Institut für
Computertechnik
Institute of
Computer Technology

Efficient Inference at the Edge

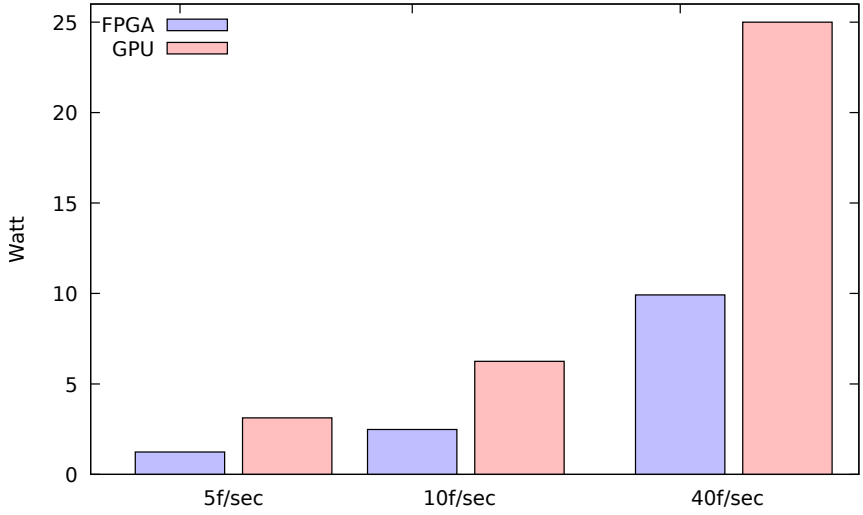
TECoSA Seminar

Center for Trustworthy Edge Computing Systems and Applications
KTH, Stockholm

Axel Jantsch

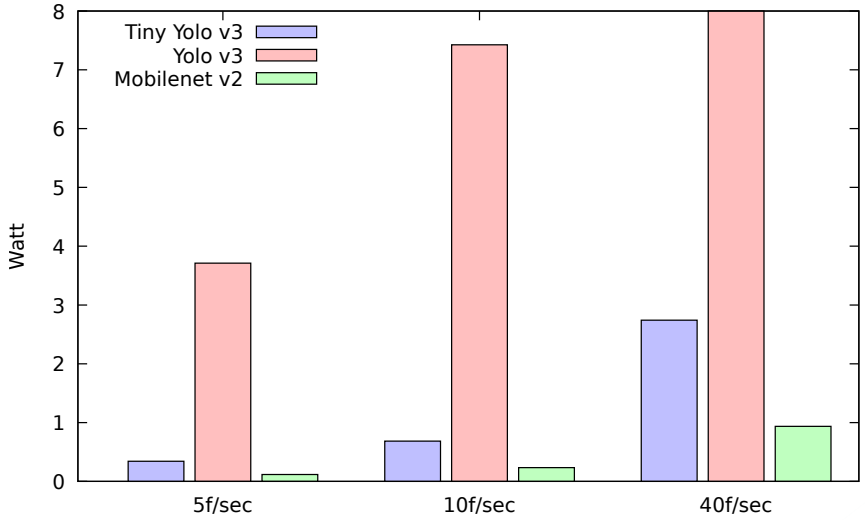
November, 3, 2022

Power Consumption in Inference



VGG16 applied to the ImageNet data set based on published papers.

Power Consumption in Inference



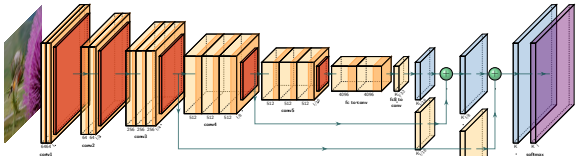
Object detection on the NCS2 platform; own measurements.

Resource limitations

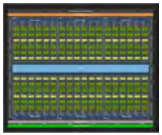
	Embedded	Data center
Computation [flop]	$30 - 1800 \cdot 10^{12}$	$86 \cdot 10^{18}$
Memory [bit]	10^{10}	10^{15}
Power [W]	5-100	$10^3 - 10^6$
Energy [Wh]	48-1000	$200 \cdot 10^6$

Computation Embedded refers to an Nvidia Jetson Nano running 1 min and 1 hour, respectively.
Computation server refers to the computation needed for the 40 day experiment with AlphaGo Zero
Energy embedded refers to a mobile phone and to a car battery, respectively.
Energy server refers to the 40 day experiment for AlphaGo Zero.

Design Space



ARM NN



Nvidia Turing

DNN Choices

- Convolutional layers
- Filter kernels
- Number of filters
- Pooling layers
- Filter shape
- Stride
- Fully connected layer
- Number of layers
- Regularization
- etc.

Mapping Choices

- Neuron pruning
- Data type selection
- Approximation
- Retraining
- Connection pruning
- Weight sparsifying
- Regularization
- etc.

Platform Choices

- Platform Selection
- Reconfiguration
- Batch processing
- Deep pipelining
- Resource reuse
- Hierarchical control
- Processing unit selection
- Memory allocation
- Memory reuse
- etc.

Outline

1. Estimation
2. Power Profiling
3. Traffic Light Controller Case Study
4. Shunt Connections
5. Ragweed Detection
6. DNN Partitioning for Inference

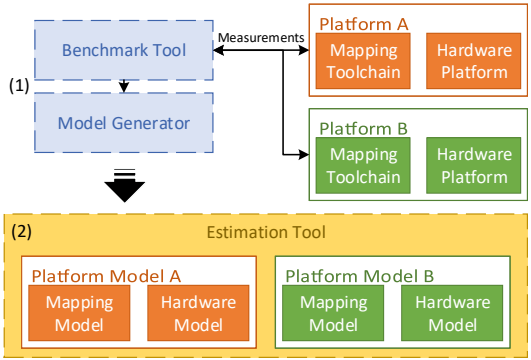
Outline

1. Estimation
2. Power Profiling
3. Traffic Light Controller Case Study
4. Shunt Connections
5. Ragweed Detection
6. DNN Partitioning for Inference

ESTIMATION

Estimation

- Two leading performance estimation tools: ANNETTE and Blackthorn
- For NCS2, Xilinx FPGA, and Jetson
- Combine analytic, statistical model and partial measurements



M. Wess, M. Ivanov, C. Unger, A. Nookala, A. Wendt, and A. Jantsch. "ANNETTE: Accurate Neural Network Execution Time Estimation With Stacked Models". In: *IEEE Access* 9 (2021), pages 3545–3556

Martin Lechner and Axel Jantsch. "Blackthorn: Latency Estimation Framework for CNNs on Embedded Nvidia Platforms". In: *IEEE Access* (2021)

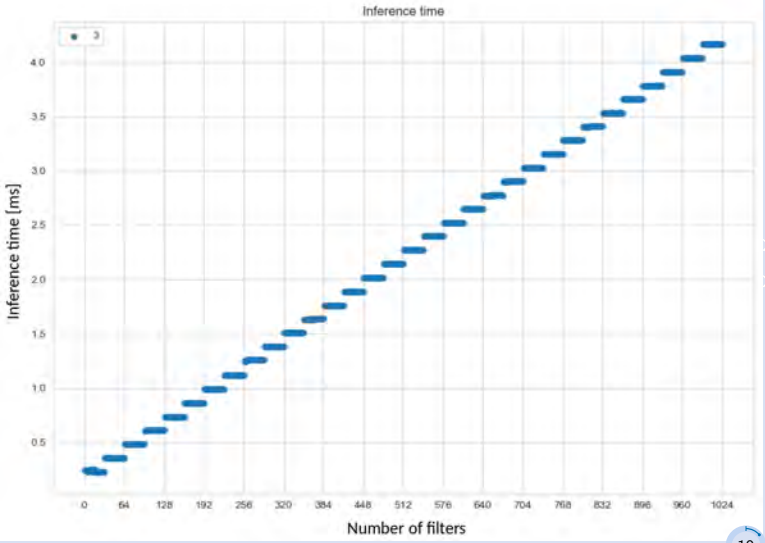
Inference Run Time Estimation

Assumption:

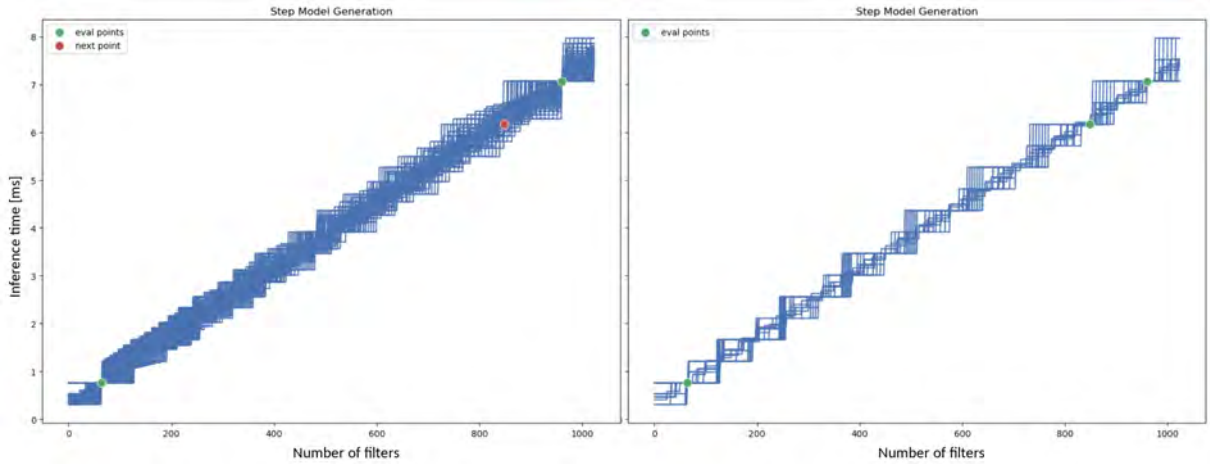
- Inference time as a function of problem size is a combination of step and linear functions due to limited parallel resources.
- Few measurement points are required

Example:

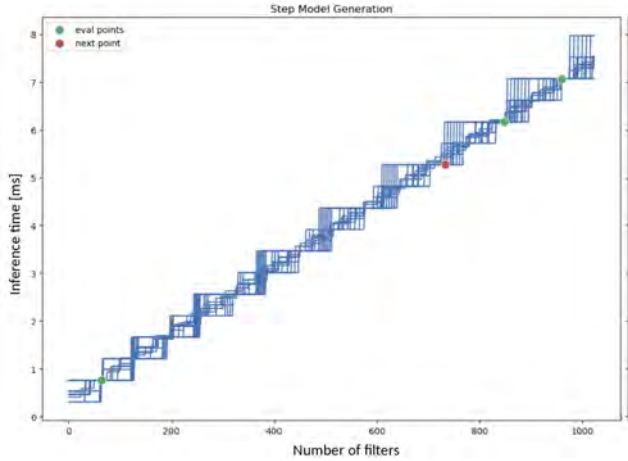
- Single convolutional layer sweep
- 32x32x64 with k filter and kernel size 3

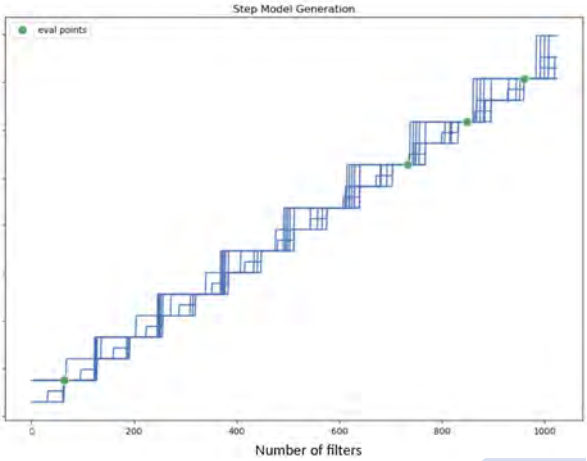


Iterative Refinement

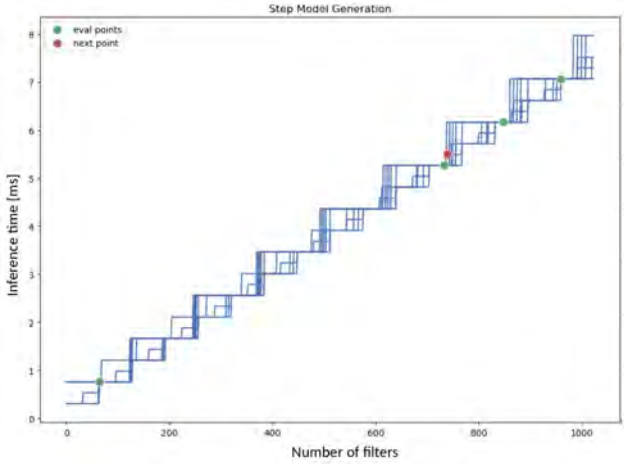


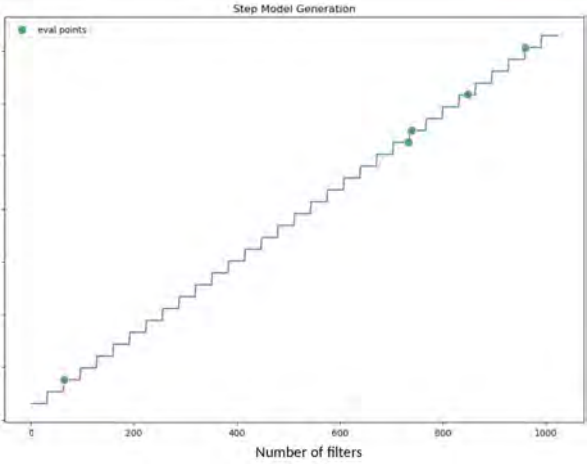
Iterative Refinement





Iterative Refinement



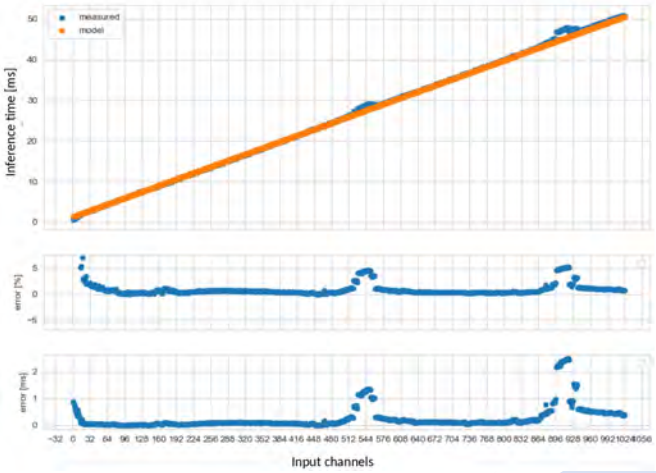


2D Example - Error

Slice through 2D plane at no of filters $k = 1024$

$$
\begin{aligned}
&f(d_{\text{in}}, k) \\
&\quad = 0.1418 + \lfloor \frac{d_{\text{in}} - 1}{8} \rfloor 0.0106 \\
&\quad + \lfloor \frac{k-1}{32} \rfloor \left(0.044 + \lfloor \frac{d_{\text{in}} - 1}{8} \rfloor 0.0121 \right)
\end{aligned}
$$

$$
\begin{aligned}
&f(d_{\text{in}}, 1024) \\
&\quad = 1.5058 + \lfloor \frac{d_{\text{in}} - 1}{8} \rfloor 0.3857
\end{aligned}
$$

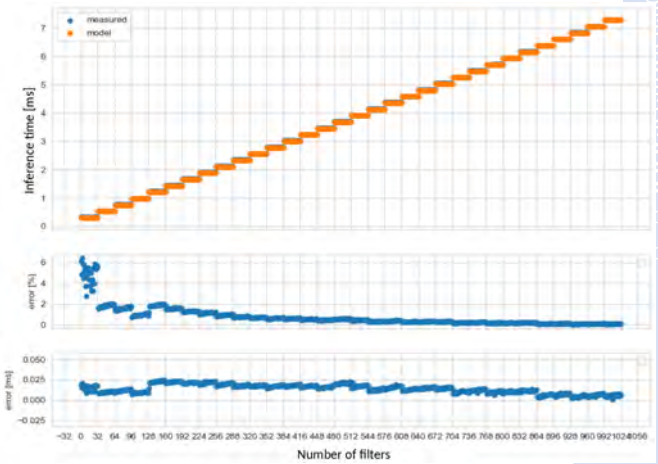


2D Example - Error

Slice through 2D plane at nr of channels $d_{\text{in}} = 128$

$$
\begin{aligned}
&f(d_{\text{in}}, k) \\
&\quad = 0.1418 + \lfloor \frac{d_{\text{in}} - 1}{8} \rfloor 0.0106 \\
&\quad + \lfloor \frac{k-1}{32} \Big(0.044 + \lfloor \frac{d_{\text{in}} - 1}{8} \rfloor 0.0121 \Big)
\end{aligned}
$$

$$
\begin{aligned}
&f(128, k) \\
&\quad = 0.3008 + \lfloor \frac{k-1}{32} \rfloor 0.2255
\end{aligned}
$$



Latency Estimation

Network	Estimation Error [%]			
	NCS2	ZCU102	Jetson Nano	Jetson TX2
YoloV3	4.1	3.2	-	-
MobileNetV2	4.3	4.2	3.6	4.2
ResNet50	8.2	1.2	2.4	2.8
FPN Net	9.3	7.5	-	-
AlexNet	5.2	4.8	5.5	6.6
VGG16	11.3	6.2	0.5	1.4

Latency Estimation - Summary

Summary

- Exploiting the discrete nature of HW resources

Latency Estimation - Summary

Summary

- Exploiting the discrete nature of HW resources
- Fast estimation function for latency based on linear and step functions

Latency Estimation - Summary

Summary

- Exploiting the discrete nature of HW resources
- Fast estimation function for latency based on linear and step functions
- Semi-automatic derivation of estimation for a new platform

Latency Estimation - Summary

Summary

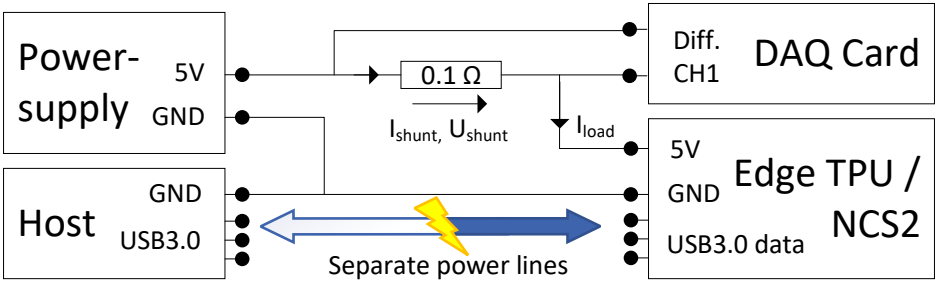
- Exploiting the discrete nature of HW resources
- Fast estimation function for latency based on linear and step functions
- Semi-automatic derivation of estimation for a new platform
- Results for several platforms are robust

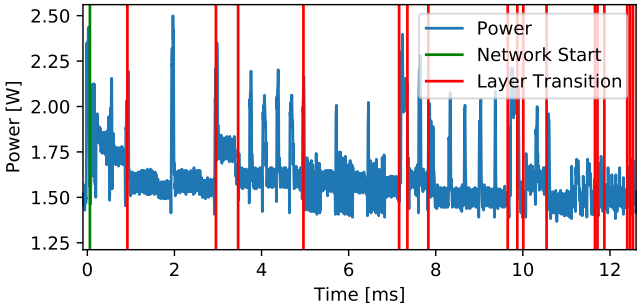
Outline

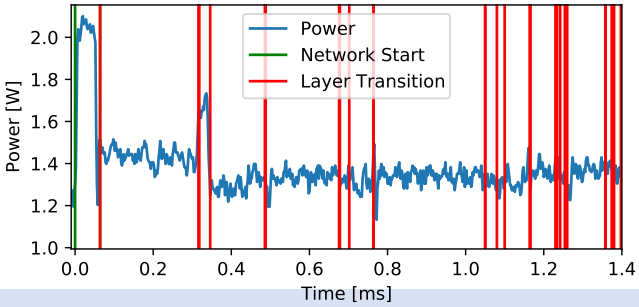
1. Estimation
2. Power Profiling
3. Traffic Light Controller Case Study
4. Shunt Connections
5. Ragweed Detection
6. DNN Partitioning for Inference

Power Profiling

Experimental Setup





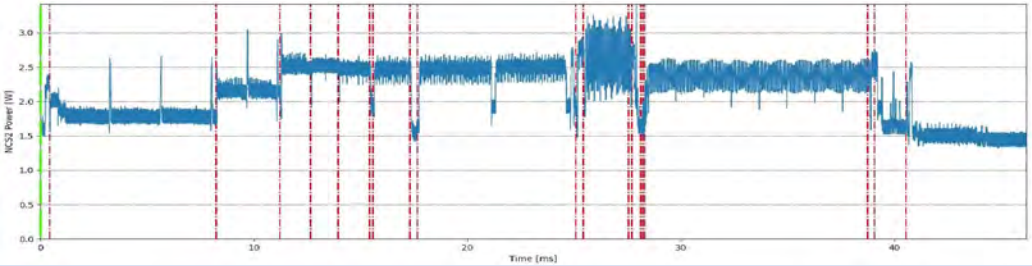


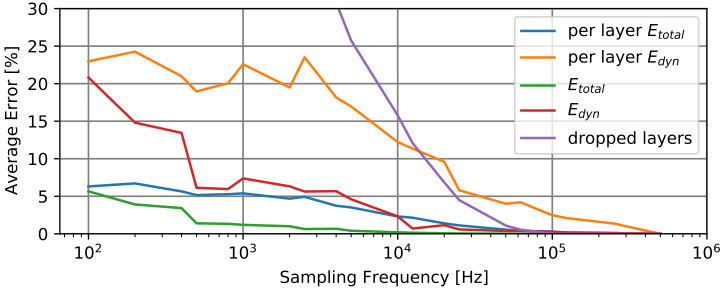
MobileNetV2 on NCS2 and Coral Edge TPU

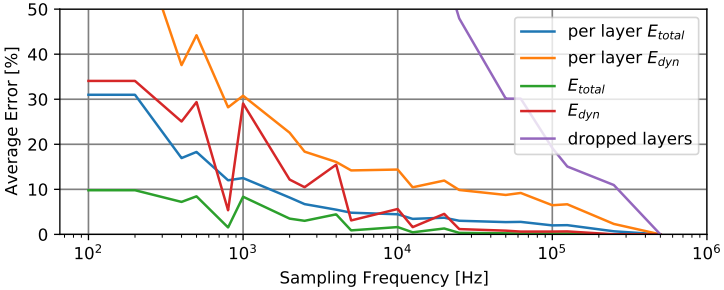
Power and Performance Profiling

Yolov3-tiny power profile on NCS2

00.000 ms

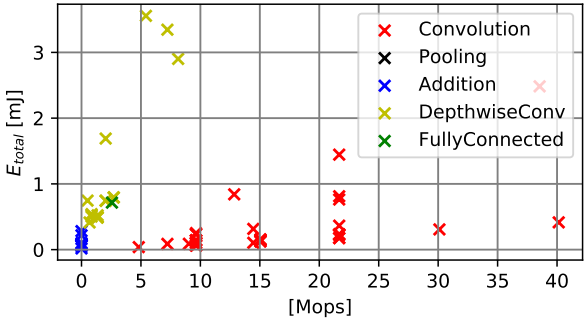


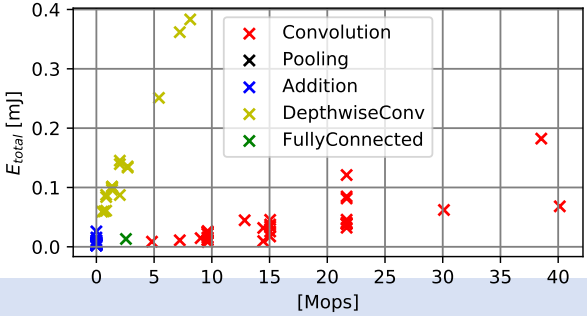




MobileNetV2 on NCS2 and Coral Edge TPU

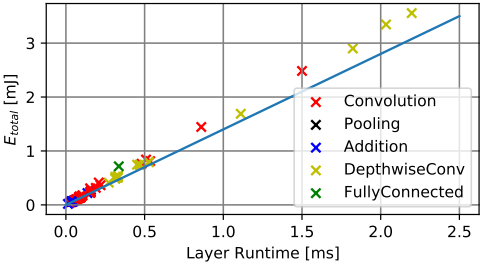
The error in % with respect to 500 kHz sampling frequency.

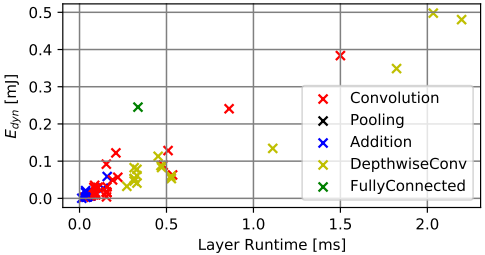


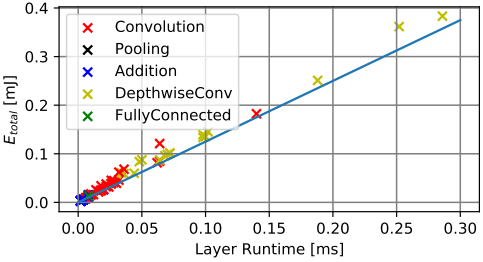


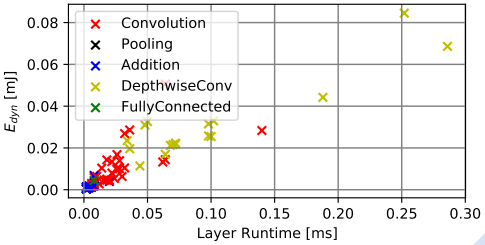
MobileNetV2 on NCS2 and Coral Edge TPU

Energy versus number of operations.









MobileNetV2 on NCS2 and Coral Edge TPU; Energy versus latency.

Power and Performance Profiling - Summary

Summary

- NCS2, Edge TPU and Nvidia platforms

Power and Performance Profiling - Summary

Summary

- NCS2, Edge TPU and Nvidia platforms
- Detailed, per layer latency and power profiling

Power and Performance Profiling - Summary

Summary

- NCS2, Edge TPU and Nvidia platforms
- Detailed, per layer latency and power profiling
- Hardware settings have significant influence

Power and Performance Profiling - Summary

Summary

- NCS2, Edge TPU and Nvidia platforms
- Detailed, per layer latency and power profiling
- Hardware settings have significant influence
- 100 kHz sampling frequency is required for 5 % accuracy

Power and Performance Profiling - Summary

Summary

- NCS2, Edge TPU and Nvidia platforms
- Detailed, per layer latency and power profiling
- Hardware settings have significant influence
- 100 kHz sampling frequency is required for 5 % accuracy
- **Number of operations is a poor predictor for latency and energy**

Power and Performance Profiling - Summary

Summary

- NCS2, Edge TPU and Nvidia platforms
- Detailed, per layer latency and power profiling
- Hardware settings have significant influence
- 100 kHz sampling frequency is required for 5 % accuracy
- **Number of operations is a poor predictor for latency and energy**
- **Latency and energy usage correlate fairly well in total**

Power and Performance Profiling - Summary

Summary

- NCS2, Edge TPU and Nvidia platforms
- Detailed, per layer latency and power profiling
- Hardware settings have significant influence
- 100 kHz sampling frequency is required for 5 % accuracy
- **Number of operations is a poor predictor for latency and energy**
- **Latency and energy usage correlate fairly well in total**
- Per-layer power analysis gives more insight in HW inefficiencies

Outline

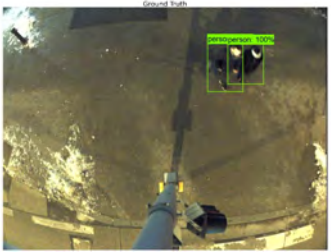
1. Estimation
2. Power Profiling
3. Traffic Light Controller Case Study
4. Shunt Connections
5. Ragweed Detection
6. DNN Partitioning for Inference

Traffic Light Controller Case Study

Traffic Light Controller

Data set:

- training: 19087 images
- positive examples 47%
- validation: 13184
- positive examples 26%
- Resolution: 1280x720
- Issue: Validation 4h/network
  $\rightarrow$ validation set: 1319



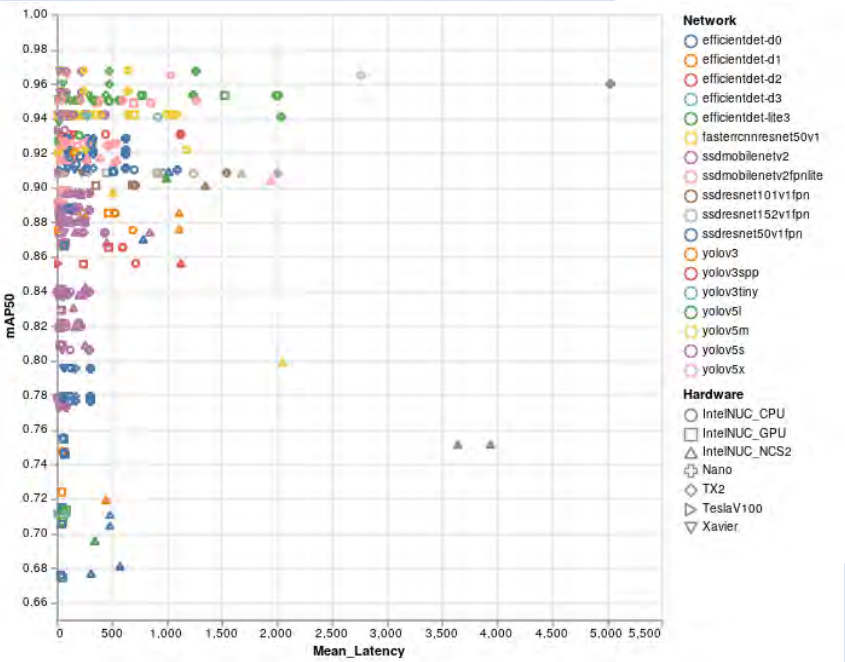
Platforms under Study

Name	Performance [T op/s]	Memory [GB]	Power [W]	Cost [€]
NVIDIA Xavier AGX	32	16	10–30	800
NVIDIA Jetson TX2	1.3	4	7.5–15	260
NVIDIA Jetson Nano	0.5	4	5–10	120
Intel NCS2	1	0.5	5	80
Intel NUC CPU (i7-8650U)	22.4	32	15	600
Intel NUC GPU (Intel UHD 620)	0.8	32	15	600
Tesla V100	130	32	250	>1000

Networks under Study

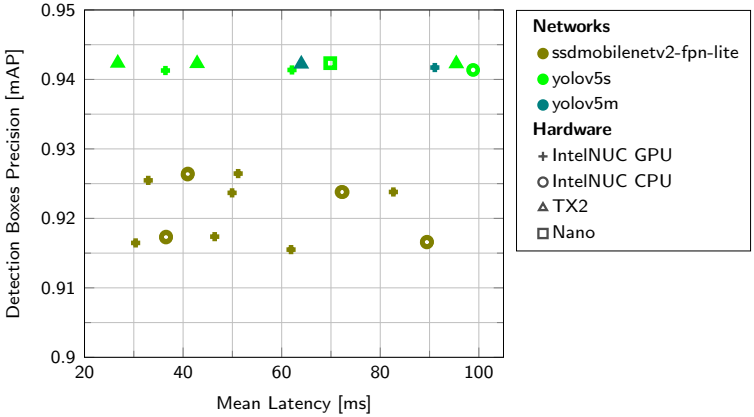
Name	Framework used	No of parameters (10^6)
ssdmobilenetv2fpnlite	Tensorflow	2.8
efficientdet-d0	Tensorflow	3.9
ssdmobilenetv2	Tensorflow	4.5
yolov5s	Pytorch	7.0
yolov3tiny	Pytorch	8.6
yolov5m	Pytorch	21.0
yolov5l	Pytorch	46.6
ssdresnet50v1fpn	Tensorflow	50.7
yolov3	Pytorch	61.4
yolov3spp	Pytorch	62.5
ssdresnet101v1fpn	Tensorflow	69.7
ssdresnet152v1fpn	Tensorflow	85.3
yolov5x	Pytorch	87.1

All solutions



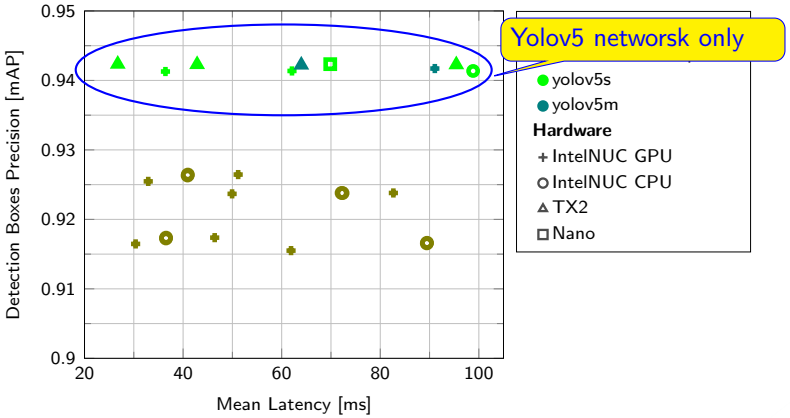
Solutions under cost constraints

latency
≤ 100 ms
and
mAP50
≥ 0.9.



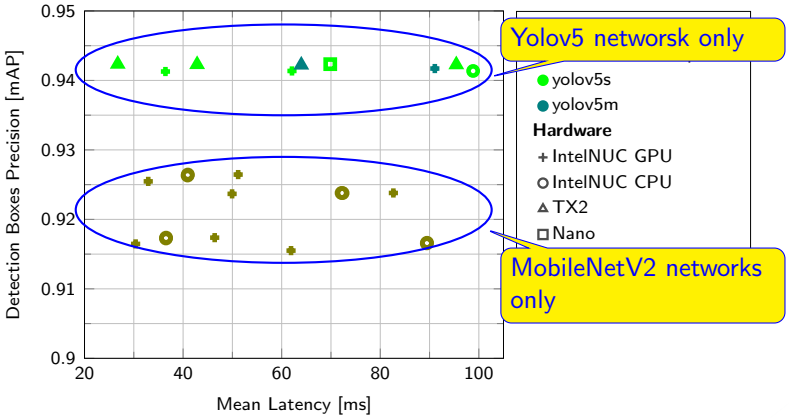
Solutions under cost constraints

latency
≤ 100 ms
and
mAP50
≥ 0.9.

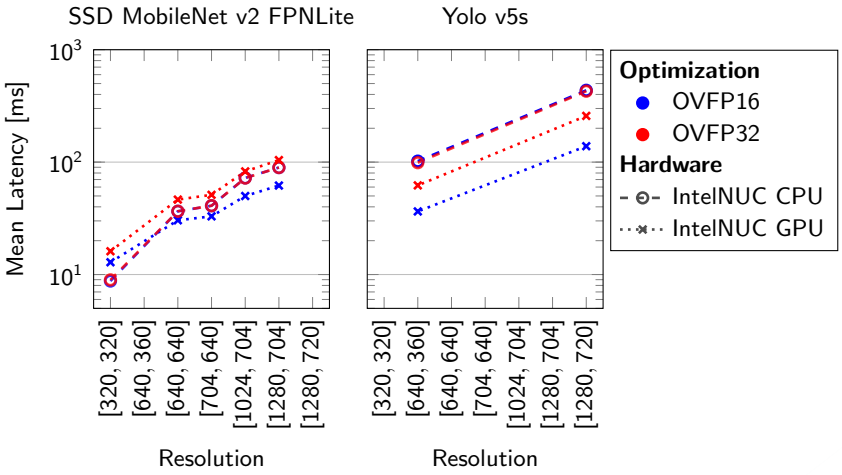


Solutions under cost constraints

latency
≤ 100 ms
and
mAP50
≥ 0.9.

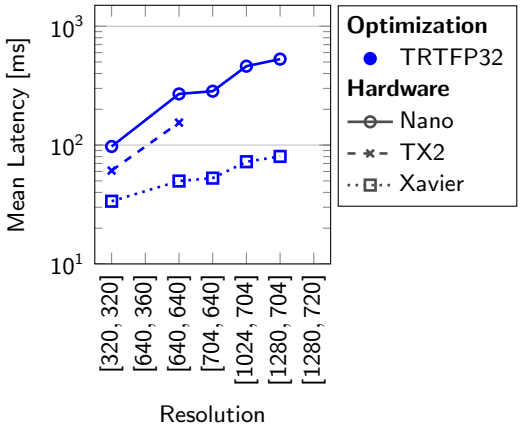


Impact of resolution and quantization on the Intel NUC platform

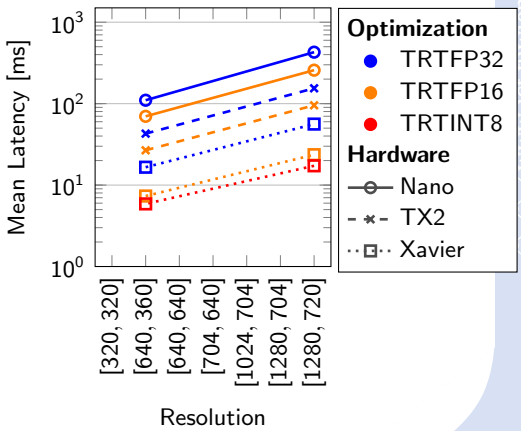


Impact of resolution and quantization on the Nvidia platform

SSD MobileNet v2 FPNLite



Yolo v5 small



Traffic Light Case Study - Summary

Summary

- **Yolo v5s** is the most suitable network;

Traffic Light Case Study - Summary

Summary

- **Yolo v5s** is the most suitable network;
- **Nvidia Jetson Nano and TX2** are most suitable platforms

Traffic Light Case Study - Summary

Summary

- **Yolo v5s** is the most suitable network;
- **Nvidia Jetson Nano and TX2** are most suitable platforms
- Yolo v5m and MobileNetV2 are reasonable networks;

Traffic Light Case Study - Summary

Summary

- **Yolo v5s** is the most suitable network;
- **Nvidia Jetson Nano and TX2** are most suitable platforms
- Yolo v5m and MobileNetV2 are reasonable networks;
- IntelNUC GPU, IntelNUC CPU are reasonable platforms.

Traffic Light Case Study - Summary

Summary

- **Yolo v5s** is the most suitable network;
- **Nvidia Jetson Nano and TX2** are most suitable platforms
- Yolo v5m and MobileNetV2 are reasonable networks;
- IntelNUC GPU, IntelNUC CPU are reasonable platforms.
- Latency depends linear on image resolution

Traffic Light Case Study - Summary

Summary

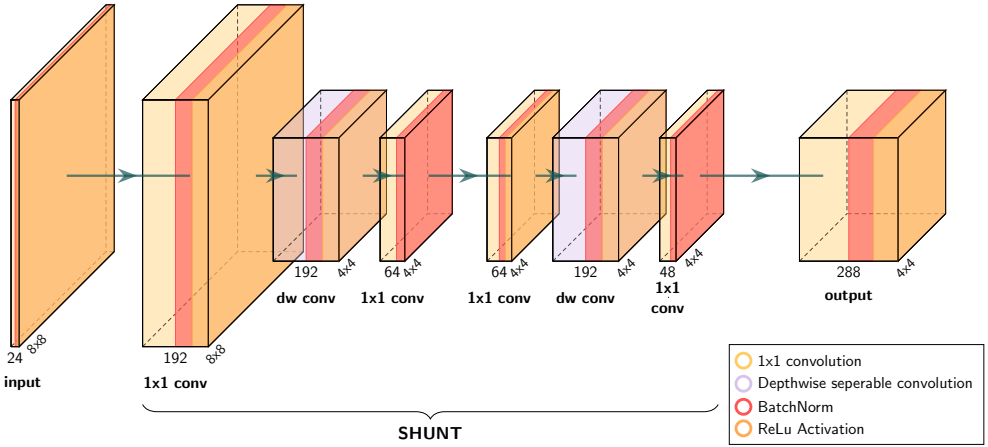
- **Yolo v5s** is the most suitable network;
- **Nvidia Jetson Nano and TX2** are most suitable platforms
- Yolo v5m and MobileNetV2 are reasonable networks;
- IntelNUC GPU, IntelNUC CPU are reasonable platforms.
- Latency depends linear on image resolution
- FP16 quantization is a sweet spot compared to FP32 and INT8

Outline

1. Estimation
2. Power Profiling
3. Traffic Light Controller Case Study
4. Shunt Connections
5. Ragweed Detection
6. DNN Partitioning for Inference

Shunt Connections

Shunt Architecture



Brijraj Singh, Durga Toshniwal, and Sharan Kumar Allur. "Shunt connection: An intelligent skipping of contiguous blocks for optimizing MobileNet-V2". In: *Neural Networks* 118 (2019), pages 192–203

Bernhard Haas, Alexander Wendt, Axel Jantsch, and Matthias Wess. "Neural Network Compression Through Shunt Connections and Knowledge Distillation for Semantic Segmentation Problems". In: *17th International Conference on Artificial Intelligence Applications and Innovations (AIAI)*. June 2021

Shunt Architecture on Classification MobilNet V3 Small

Block ID	MAdds	KQ CIFAR10	KQ CIFAR100
0	918k	-	-
1	584k	-	-
2	588k	0.12	0.25
3	465k	-	-
4	683k	0.13	0.54
5	683k	0.06	0.16
6	339k	0.02	0.09
7	339k	0.01	0.10
8	400k	-	-
9	599k	0.03	0.24
10	599k	0.03	0.15
Head	567k	-	-
	$\sum$ 6.8M		
flops reduction		$\sim$ 42%	$\sim$ 31%

Shunt Architecture on Classification

	Acc. CIFAR10 [%]	Acc. CIFAR100 [%]
Original model	91.93	67.10
Shunt-inserted models	79.78	53.56
Fine-tuned models with shunt		
Standard	88.09	64.63
Partial model freezing	85.66 (-2.43)	60.04 (-4.59)
Knowledge distillation (T=5, λ=2)	**91.36 (+3.27)**	**67.54 (+2.91)**

Shunt Architecture on Segmentation

- MobilNet V3 Small Segmentation architecture
- Cityscapes data set

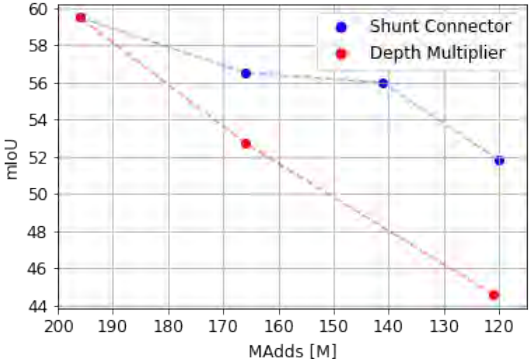
Block ID	MAdds	KQ
0	227 M	-
1	53 M	-
2	230 M	-
3	166 M	-
4	128 M	0.13
5	211 M	0.15
6	211 M	-
7	114 M	0.20
8	146 M	-
9	160 M	0.20
10	75 M	0.19
Head	247 M	-
	$\sum$ 2.0 B	

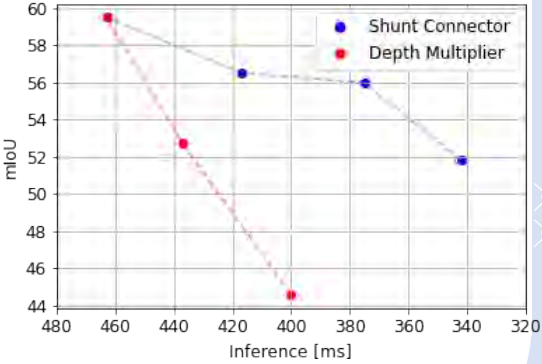
Shunt Architecture on Segmentation

Reference mIoU: 59.6

	7-10-ARCH4	5-10-ARCH1	4-10-ARCH1
MAdds	-15%	-28%	-39%
NCS2	111 ms (-12.6%)	98 ms (-22.8%)	92 ms (-27.6%)
Shunt-inserted models - MIOU:			
	34.84	37.97	35.13
FINE-TUNED MODELS WITH STANDARD SHUNT - MIOU:			
Standard	**56.52**	54.91	**51.83**
Knowledge distillation	56.03 (-0.49)	**55.98 (+1.07)**	51.14 (-0.69)

Shunt Architecture on Segmentation





Depth Multiplier: built in parameter of MobileNet for controlling the number of channels.

Shunt Connection - Summary

Summary

- Shunt connection is effective to decrease model size

Shunt Connection - Summary

Summary

- Shunt connection is effective to decrease model size
- Knowledge Quotient for identifying promising skip connections

Shunt Connection - Summary

Summary

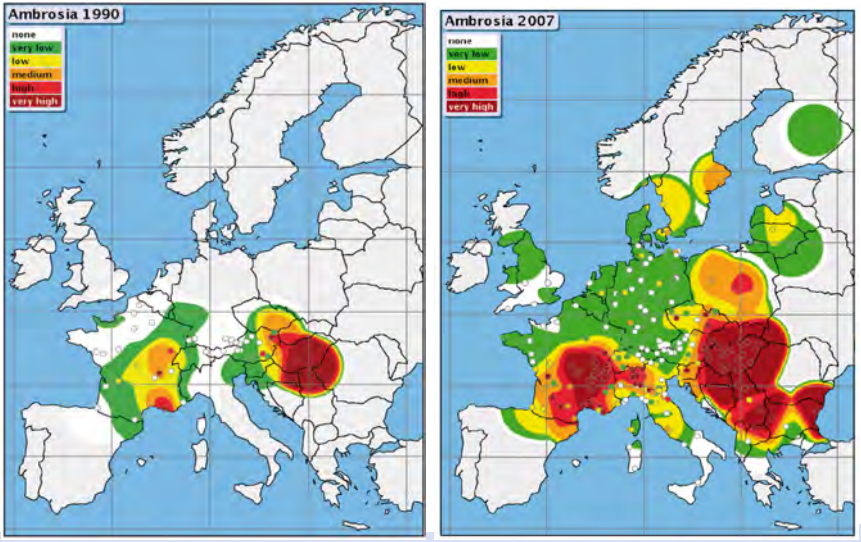
- Shunt connection is effective to decrease model size
- Knowledge Quotient for identifying promising skip connections
- Knowledge distillation for training the replacing shunt block

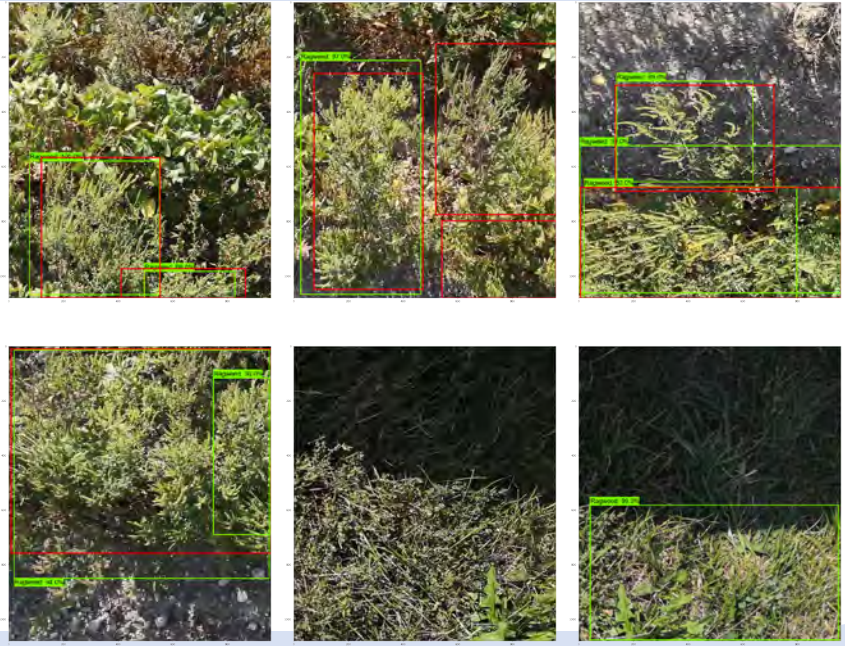
Outline

1. Estimation
2. Power Profiling
3. Traffic Light Controller Case Study
4. Shunt Connections
5. Ragweed Detection
6. DNN Partitioning for Inference

Ragweed Detection

Ragweed Invasion





Scenarios

- S1: to detect small ragweed plants that are $10\,\mathrm{cm}^2$ in size.
  $\Rightarrow$ Ground Sampling Distance GSD $= 0.1\,\mathrm{cm/pixel}$
- S2: to detect large ragweed $1\,\mathrm{m}^2$ in size.
  $\Rightarrow$ GSD $= 1\,\mathrm{cm/pixel}$

Scenario	S1			S2		
Drone	DJI Phantom	DJI Matrice	Wingtra One	DJI Phantom	DJI Matrice	Wingtra One
Efficiency Metrics						
Groundwidth [m/img]	5.47	8.19	5.46	54.72	81.92	54.56
Groundlength [m/img]	3.65	5.46	3.63	36.48	54.60	36.32
Ground area [km²/flight]	0.2	0.62	0.31	1.97	6.22	3.09
Flights per km²	5.08	1.61	3.24	0.51	0.16	0.32
Time [h/km²]	**2.54**	**1.48**	**3.19**	**0.26**	**0.15**	**0.31**
Costs [EUR/km²]	**152.3**	**176.9**	**145.6**	**15.2**	**17.7**	**14.6**
Optimal Flight Parameters						
Altitude [m]	3.65	11.41	4.7	36.48	114.09	47.03
Speed [m/s]	20	23	16	20	23	16
Images per flight	9,869	13,902	15,595	987	1,391	1,560
Framerate [images/s]	5.48	4.21	4.41	0.55	0.42	0.44

Segmentation for Ragweed Identification

- Network: Deeplab V3+, based on MobileNet V3, with several layers modified
- Pretrained on Cityscapes dataset
- Trained with drone based recorded image set: 130 minutes of video data, height: 2.5 - 4m
  971 images, 960x1080 pixel
- Platform: Nvidia Jetson TX2
- Optimizations: Shunt connections
- Tensor RT Optimization

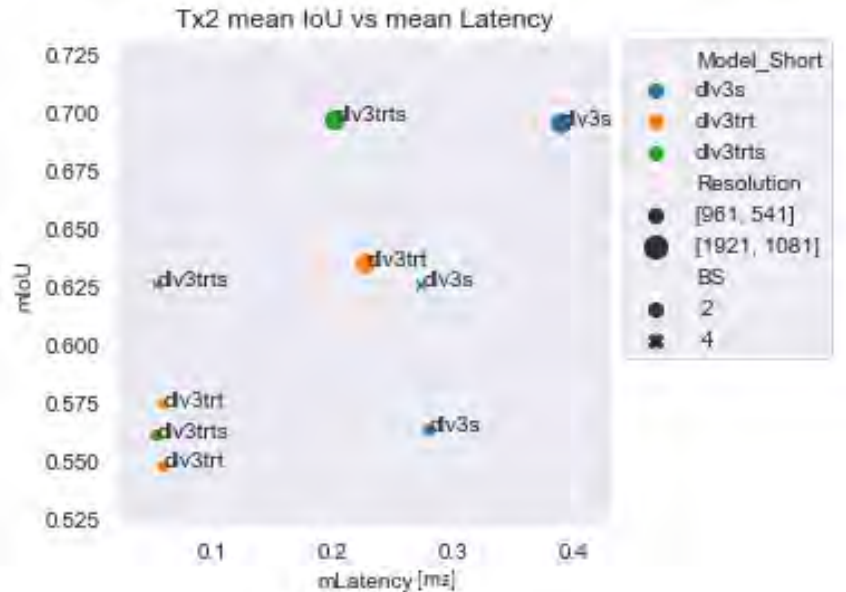
Segmentation accuracy

	mIoU			
Resolution	dlv3	dlv3s	dlv3trt	dlv3trts
961×541	0.588	0.625	0.574	0.625
1921×1081	0.670	0.695	0.634	0.695

dlv3	DeepLab V3 based reference model
dlv3s	dlv3 with shunt connection
dlv3trt	dlv3 optimized with Tensor RT framework
dlv3trts	dlv3 optimized with Tensor RT framework and with shunt connection


0
250
500
750
1000
1250
1500
1750
2000
0
500
1000
1500
2000
2500
3000
3500

Latency vs. Accuracy



Ragweed Detection - Summary

Summary

- Drone based detection is cost efficient
  5-50 x more cost efficient, 8-100x more time efficient than manual methods

Ragweed Detection - Summary

Summary

- Drone based detection is cost efficient
  5-50 x more cost efficient, 8-100x more time efficient than manual methods
- Segmentation is superior over object detection

Ragweed Detection - Summary

Summary

- Drone based detection is cost efficient
  5-50 x more cost efficient, 8-100x more time efficient than manual methods
- Segmentation is superior over object detection
- DeepLabV3+ is a suitable base network

Ragweed Detection - Summary

Summary

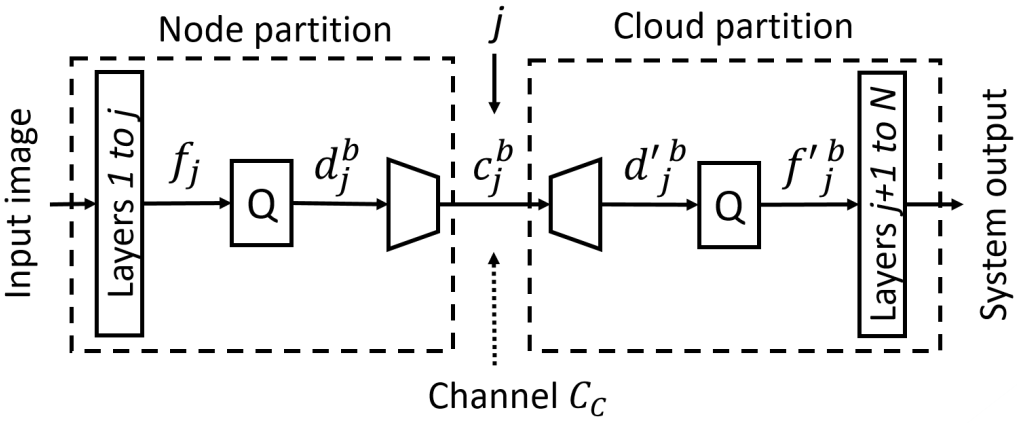
- Drone based detection is cost efficient
  5-50 x more cost efficient, 8-100x more time efficient than manual methods
- Segmentation is superior over object detection
- DeepLabV3+ is a suitable base network
- Nvidia TX2 is a reasonable platform

Outline

1. Estimation
2. Power Profiling
3. Traffic Light Controller Case Study
4. Shunt Connections
5. Ragweed Detection
6. DNN Partitioning for Inference

DNN Partitioning for Inference

Partitioned DNNs



Partitioning Methodology

Methodology

- Energy and delay model
- Quantization and compression at partitioning point
- Selecting the partitioning point

Energy Model

$$E_{\text{Node}} = \sum_{i=1}^{j} (E_P(d_{i-1}, t_i, P_P)) + E_C(c_j, C_C)$$

E_{Node}	Energy spent on the IoT node
j	Partitioning point
i	DNN layers
t_i	Computation in layer i
d_{i-i}	Amount of input data for layer i
c_j	Amount of data at partitioning point j
E_P	Processing energy
E_C	Communication energy
P_P	IoT Node platform

Silvia Krug and Mattias O'Nils. "Modeling and comparison of delay and energy cost of iot data transfers". In: *IEEE Access* 7 (2019), pages 58654–58675

Irida Shallari, Isaac Sánchez Leal, Silvia Krug, Axel Jantsch, and Mattias O'Nils. "Design space exploration on IoT node: Trade-offs in processing and communication". In: *IEEE Access* (2021)

Quantization

$$d = \left\lfloor \frac{f - S}{M - S} \times 2^{b-1} \right\rfloor$$

$$S = \left(\frac{M + m}{2} \right)$$

d	Quantized data
$b \in [1 \cdots 8]$	Resolution in bits
f	data to be quantized
M	Maximum of the value range of f
m	Minimum of the value range of f
μ	Mean of all f

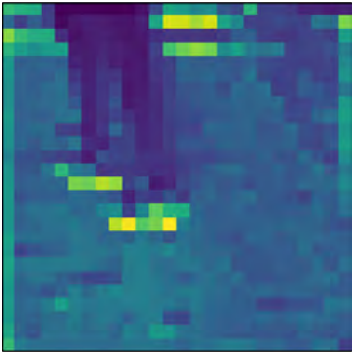
After quantization, the data is compressed with the zip algorithms.

Selection of Partitioning Point

- Candidate points have low data volume
- For each candidate point
  - Quantization and compression
  - Retraining
- Best point according to constraints and optimization criteria is selected.

Wheel Chair Steering Case Study

Input image

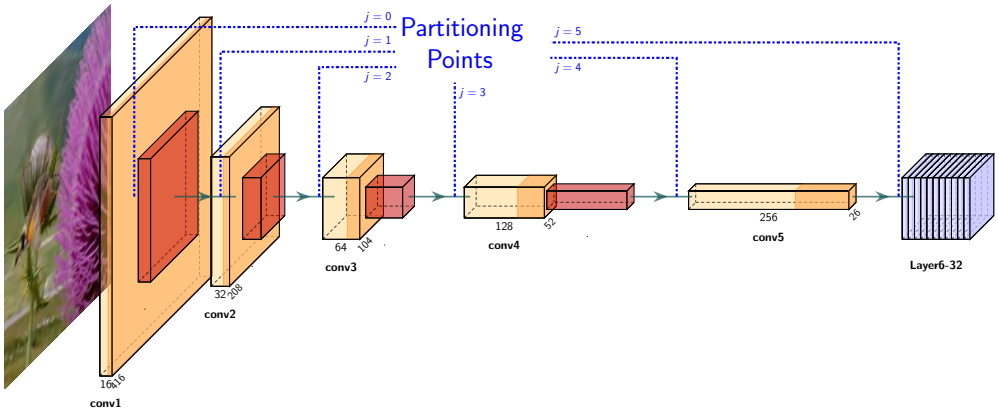


32-bits

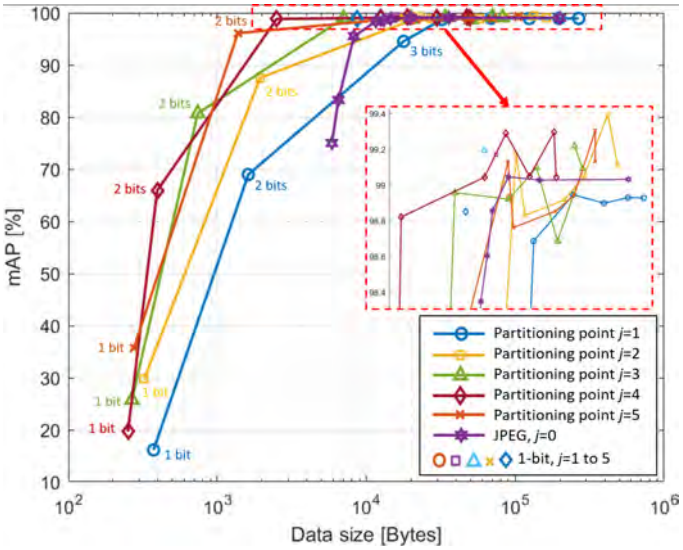
1-bit

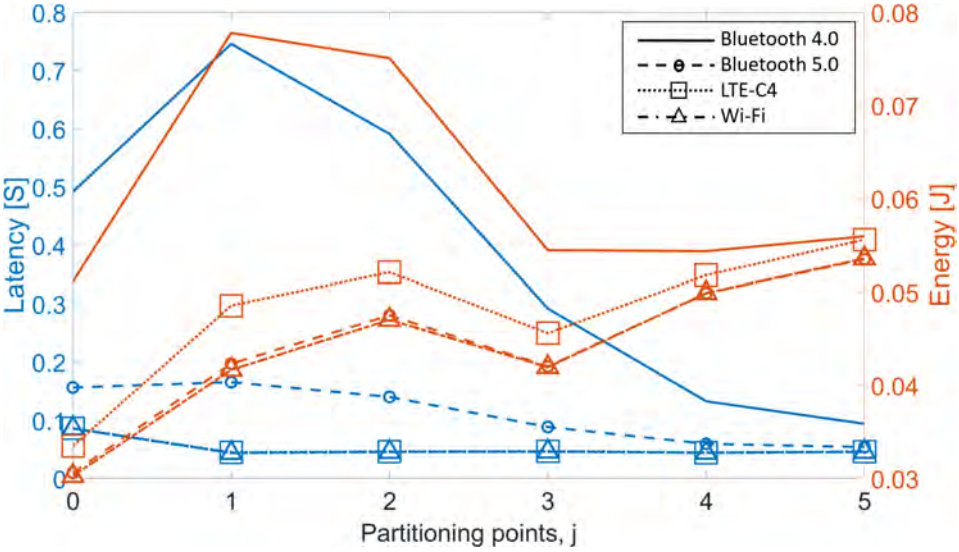
Cristian Vilar Giménez, Silvia Krug, Faisal Z. Qureshi, and Mattias O'Nils. "Evaluation of 2D-/3D-Feet-Detection Methods for Semi-Autonomous Powered Wheelchair Navigation". In: *Journal of Imaging* 7.12 (2021)

TinyYolo V3



Results





DNN Partitioning - Summary

Summary

- Effective DNN partitioning is feasible

DNN Partitioning - Summary

Summary

- Effective DNN partitioning is feasible
- DNN partitioning opens a considerable design space for DNN based IoT applications

DNN Partitioning - Summary

Summary

- Effective DNN partitioning is feasible
- DNN partitioning opens a considerable design space for DNN based IoT applications
- Next steps is to explore more aggressive DNN adaptions for partitioning

Results, publications, demos, code on

eml.ict.tuwien.ac.at

¿ Questions ?

References I

[1] Bernhard Haas, Alexander Wendt, Axel Jantsch, and Matthias Wess. "Neural Network Compression Through Shunt Connections and Knowledge Distillation for Semantic Segmentation Problems". In: *17th International Conference on Artificial Intelligence Applications and Innovations (AIAI)*. June 2021.

[2] Irida Shallari, Isaac Sánchez Leal, Silvia Krug, Axel Jantsch, and Mattias O'Nils. "Design space exploration on IoT node: Trade-offs in processing and communication". In: *IEEE Access* (2021).

[3] M. Wess, M. Ivanov, C. Unger, A. Nookala, A. Wendt, and A. Jantsch. "ANNETTE: Accurate Neural Network Execution Time Estimation With Stacked Models". In: *IEEE Access* 9 (2021), pages 3545–3556.

[4] Martin Lechner and Axel Jantsch. "Blackthorn: Latency Estimation Framework for CNNs on Embedded Nvidia Platforms". In: *IEEE Access* (2021).

[5] Cristian Vilar Giménez, Silvia Krug, Faisal Z. Qureshi, and Mattias O'Nils. "Evaluation of 2D-/3D-Feet-Detection Methods for Semi-Autonomous Powered Wheelchair Navigation". In: *Journal of Imaging* 7.12 (2021).

[6] Silvia Krug and Mattias O'Nils. "Modeling and comparison of delay and energy cost of iot data transfers". In: *IEEE Access* 7 (2019), pages 58654–58675.

[7] Brijraj Singh, Durga Toshniwal, and Sharan Kumar Allur. "Shunt connection: An intelligent skipping of contiguous blocks for optimizing MobileNet-V2". In: *Neural Networks* 118 (2019), pages 192–203.