Micron AI Solutions

©2019 Micron Technology, Inc. All rights reserved. Information, products, and/or specifications are subject to change without notice. All information is provided on an “AS IS” basis without warranties of any kind. Statements regarding products, including statements regarding product features, availability, functionality, or compatibility, are provided for informational purposes only and do not modify the warranty, if any, applicable to any product. Drawings may not be to scale. Micron, the Micron logo, and all other Micron trademarks are the property of Micron Technology, Inc. All other trademarks are the property of their respective owners.

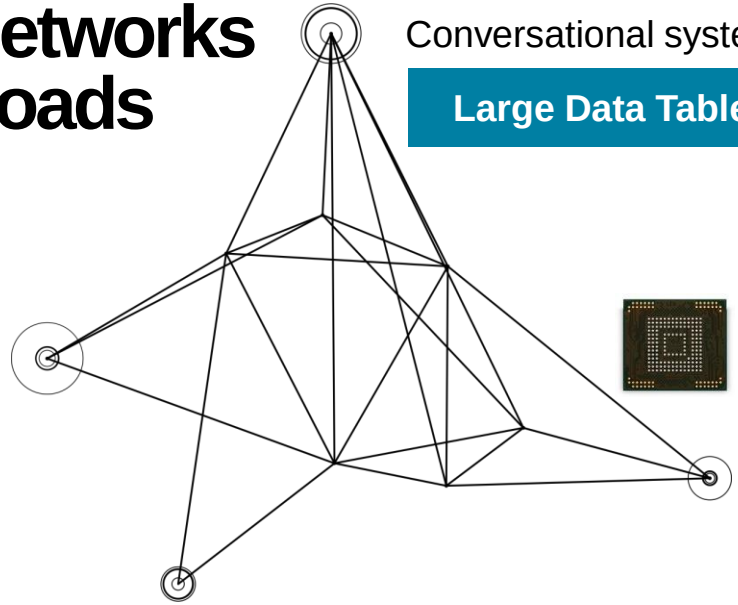
Micron Advanced Computing Solutions Group

Research and engineering team focused on:

- High performance computing customer solutions
- Exploring pathfinding technologies – machine learning & AI
- Product security leadership
- Memory research path planning

The Micron Advantage

Neural Networks AI Workloads



Storage
Recommendation engines
Conversational systems

Large Data Tables

Machine-Learning memory
IoT ultra-low-power
All neural networks!

Edge-Devices

3D NAND
Medical imaging
Satellite imaging

Large Inputs

Low-Power DDR
Categorization
Speech commands

Standard

Micron Confidential

Micron AI/DL Platform Overview

Flexible deep learning solution comprised of a modular FPGA-based architecture, powered by Micron memory, running FWDNXT's high performance inference engine tuned for a variety of neural networks

Fully Integrated SDK

Broad deep learning framework support

Easy to use toolset; software programmability

Ability to run on multiple neural networks simultaneously

Deep Learning Compiler

Compiles trained network directly to inference engine machine code

Underlying hardware is abstracted away

Feature rich & designed to optimize for specific use cases

Advanced Accelerators

FPGA solutions with high performance memory available for prototyping or early deployments

Scale with multiple cores

Integrated with SDK to fully utilize silicon resources

Scalable solution with no FPGA programming required accelerating time to working applications

Micron DLA Value Proposition & Benefits

Integration of Micron optimized memory will further differentiate the Micron DLA through lower power and higher performance for edge based workloads

Low latency & Low power

- Designed for full utilization of silicon resources
- Operates with small batch for efficient responses
- On-chip cache re-use operands for reduced memory traffic

Flexibility & Ease of Use

- Broad framework and neural network support
- Compiler designed to optimize workloads
- Scalable architecture allows to re-size engine to service edge, fog & cloud apps

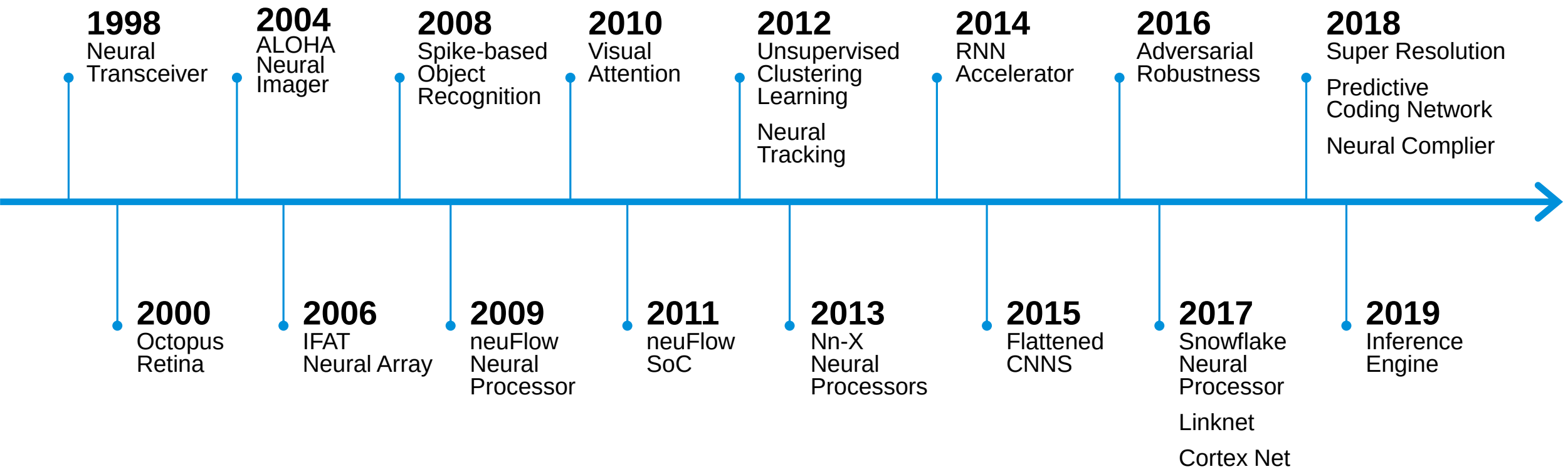
Scalability

- Ability to run multiple NNs on one instance
- Multiple matrix-vector and matrix-matrix unit are composable “cores” that provide scalability

Fast time-to-POC

- Model, train and compile directly to FPGA
- No FPGA programming required
- Simple code portability from existing CPU and GPU code

FWDNXT Innovation Timeline



Micron Confidential

Micron Deep Learning Accelerator (DLA)

Designed for:

Good performance per power

High utilization

Efficient use of memory bandwidth

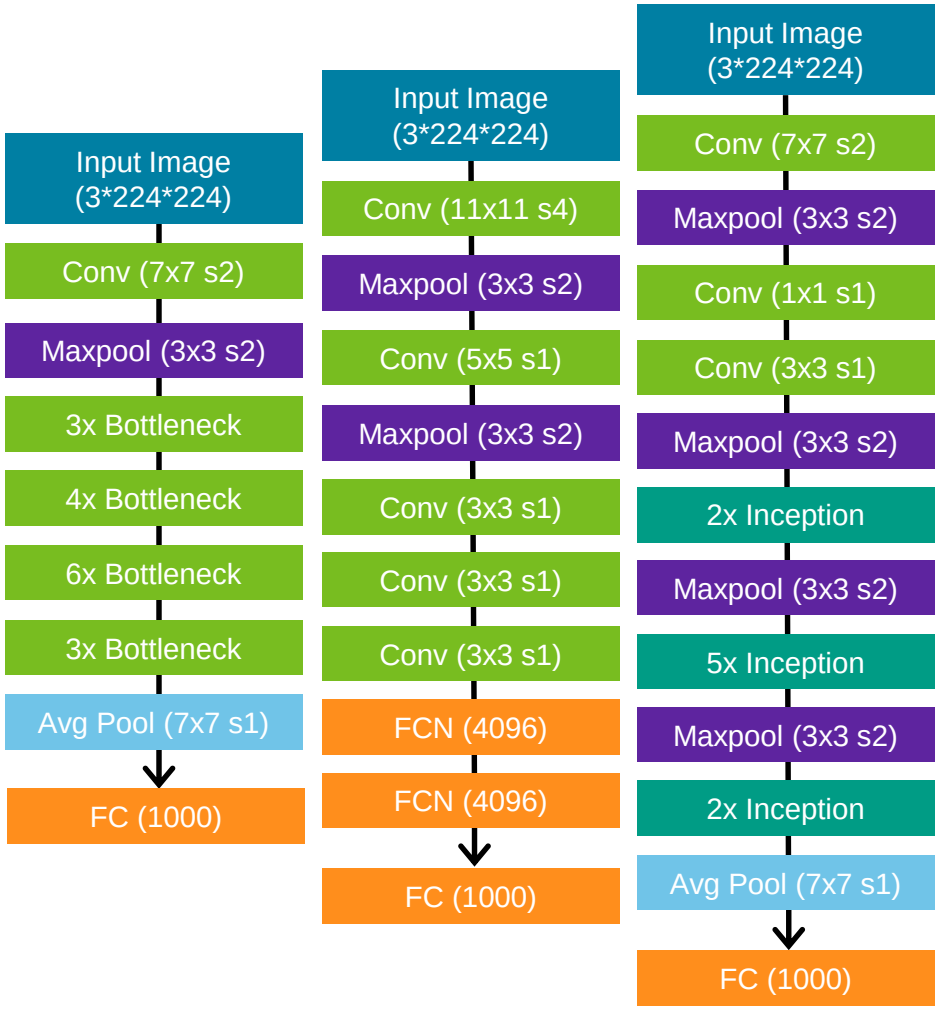
Low latency

Scalability: IoT to cloud

Implemented on Micron FPGAs

Any Neural Network. Any Framework.

AlexNet
ResNet
GoogleNet
LinkNet
…
encoder-decoder
…
RNN
LSTM
training



Import

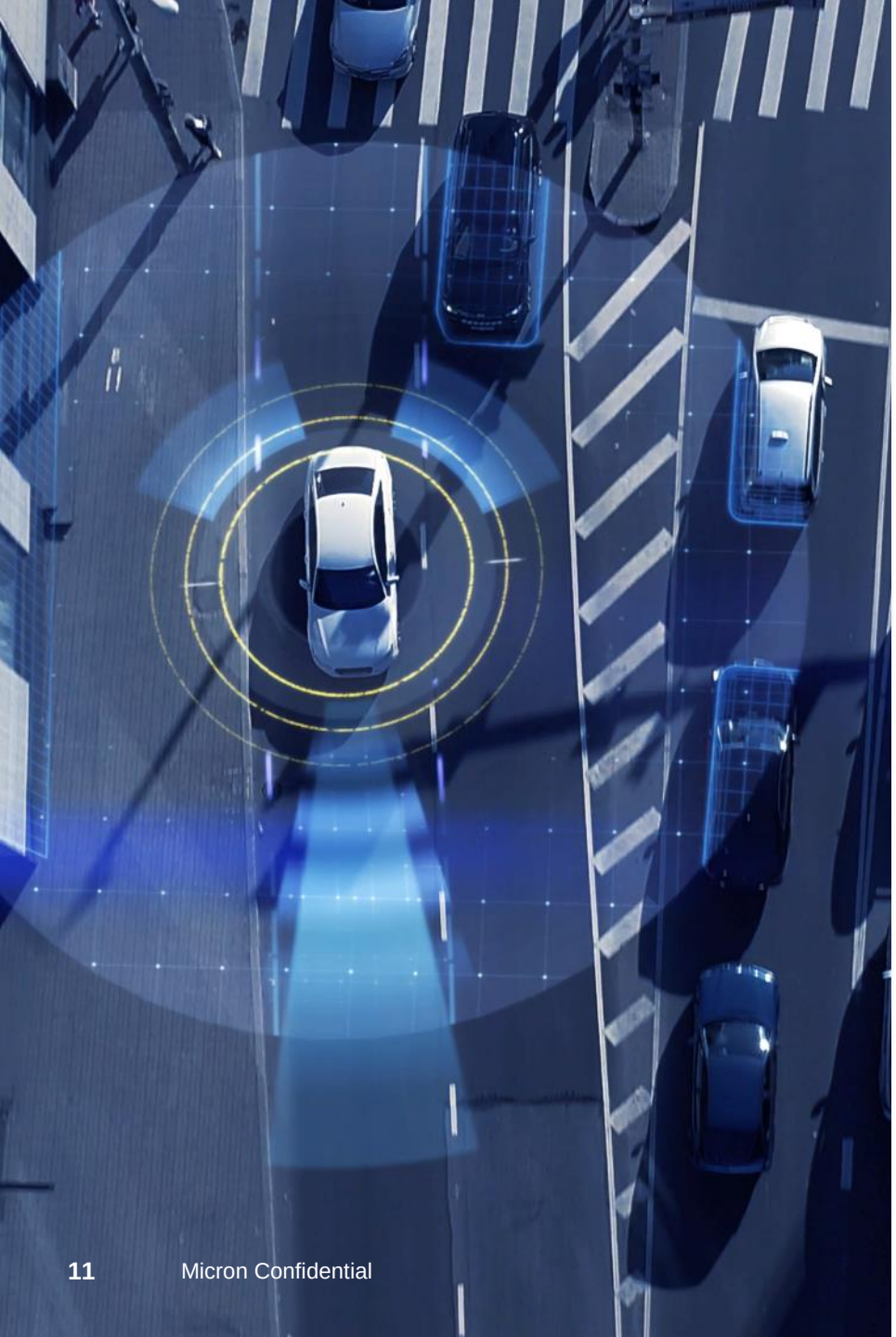
Instantiate

Run

```
 1 import Micron_DLA
 4 import sys
 5 import PIL
 6 from PIL import Image
 7 import numpy as np
 8
 9 from argparse import ArgumentParser
10 # argument Checking
11 parser = ArgumentParser(description="IE Demonstration")
12 …
19 args = parser.parse_args()
20
21 #Load image into a numpy array
22 img = Image.open(args.image)
24 #Resize it to the size expected by the network
25 img = img.resize((args.res[2], args.res[1]), resample=PIL.Image.BILINEAR)
27 #Convert to numpy float
28 img = np.array(img).astype(np.float32) / 255
30 #Transpose to plane-major, as required by our API
31 img = np.ascontiguousarray(img.transpose(2,0,1))
32
33 #Normalize images
34 stat_mean = list([0.485, 0.456, 0.406])
35 stat_std = list([0.229, 0.224, 0.225])
36 for i in range(3):
37     img[i] = (img[i] - stat_mean[i])/stat_std[i]
38
39 #Create and initialize the snowfloke object
40 fie1 = Micron_DLA.instance()
41 nresults = sf.init("{:d}x{:d}x{:d}".format(args.res[1], args.res[2], args.res[0]), args.modelpath, '')
45
46 #Create the storage for the result and run one inference
47 result = np.ndarray(nresults,dtype=np.float32)
48 fie1.Run(img, result)
49
50 #Convert to numpy and print top-5
51 idxs = (-result).argsort()
54 print('---------- Micron DLA results ----------')
55for i in range(5):
62         print(idxs[i], result[idxs[i]])
```

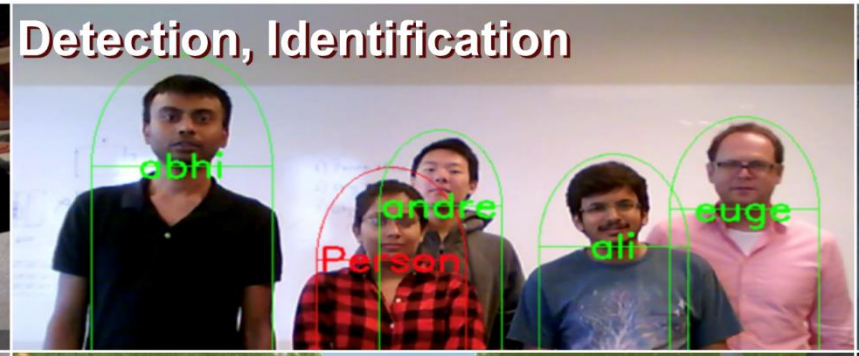
Micron

Experts in Neural Networks



Categorization
Pose estimation
Predictive networks
Speech commands
Detection, Identification
Segmentation
RNN, LSTM, GRU
GAN style-transfer
Tracking
Satellite
Speaker identification
Reinforcement learning

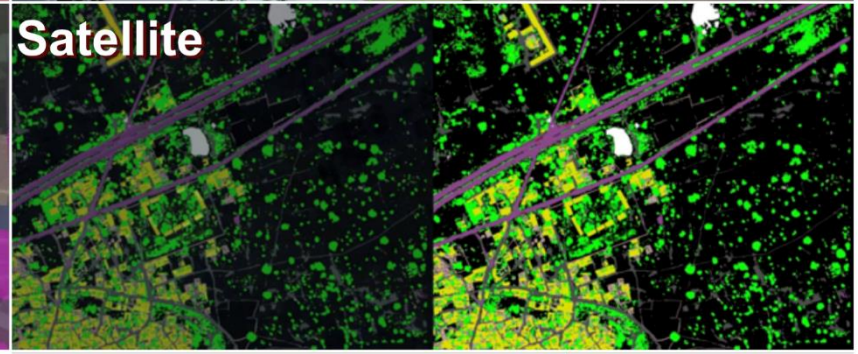

Categorization
11.70 fps
upholstered armchair lounge cushioned
11.45 fps
tote satchel sling messenger handbag


Detection, Identification
obhi
andre
Person
ali
euge

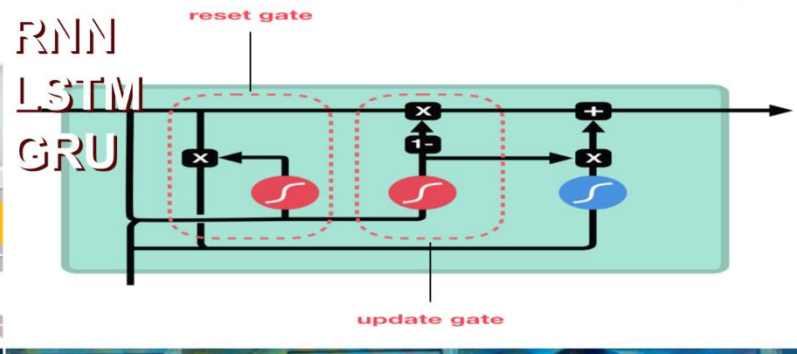

Tracking
141
140
147
149
115
131

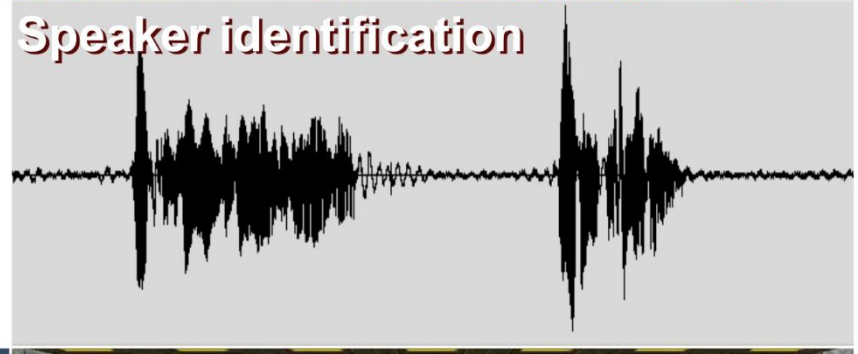

Pose estimation

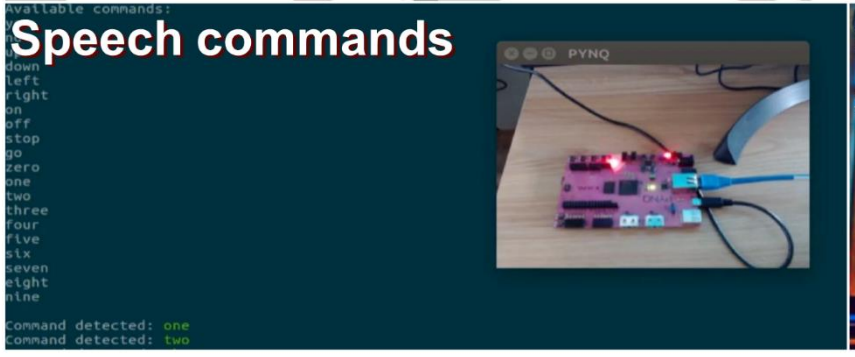

Segmentation


Satellite


Predictive
networks


RNN
LSTM
GRU
reset gate
update gate


Speaker identification


Speech commands
Available commands:
down
left
right
on
off
stop
go
zero
one
two
three
four
five
six
seven
eight
nine
Command detected: one
Command detected: two
PYNQ


GAN
style-transfer

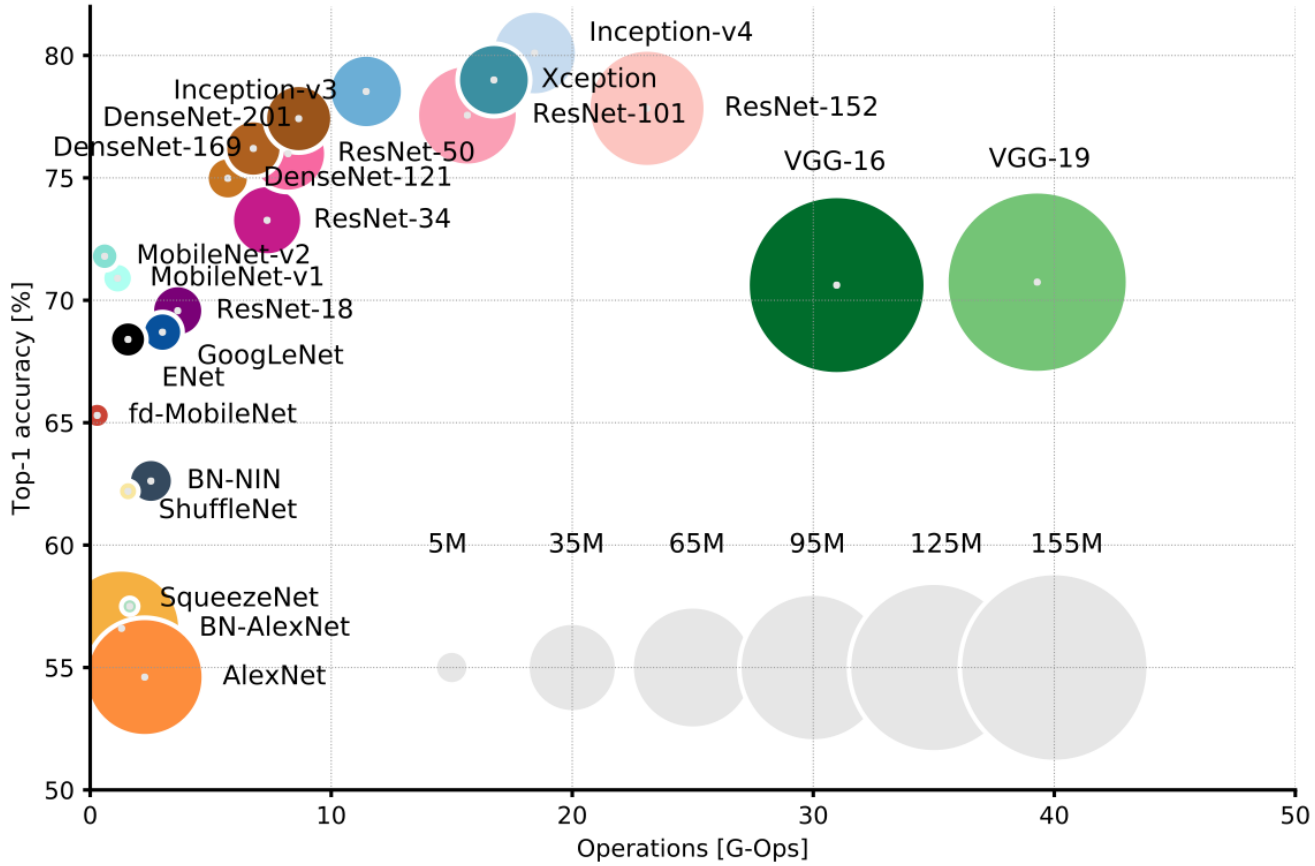

Reinforcement Learning
50
100%
0%
AMMO
HEALTH
ARMS
ARMOR

Innovation in Neural Nets: Efficiency

We compare and analyze efficient categorization neural networks and optimize:

Accuracy

Run-Time / Power / Operations

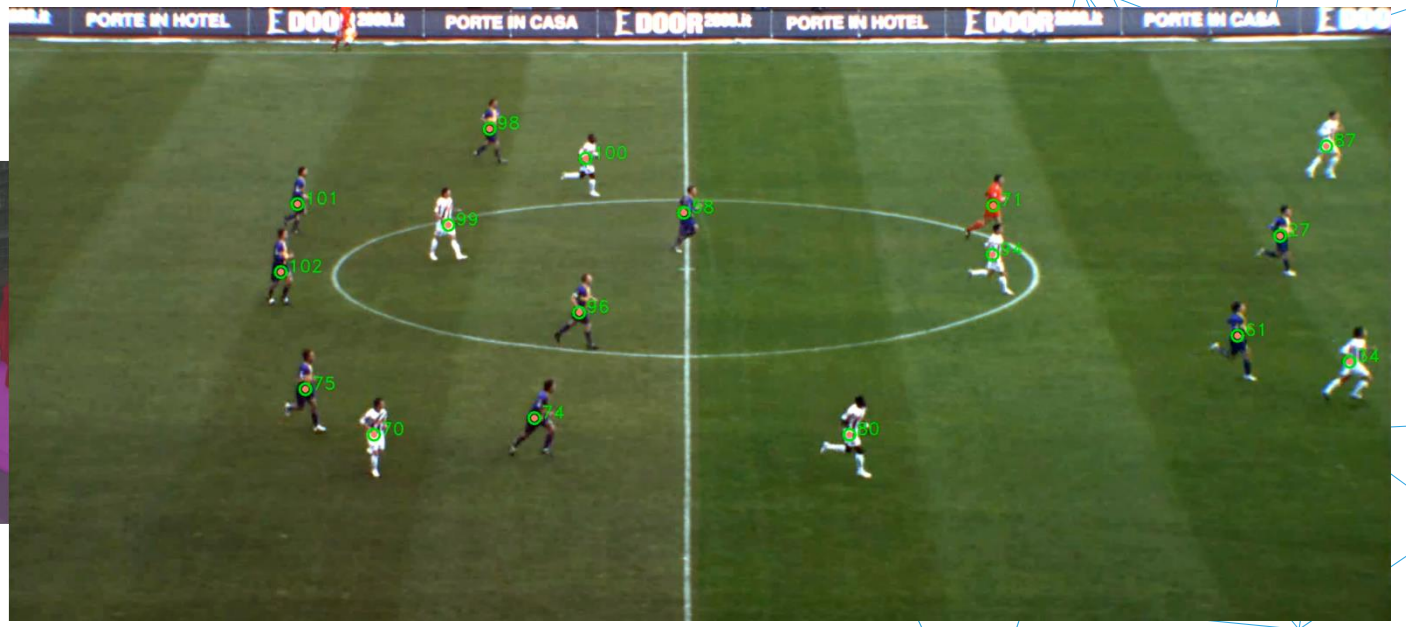


https://towardsdatascience.com/neural-network-architectures-156e5bad51ba

https://arxiv.org/abs/1605.07678

Published Neural Network Too Large for Use

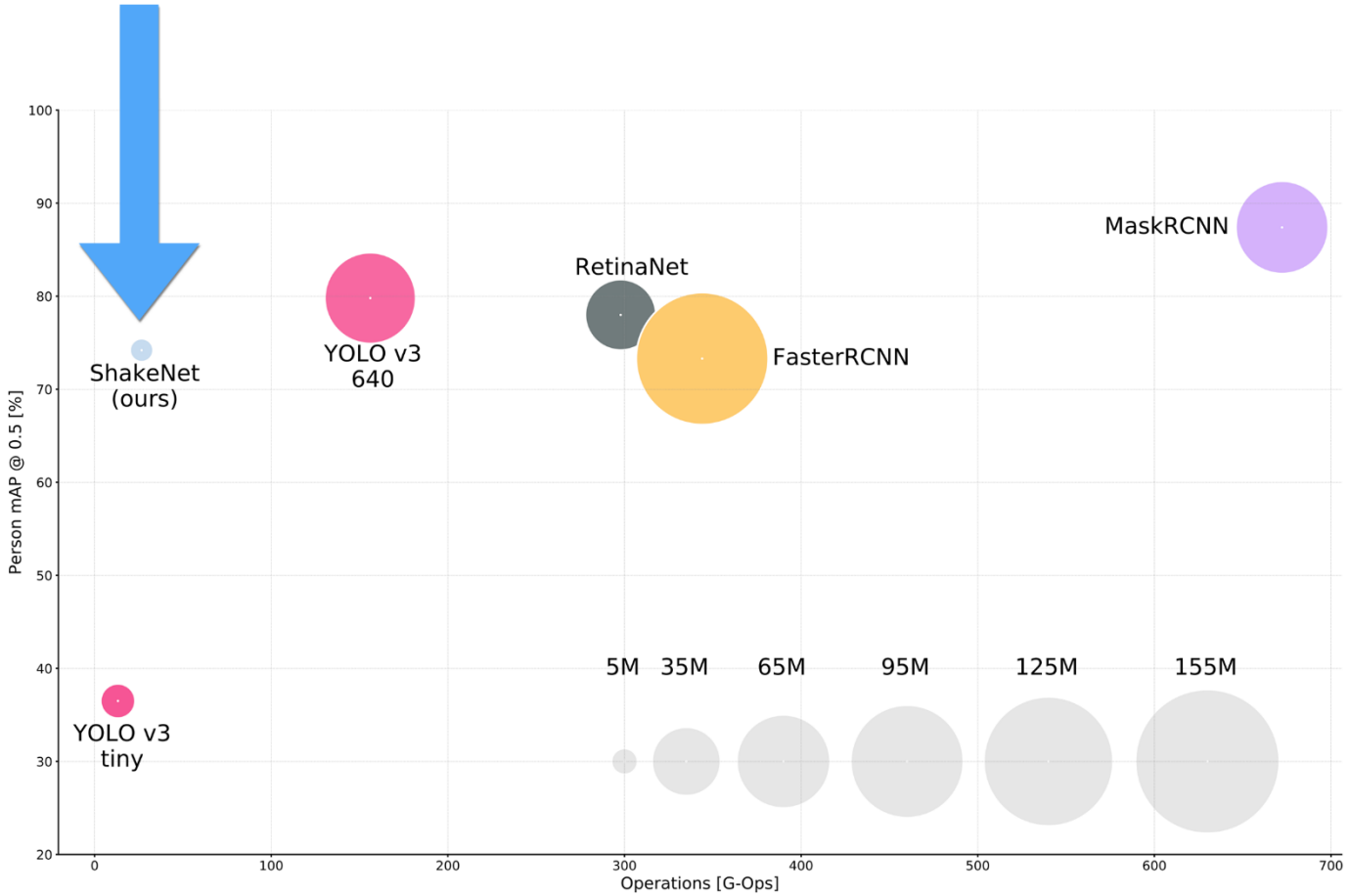
We Design Our Own Networks



Micron Confidential

Innovation Example: Pixel Networks

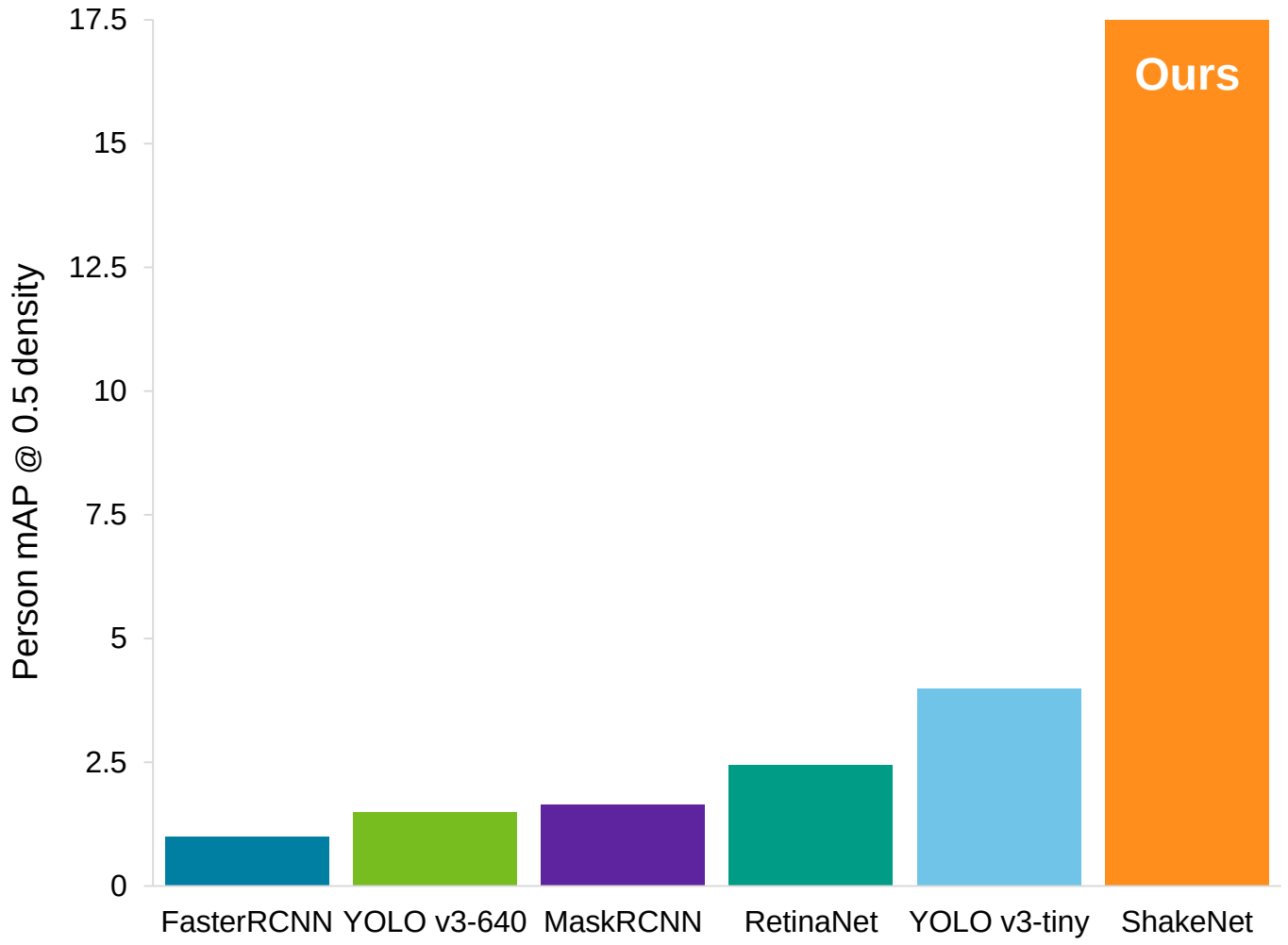
Best in perf/params



Micron Confidential

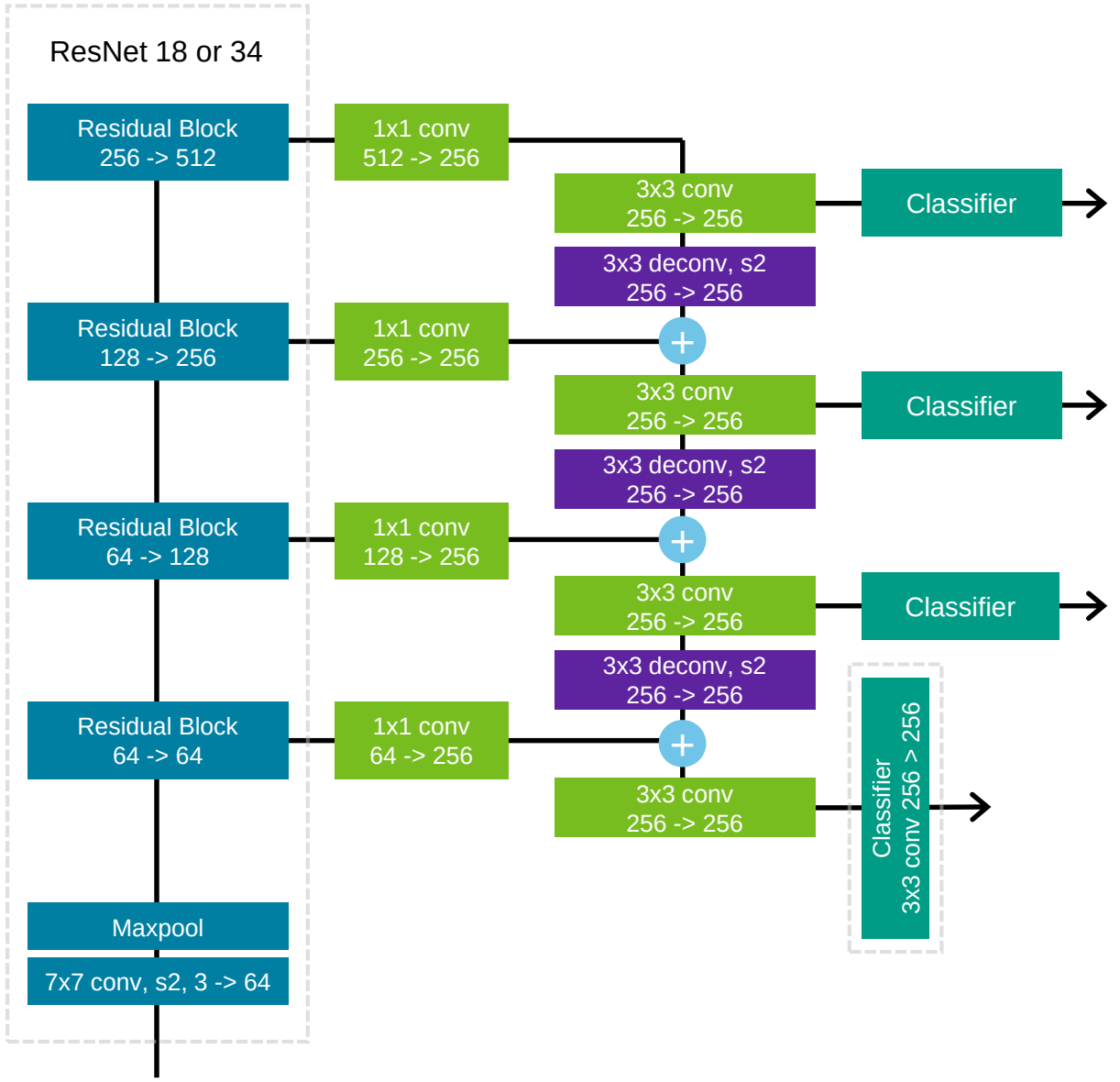
Innovation Example: Pixel Networks

Best in perf/params



Micron Confidential

Innovation in Neural Nets: Multi-task



Micron Confidential

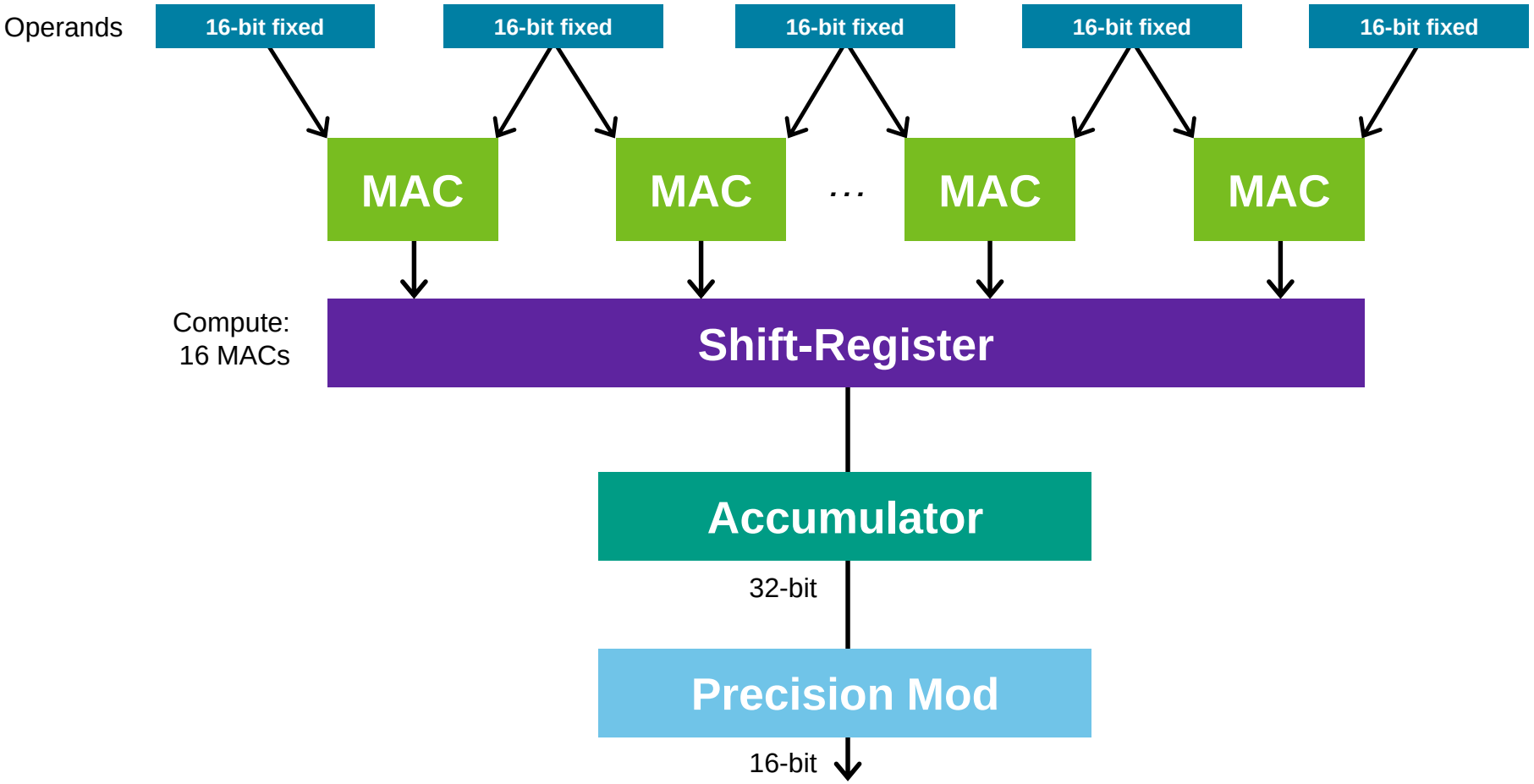
“General Purpose” AI

Multiple Networks Running on same HW

Micron Confidential

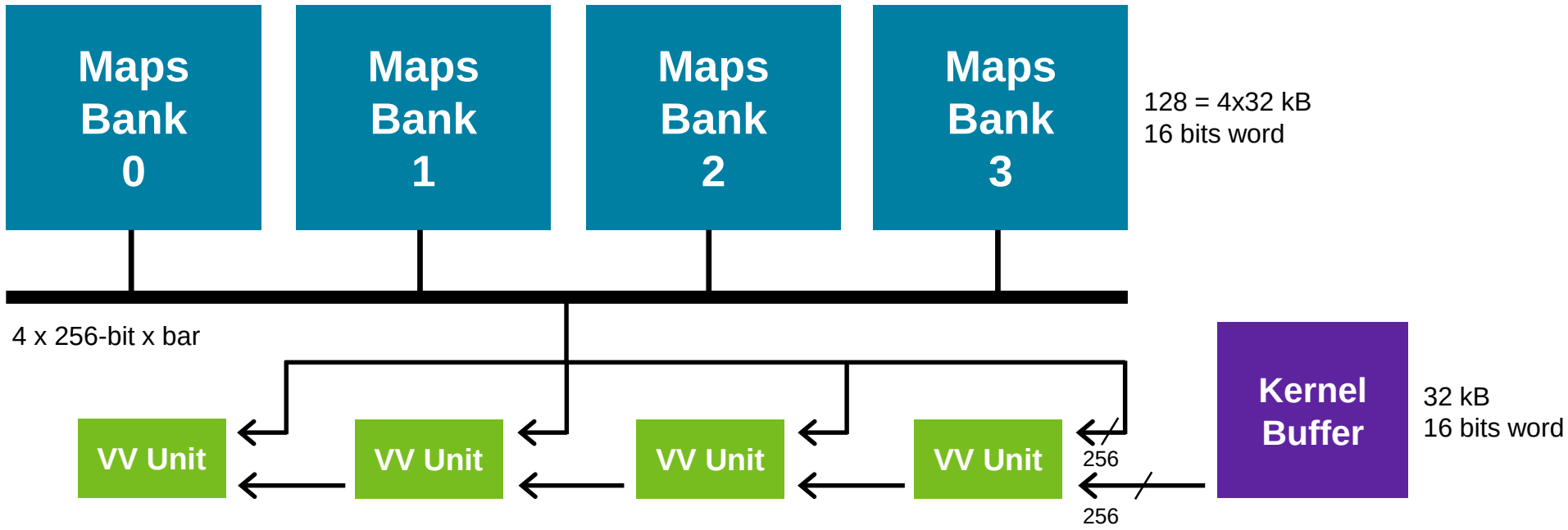
Micron Inference Engine Architecture

Vector-Vector (VV) Unit



Micron Confidential

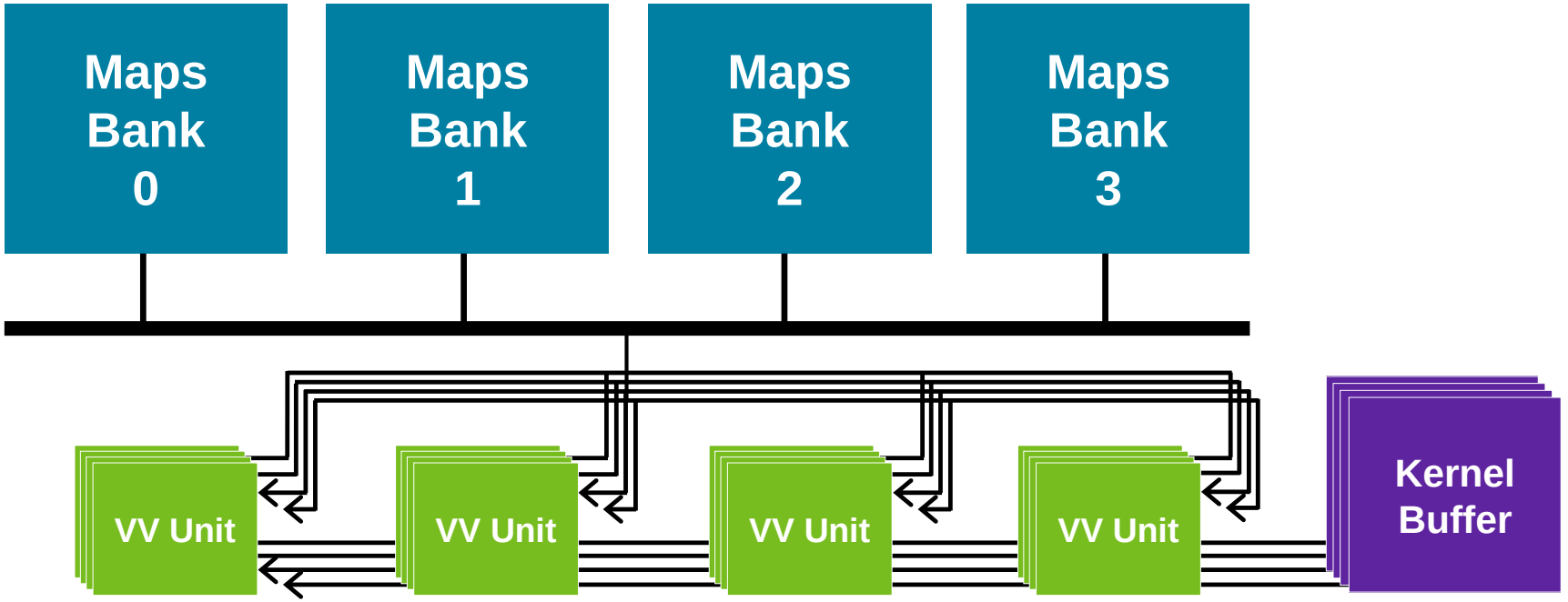
Matrix-Vector (MV) Unit



Total compute = 64 MACs

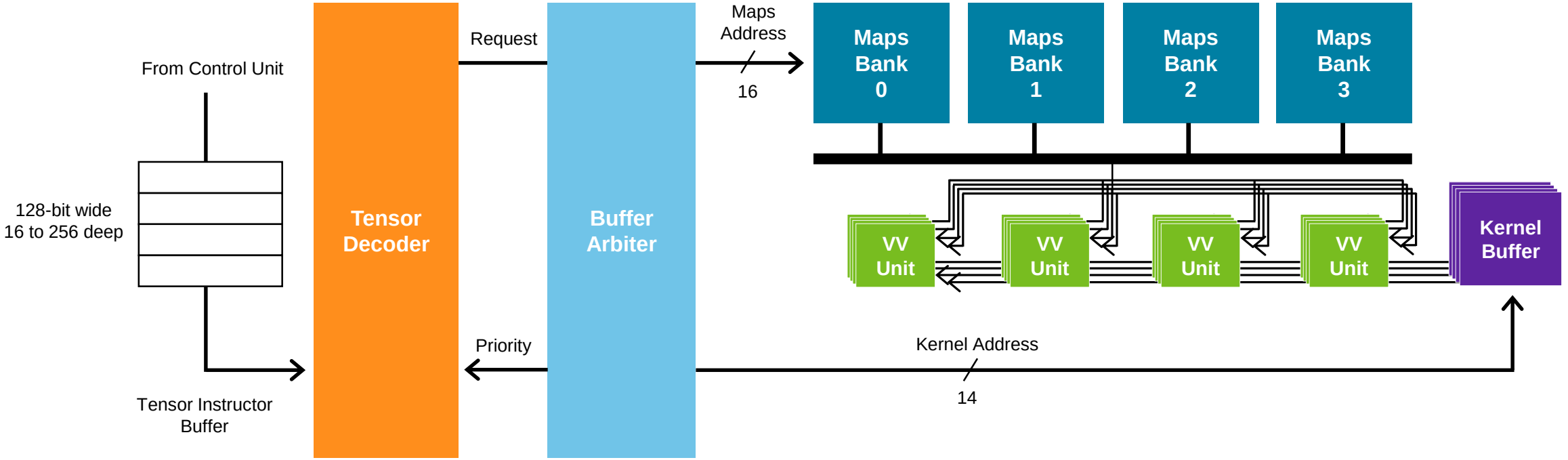
Micron Confidential

Matrix-Matrix (MM) Unit



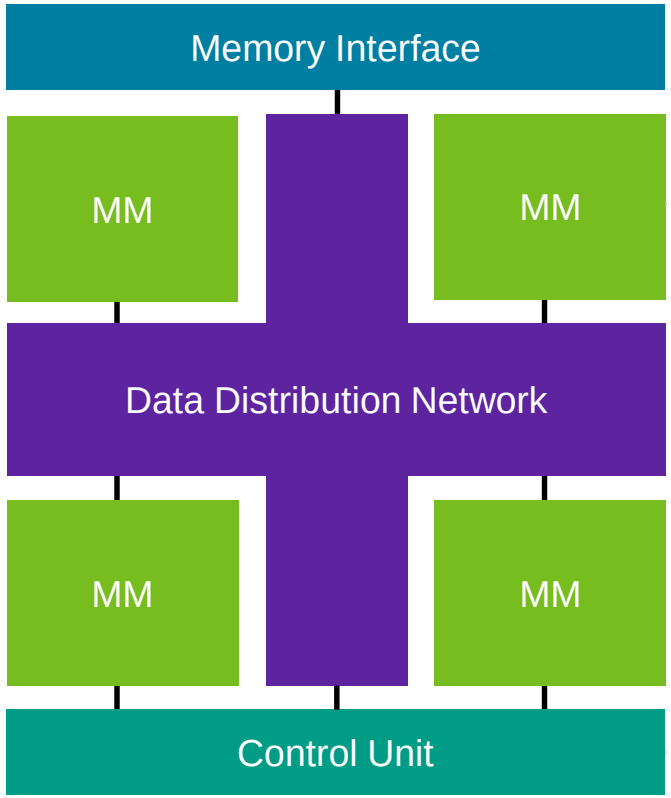
Total memory = 256 kB
Total compute = 256 MACs

Matrix-Matrix (MM) Unit

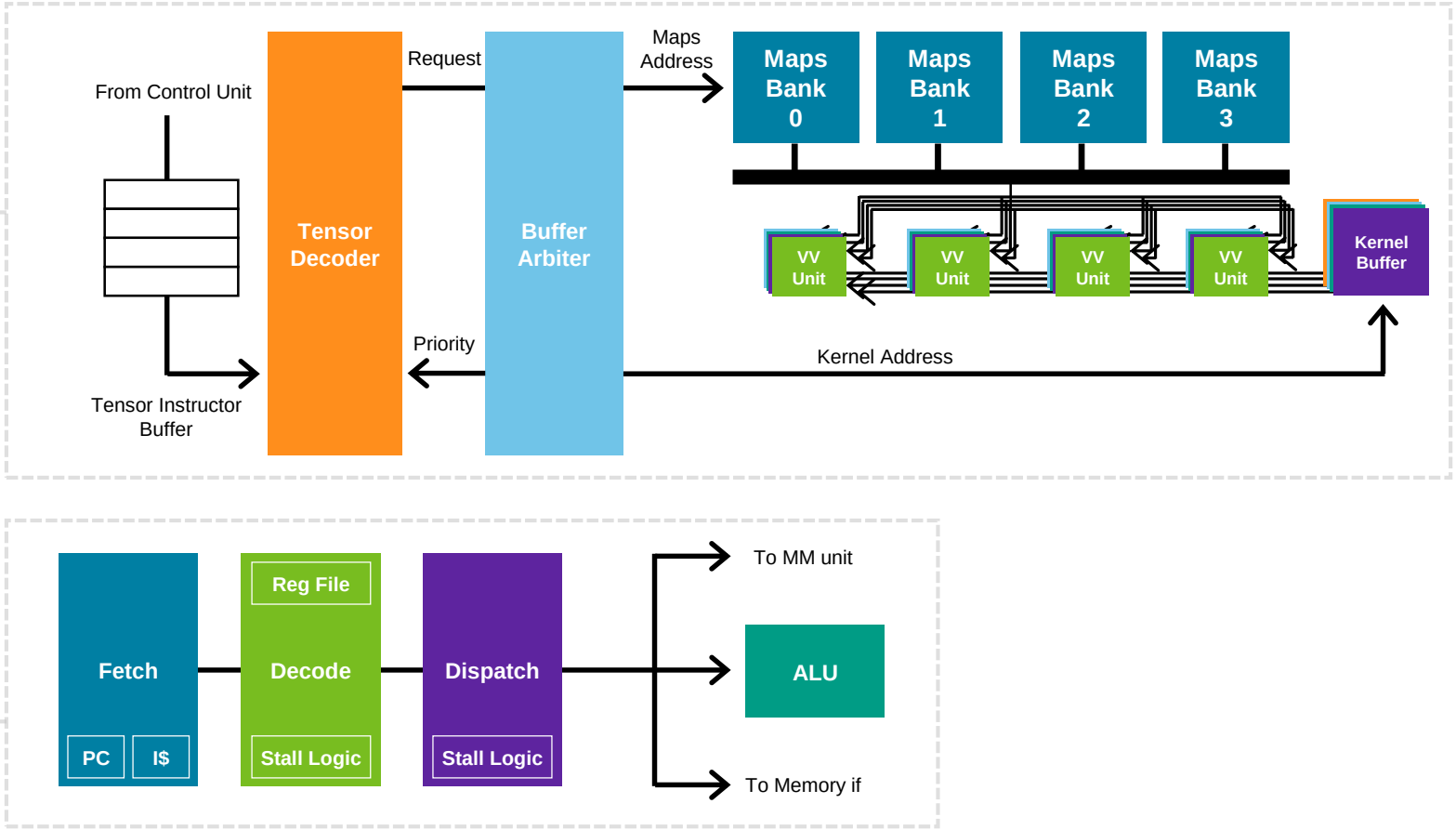


Micron Confidential

Micron Inference Engine Architecture



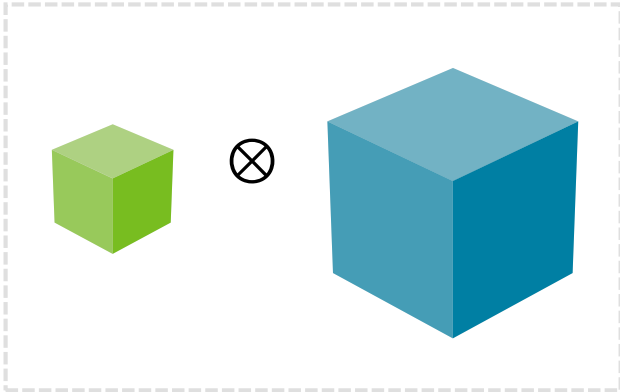
Total memory = 1 MB
Total compute = 1024 MACs



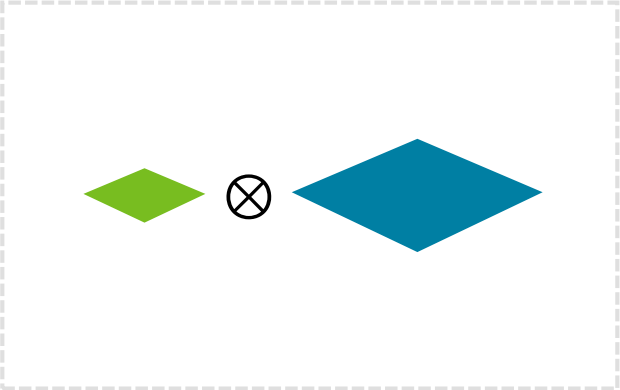
Micron Confidential

Efficient Hardware Utilization

Deep Learning Ops



Convolutions



Matrix by Vector(s)

Micron Inference Engine: Key to Occupancy

1. Convolutions

2. Maxpool

⊗ replaced by “compare”

3. ResAdd

Added values added to 1st vv-unit

4. Modes

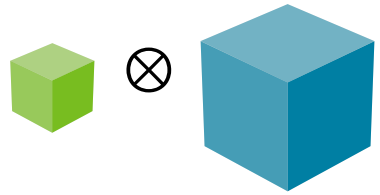
Works for:
Row stationary
Column stationary
Any other mode

Inference Engine Occupancy

- K-Plane tensors
- Slice tensors into small pieces
- Small granularity of compute
- More complex compiler
- More flexibility to new algorithms

Micron Inference Engine: Example Convolution

Convolution, Stride = 1



12x12x64 Map
3x3x64 Kernel

Kernels ⊗ Maps

Partitioning

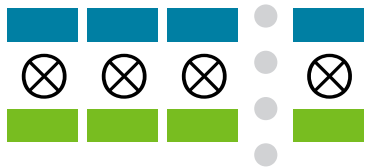
K-Tall Slice
12x3x64

Kernel
3x3x64

Compute

Unroll Tensor
3x3x64 products

Splice Tensor



Each product: 16 MAC
3x3x64 / 16 =
36 vectors multiplied

Compiler

Micron Confidential

Models	CNN	RNN	Custom
Framework	  		Others
Format	ONNX		
SDK	Python API	C API	Installer and Docs
Compiler	AI Model Parser	AI Model Quantizer	
	DLA Optimizer	DLA Assembler	
	DLA Run Time		
Hardware	FPGA DLA	ASIC DLA	

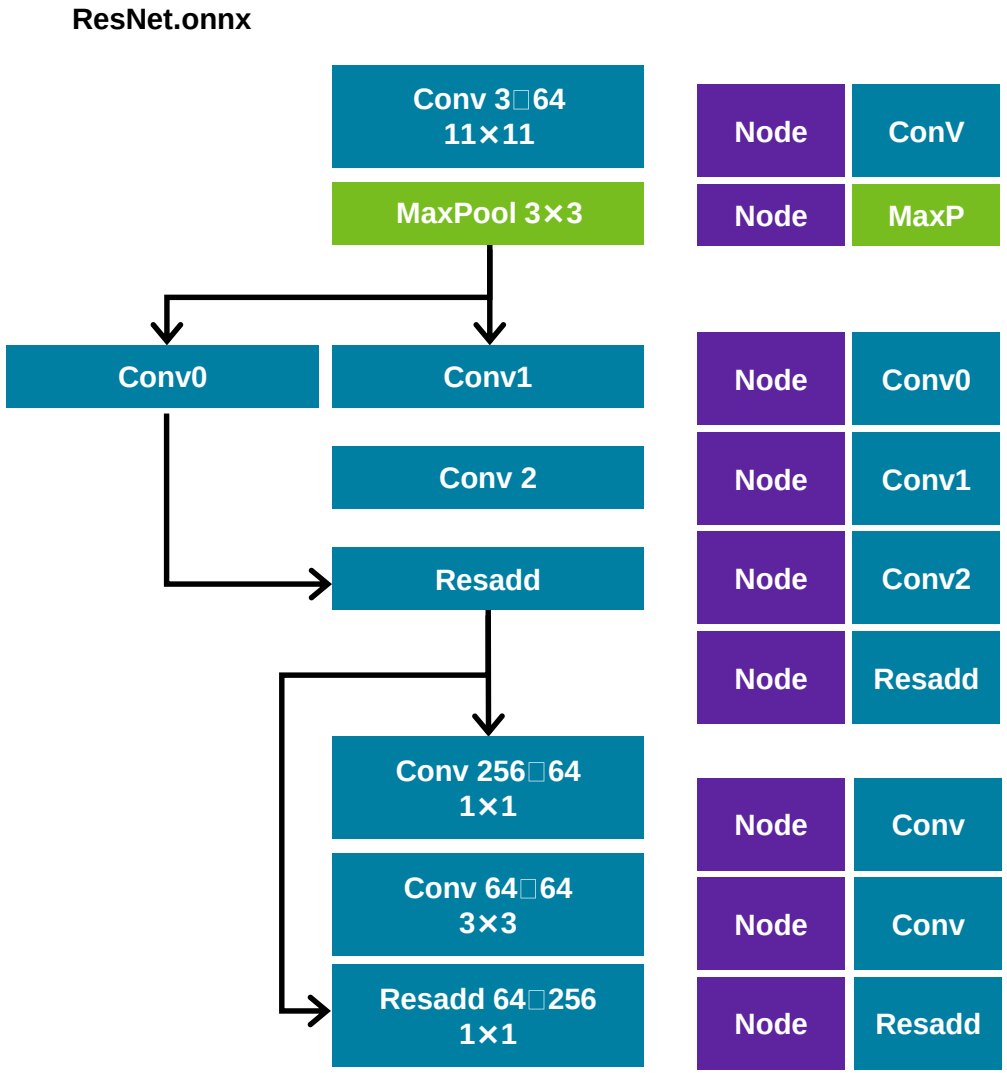
Micron Confidential

Parsing Neural Networks

Parse

Partition

Code Generation



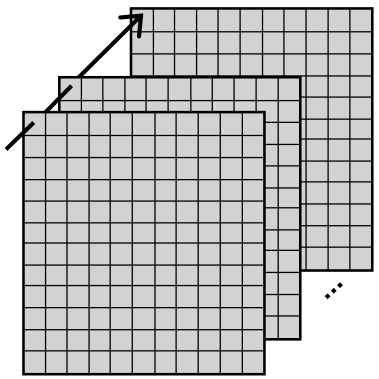
Micron Confidential

Partitioning the workload

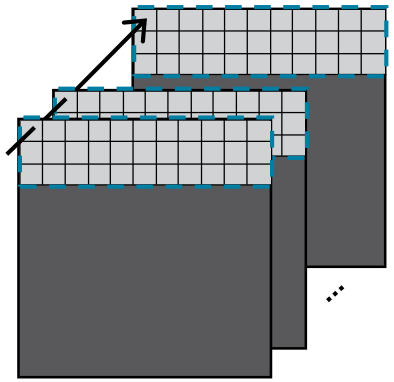
Parse

Partition

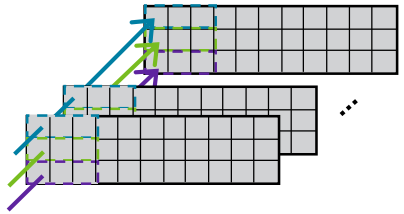
Code Generation



Tensors: Depth minor



TMOV instruction loads a tensor from memory to on-chip cache



Compute the entire filter

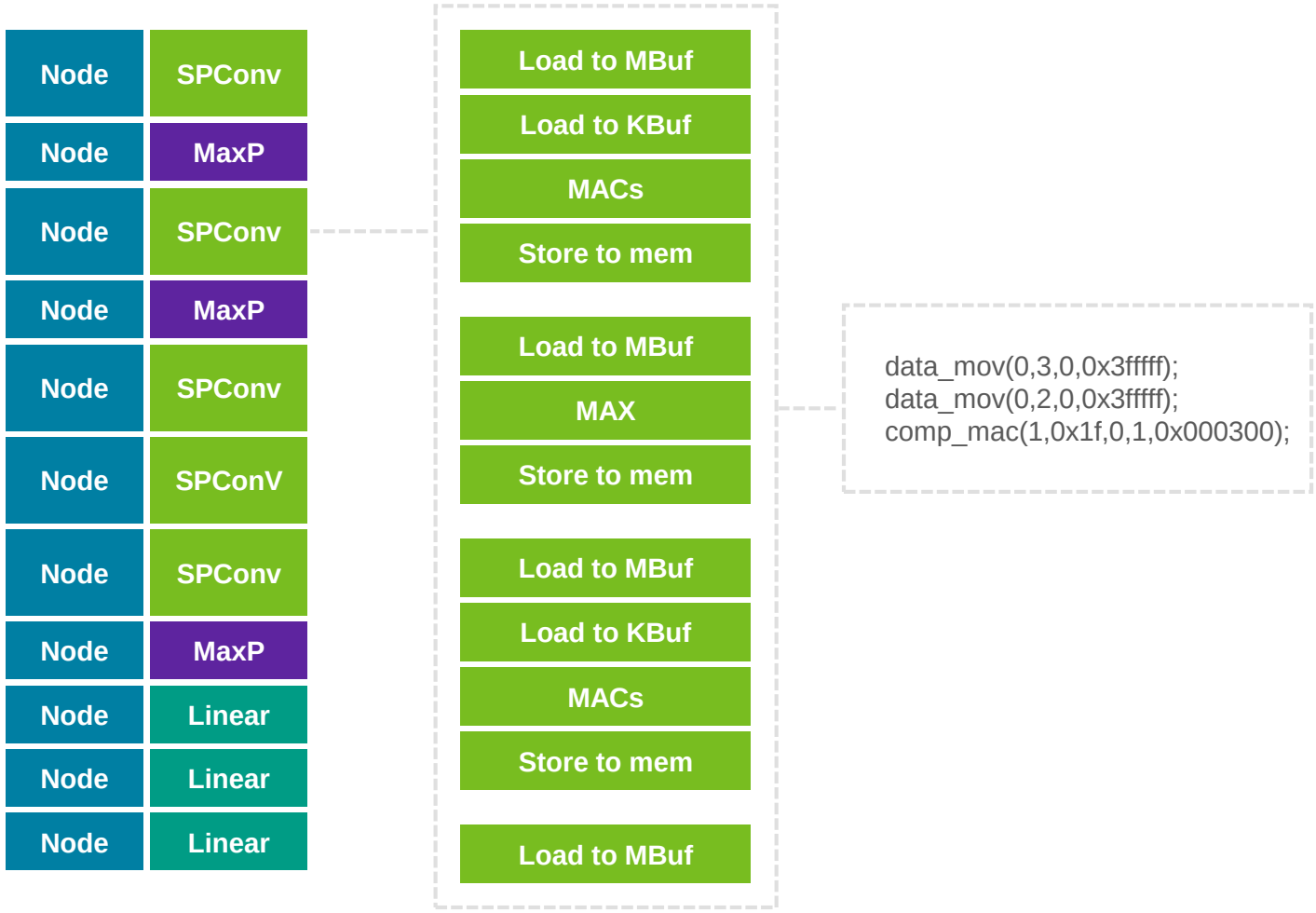
Micron Confidential

Generating Code

Parse

Partition

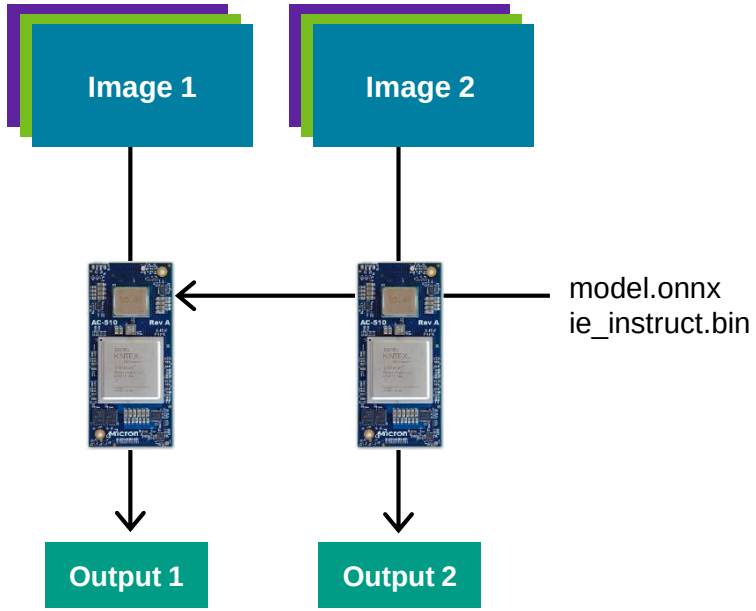
Code Generation



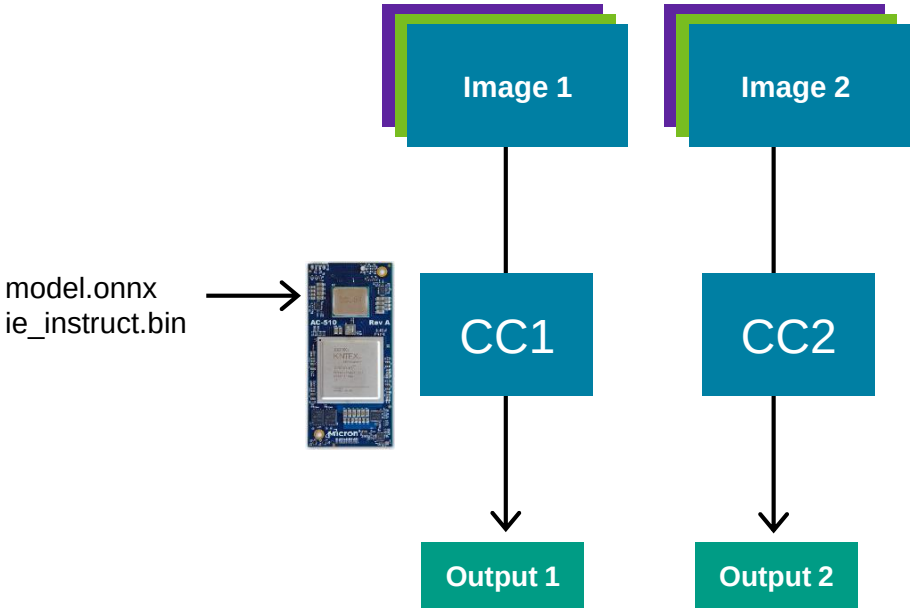
Micron Confidential

Efficient Compiler

2 DLAs, 2 images



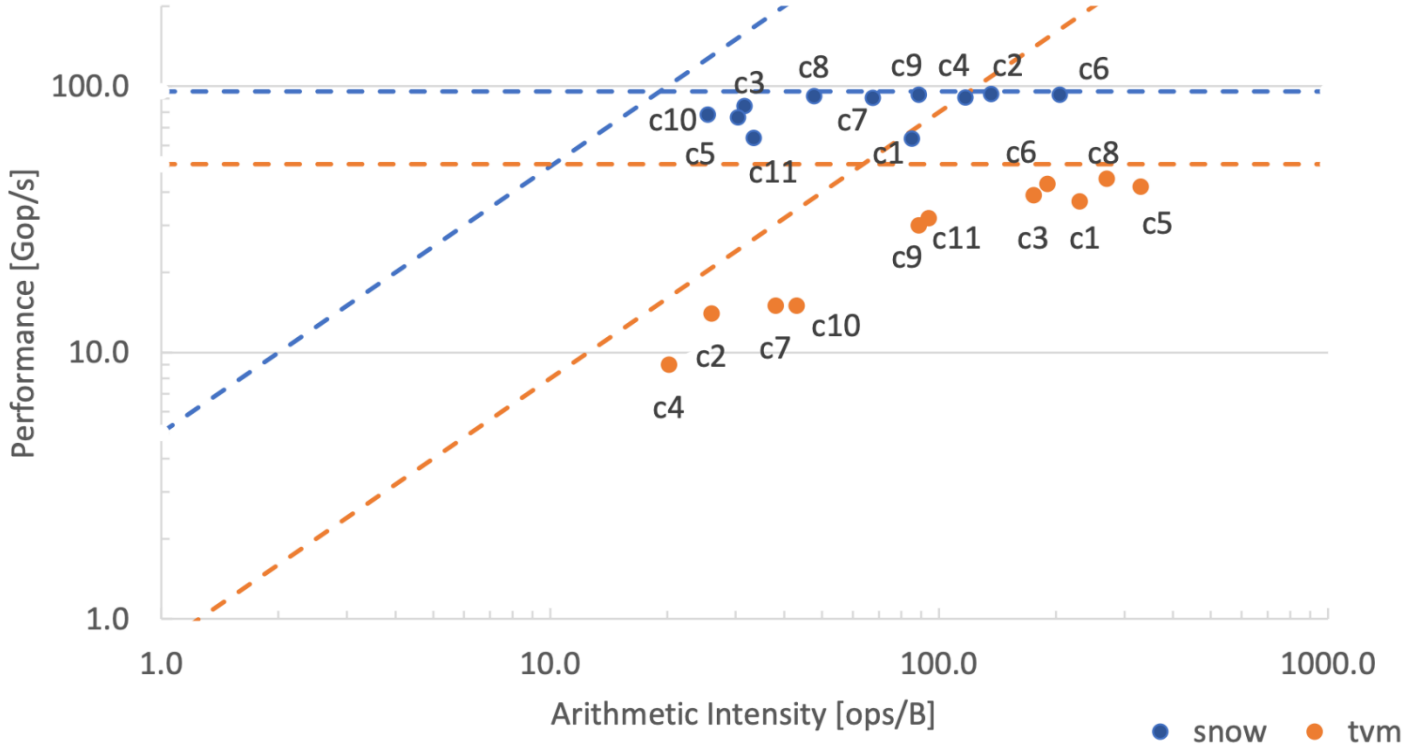
1 DLAs, 2 images



Micron Confidential

Efficient Compiler

Superior performance when compared to the popular open-source TVM complier!



C1 conv in=(H224W224P3) out=(H112W112P64) k=7x7 stride=2x2 pad=(3,3)
C2 conv in=(H56W56P64) out=(H56W56P64) k=3x3 stride=1x1 pad=(1,1)
C3 conv in=(H56W56P64) out=(H56W56P64) k=1x1 stride=1x1 pad=(0,0)
C4 conv in=(H56W56P64) out=(H28W28P128) k=3x3 stride=2x2 pad=(1,1)
C5 conv in=(H56W56P64) out=(H28W28P128) k=1x1 stride=2x2 pad=(0,0)
C6 conv in=(H28W28P128) out=(H28W28P128) k=3x3 stride=1x1 pad=(1,1)
C7 conv in=(H28W28P128) out=(H14W14P256) k=3x3 stride=2x2 pad=(1,1)
C8 conv in=(H28W28P128) out=(H14W14P256) k=1x1 stride=2x2 pad=(0,0)
C9 conv in=(H14W14P256) out=(H14W14P256) k=3x3 stride=1x1 pad=(1,1)
C10 conv in=(H14W14P256) out=(H7W7P512) k=3x3 stride=2x2 pad=(1,1)
C11 conv in=(H14W14P256) out=(H7W7P512) k=1x1 stride=2x2 pad=(0,0)

Moreau, Thierry, et al. "VTA: An Open Hardware-Software Stack for Deep Learning." *arXiv preprint arXiv:1807.04188*(2018).


Micron®