PSVL

Embedded Computer Vision for Autonomous Systems

Denis Gudovskiy
Panasonic AI Lab
Mountain View, CA
denis.gudovskiy@us.panasonic.com

Panasonic AI

Panasonic Business Introduction

Appliance Company

Air Conditioner

TV

Eco Solutions Company

System Kitchen

Solar Panel System

Connected Solutions Company

Avionics System

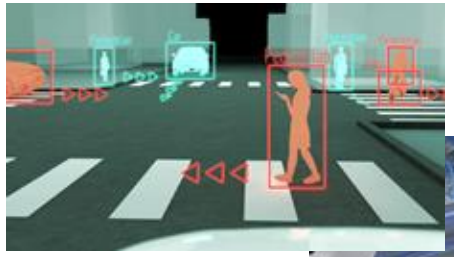
Factory Automation

Automotive & Industrial systems Company

Li-ion Battery

Next Gen. Cockpit System

ADAS system for Car OEM

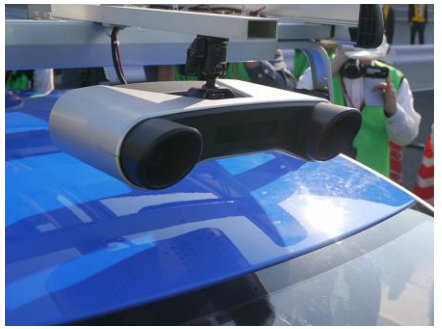


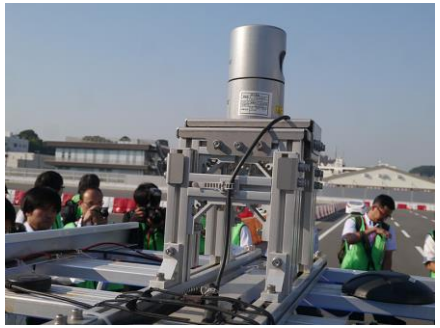
Robots

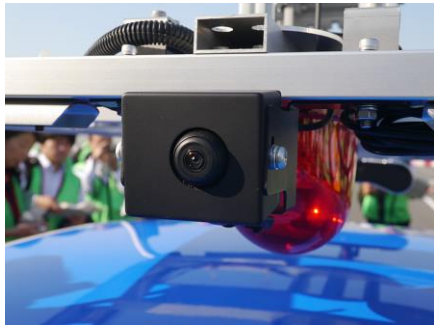
Security

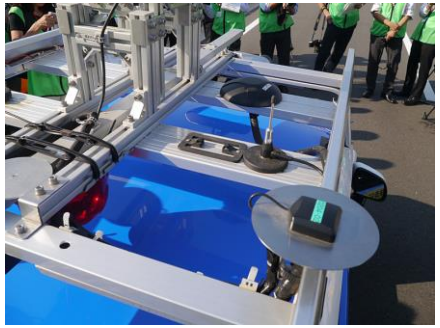
Imaging

Autonomous Commuter Project







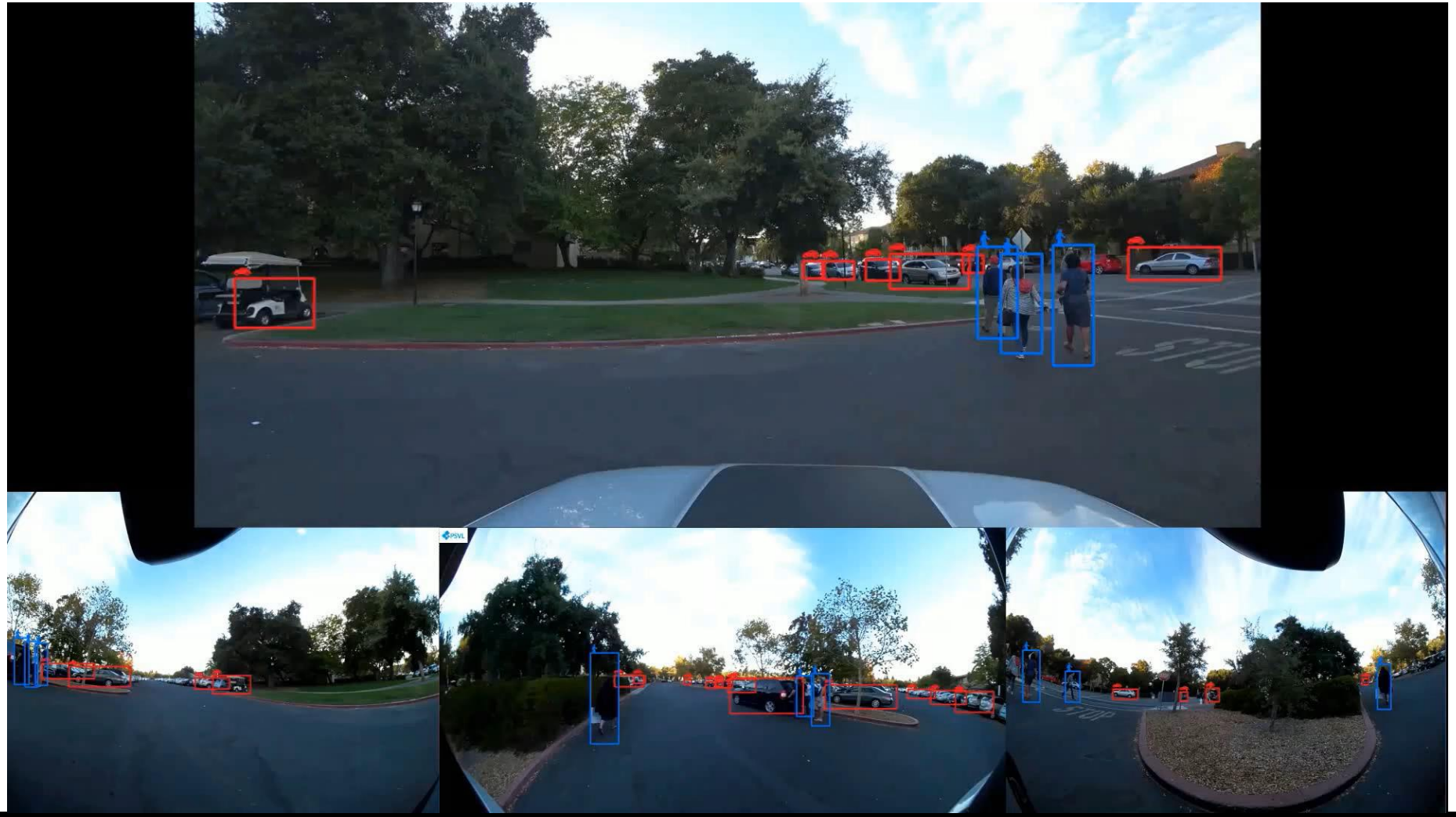


Platform to check our OEM solutions for car manufacturers:

- IVI (In-Vehicle Infotainment) and cockpit
- Car electrification and batteries
- **Perception, planning and control**

Computer Vision for Autonomous Systems

- Task: **DNN-based object detection** and tracking for 360° perception
- Target: decrease embedded **hardware cost** and **power consumption**

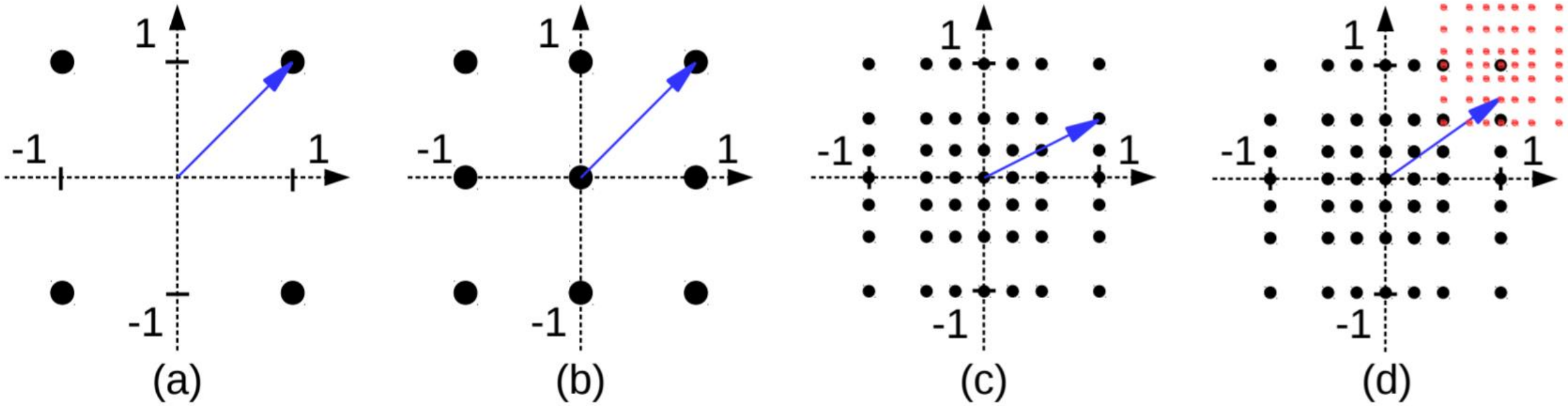


Embedded Object Detection

- Efficient detector DNN models
  - Single-stage or two-stage detector, off-the-shelf or NAS backbone
- Hardware-friendly quantization
  - Common schemes: integer, binary and ternary
- Off-chip memory traffic
  - Bottleneck for speed and power consumption
- DSP/FPGA/ASIC chips
  - Spec-dependent: power, latency, cost, programmability etc.

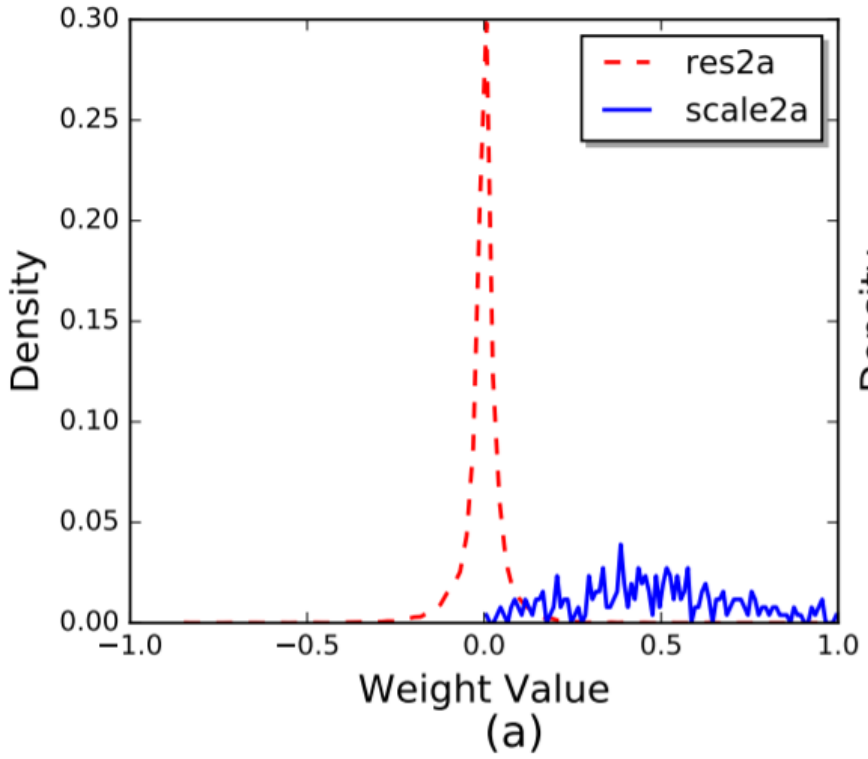
Efficient Quantization: ShiftCNN (arXiv:1706.02393)

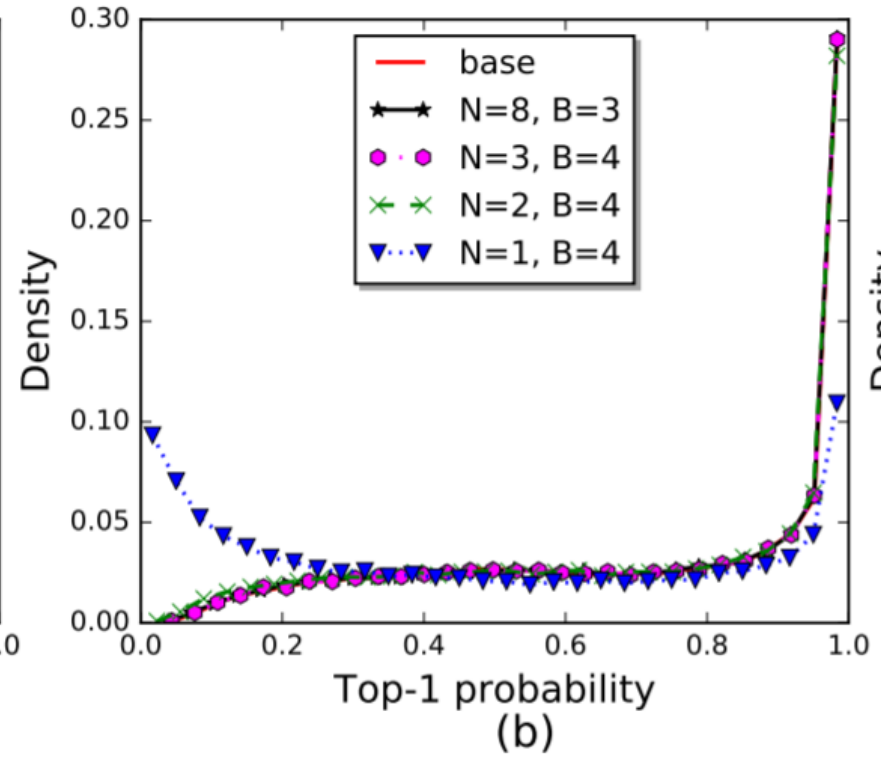
- Tradeoff: uniform integers and binary/ternary(a,b) quantization
- Nonuniform **$\log_2$** quantization for embedded hardware (c)
- ShiftCNN: sum of *N* $\log_2$ values each of *B*-bits (d)
- “No-retraining” mode to support existing models

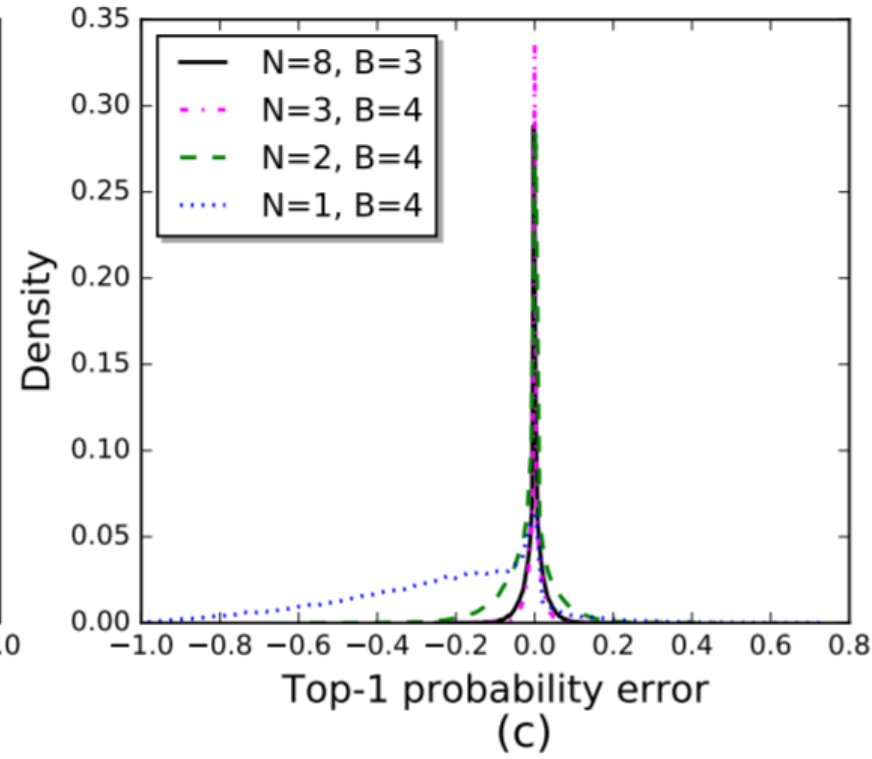


Efficient Quantization: ShiftCNN (arXiv:1706.02393)

- Weight prior has Gaussian distribution due to L_2 regularization
- Distribution of weights and batch normalization scales (a)
- Most of the DNNs work well when N=2 and B=4 (b,c)
- Why ShiftCNN: ALU performs only inexpensive shifts and adds







Efficient Quantization: ShiftCNN (arXiv:1706.02393)

Image classification results and power simulations for Xilinx FPGA

Table: 1. ImageNet accuracy of baseline models and corresponding ShiftCNN variants.

Network	Shifts N	Bit-width B	Top-1 Accuracy, %
SqueezeNet	base	32	58.4
SqueezeNet	1	4	23.01
SqueezeNet	2	4	57.39
SqueezeNet	3	4	58.39
GoogleNet	base	32	68.93
GoogleNet	1	4	57.67
GoogleNet	2	4	68.54
GoogleNet	3	4	68.88
ResNet-18	base	32	64.78
ResNet-18	1	4	24.61
ResNet-18	2	4	64.24(61.57)
ResNet-18	3	4	64.75(64.69)

Table: 2. FPGA utilization and power consumption estimates

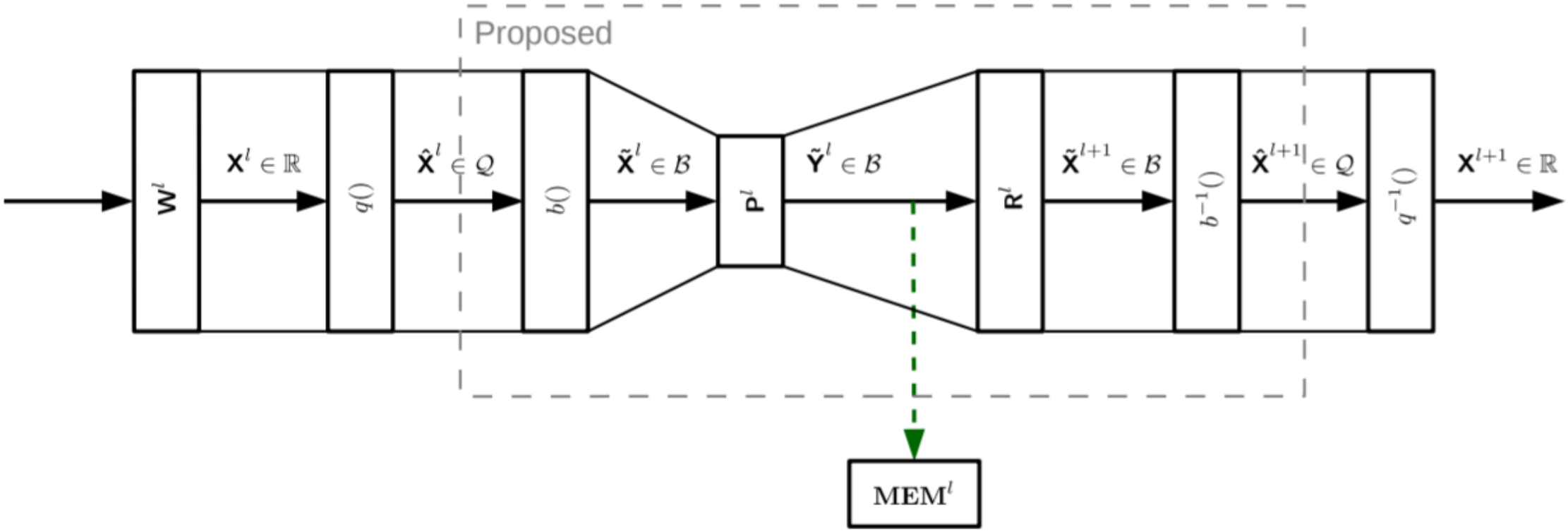
ALU	LUTs	FFs	DSPs	Power, mW
Add. only	1644	1820	0	76
ShiftCNN	4016	2219	0	102
Mult. DSP	0	2048	191	423
Mult. LUT	10064	4924	0	391

ShiftCNN experiments:

- No accuracy decrease without retraining
- N=2, B=4 config works well for classification
- **4× power reduction** for FPGAs

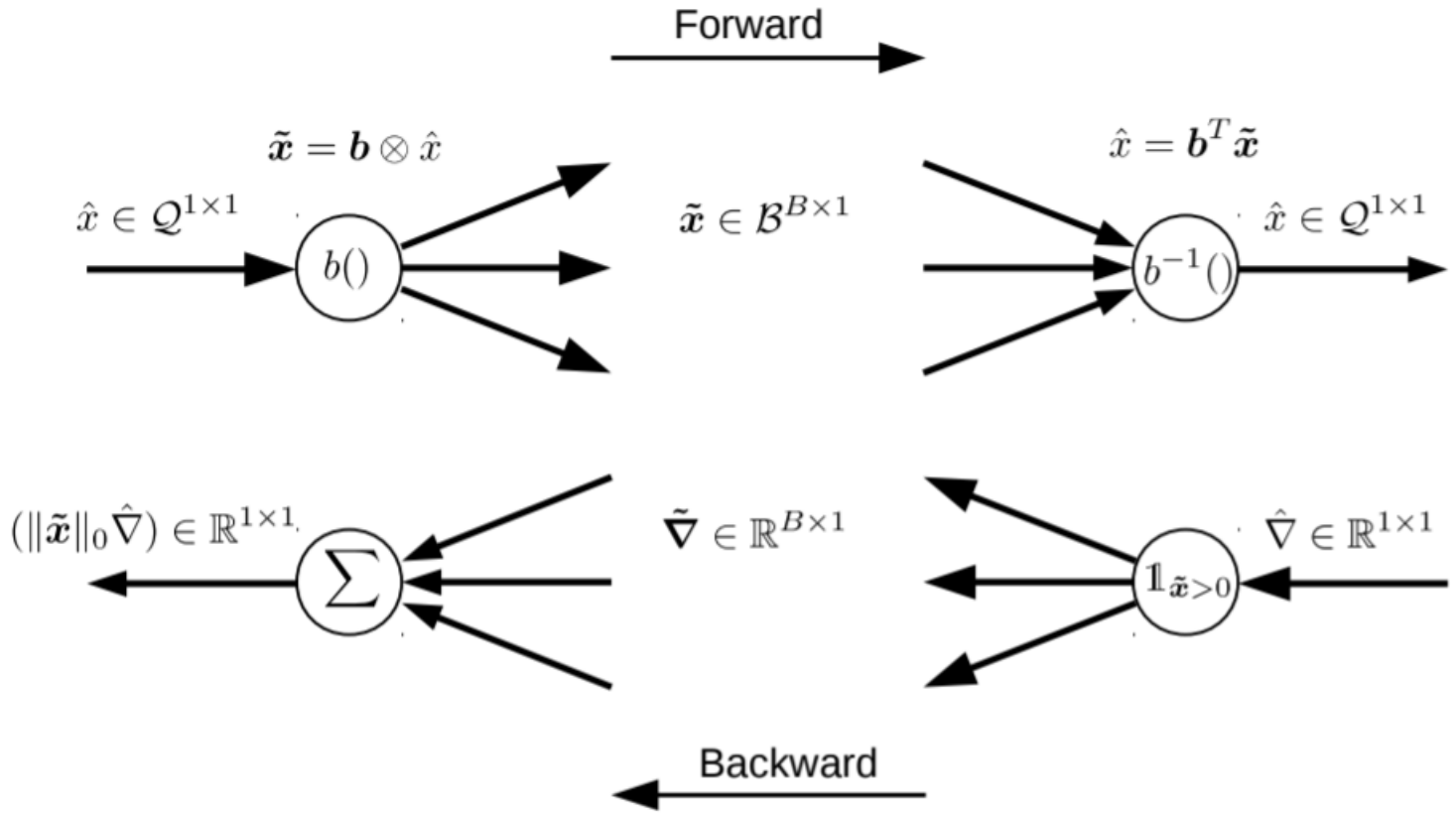
DNN Memory Compression (arXiv:1808.05285)

- Motivation: DRAM memory access is **100×** more power-hungry than on-chip
- Conventional methods: layer fusion, bottleneck layers, quantization, sparsification
- Our idea: project integers into binary vector space and compress binary vectors
- Outcome: keep on-chip only dense compressed feature maps



DNN Memory Compression (arXiv:1808.05285)

- Compression requires DNN finetuning
- Problem: quantization is not differentiable operation
- Solution: straight-through estimator, identity initialization etc.
- Recent research: better solutions exist!



DNN Memory Compression (arXiv:1808.05285)

- SSD512 detector with SqueezeNet backbone
- 512×512x3 = 768 KB input image
- Experiments on Pascal VOC2007 dataset
- We compress 1st two layers with the highest memory consumption
- Factor of **2×** compression can be achieved!

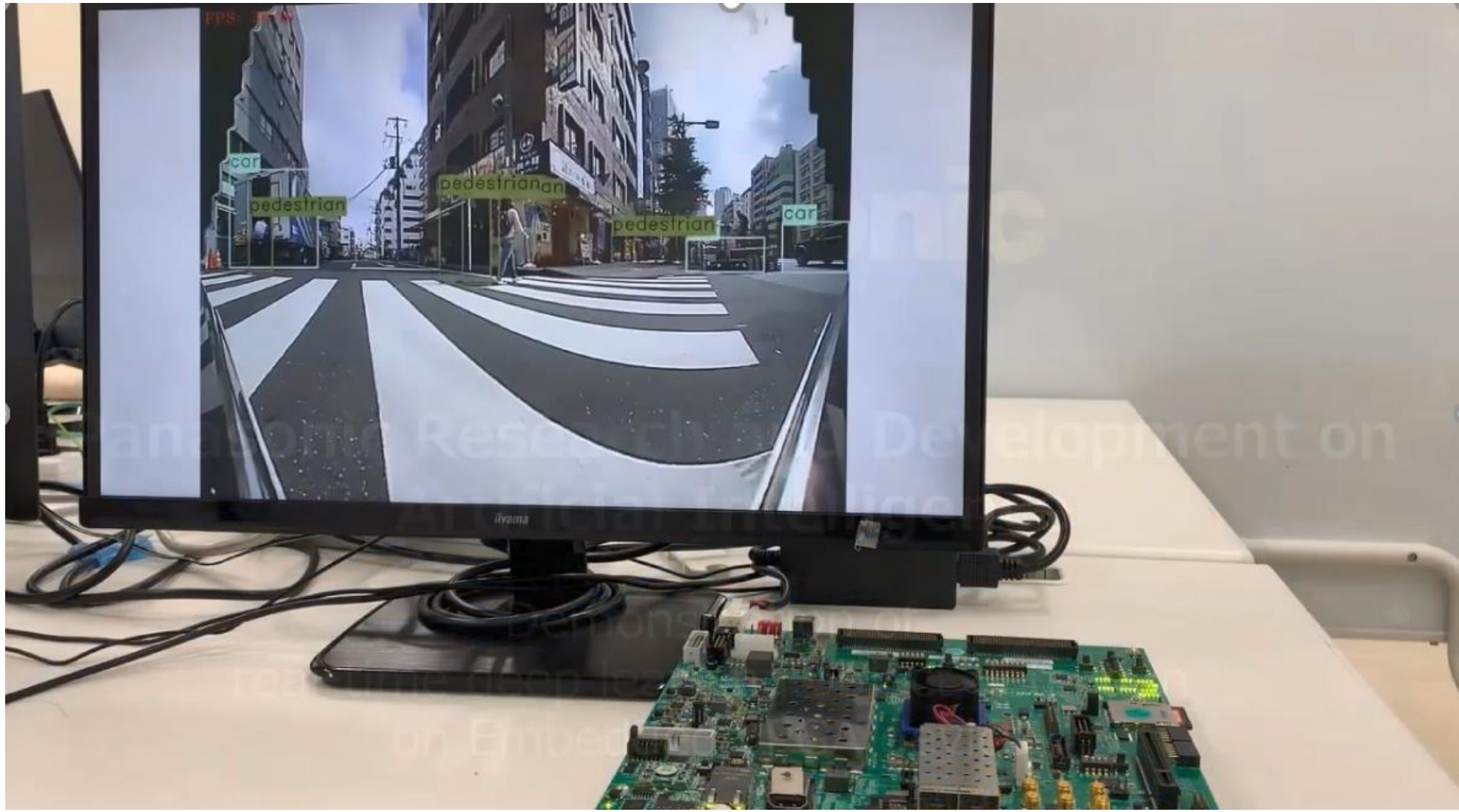
A size, KB	Base, fp32	F, fp32	F+Q, uint8	F+Q, uint4	F+Q+C, uint8
input (int8)	768	768	768	768	768
conv1	16384	0	0	0	0
mpool1	4096	0	0	0	0
fire2,3/squeeze	2048	2048	512	256	128
fire2,3/expand	16384	0	0	0	0
Total	38912	2048	512	256	128
mAP, %	68.12	68.12	68.08	65.92	64.39
Compression	-	19×	76×	152×	304×

Model	W size, MB	A size, KB	mAP, %
fp32	23.7	2048	68.12
Quantized			
uint8	23.7	512	68.08(68.04)
uint6	23.7	384	67.66(67.14)
uint4	23.7	256	65.92(44.13)
uint2	23.7	128	55.86(0.0)
Proposed $b() \rightarrow 1\times1 \rightarrow 1\times1 \rightarrow b^{-1}()$			
uint6	23.9	384	68.17
uint4	23.8	256	65.42
Proposed: $b() \rightarrow 3\times3/2 \rightarrow 3\times3*2 \rightarrow b^{-1}()$			
uint8	26.5	128	63.53
uint6	25.9	96	62.22
Proposed: $b() \rightarrow 2\times2/2 \rightarrow 2\times2*2 \rightarrow b^{-1}()$			
uint8	24.9	128	64.39
uint6	24.6	96	62.09

FPGA Object Detector

Specs:

- Device: ZCU102
- Speed: 30 fps
- Power: 14W
- Accuracy: 71% mAP (Pascal VOC2007)

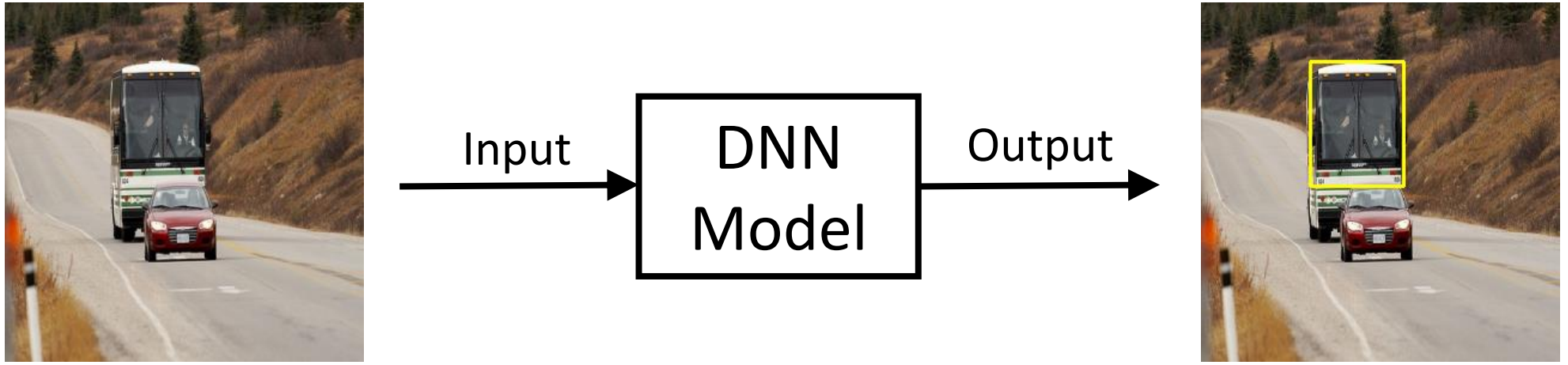


Are We Done?

- We got power-efficient and fast object detector
  - ❖ How to make it 100% accurate?
  - ❖ Tune model knobs to improve accuracy
  - ❖ But, usually, it is not about the model!
- Dataset selection
  - ❖ Critical to achieve target accuracy
- Debugging tools
  - ❖ Analyze detector predictions for customers and yourself
- More dataset selection and debugging
  - ❖ Machine learning is about the data

Object Detection: Explain to Fix (arXiv:1811.08011)

- DNNs are ***black boxes*** due to complex nonlinear models
- Customers of DNN models would like to understand output predictions

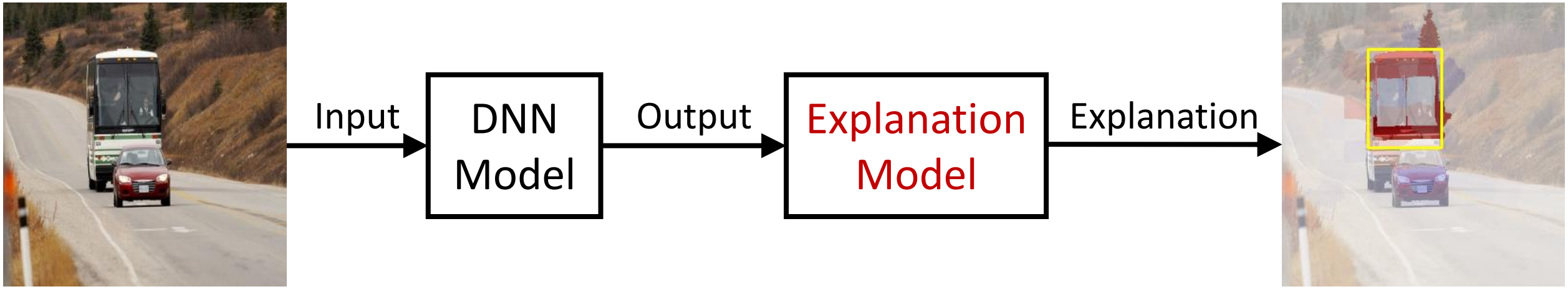


{class, x_1, y_1, x_2, y_2}

- Challenges:
  - ✓ Can we explain DNN predictions for our customers?
  - ✓ Can we estimate explanation at reasonable time?
  - ✓ Can we improve predictions using explanation?

Object Detection: Explain to Fix (arXiv:1811.08011)

- ***Explanation models*** are able to address these questions



- What is explanation:
  - ✓In computer vision, explanation is a spatial feature importance map
  - ✓Importance values attribute the output effect to the input feature
  - ✓Important values can be either **positive** or **negative** or neutral
  - ✓Explanations can identify missing features in the training dataset!

Object Detection: Explain to Fix (arXiv:1811.08011)

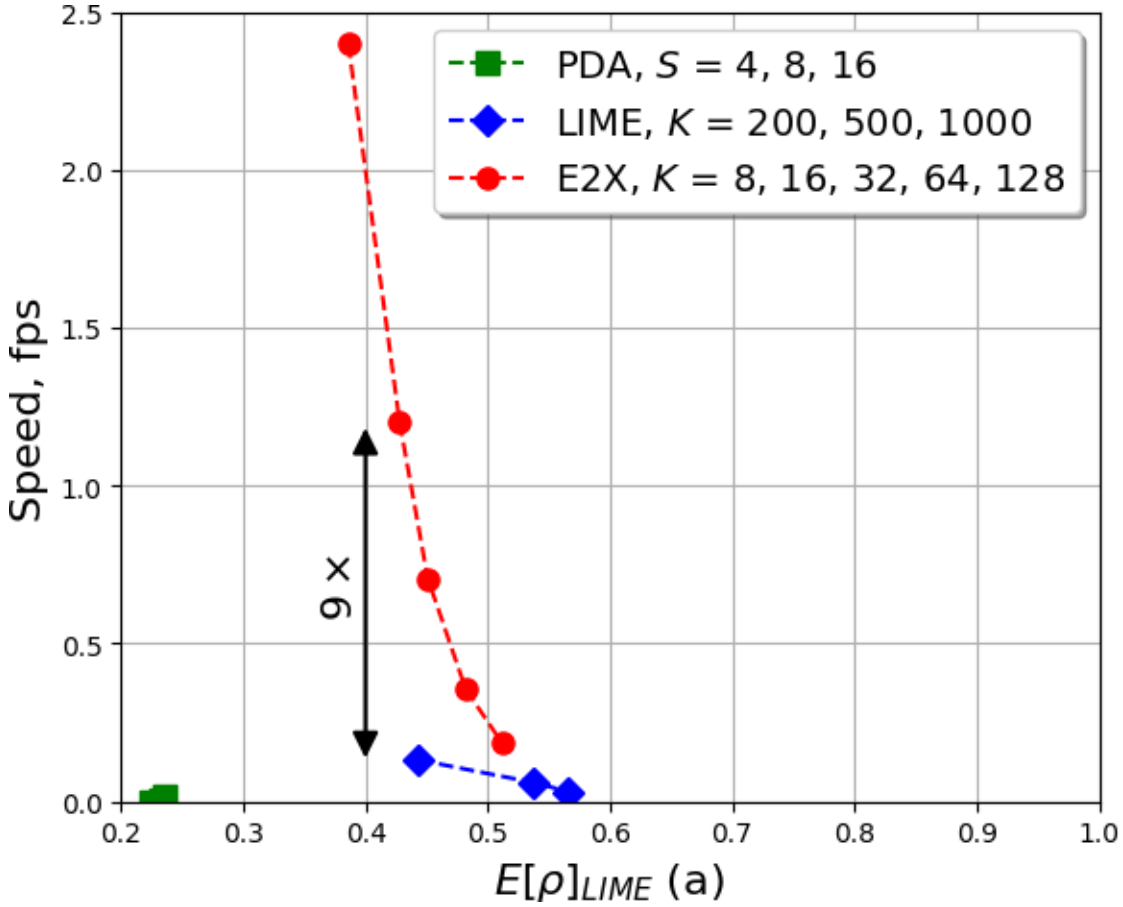
- Explain to Fix (E2X) is an ***explanation method*** combined into ***software framework***:
  - ✓High-fidelity explanations comparable to state-of-the-art
  - ✓10× higher computational efficiency with the similar quality
  - ✓Can be used to process large datasets on a GPU cluster
- E2X is a *local backpropagation-based attribution* method:
  - ✓All M feature importance values are estimates simultaneously

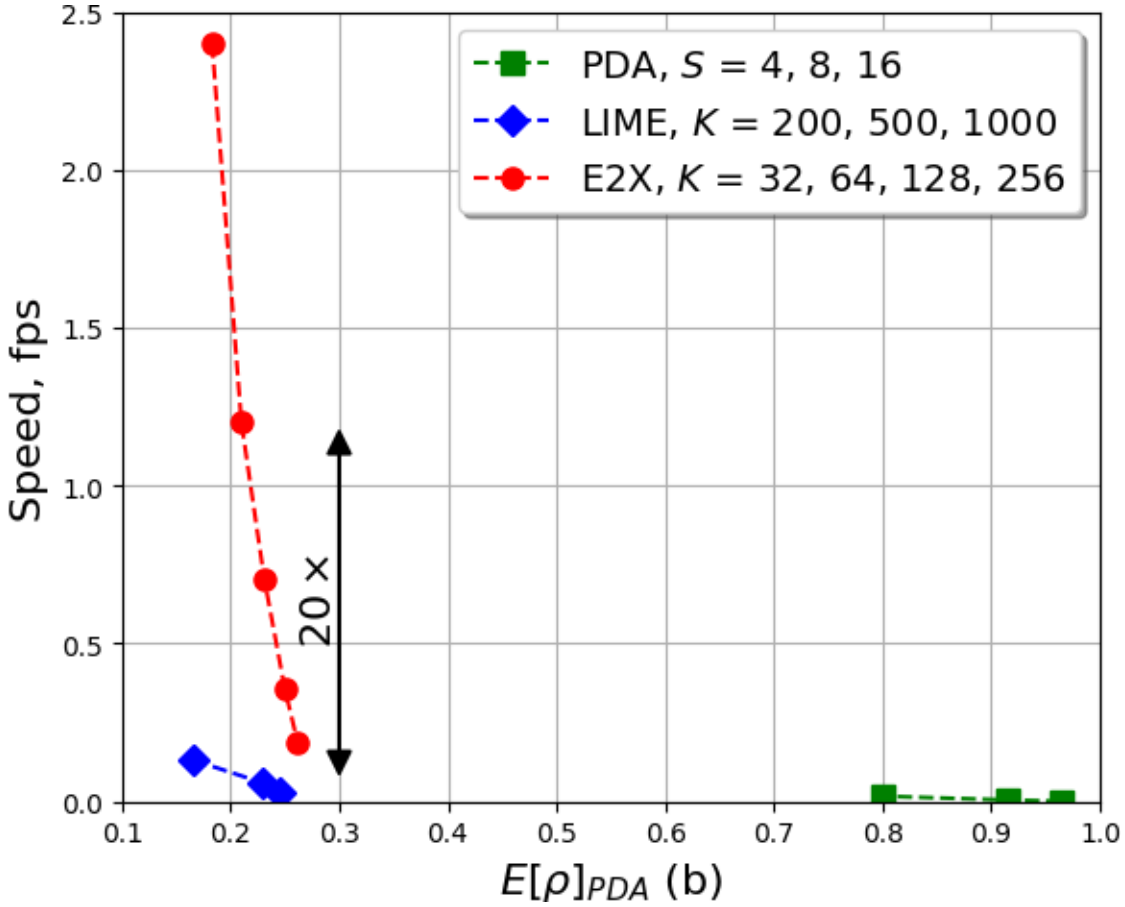
$$\phi = s_x\left(\frac{x}{K}\sum_{k=1}^{K}\frac{\partial f\{u_x(k)\}}{\partial u_x(k)}\right)$$

  - ✓K is the number of samples (forward and backward passes)
  - ✓Path function $u_x(k)_i = w_i(k)h_x(\acute{x})_i$
  - ✓Weights $w_i(k) \sim U\{0,1\}$

Object Detection: Explain to Fix (arXiv:1811.08011)

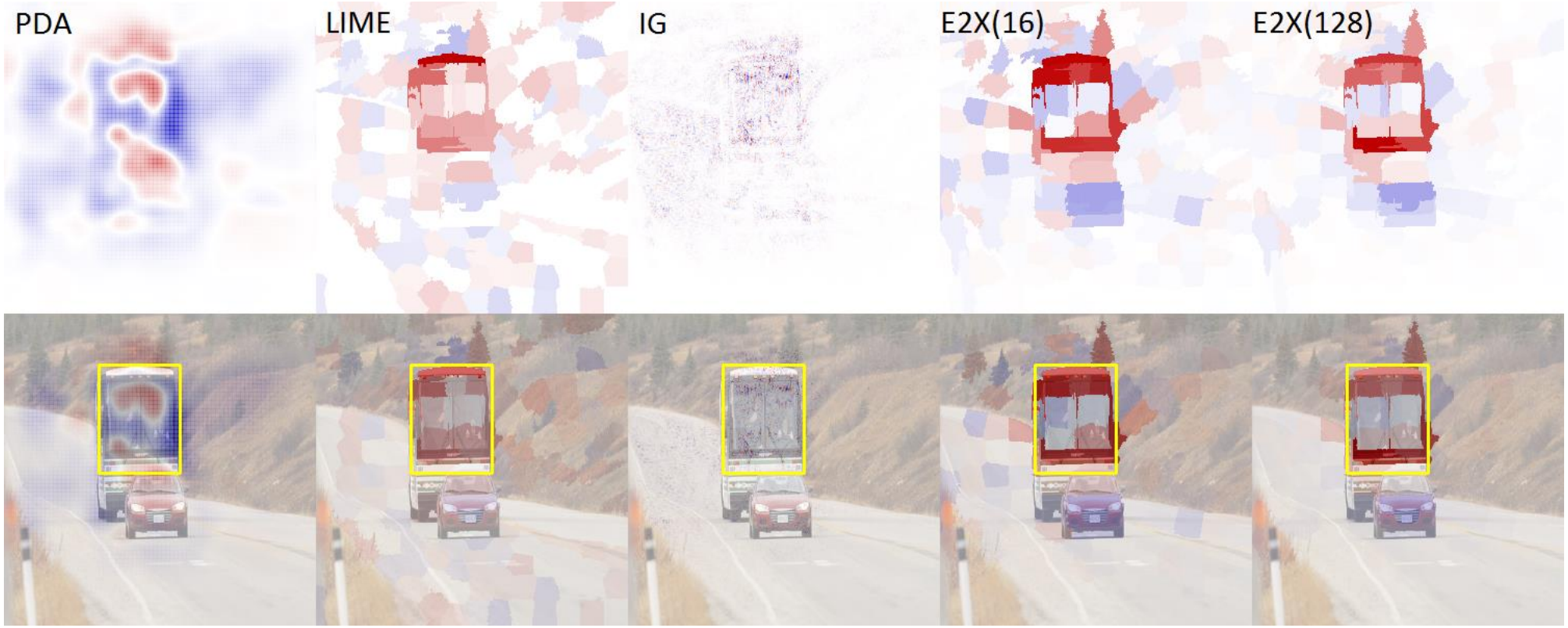
- We apply E2X to ***SSD300 object detector*** and calculate correlation $E[\rho]$ to reference model
- Correlation to state-of-the-art vs. computational speed





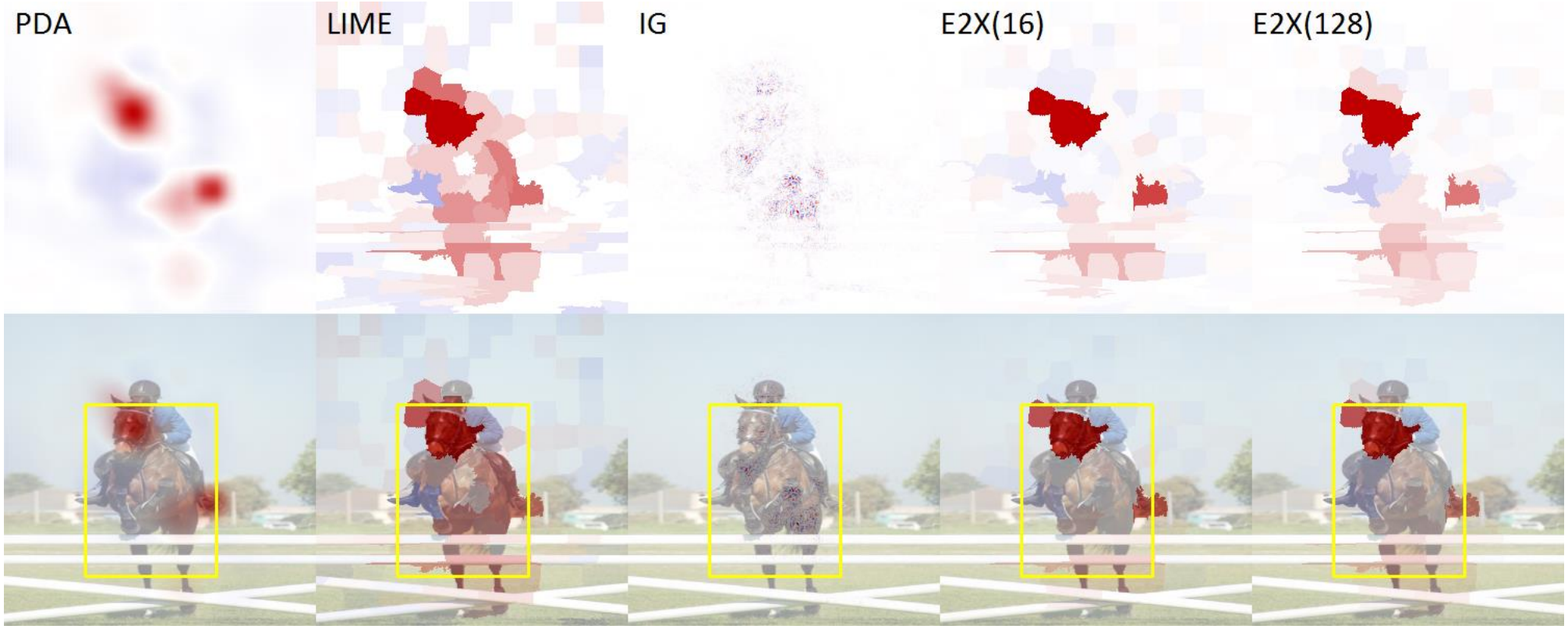
Object Detection: Explain to Fix (arXiv:1811.08011)

- We apply E2X to ***SSD300 object detector*** and analyze detections
- False negatives can be explained using E2X framework
- False negative **bus** here is occluded by a car that decreases detector confidence



Object Detection: Explain to Fix (arXiv:1811.08011)

- The most important features for a horse: head, legs and a tail
- Rider may decrease confidence if not present in the training dataset
- DNN is confused by a rider's left foot: model thinks it is horse tail!



Conclusions

- Hardware-efficient DNN quantization for embedded chips
  - ✓ 4× power reduction using ShiftCNN
- Feature map compression for on-chip processing
  - ✓ 2× memory drop in addition to prior methods
- Implementation of FPGA-based object detector
  - ✓ Better than GPU power efficiency with low latency
- Explainability tools for DNN detectors
  - ✓ Make your model transparent and accurate

Questions?

Panasonic AI