Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

Embedded Deep Learning for Image-Based Grasp Verification

R&D Colloquium

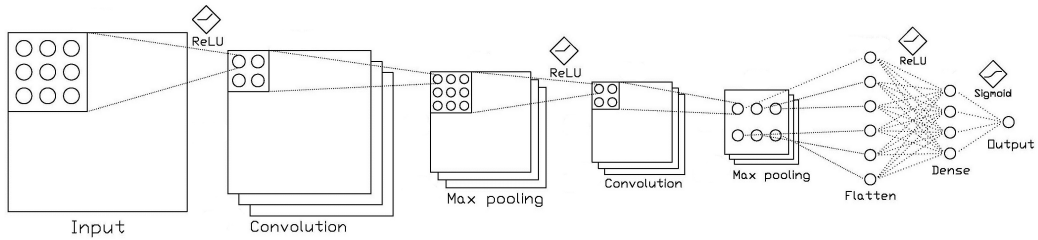
24.03.2020

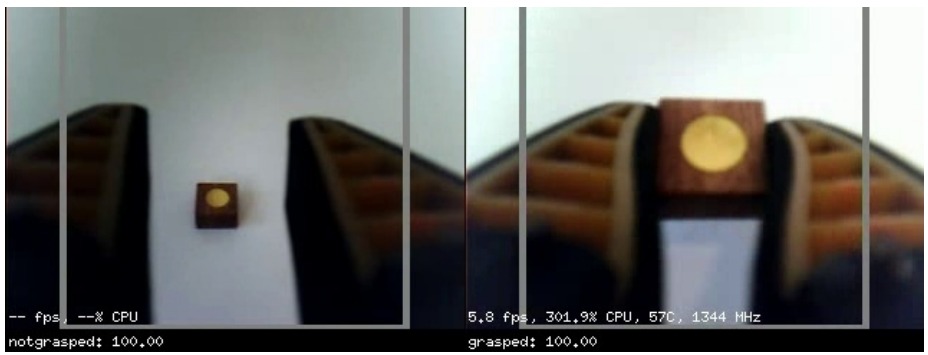
Amirhossein Pakdaman

Contents

Introduction

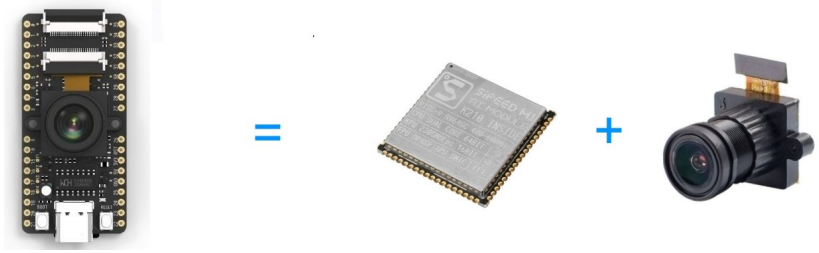
This project studies the implementation of **deep learning algorithms** on **smart cameras**, for the purpose of **grasp verification**.





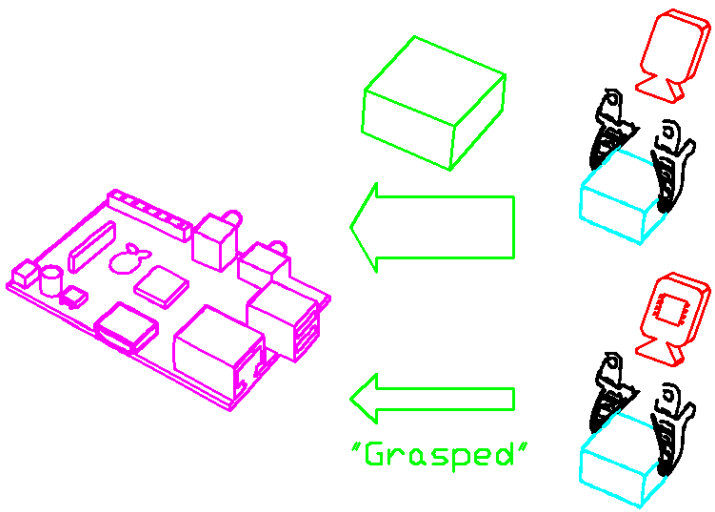
Introduction: Smart Cameras

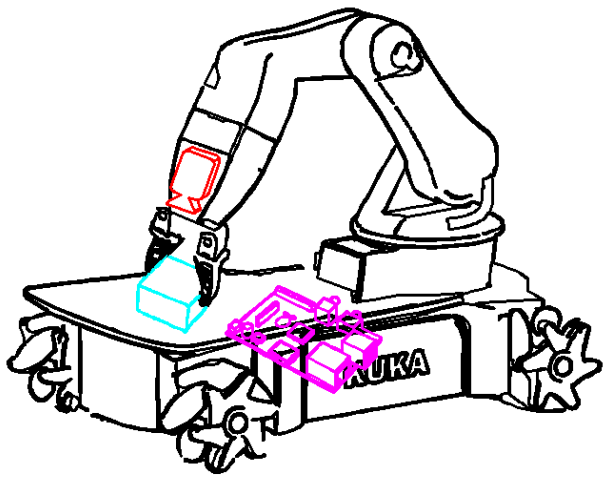
Implementation of **deep learning algorithms** on **smart cameras**, for the purpose of **grasp verification**.



Embedded Device = Processor + Camera Sensor

Edge Computing:





Introduction: Deep Learning Algorithms

Implementation of **deep learning algorithms** on **smart cameras**, for the purpose of **grasp verification**.

Convolutional Neural Networks:

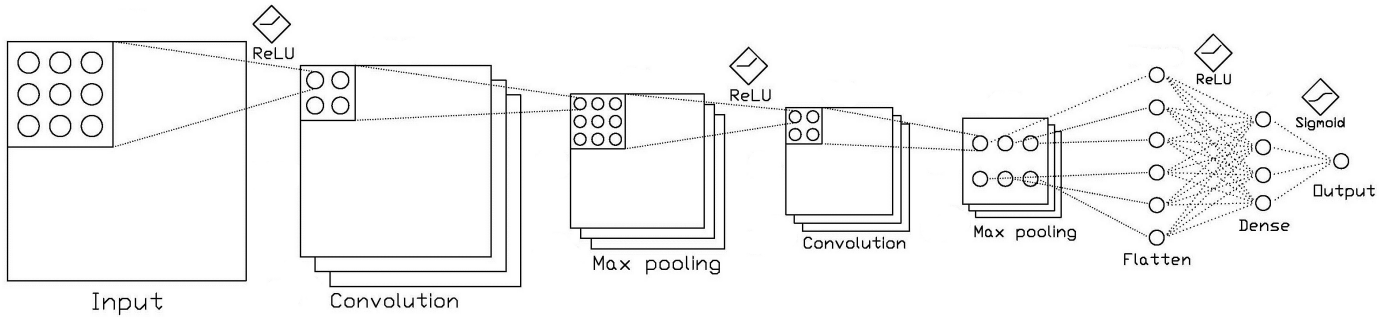
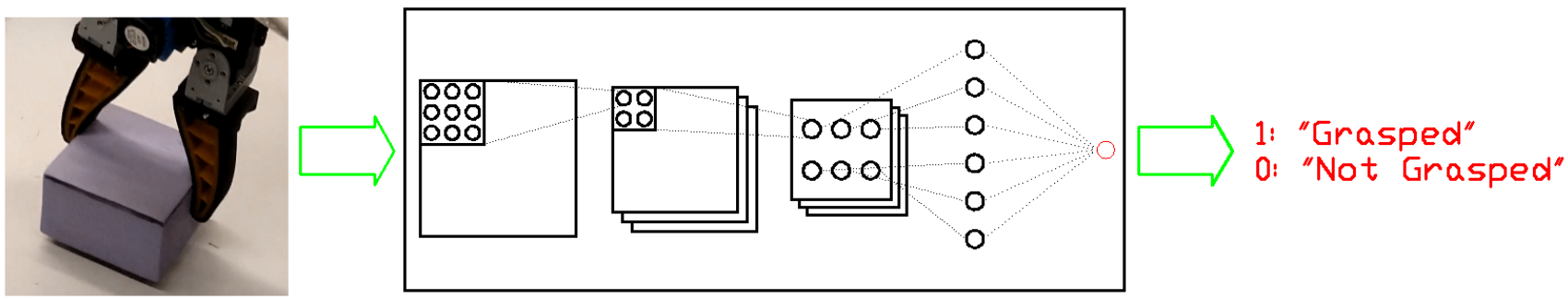
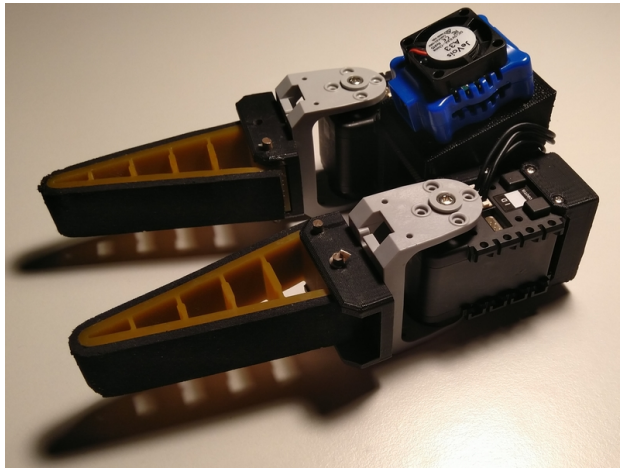


Image Classification:

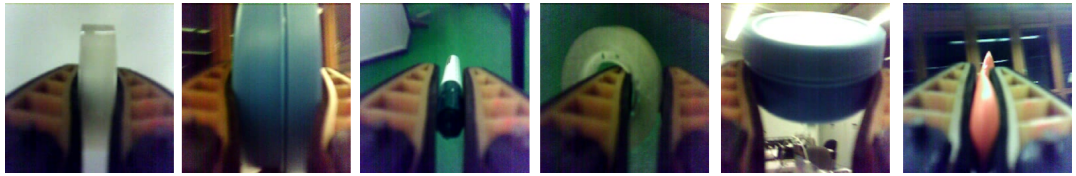


Introduction: Grasp Verification

Implementation of **deep learning algorithms** on **smart cameras**, for the purpose of **grasp verification**.



Grasped:



Not Grasped:

Contents

Smart Cameras

OpenMV H7[1]

Sipeed Maix Bit[2]

JeVois A33[3]

- Hardware Performance
- Software Capabilities

Smart Cameras: Hardware Performance

	JeVois	Sipeed	OpenMV
Main Processor	ARM A7 1.3 GHz	Kendryte K210 400 MHz	ARM M7 400 MHz
RAM	256 MB	8 MB	1 MB
Additional Processor	GPU[1]	KPU[2]	FPU[3]
Power	3.5 W	2.8 W	0.7 W

[1] Graphics Processing Unit
[2] Kendryte Processor Unit
[3] Floating-point Processing Unit

Smart Cameras: Software Capabilities



	JeVois	Sipeed	OpenMV
Programming language	Python, C++	MaixPy	MicroPython
Running DL algorithms	√	√	√
Running custom architectures	√	√	×
Unlimited model size	√	×	×
Supported library	keras	keras	Caffe
Model file format	.tflite	.kmodel	.network
Model converter	(keras)	nncase	CMSIS-NN

Smart Cameras: Software Capabilities



	JeVois	Sipeed	OpenMV
Programming language	Python, C++	MaixPy	MicroPython
Running DL algorithms	√	√	√
Running custom architectures	√	√	×
Unlimited model size	√	×	×
Supported library	keras	keras	Caffe
Model file format	.tflite	.kmodel	.network
Model converter	(keras)	nncase	CMSIS-NN

Benchmarking experiments:

- MobileNets
- YOLO
- Custom CNN models

Contents

Benchmarking Experiment

Reason: to explore and compare

- Capabilities
- Limitations

Smart cameras:

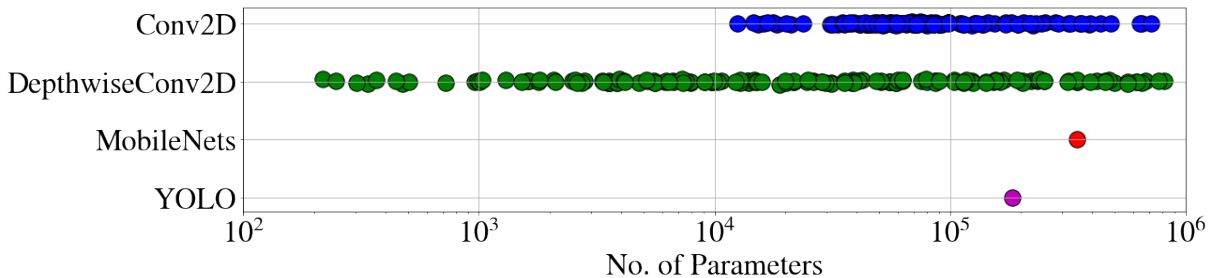
JeVois

Sipeed

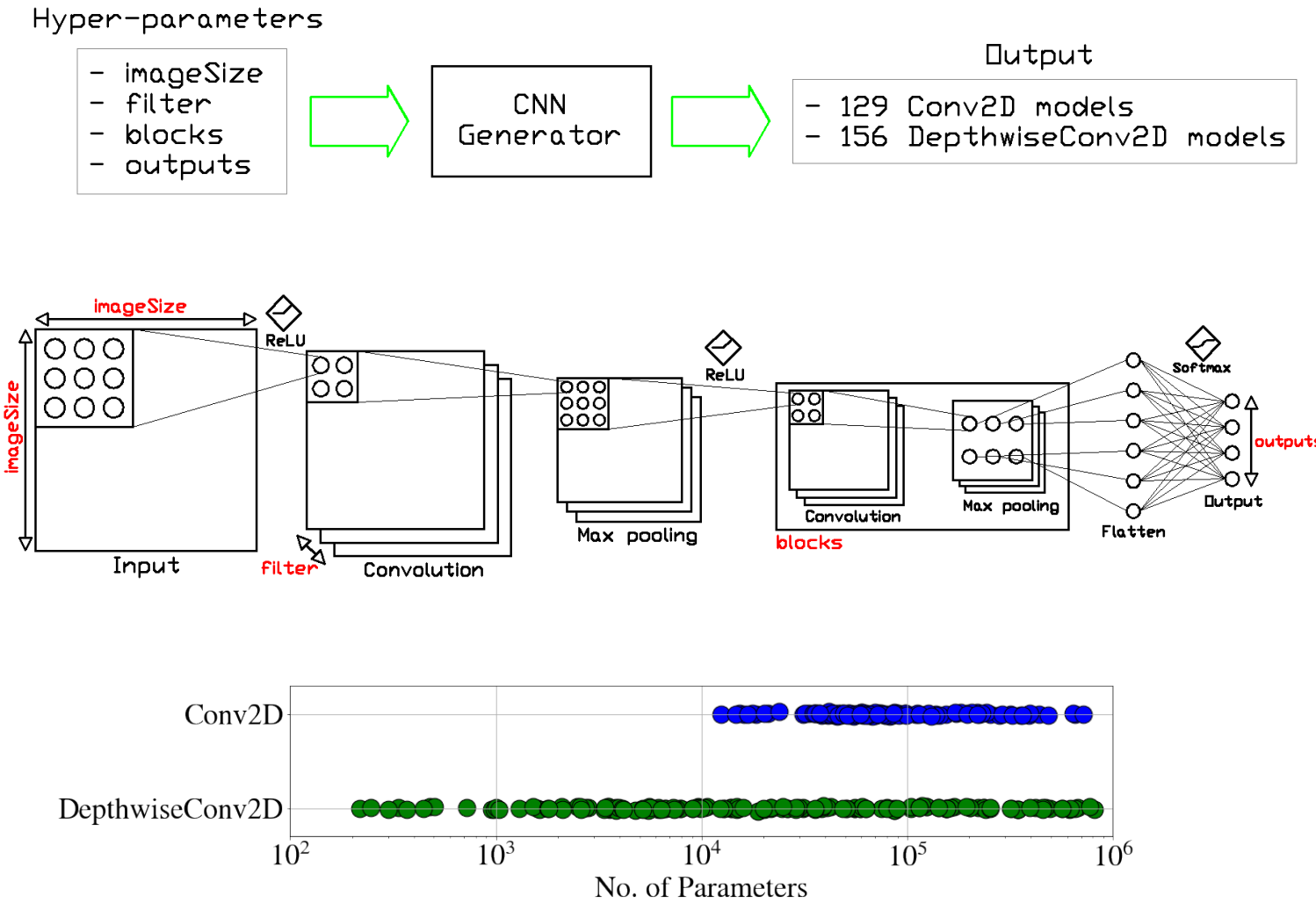
Measured values:

- Runtime (Latency)
- No. of frames per second (Throughput)[4]

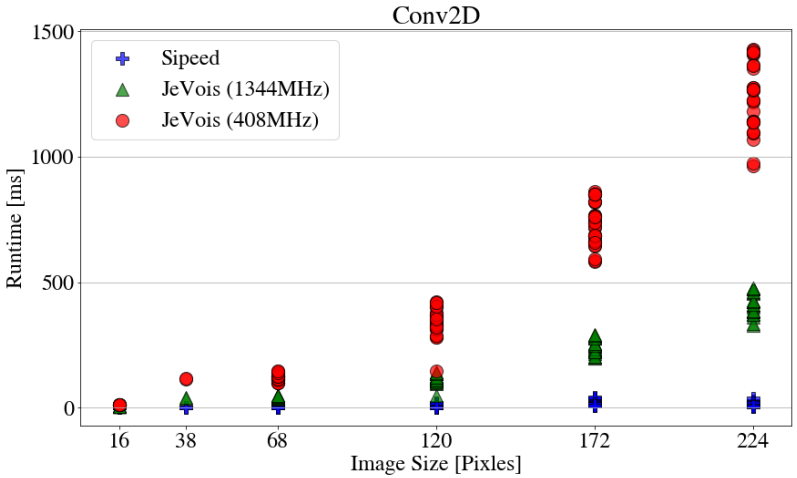
CNN models:

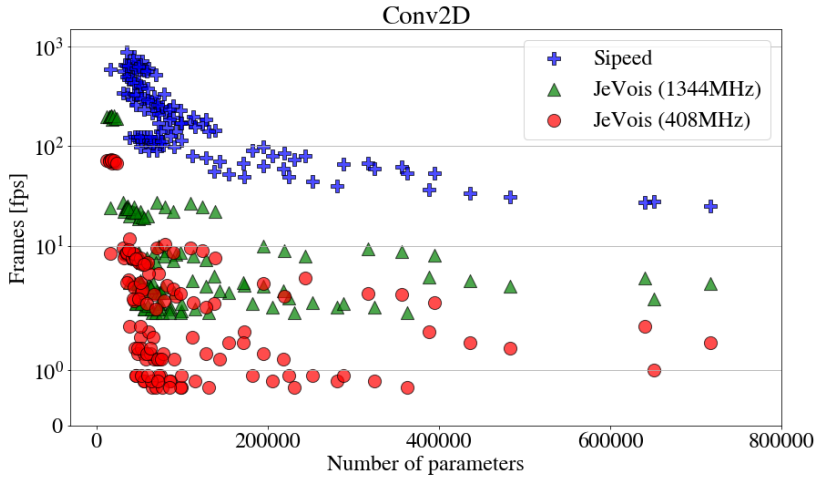


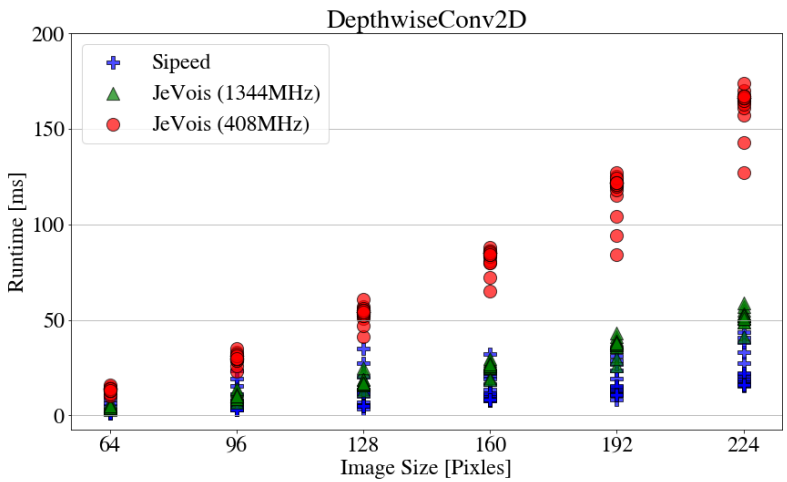
Benchmarking Experiment: Custom CNN Models

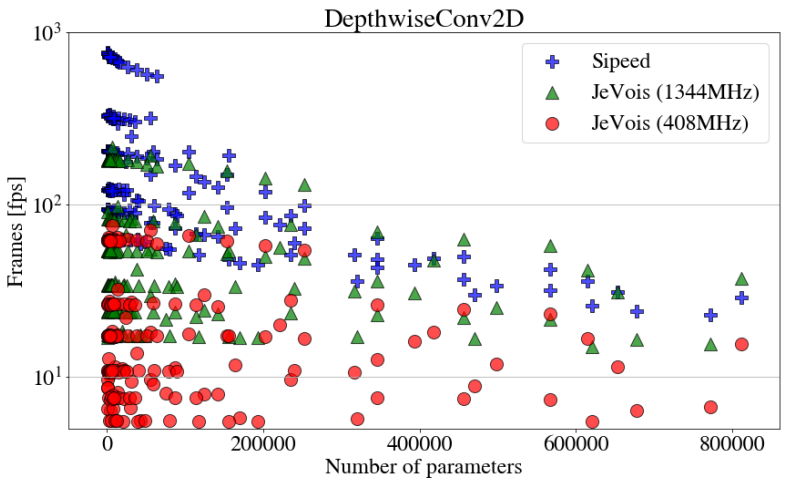


Benchmarking Experiment: Custom CNN Models

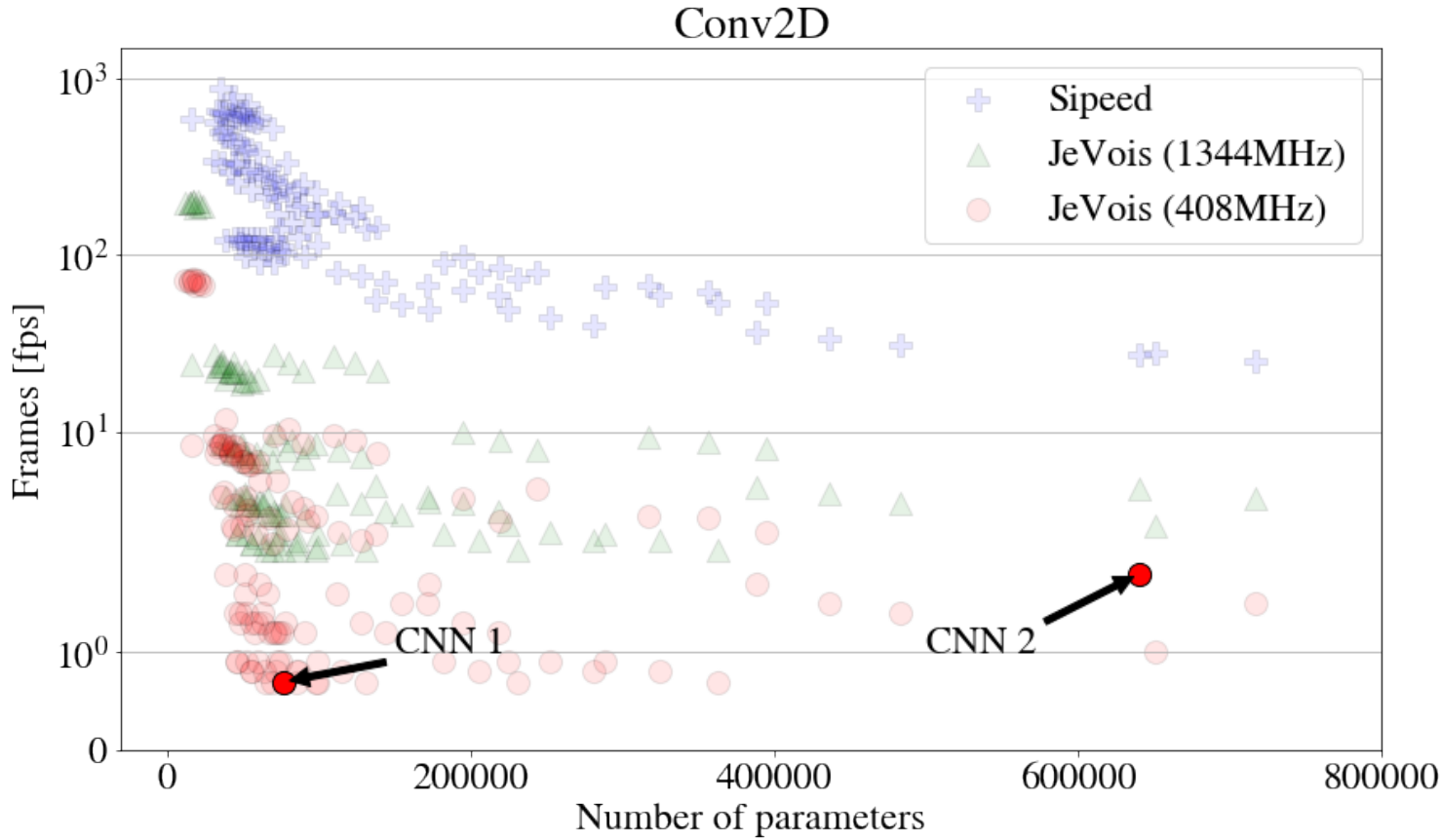






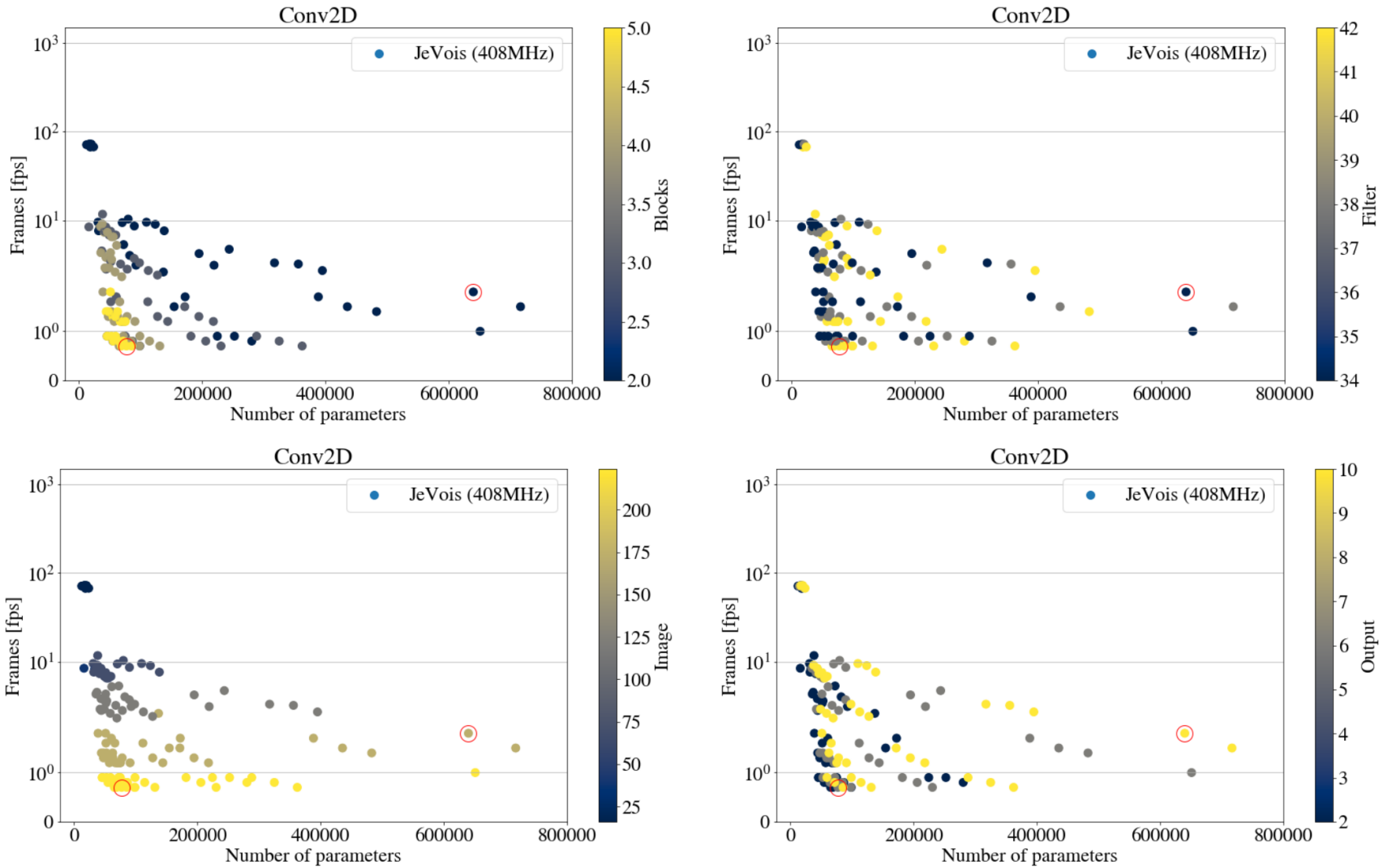


Benchmarking Experiment: Custom CNN Models



Attributes	CNN 1	CNN 2
No. of Parameters	77,202	640,060
Frames [fps]	0.7	1.8
No. of Blocks	5	2
No. of Filters	42	34
Image Size	224	172
No. of Outputs	6	10

Benchmarking Experiment: Custom CNN Models



Benchmarking Experiment: Custom CNN Models

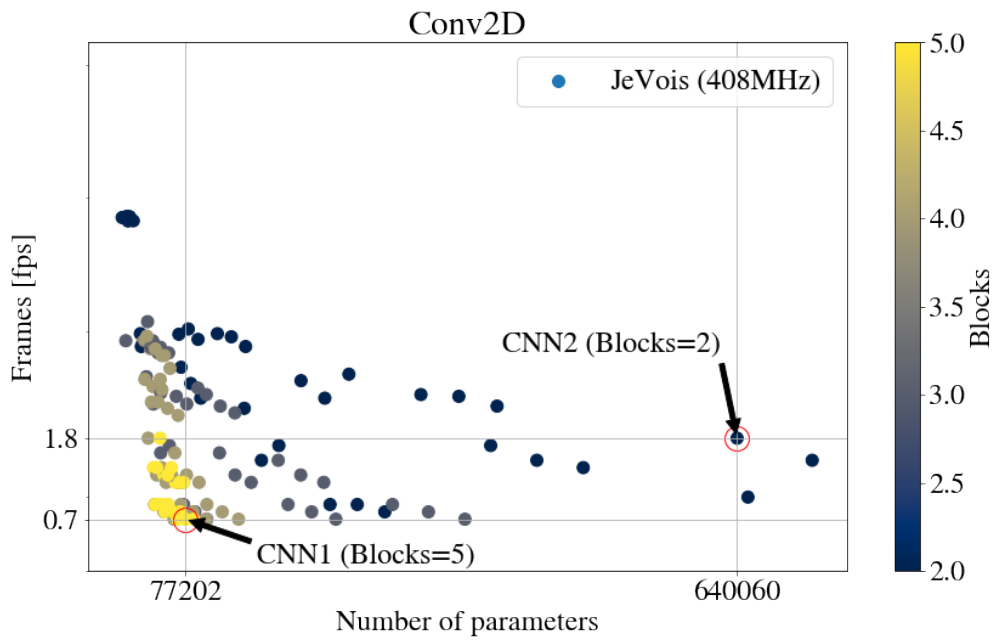
In general:

- (No. of parameters ↑) ⇒ (Frames ↓)
- Bigger models ⇒ Slower inference

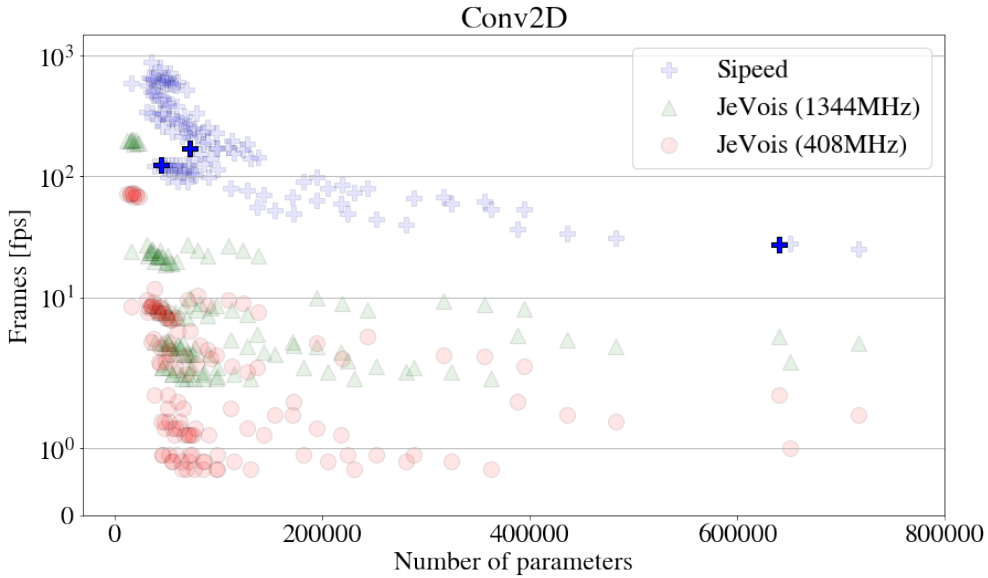
New insight:

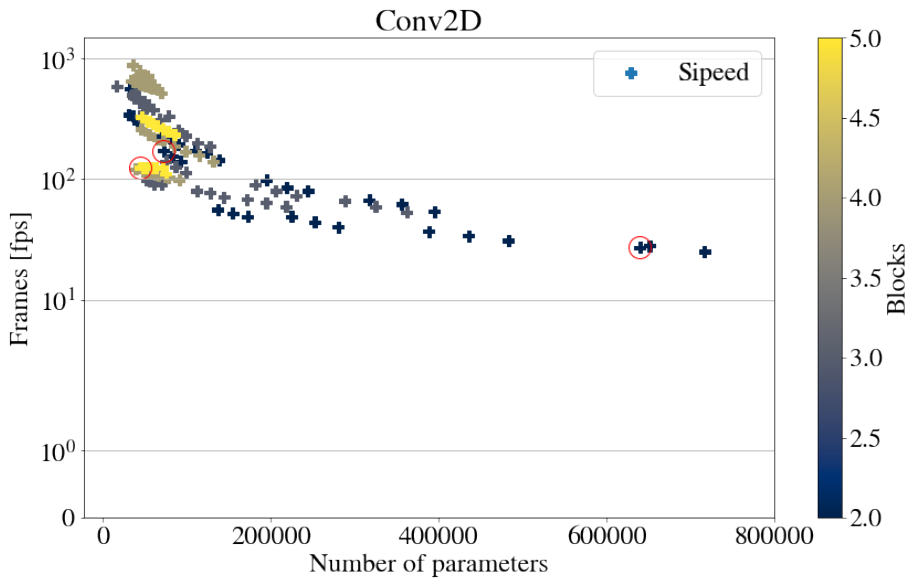
- (No. of parameters ↓) , (No. of blocks ↑) ⇒ (Frames ↓)
- Smaller models with more blocks ⇒ Slower inference

Attributes	CNN 1	CNN 2
No. of Parameters	**77,202**	**640,060**
Frames [fps]	**0.7**	**1.8**
No. of Blocks	**5**	**2**
No. of Filters	42	34
Image Size	224	172
No. of Outputs	6	10



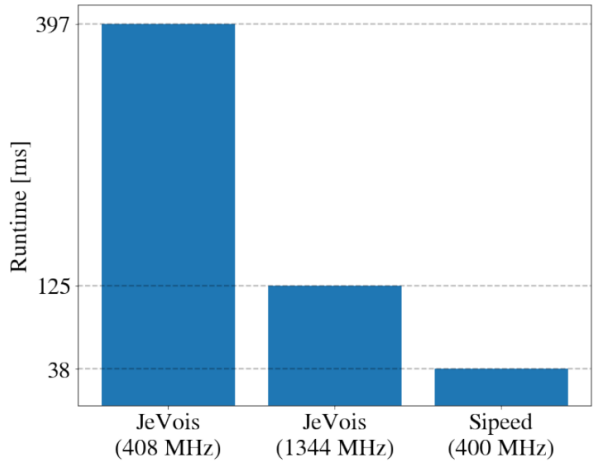
Benchmarking Experiment: Custom CNN Models

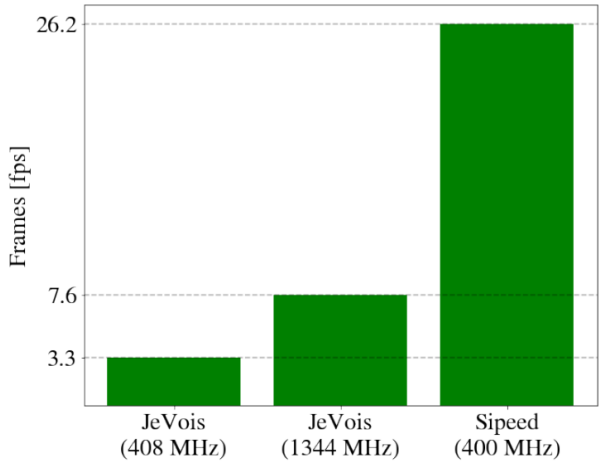




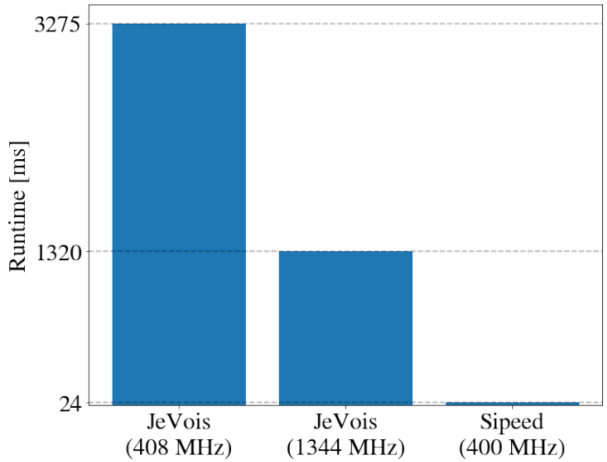
Benchmarking Experiment: MobileNets & YOLO

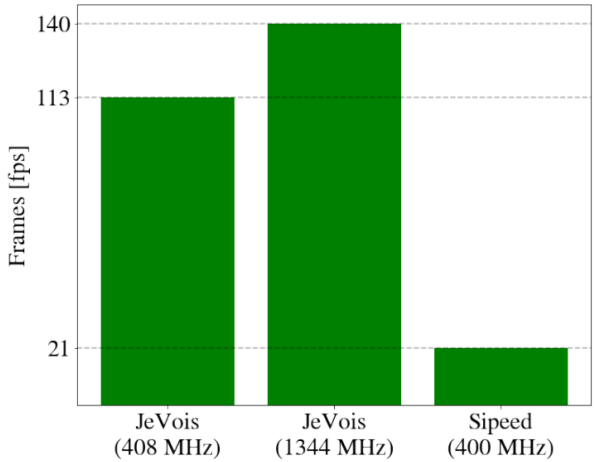
MobileNets:





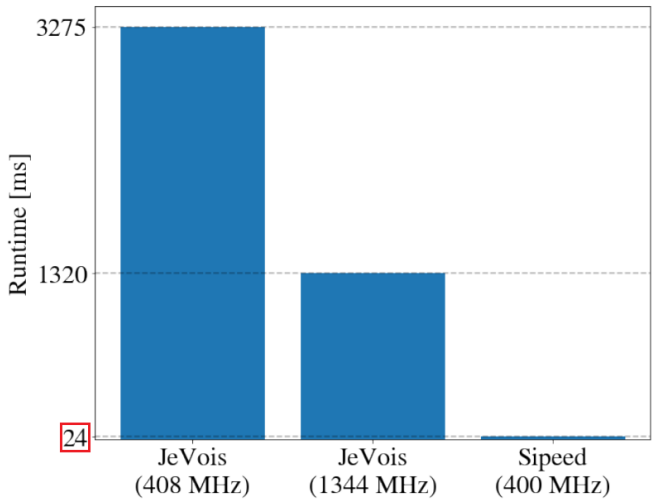
YOLO:

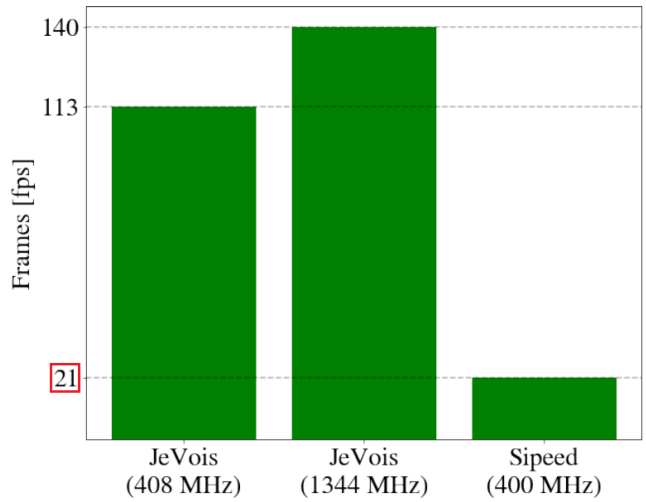


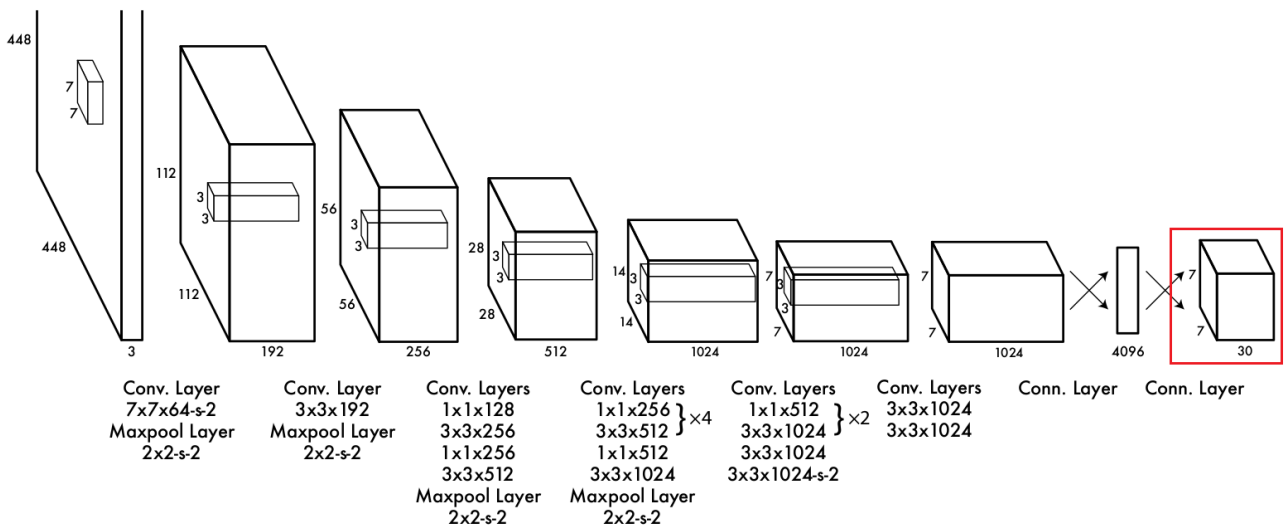


Benchmarking Experiment: MobileNets & YOLO

YOLO:





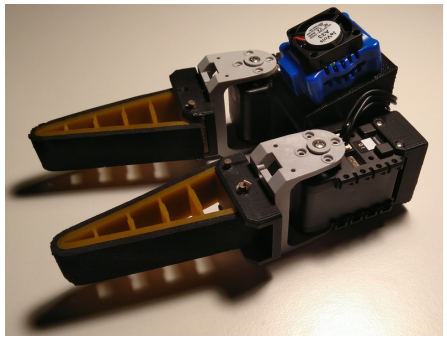


[5]

Contents

Grasp Verification

- Mounting:

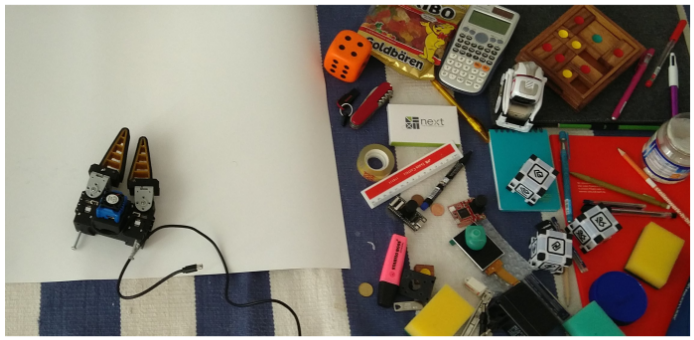


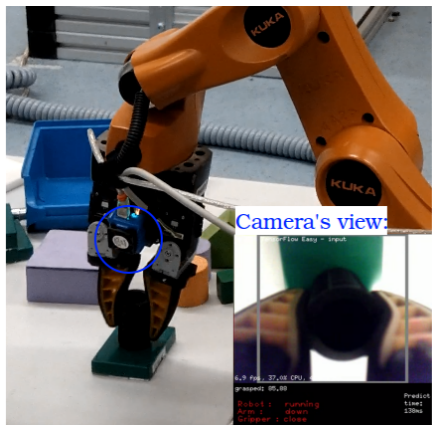
- Dataset:

Grasped:

Not Grasped:

- CNN Architectures:
  - MobileNets
  - ResNet
  - Conv2D
  - DepthwiseConv2D
- Evaluation in reality.
- Experiments:





Grasp Verification

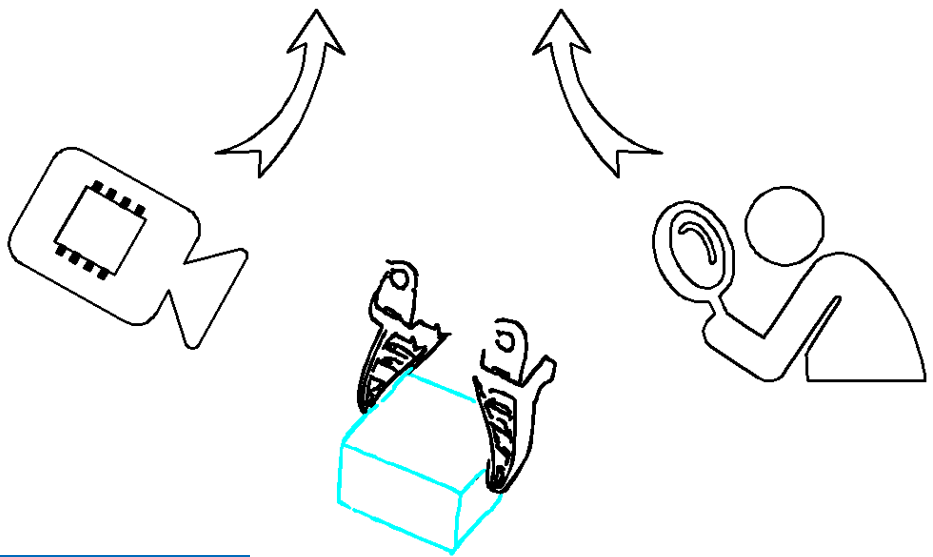
- CNN Architectures:

Accuracy	MobileNets	ResNet	Conv2D	DepthwiseConv2D
Train	0.9889	0.9470	0.9872	0.8389
Validate	0.9750	0.9750	1.0000	0.9500
Real world	?	?	?	?

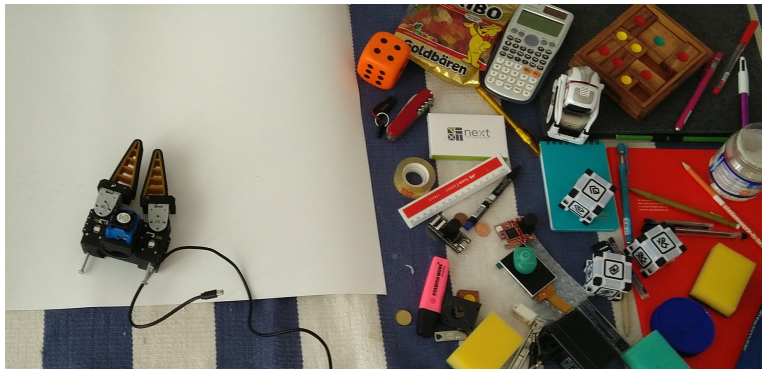
- Evaluation in reality:

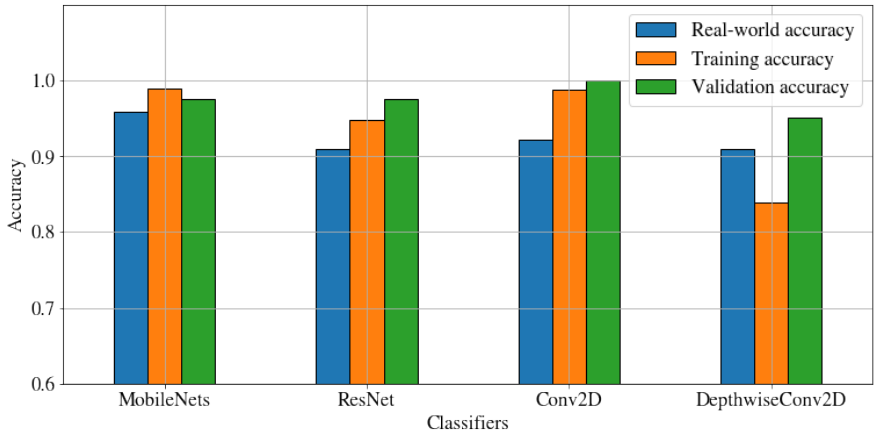
Frame	Camera	Observer	Eval.
1	Grasped	Grasped	✓
2	Grasped	Not grasped	×
3	Not grasped	Not grasped	✓
4	Grasped	Grasped	✓

$\Rightarrow$ Real world accuracy=75%



Grasp Verification: Manual Grasp Experiment

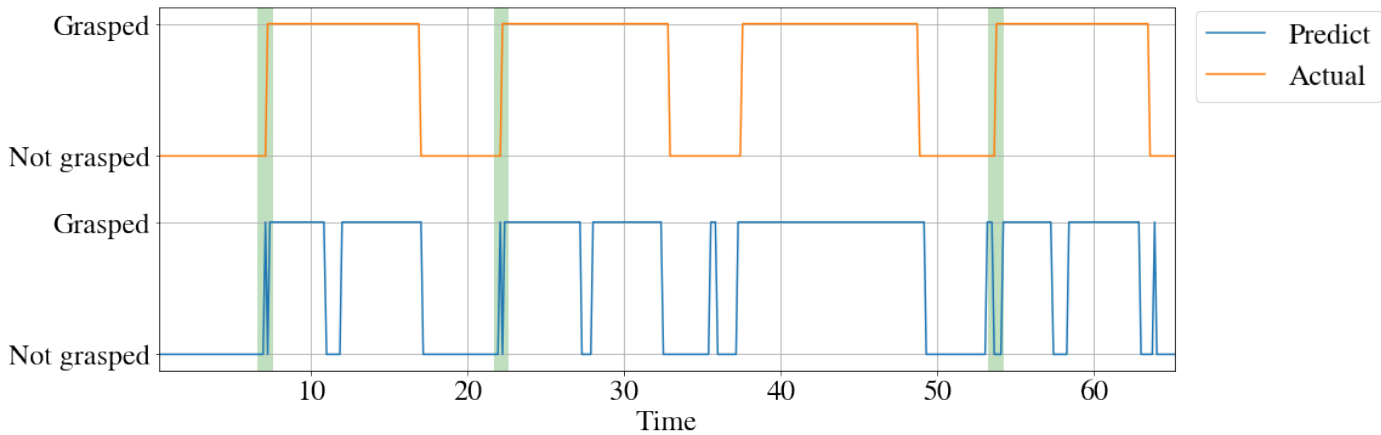




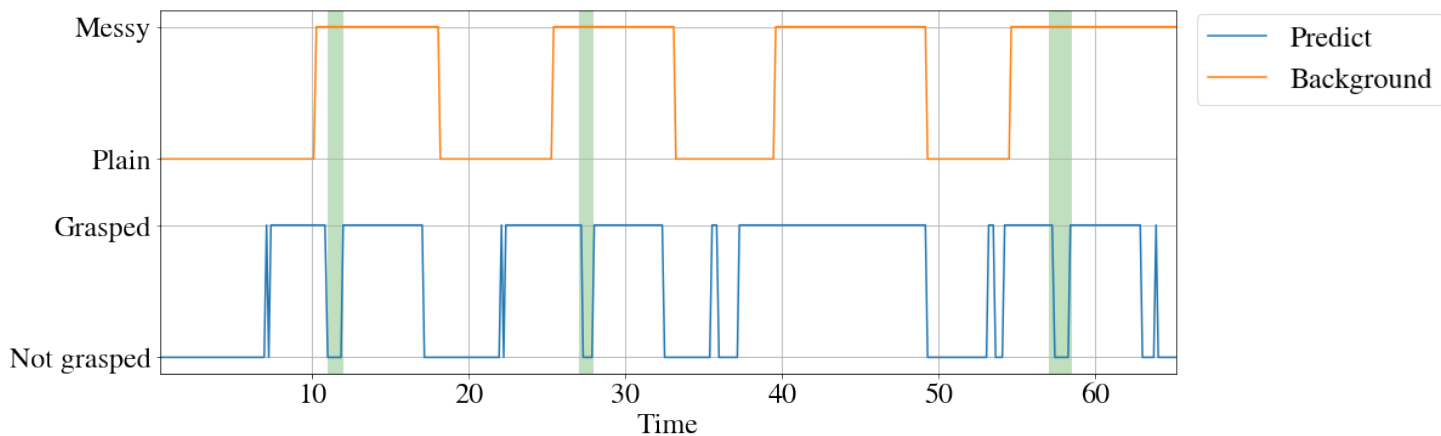
Grasp Verification: Manual Grasp Experiment

Sources of misclassification:

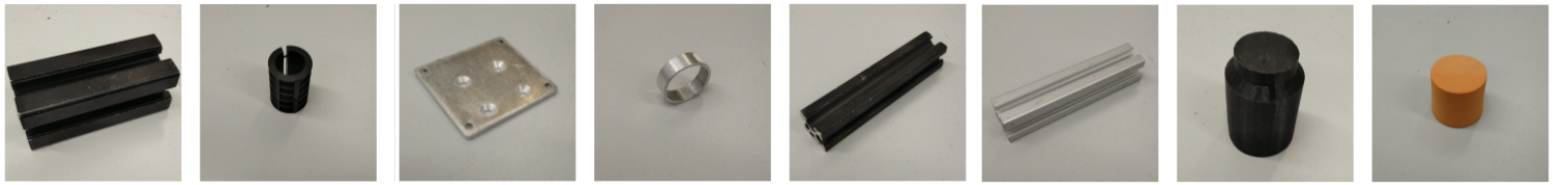
- At pick or drop moments:



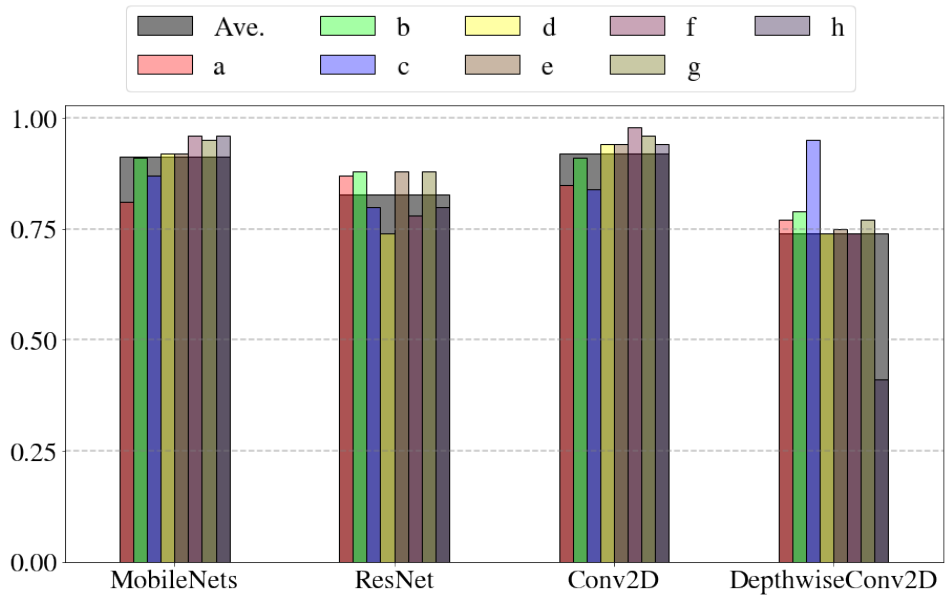
- By messy background:



Grasp Verification: Experiment with YouBot



(a) (b) (c) (d) (e) (f) (g) (h)

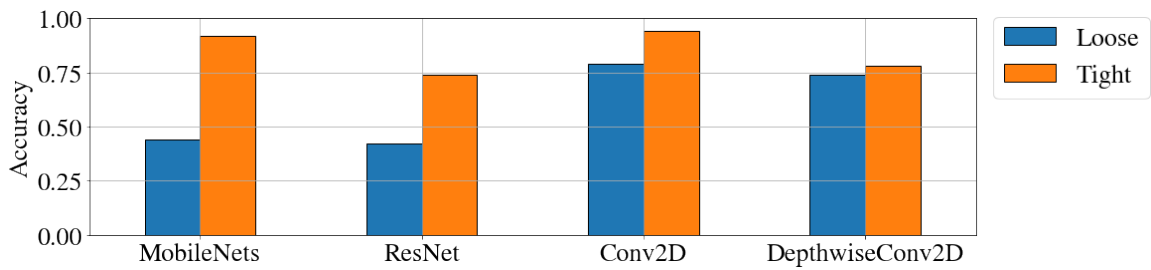


Grasp Verification: Experiment with YouBot

Drawbacks:

- Loose grasp:

[6]

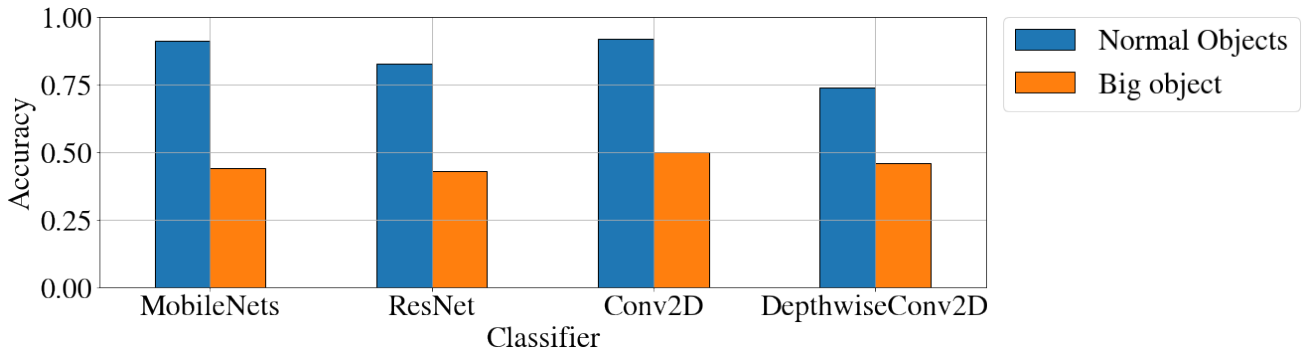


Grasp Verification: Experiment with YouBot

Drawbacks:

- Big object:

Not grasped:

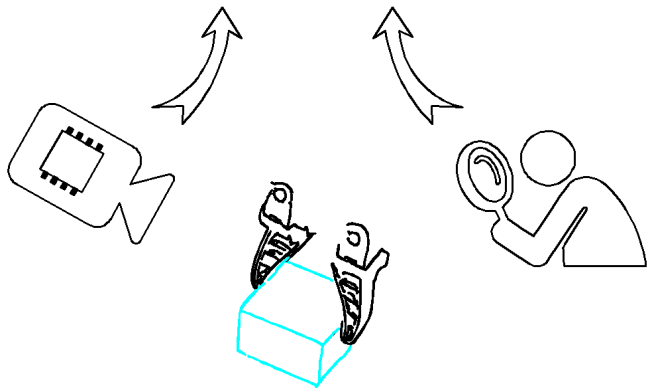


Grasp Verification: Experiment with YouBot

Adding suggestions to dataset:

Frame	Camera	Observer	Eval.
1	Grasped	Grasped	✓
2	Grasped	Not grasped	×
3	Not grasped	Not grasped	✓
4	Grasped	Grasped	✓

⇒ Add frame 2 to dataset - Not grasped



New dataset:

Class	Original dataset	New dataset
Grasped	2134	2376
Not grasped	2191	2307

Train with new dataset:

Accuracy	MobileNets	ResNet	Conv2D	DepthwiseConv2D
Train	0.9937	0.9507	0.9915	0.8010
Validate	0.1000	0.5000	0.7250	0.5500

Contents

Conclusion

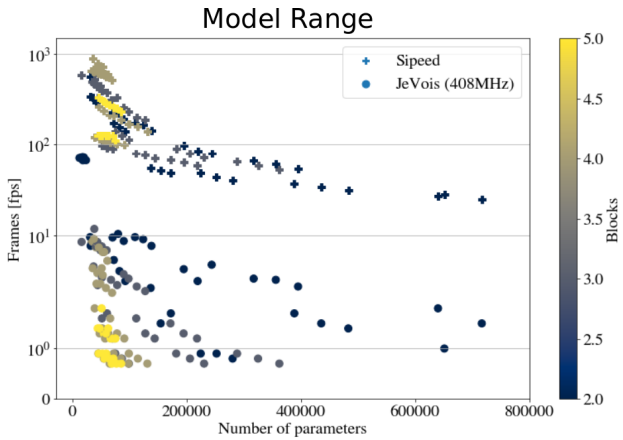
Smart cameras:

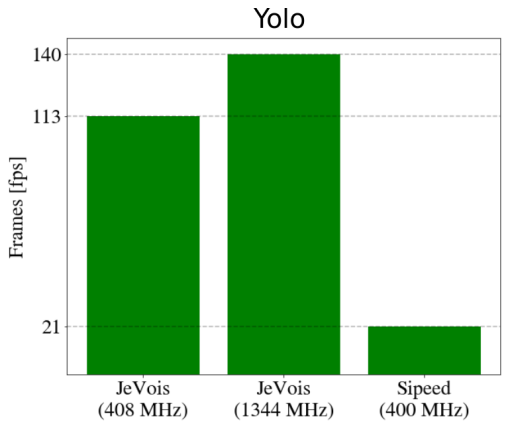


	JeVois	Sipeed	OpenMV
Running custom architectures	√	√	×
Accepting all CNN features	√	×	×
Inference time*	397 ms	38 ms	-

* Based on MobileNets benchmarking experiment

Benchmarking:





Grasp verification:

Experiment	Best accuracy	Drawbacks
Manual grasp	0.96	-Pick or drop moments -Messy background
Exp. with YouBot	0.92	-Loose grasps -Big objects

Conclusion

Future work:

- More robotic applications with smart cameras.
- Examining other CNN architectures.
- Adding more sensory data to the verification system.

Conclusion

Future work:

- More robotic applications with smart cameras.
- Examining other CNN architectures.
- Adding more sensory data to the verification system.

With special thanks to: Prof. Plöger and Deebul Nair.

Conclusion

Future work:

- More robotic applications with smart cameras.
- Examining other CNN architectures.
- Adding more sensory data to the verification system.

With special thanks to: Prof. Plöger and Deebul Nair.

References:

[1] wwww.openmv.io/products/openmv-cam-h7

[2] www.seeedstudio.com/Sipeed-MAix-BiT-for-RISC-V-AI-IoT-p-2872.html

[3] www.jevoisinc.com/

[4] www.github.com/eembc/mlmark

[5] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[6] www.github.com/raghakot/keras-vis

* GitHub repository of this project: www.github.com/amirhpd/grasp_verification