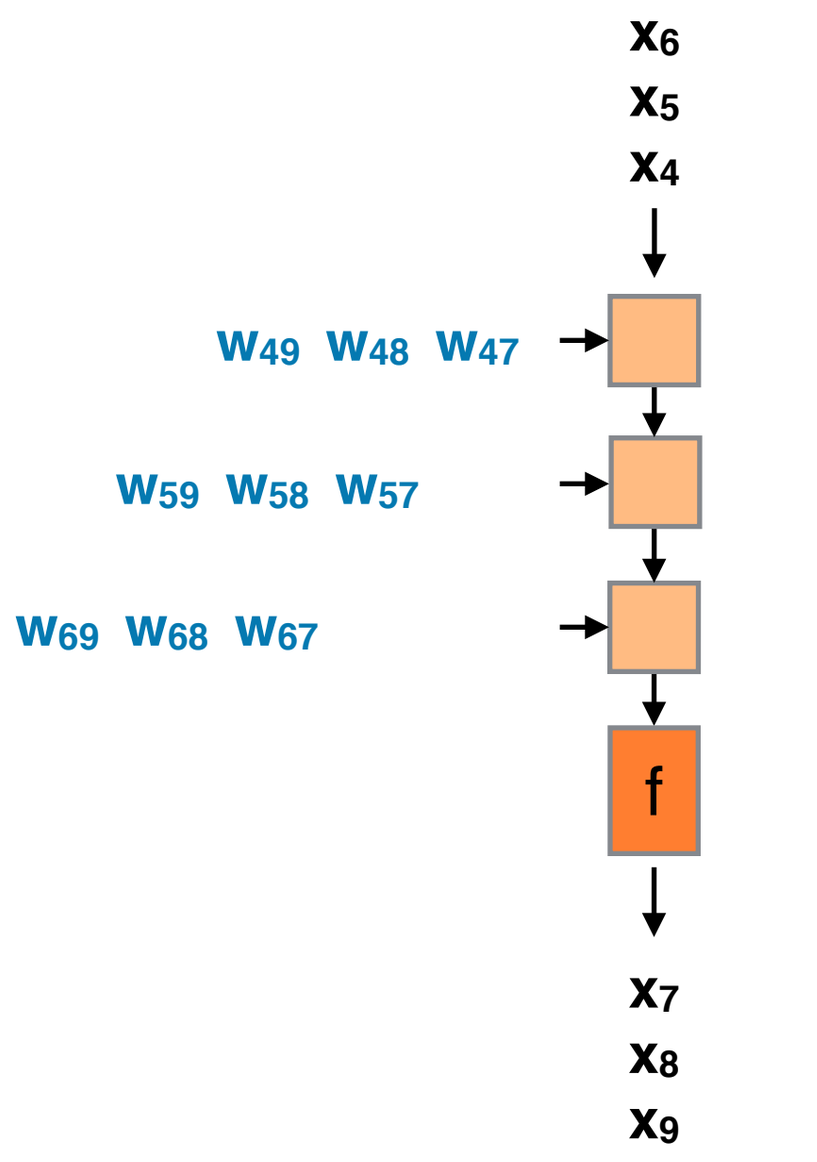


Background: Systolic Arrays

computing a single layer

$$\begin{bmatrix} x_7 \\ x_8 \\ x_9 \end{bmatrix} = f\left(\begin{bmatrix} w_{67} & w_{57} & w_{47} \\ w_{68} & w_{58} & w_{48} \\ w_{69} & w_{59} & w_{49} \end{bmatrix} \begin{bmatrix} x_6 \\ x_5 \\ x_4 \end{bmatrix} \right)$$

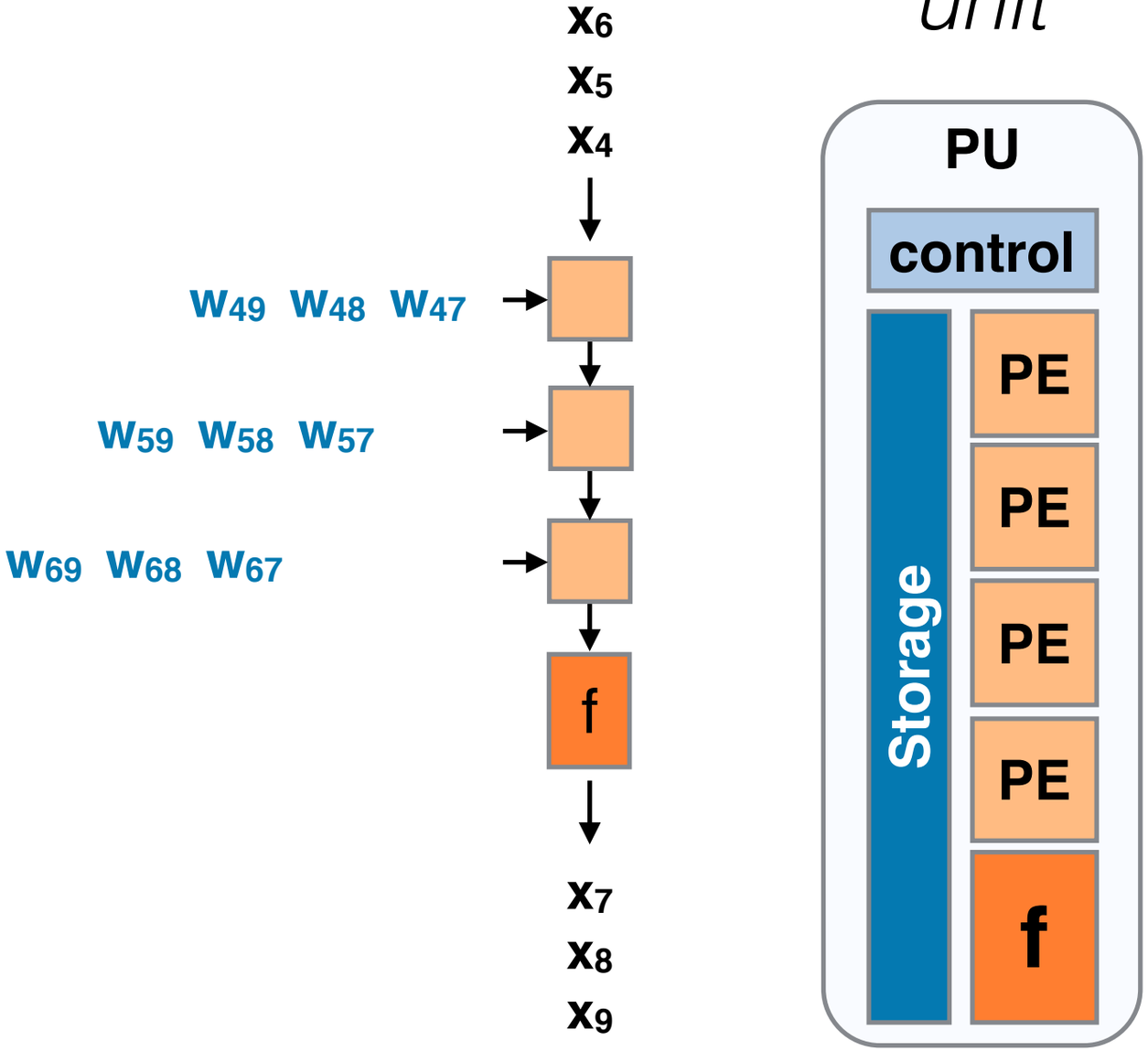
systolic array



PU Micro-Architecture

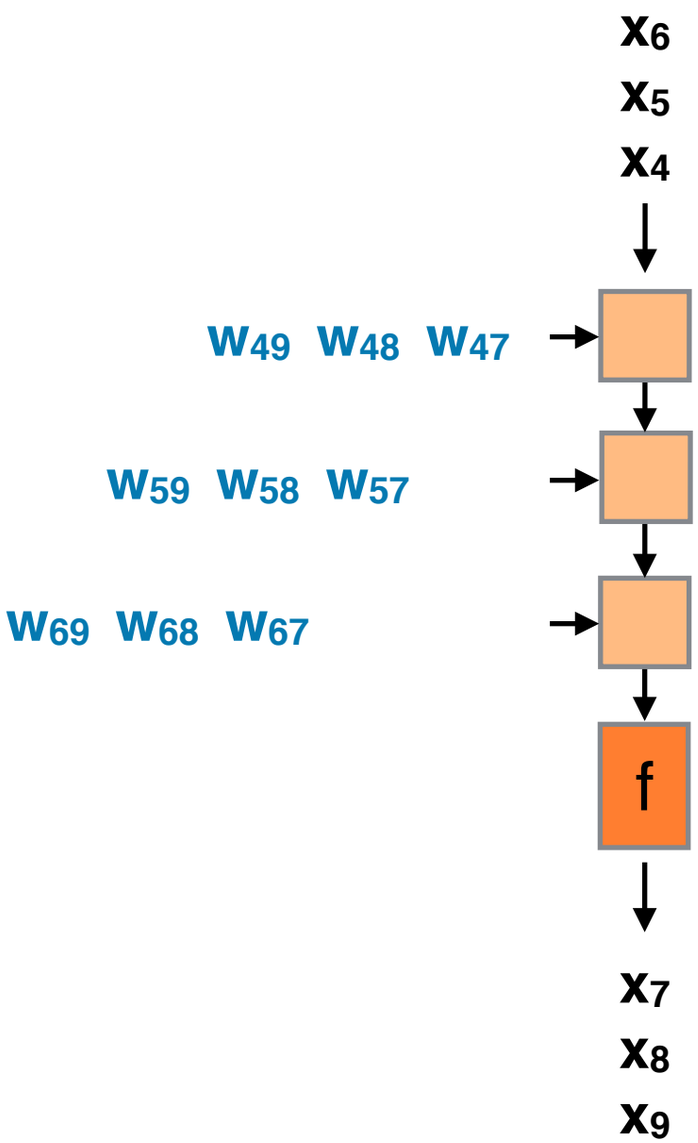
systolic array

processing unit

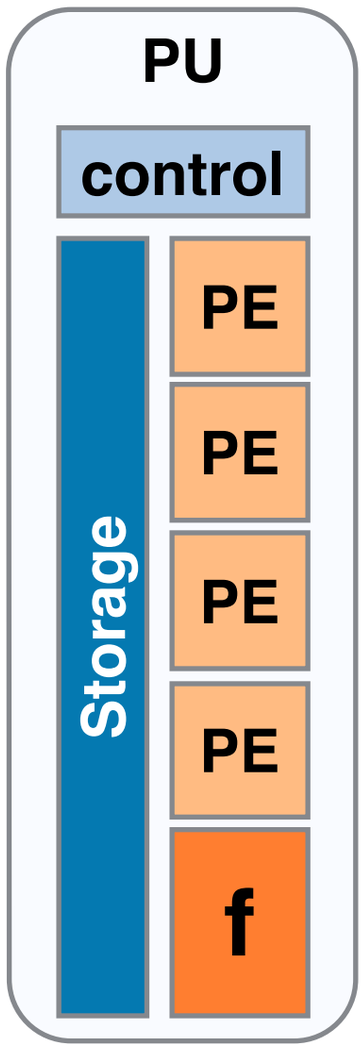


PU Micro-Architecture

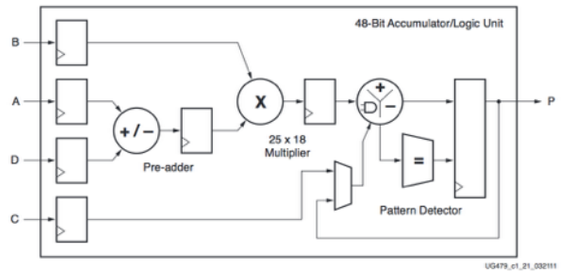
systolic array



processing unit

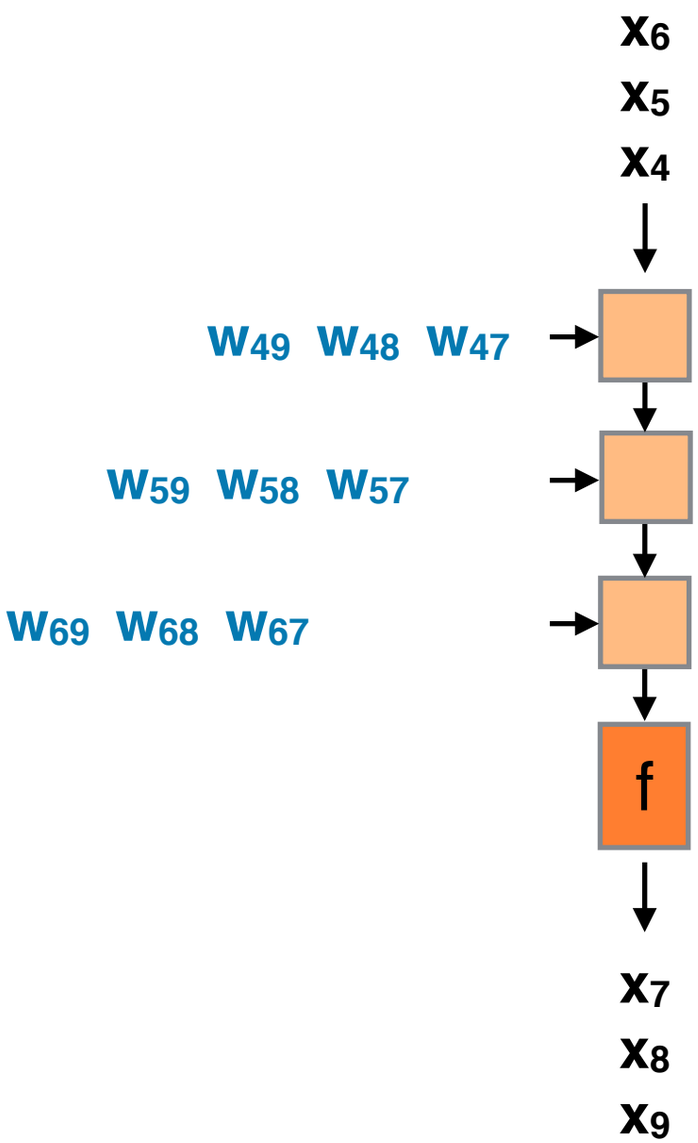


1 - processing elements in DSP logic

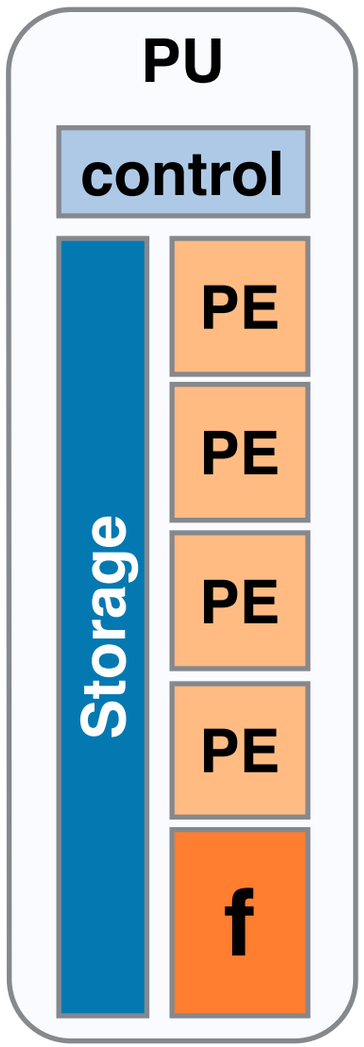


PU Micro-Architecture

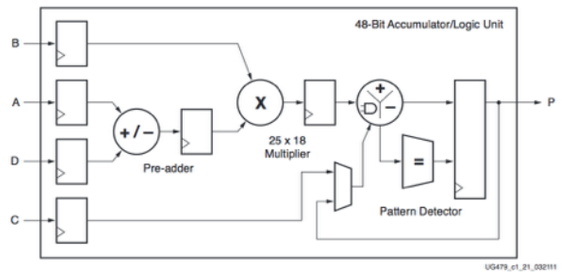
systolic array



processing unit



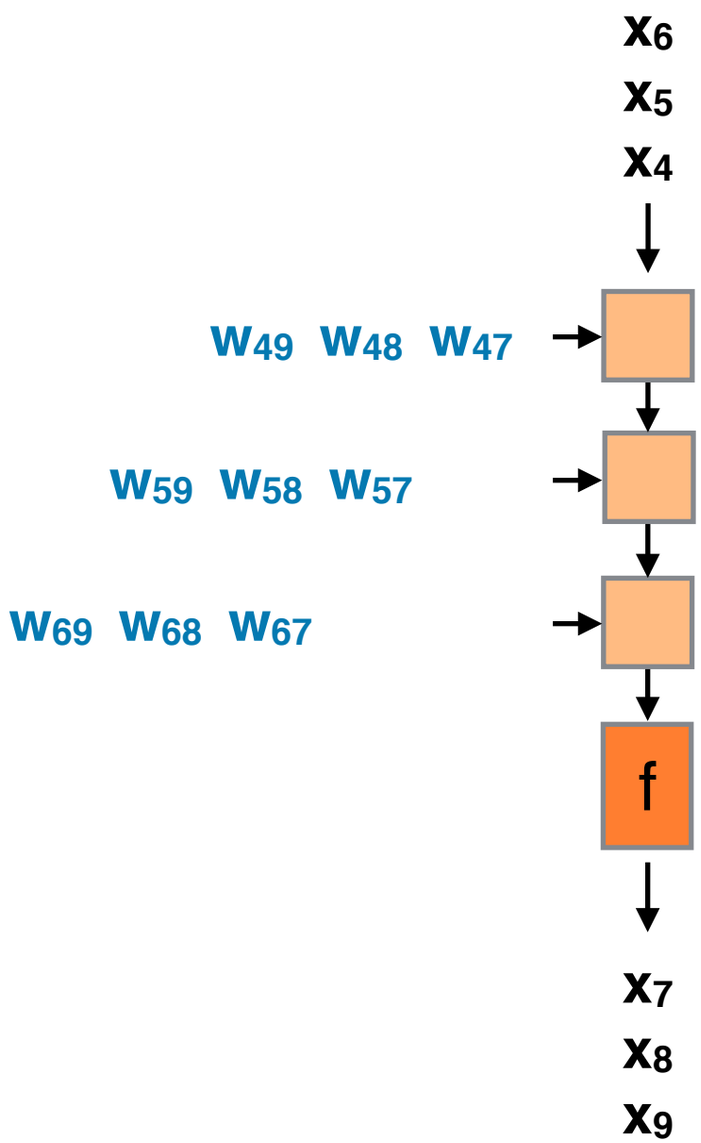
1 - processing elements in DSP logic



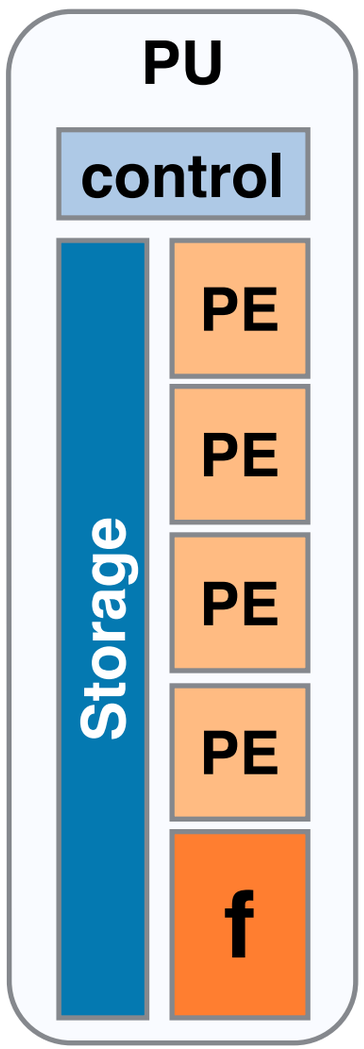
2 - local storage for synaptic weights

PU Micro-Architecture

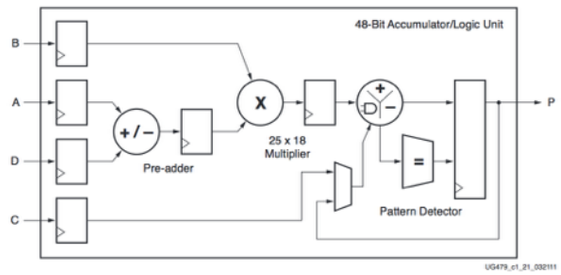
systolic array



processing unit

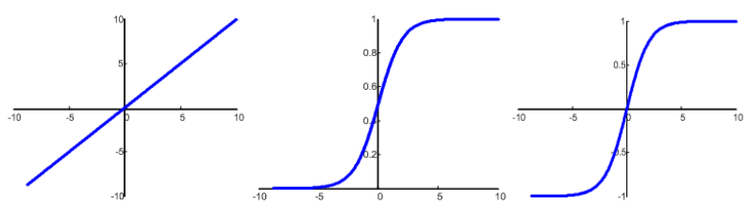


1 - processing elements in DSP logic



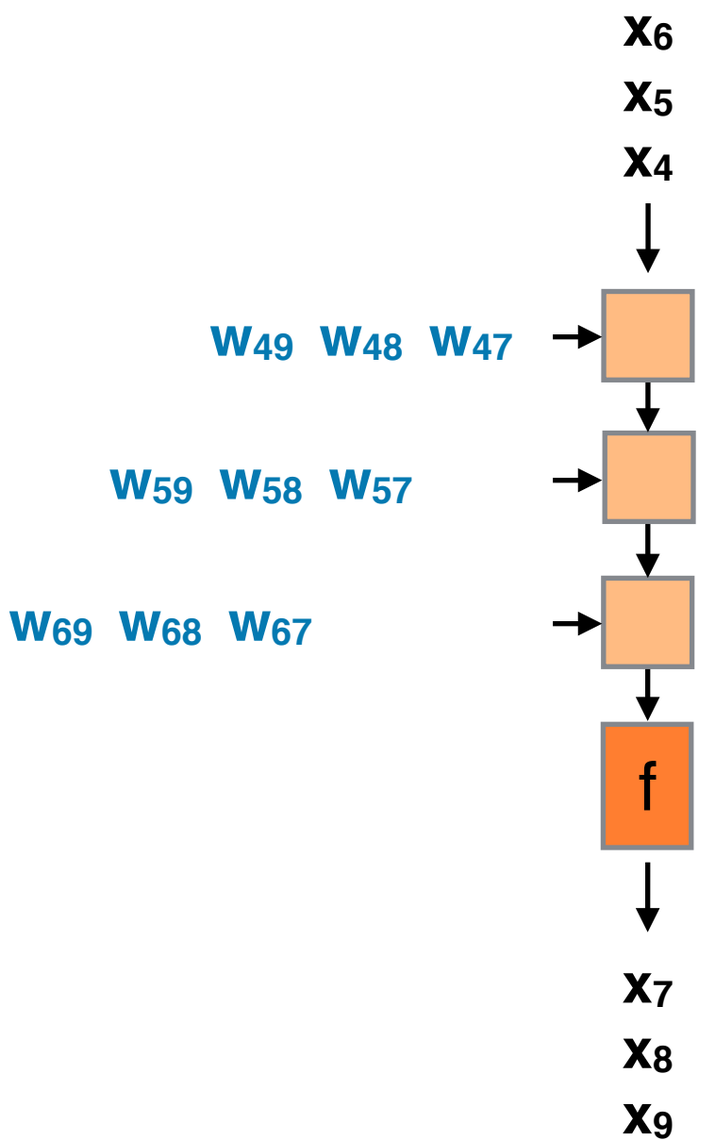
2 - local storage for synaptic weights

3 - sigmoid unit implements non-linear activation functions

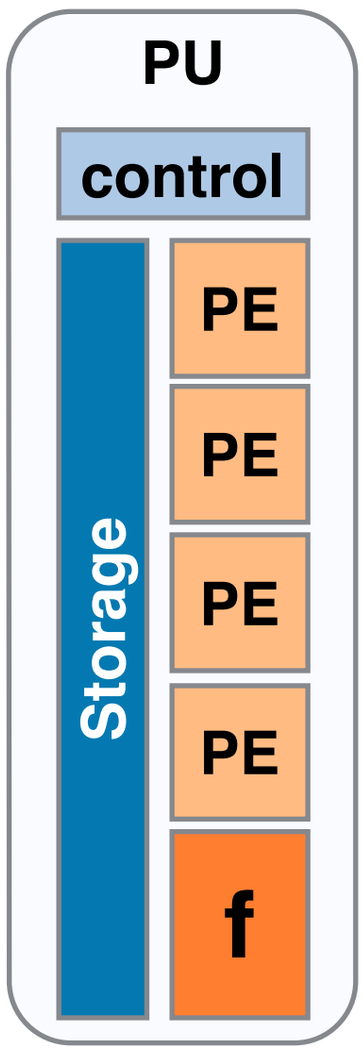


PU Micro-Architecture

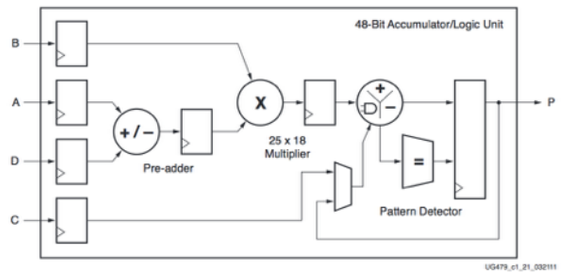
systolic array



processing unit

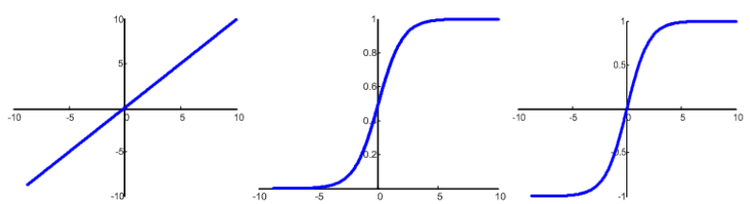


1 - processing elements in DSP logic



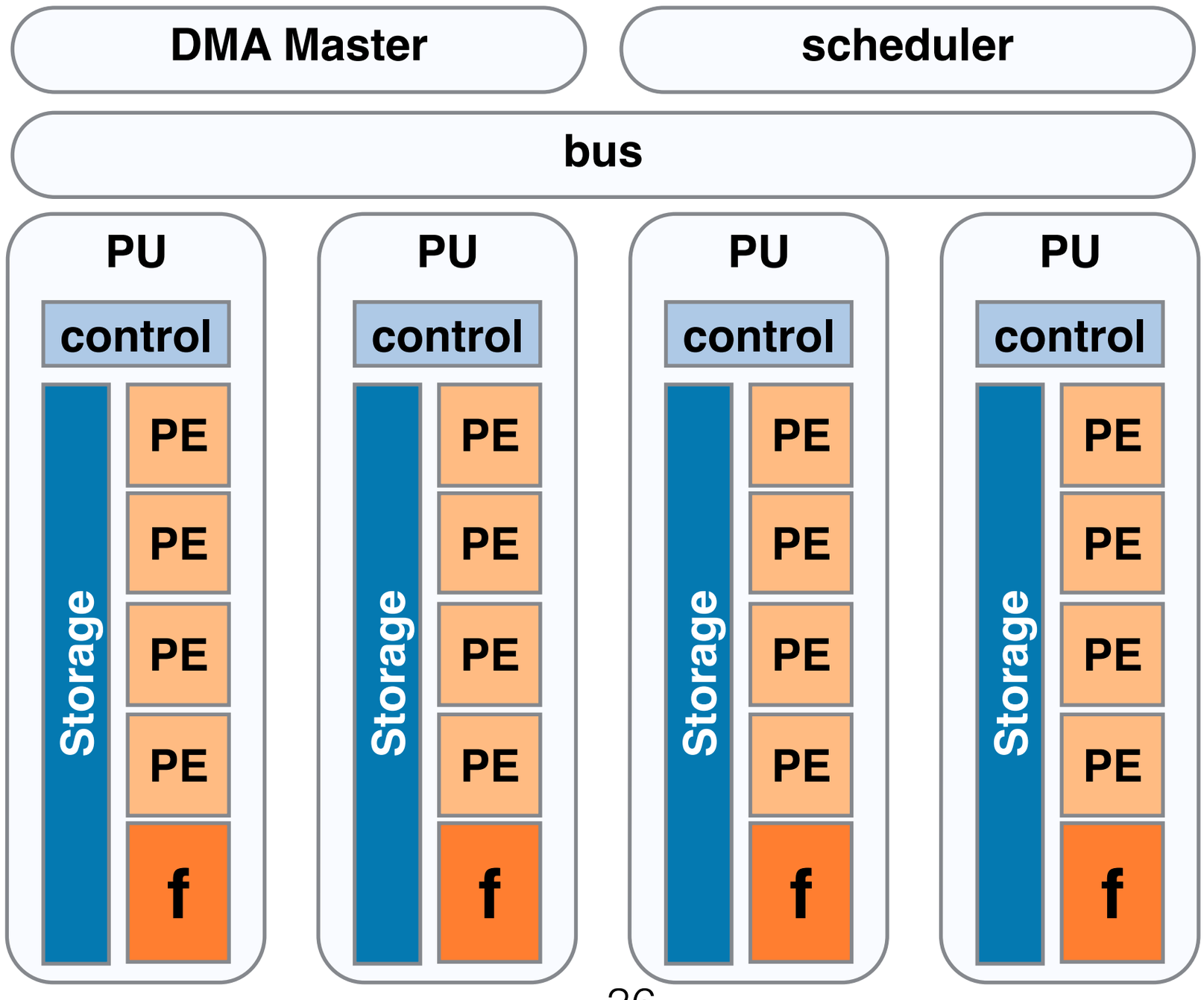
2 - local storage for synaptic weights

3 - sigmoid unit implements non-linear activation functions



4 - vertically micro-coded sequencer

Multi-Processing Units

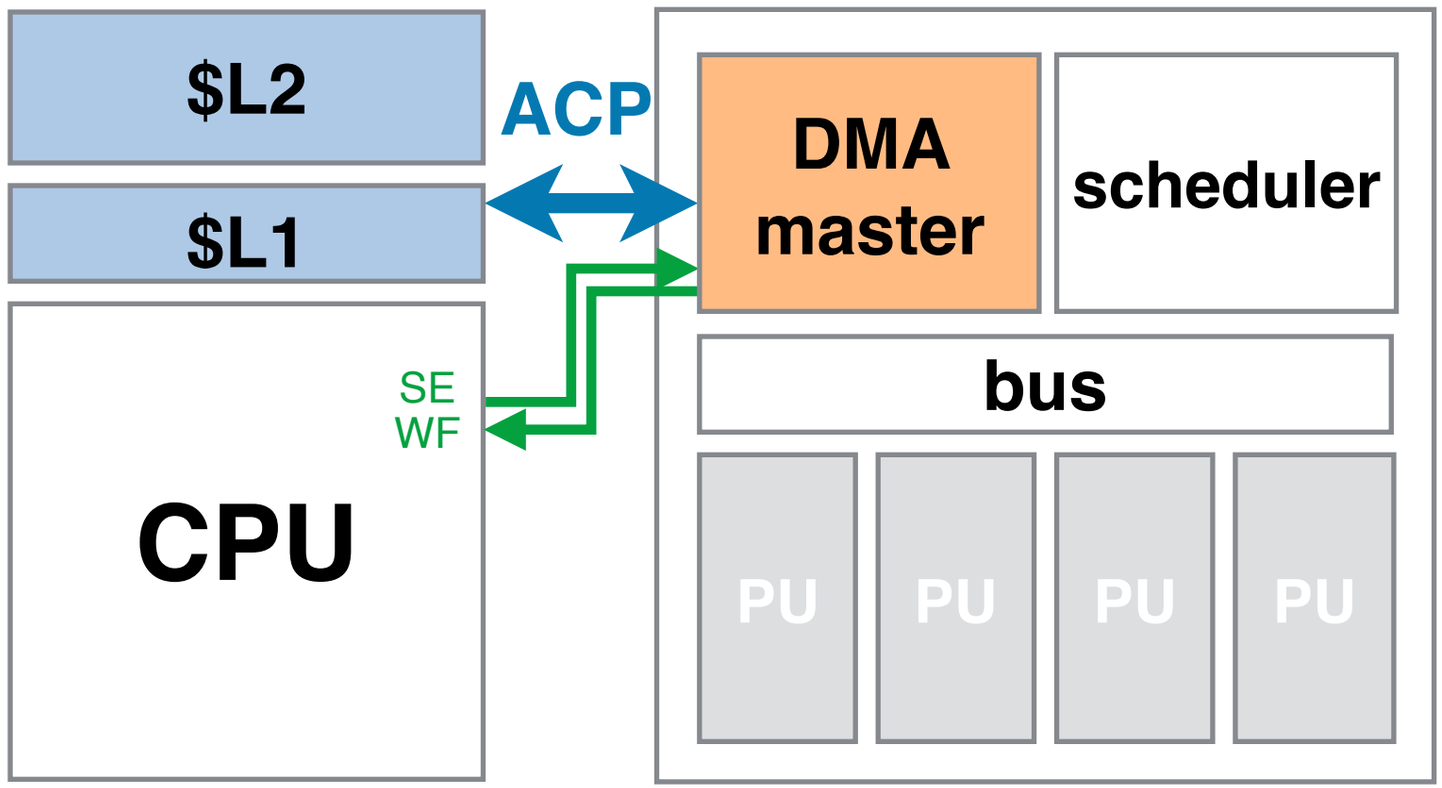


CPU-SNNAP Integration

coherent reads & writes with accelerator coherency port

custom mastering interface

low-latency event signaling, sleep & wakeup



Talk Outline

Introduction

Programming model

SNNAP design:

- Efficient neural network evaluation
- Low-latency communication

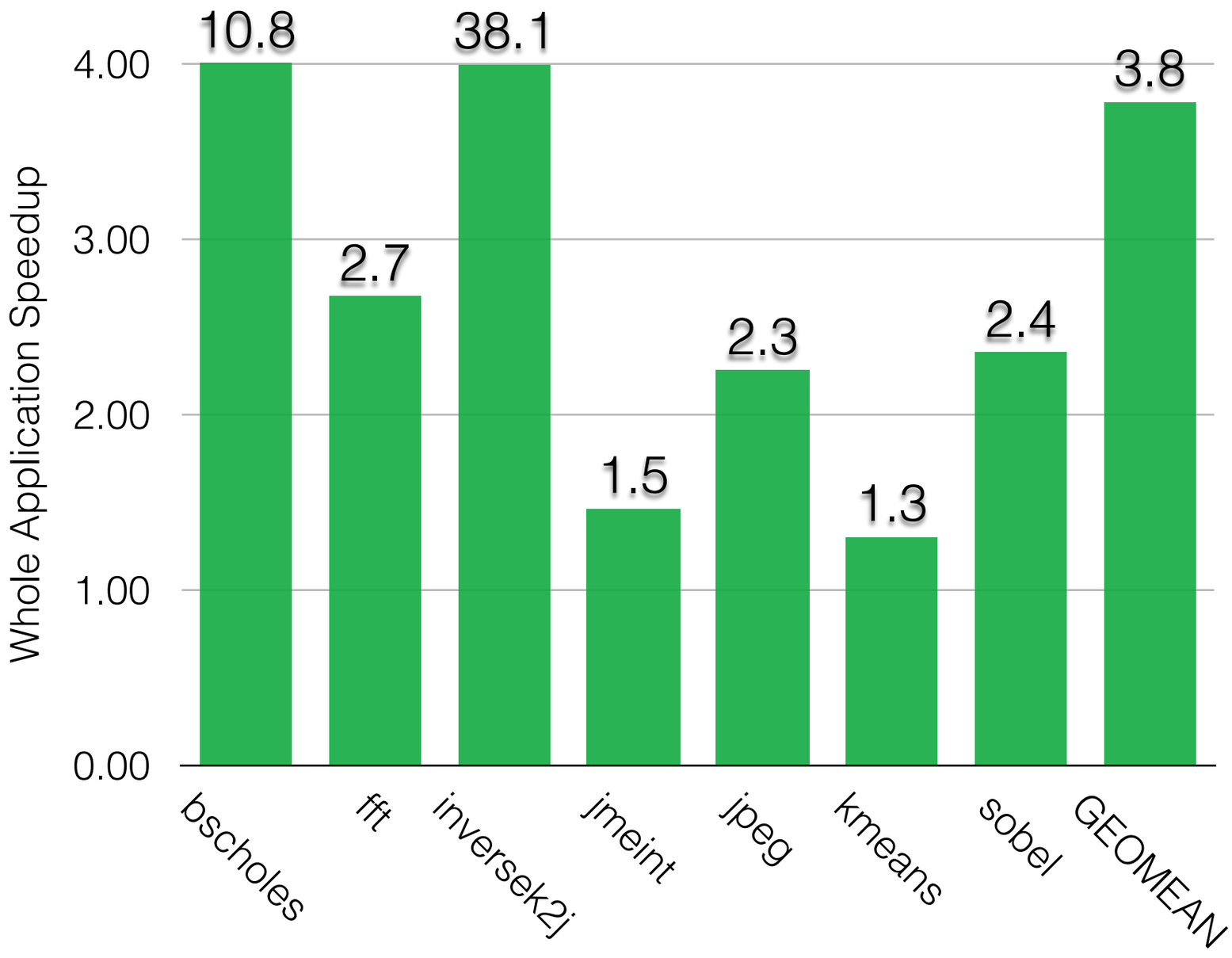
Evaluation & Comparison with HLS

Evaluation

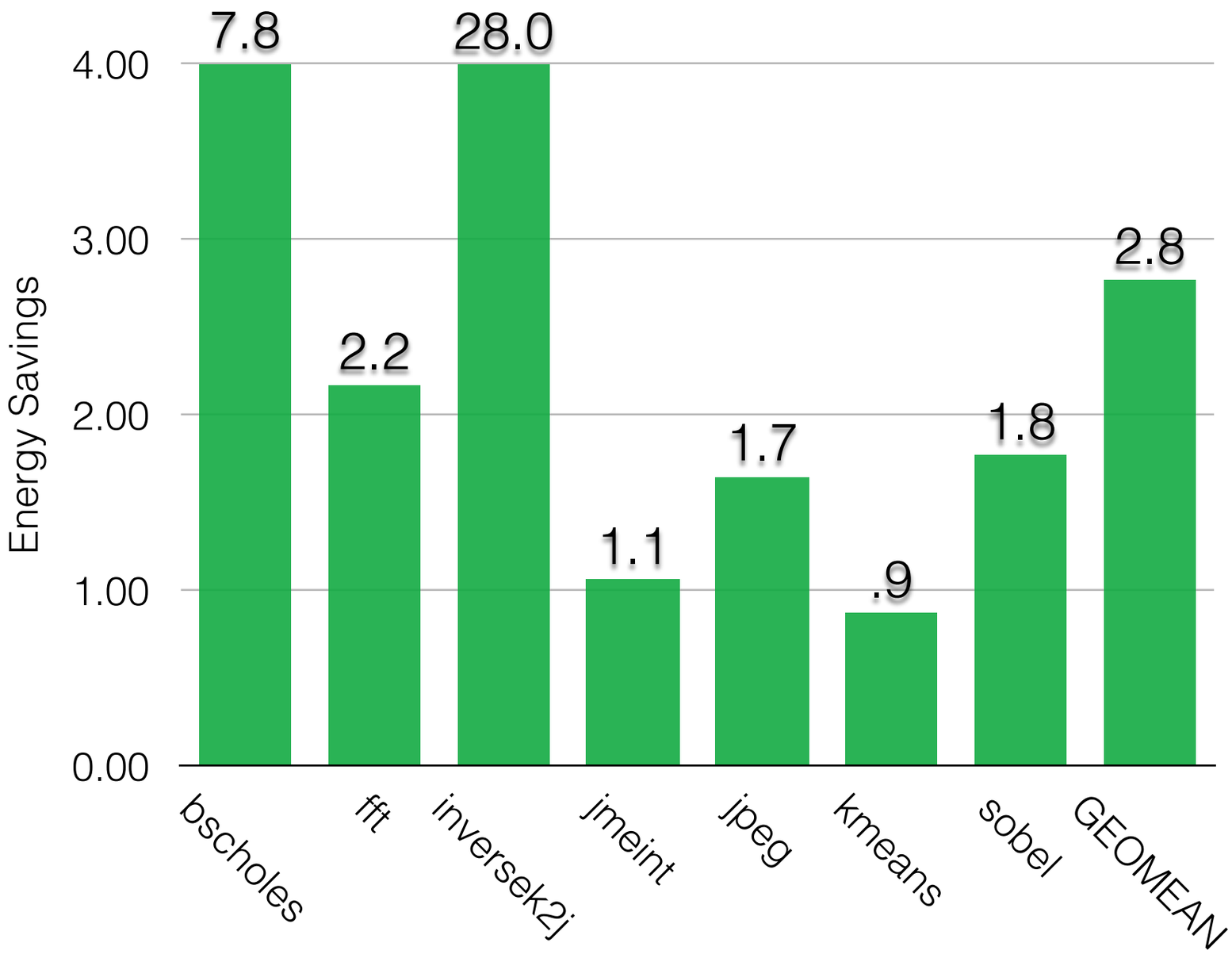
Neural acceleration on SNNAP (8x8 configuration, clocked at 1/4 of f_{CPU}) vs. precise CPU execution

application	domain	error metric
blackscholes	option pricing	MSE
fft	DSP	MSE
inversek2j	robotics	MSE
jmeint	3D-modeling	miss rate
jpeg	compression	image diff
kmeans	ML	image diff
sobel	vision	image diff

Whole-Application Speedup



Energy Savings



+36%

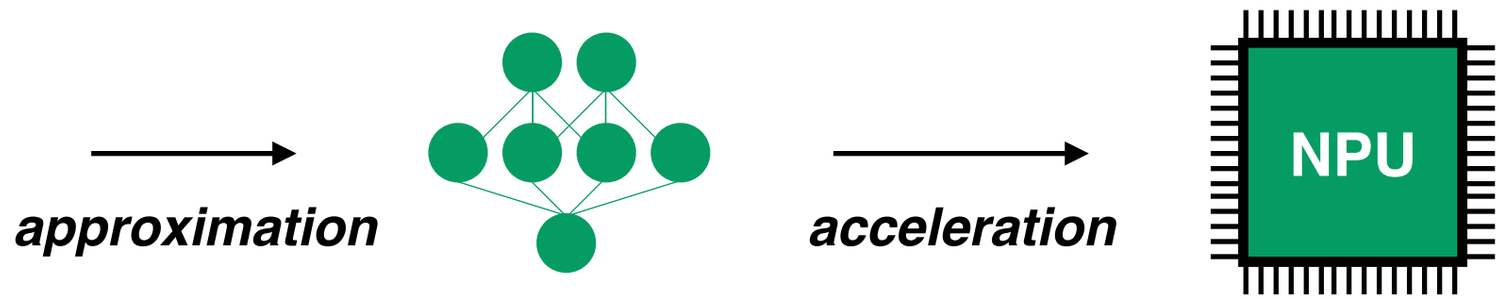
Energy = Power * Runtime

on

(DRAM + SoC)

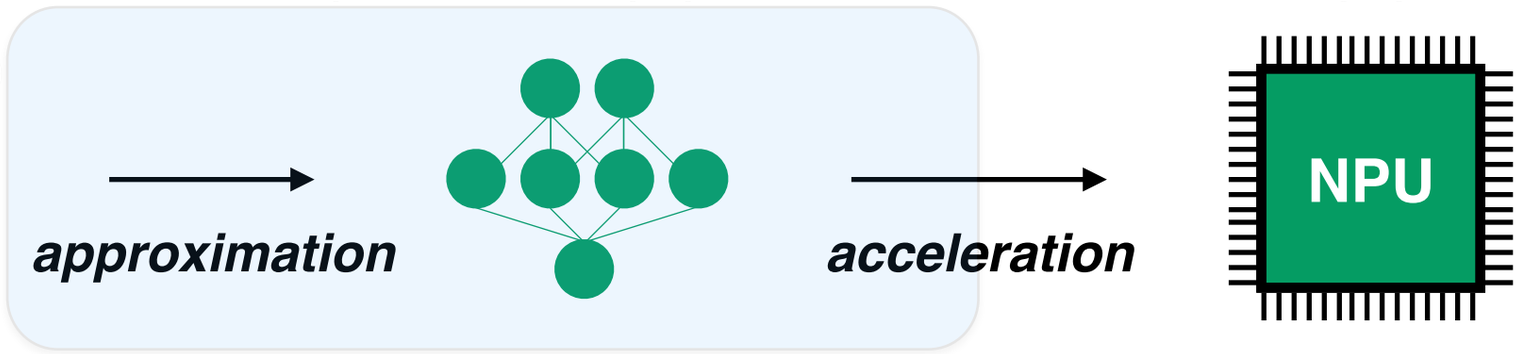
Conclusion

```
float foo (float a, float b)
{
    …
    return val;
}
```



Conclusion

```
float foo (float a, float b)
{
   …
   return val;
}
```



ACCEPT

Conclusion

```
float foo (float a, float b)
{
    …
    return val;
}
```


3.8x speedup & 2.8x energy savings

Compilation and Hardware Support for Approximate Acceleration

Thierry Moreau, Adrian Sampson, Andre Baixo,
Mark Wyse, Ben Ransford, Jacob Nelson,
Luis Ceze and Mark Oskin
University of Washington
moreau@uw.edu

ACCEPT: http://accept.rocks

SNNAP: upon request