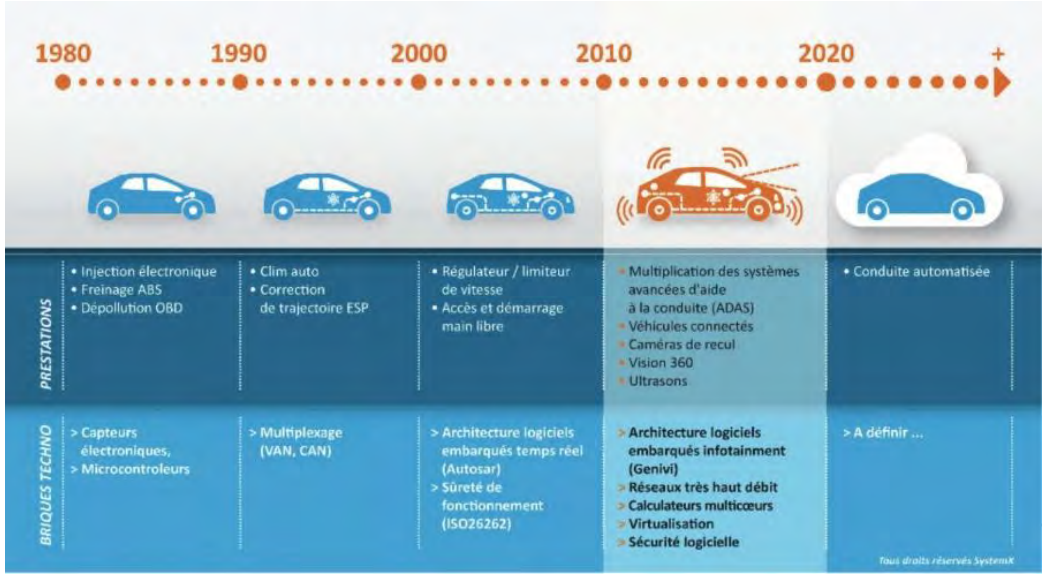

FORUM
MobiliT.AI


AI AND THE EMBEDDED SYSTEMS
NEW CHALLENGES FOR THE SOFTWARE ENGINEERING
RENAULT SOFTWARE LABS / JEAN-MARC GABRIEL

Embedded systems

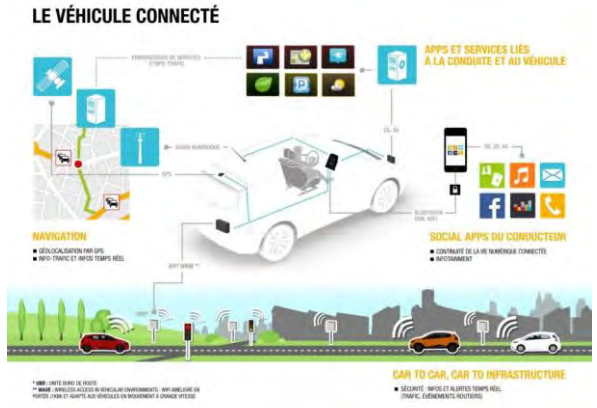
Embedded systems :
+ Autonomous
+ Real time
≈ Dedicated to a specific task
≈ Integrate processor and memory
- Very limited resources (cost, energy, space)



→ The automotive industry is a key actor for embedded innovations relying on AI technologies !

AI based services for the cars

- **On board services**
  - Voice recognition
  - Recommender systems
  - …
- **ADAS/AD**
  - Perception
  - Trajectory planning
  - Motion planner
- **Support services**
  - Predictive maintenance
  - Fleet management
  - …



AI based services for the cars : Embedded ML

- **On board services**
  - Voice recognition
  - Recommender systems
  - …
- **ADAS/AD**
  - Perception
  - Trajectory planning
  - Motion planner
- **Support services**
  - Predictive maintenance
  - Fleet management
  - …

Most of these services relies on **Machine Learning** :

- Neural Network (perception)
- Symbolic ML (predictive maintenance)
- *Reinforcement Learning*

⚠ *Other AI fields can be involved (rules based systems, multi-agent systems, …) but they are not considered here (no particular technical problems, no yet mature for services on production).*

Many of them must be embedded for various reasons :

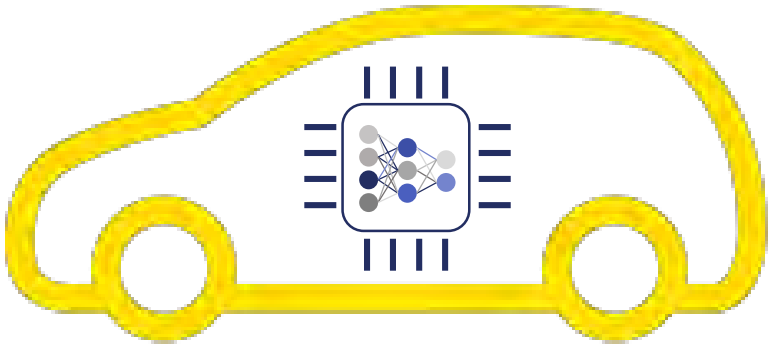
- Real time decision
- Cost of data transfer
- Network robustness not guarantee

⚠ *Edge computing will help but it is not yet ready at the industrial grade.*

Non functional requirements

Energy
Electricity consumption
Thermal

Performance
Inference time (→ real time)



Maintenabilité
Non regression
Over The Air update

Safety
Minimal defect rates
Mode backup
Redundancy on the whole chain

« Certificability »
Severity x Exposure x Controlability
Data privacy (RGPD)

Cyber-security
Data protection
Protection against malicious code injection
Robustness against attacks

Current focal point : inference time for video

Table 1: ConvNet models in the literature

ConvNet	Naming Convention in graphs	Top-5 accuracy (%)	Dataset	# Layers	Parameters	Model Size
AlexNet	alexNet [42]	80.3	ImageNet	5 Conv + 3 FC	62 M	244 MB
GoogleNet	googleNet [56]	90.85	ImageNet	57 Conv + 1 FC	6.9 M	54 MB
Residual Net	resNet50 [32]	93.29	ImageNet	53 Conv + 1 FC	25 M	103 MB
SqueezeNet	squeezeNet [36]	80.3	ImageNet	26 Conv	1.2 M	5 MB
SqueezeNet with Deep Compression	sqCompressed [36]	80.3	ImageNet	26 Conv	1.2 M	675.8 KB
SqueezeNet with Residual Connections	squeezeNetRes [36]	82.5	ImageNet	26 Conv	1.2 M	6.3 MB
VGG	vgg- small [53]	86.9	ImageNet	5 Conv + 3 FC	102 M	393 MB
MobileNet	mobileNet [34]	70.6	ImageNet	27 Conv	29 M	17 MB
Places-CDNS-8s	Places-CDNS-8s [59]	86.8	ImageNet	8 Conv + 3 FC	60 M	241.6 MB
Inception-BN	Inception-BN [37]	89.0	ImageNet	69 Conv + 1 FC	1.4 B	134.6 MB
ALL-CNN-C	ALL-CNN-C [54]	90.92	CIFAR 10	9 Conv	1.3 M	5.5 MB

1,5 Gflops

20 Gflops

VS

Nvidia : *≈ 10 TFLOPS (perception with many cameras), 50 TFLOPS for the whole AD chain*

HARDWARE
- Processor and memory type
- Parallelism strategy

EMBEDDED SYSTEM
↑ Computation speed
↓ Memory access
↓ Energy consumption

SOFTWARE
- NN framework
- NN libraries

⚠ Speech analysis relies more and more on CNN leading to networks holding more than 100M parameters.

DNN at the software level

- Many approaches are proposed for reducing the NN complexity for inference
  - *Reduce precision of operations*
    - **Quantization****
      - Reduce the number of bits for representing the parameter weight (FP32/FP16 → Int32 … Int5)
      - Very good accuracy loss/size reduction ratio but it depends on weight sharing
    - **Weight sharing**
      - Gather (with clustering) similar weights and index the closest value
      - Very good accuracy loss/size reduction ratio but it depends on quantization
  - *Reduce amount of operations*
    - **Network pruning**
      - Remove parameters (*fine grain tuning*) or group of parameters (*coarse-grained tuning*) by considering : low weight (directly or through *l1, l2* regularization), mutual information, remove filters and channels
      - Very good accuracy loss/size reduction can be obtain (but the effort may be important) most often if specialized hardware are adapted (mainly for managing sparse layers)
      - Not sure a pruned neural network performs better than a dense network with the same « size »* !
    - **Leaner architecture**
      - Reduce convolution on channels (MobileNet)
      - …

DNN at the hardware level

Temporal parallelism

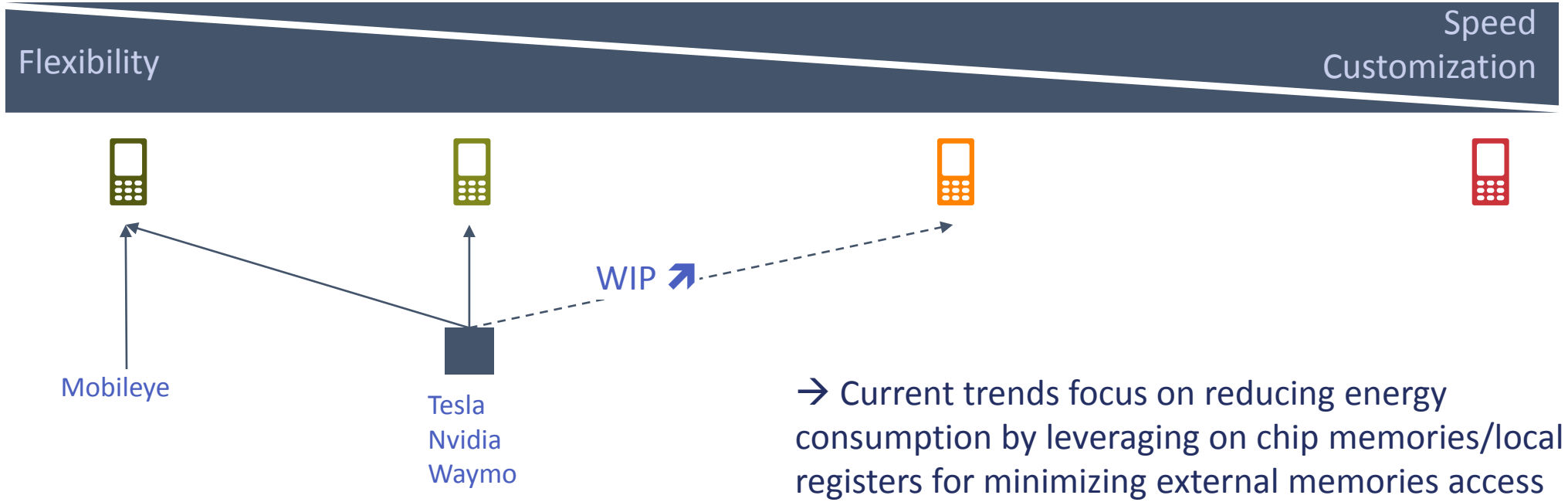
Spatial parallelism

CPU
(sequential ALUs + mem access)

GPU
(parallel ALUs + mem access)

FPGA
(interconnected ALUs with associated memories)

ASIC.TPU
(highly connected DPU aggregating multiple ALUs for a single memory access)

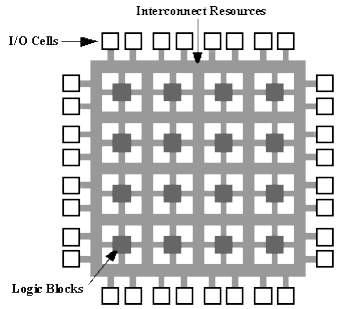


→ Current trends focus on reducing energy consumption by leveraging on chip memories/local registers for minimizing external memories access

FPGA

- « reconfigurable » logical array wich may provide an adhoc data flow adapted to a DNN shape
  - requires much less power
  - if well designed, it provides significant peformance gain
  - quite expensive (for now)
- Design :
  - Enable irregular parallelism (for sparsity)
  - Support custom data types (*cf* compact data types)
  - Depends on the optimization at the soft. level
- Require the use some CNN-to-accelerator toolflows

High density of interconnectable logical gates with volatile memory



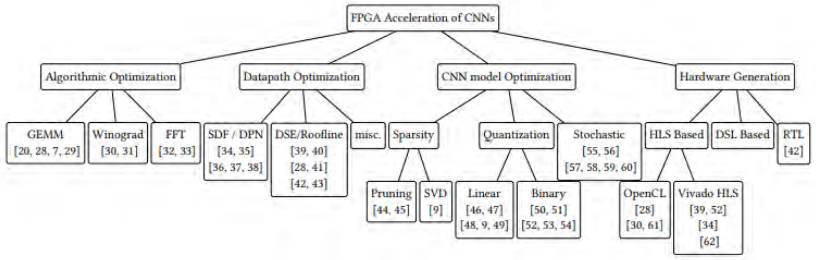


Figure 2: Main Approaches to Accelerate CNN inference on FPGAs

Support tools

- Several frameworks are available for free :
  - Tensorflow
  - Caffe
  - PyTorch

  } Interoperables with ONNX 👍

- Several (Soft/HW) vendors propose additional libraries/tools for accelerating NN processing
  - Nvidia with cuDNN and TensorRT
  - Qualcomm with Neural Processing SDK
  - Intel with Math Kernel Libraries for its CPU based chips
  - Google with Tensorflow extended and Tensorflow Lite
  - Facebook with NNPack and QNNPack

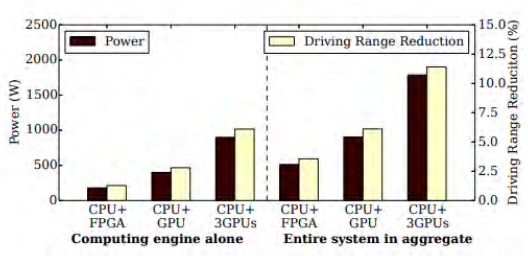
  } Wide benchmarking perimeter

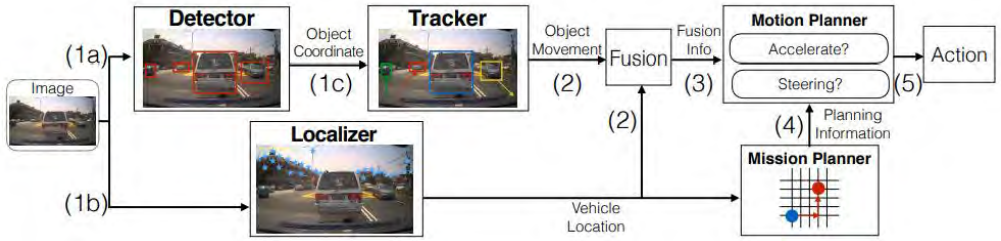
  DawnBench
  DeepBench
  MLPerf

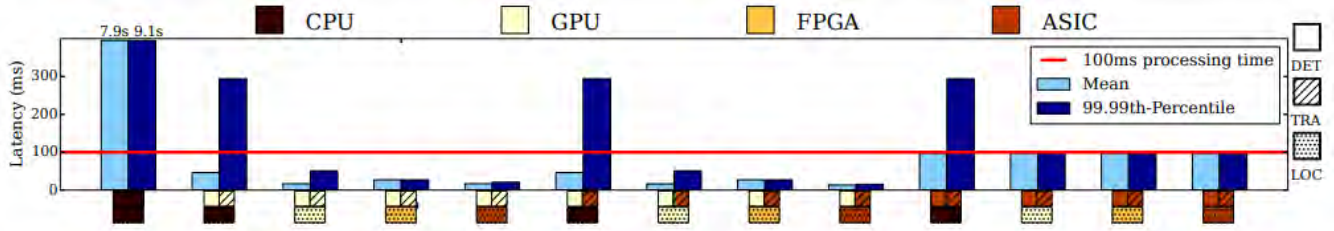
  } Some criteria are missing (power for instance)

Design concerns : integration of differents NN

The Architectural Implications of Autonomous Driving: Constraints and Acceleration [Lin & al., 2018]



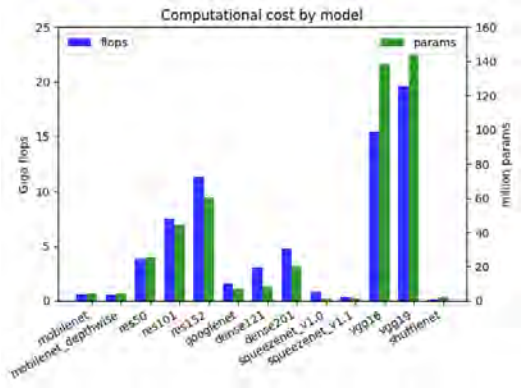




+ Safety constraints lead to integrate several NNs for the same service (eg. perception on various sensors)

Design concerns : arch. perf. not predictable

- No direct relationship between architecture and inference time



JoliBrain's DeepDetect platform evaluation

Table 2: Popular CNN models with their computational workload. Accuracy measured on single-crops of ImageNet test-set.

Model	AlexNet [14]	GoogleNet [18]	VGG16 [6]	VGG19 [6]	ResNet50 [19]	ResNet101 [19]	ResNet-152 [19]
Top1 err	42.9 %	31.3 %	28.1 %	27.3 %	24.7%	23.6% %	23.0%
Top5 err	19.80 %	10.07 %	9.90 %	9.00 %	7.8 %	7.1 %	6.7 %
conv layers	5	57	13	16	53	104	155
conv workload (MACs)	666 M	1.58 G	15.3 G	19.5 G	3.86 G	7.57 G	11.3 G
conv parameters	2.33 M	5.97 M	14.7 M	20 M	23.5 M	42.4 M	58 M
Activation layers	ReLU						
pool layers	3	14	5	5	2	2	2
FC layers	3	1	3	3	1	1	1
FC workload (MACs)	58.6 M	1.02 M	124 M	124 M	2.05 M	2.05 M	2.05 M
FC parametrs	58.6 M	1.02 M	124 M	124 M	2.05 M	2.05 M	2.05 M
Total workload (MACs)	724 M	1.58 G	15.5 G	19.6 G	3.86 G	7.57 G	11.3 G
Total parameters	61 M	6.99 M	138 M	144 M	25.5 M	44.4 M	60 M

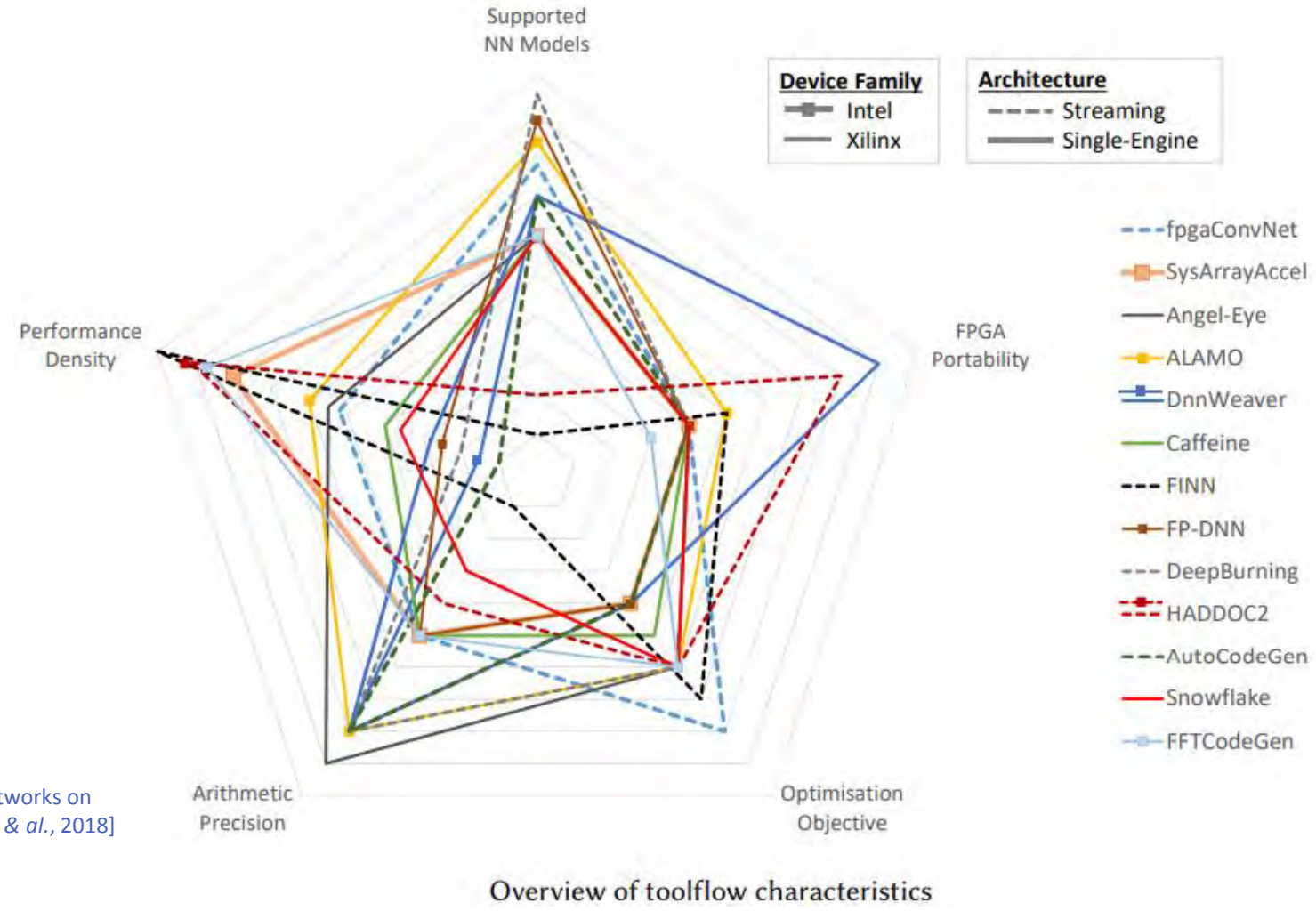
Accelerating CNN inference on FPGAs: A Survey [Abdelouahab *& al.*, 2018]

Table 1: Execution time of convolutional layers with 3× 3 kernel size, stride 1, same padding, and 224×224 input image size on the Nexus 5 phone.

	in_channel	out_channel	FLOPs	Time (ms)
CNN1	8	32	452.4 M	114.9
CNN2	32	8	452.4 M	300.2
CNN3	66	32	3732.3 M	908.3
CNN4	43	64	4863.3 M	751.7

FastDeepIoT: Towards Understanding and Optimizing Neural Network Execution Time on Mobile and Embedded Devices [Yao *& al.*, 2018]

Design concerns : FPGA mapping tools rely on various strategies

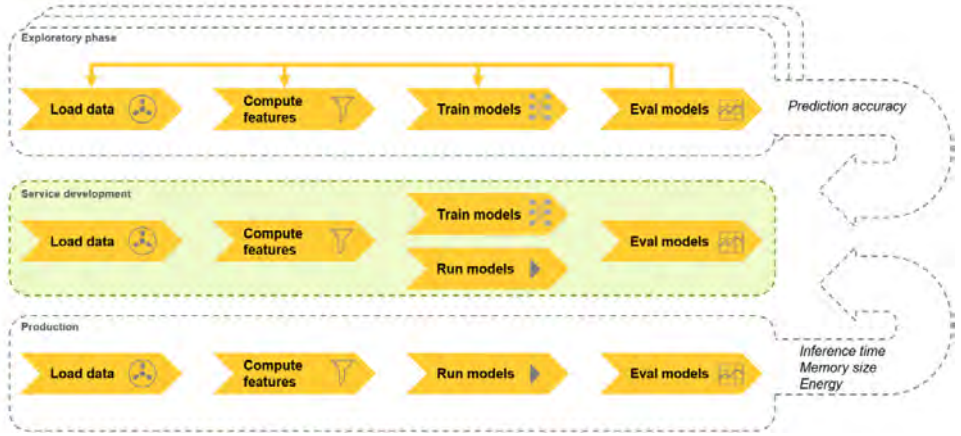


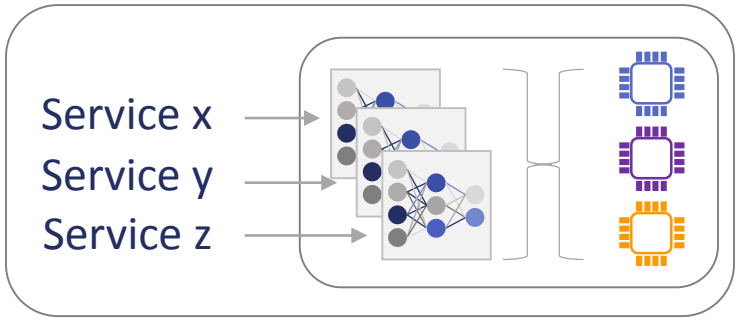
Overview of toolflow characteristics

Toolflows for Mapping Convolutional Neural Networks on FPGAs: A Survey and Future Directions [Venieris *& al.*, 2018]

New challenges for soft. engineering

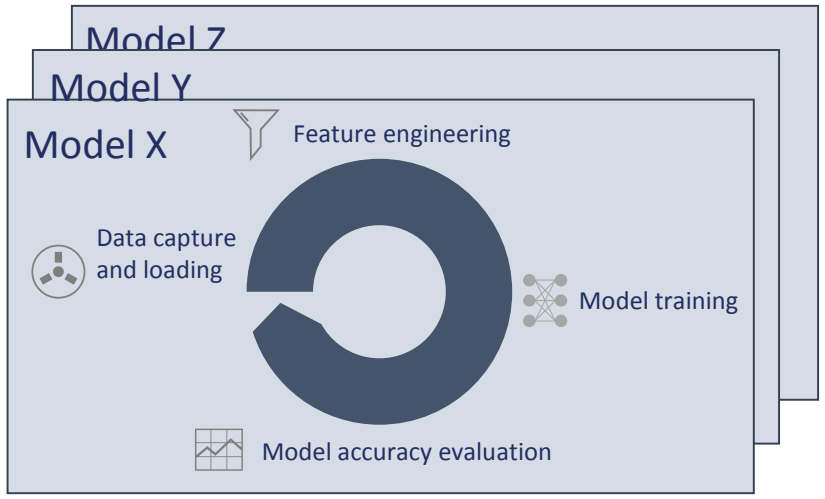
- **SE objective :** fast iterations for fast and secured moves from POC to production
- The basics :
  - DevOps/Conf management/Automated testing
  - Use of ONNX compliant frameworks
  - Set up a dashboard tracking metrics evolution
- Platforming : DYBY (« Do Your Bench Yourself ») for fast exploration of many configurations
  - Make clear production requirements (*specifications ?*) for all ML based components
  - Establish and maintain a reference baseline
  - Negociate new vendors relationship (upstream assessment)
  - Simulation is also an option
  - **Accept the cost and the latency !**





Two optimization cycles to integrate

- *Integrate* means having the ability to take decisions for solving one concern with a clear idea of their impact on other concerns.
- … instead of having to check at each development step that all concerns are adressed

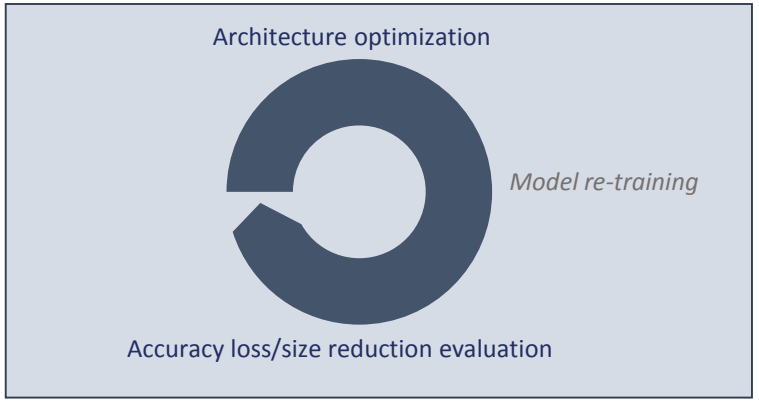


Accuracy optimization

No idea of HW constraints/possibilities
→ continuous checks

Some hints about NN resource needs
→ occasional checks

A - configurable and consistent - model establishes the boundaries of a NN *wrt* a given HW configuration
→ sequential optimization phases



Performance optimization

MERCI !