NEURAL ARCHITECTURE SEARCH

PUBLIC

Willem Sanberg
Sr. Research Engineer Embedded AI Systems
NXP - CTO Automotive System Innovations

MARCH 2022

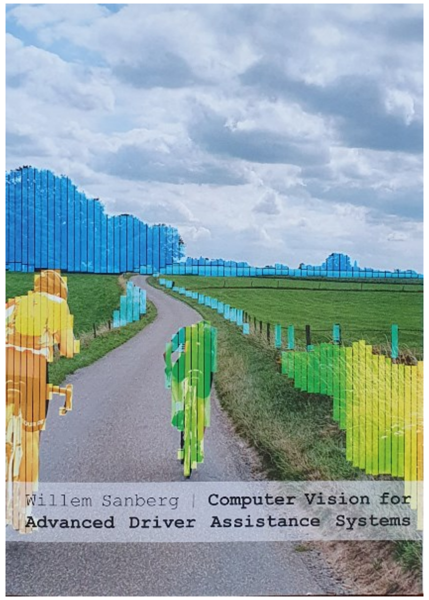
SECURE CONNECTIONS FOR A SMARTER WORLD

Intro & Background

SECURE CONNECTIONS
FOR A SMARTER WORLD

ABOUT WILLEM SANBERG, REGARDING THIS TALK:

- PhD in "Computer Vision for Advanced Driver Assistance Systems"
  - TU/e - Electrical Engineering - Mobile Perception Systems Lab
  - Stereo Camera-based environment perception
  - Algorithm/Software-oriented with several in-vehicle prototypes


Willem Sanberg | Computer Vision for Advanced Driver Assistance Systems

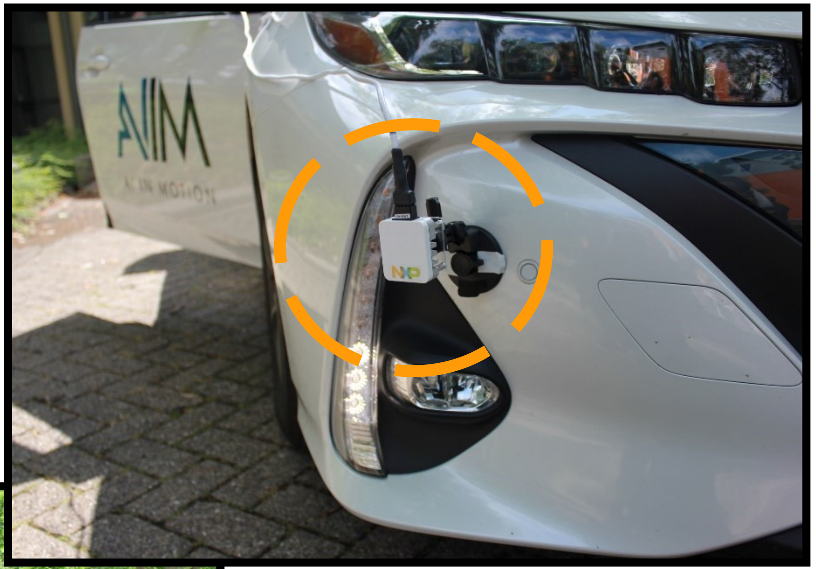
- Sr. Research Engineer for embedded AI Systems
  - NXP Semiconductors
  - CTO Automotive System Innovations ('R&D department')
  - My job in a nutshell:
    - Scouting & analysing AI research (in-house and university collaborations)
    - Translate to NXP requirements & research projects
    - Execute such a project in a team (small, but cross-NXP)

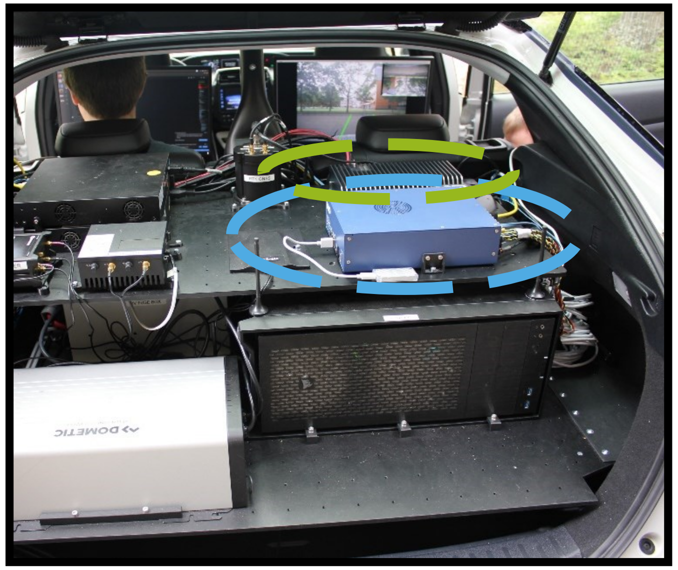
NXP headquarters in Eindhoven (High Tech Campus)

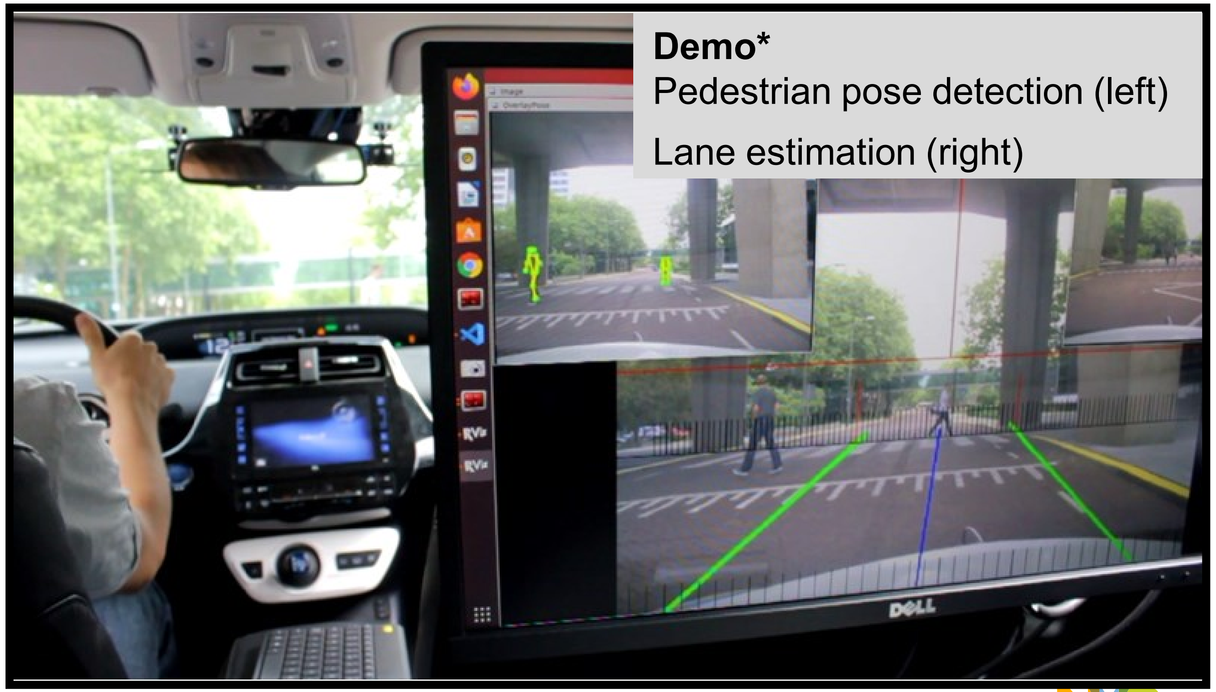
NXP CTO ('R&D')

Automotive System Innovations (ASI)

- Prototyping systems with NXP solutions, e.g.:
- Radar, AI/ML 'brain', Network
- In-house & collaborations







Demo*
Pedestrian pose detection (left)
Lane estimation (right)

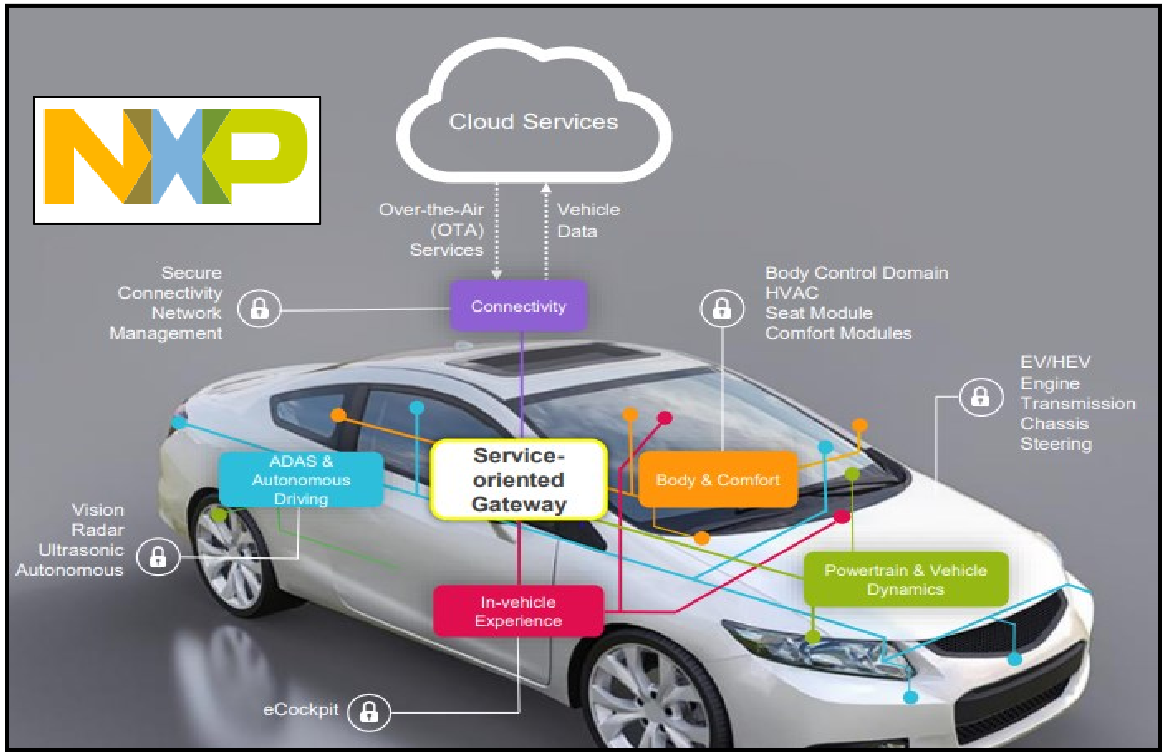
*Open source TPU-PoseNet code runs on the EdgeTPU, connected to NXP's BlueBox2, where an optimized TU/e lanetracker algorithm runs on NXP Layerscape CPU

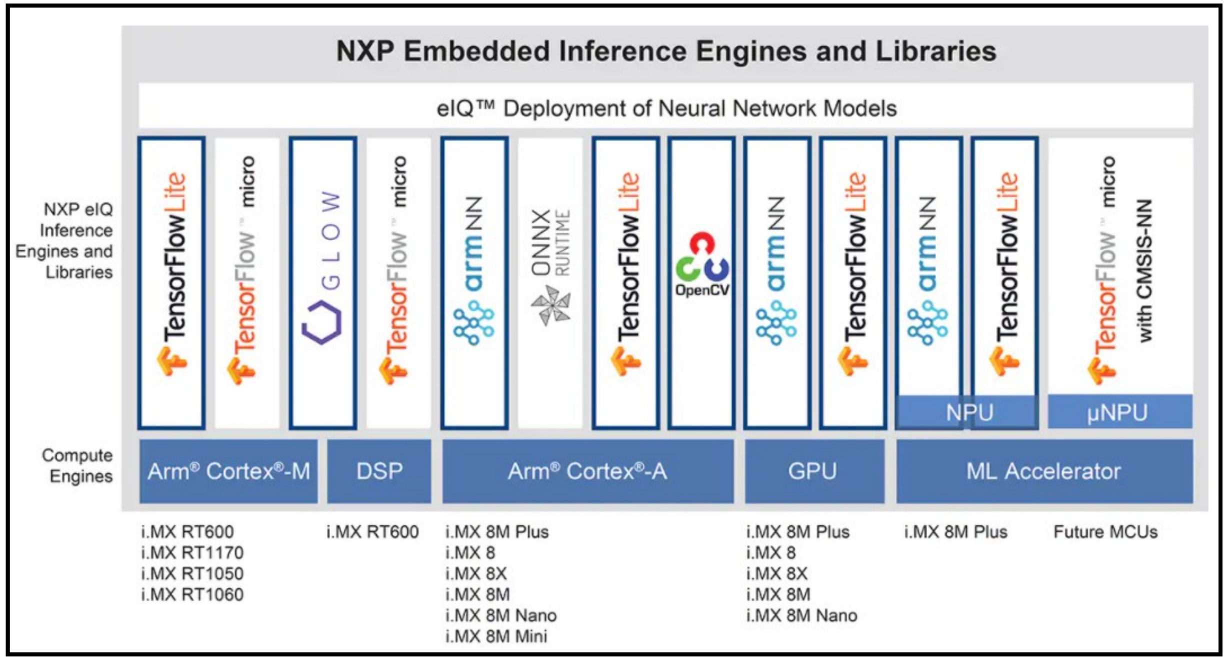
THE PORTFOLIO OF NXP IS BROAD OVER SEVERAL DIMENSIONS

- Functionality:
  - Compute
  - Connectivity
  - HMI
- Data:
  - Radar
  - UWB
  - Analytics
  - Vision
- Applications:
  - Automotive
  - IoT/edge
  - Industrial automation
  - Drones

For AI deployment:
- Applications
- Chips
- Constraints

→ **different requirements on neural network architectures**





Slide courtesy: Gerardo Daalderop gerardo.daalderop@nxp.com

CONTENTS

Neural Architecture Search: overview

SECURE CONNECTIONS
FOR A SMARTER WORLD

NEURAL ARCHITECTURES

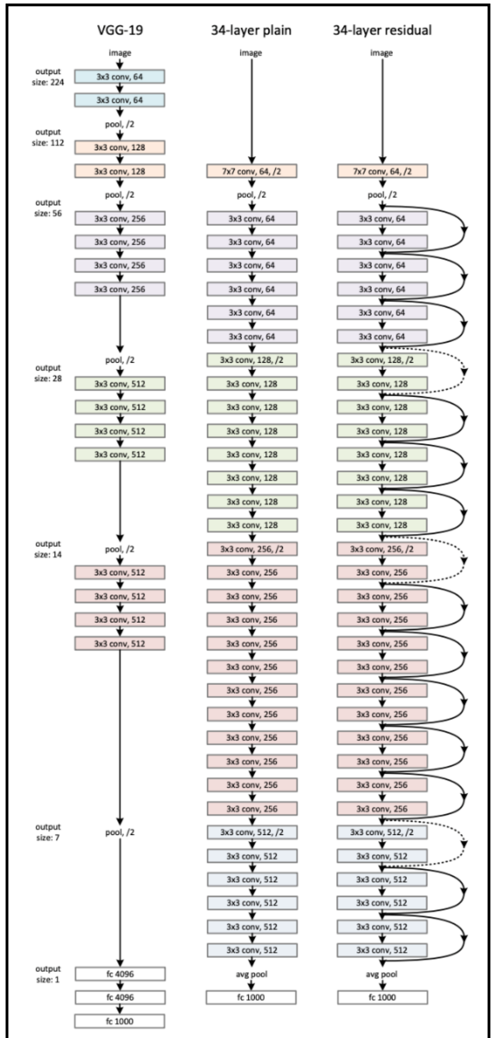
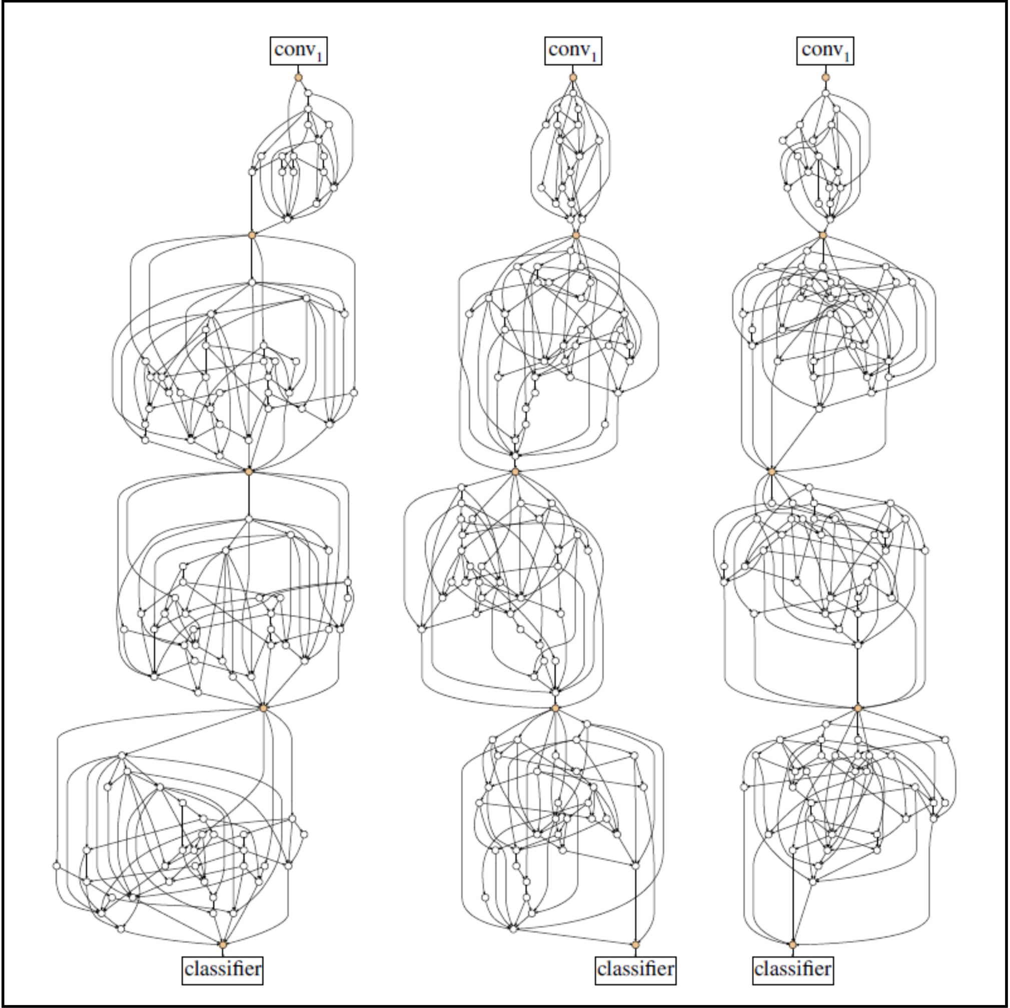


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

- Neural architectures:
  - well-structured patterns
  - operations and connections,
  - manually designed by D.L. experts
- However:
  - Search space is much bigger
  - Variety/constraints intractable for manual design
    - Accuracy, speed, energy, memory…
- NAS aims at automatically finding 'better' solutions in a scalable way



Several randomly generated networks outperform ResNet-50 on ImageNet classification (+2% top-1 acc @ same #FLOPS)

He et al., 2016, Deep Residual Learning for Image Recognition
Simonyan et al., 2014, Very Deep Convolutional Networks for Large-Scale Image Recognition
Xie et al., 2019, Exploring Randomly Wired Neural Networks for Image Recognition

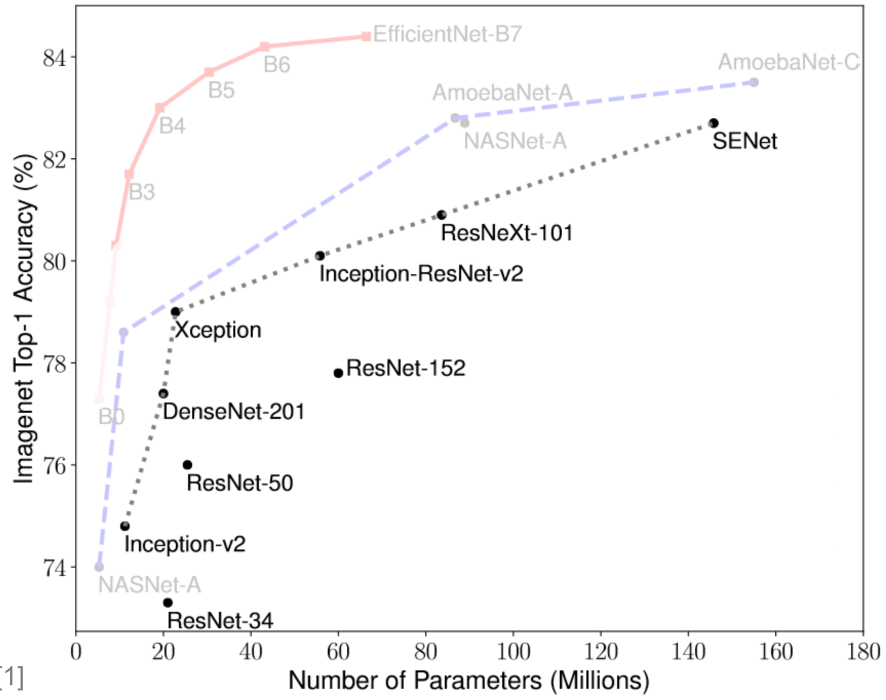
NEURAL ARCHITECTURE SEARCH FINDS EFFECTIVE & EFFICIENT SOLUTIONS

- Automated NAS finds significantly more accurate yet efficient solutions than expert data scientist have created

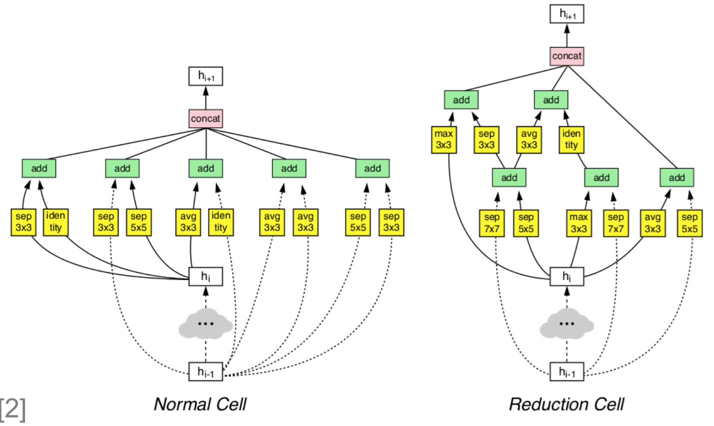
┈┈┈	Hand-crafted architectures (most competitive solutions)
╌╌╌	Early Neural Architecture Search (2017) [2]
──	Recent Neural Architecture Search (2019) [1]

- NAS offers an automated approach especially suitable for multi-objective optimization of DNNs
  - Interesting hardware-aware secondary objectives: *#params, #ops,* latency, energy, bandwidth, compute utilization, etc.

- Found solutions may incorporate complex structures human experts would not intuitively design



[1]



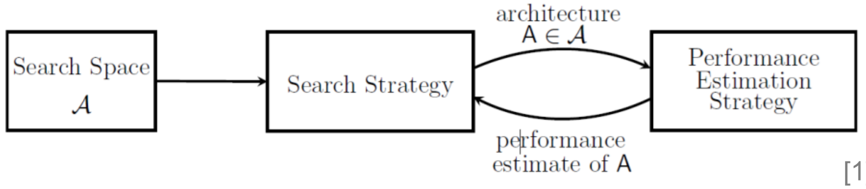
[2]

[1] EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, ICML 2019

[2] Learning Transferable Architectures for Scalable Image Recognition, CVPR 2018

HOW DOES NAS WORK? NAS IS COARSELY DEFINED BY THREE ASPECTS: 1) SEARCH SPACE, 2) SEARCH STRATEGY, 3) PERFORMANCE ESTIMATION

- Methods differ based on three main aspects of NAS
  1) Search space → Which architectures & HPs* can be found?
  2) Search strategy → How to explore possible solutions?
  3) Performance estimation → How does a solution perform wrt. accuracy, hardware-cost, etc.?



Search Space $\mathcal{A}$
Search Strategy
architecture $A \in \mathcal{A}$
Performance Estimation Strategy
performance estimate of A

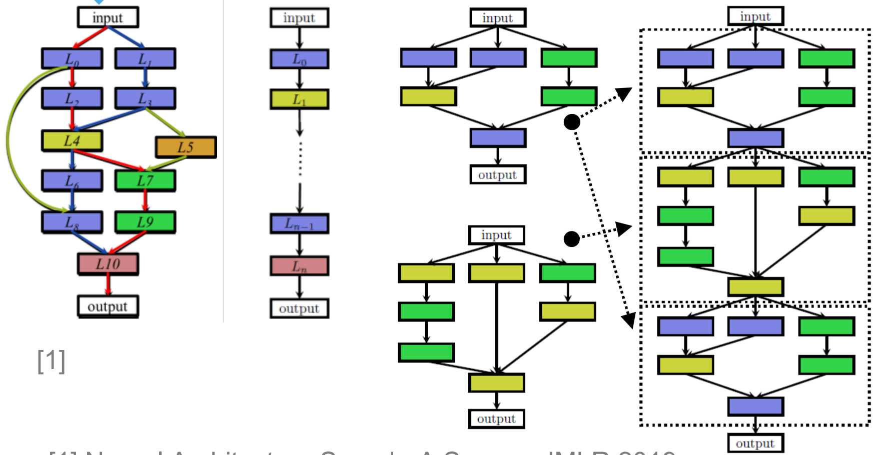
[1]

Design decisions w.r.t these 3 aspects impact on NAS resource requirements and evaluation time

1) Search Space

Smaller space

- Multi-branch networks incl. HPs*
- Chain structured DNNs**
- Cell-based
- Macro-architecture/One-shot model

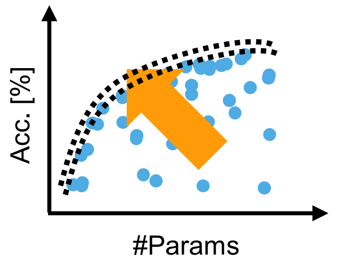


[1]

2) Search strategy

Faster convergence

- Random search
- Reinforcement learning
- Evolutionary/Genetic Alg.
- Bayesian optimization esp. for HPs*
- Differentiable architecture search



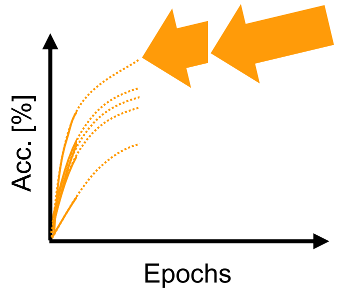
1) Performance Estimation

Faster estimation

- Full training
- Learning curve extrapolation

Additionally, for 2ndary objectives:

- *Profiling, MIL#, HIL##*
- *Surrogate-model-based evaluation*

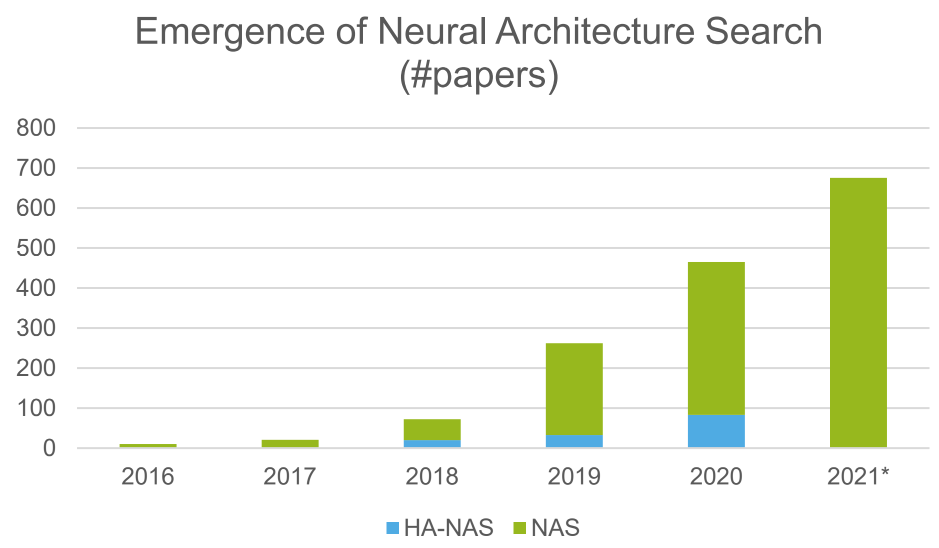


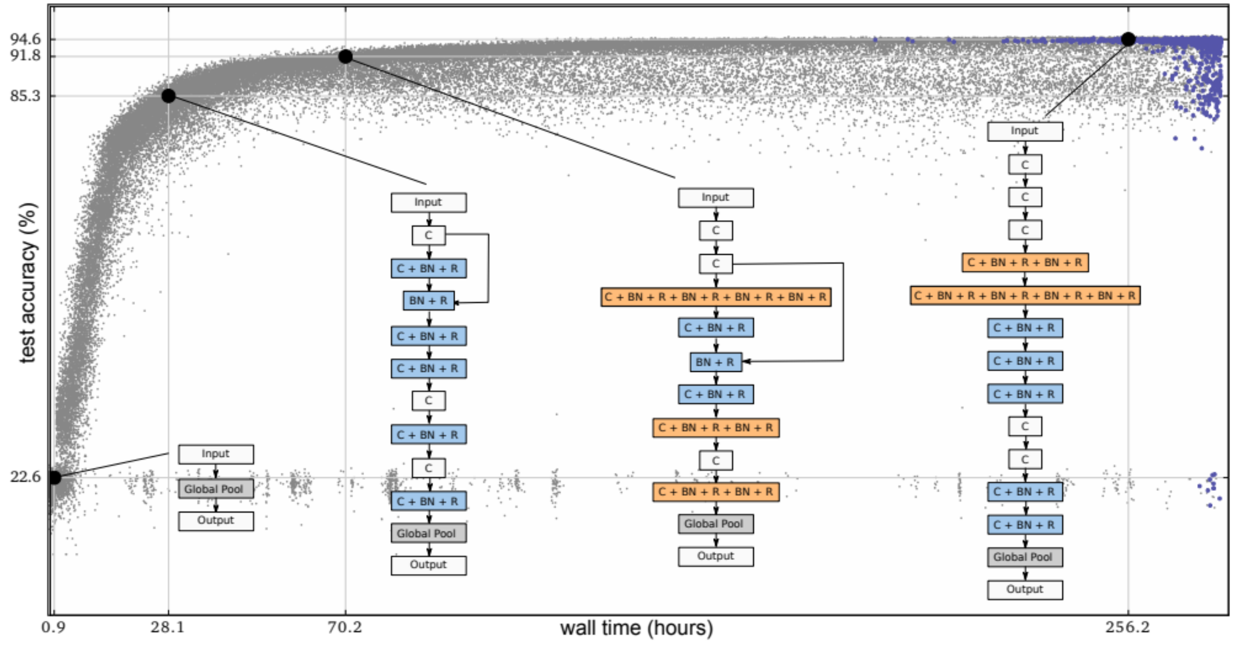
* HP – hyper parameters
** DNN – deep neural network
# MIL – model in the loop
## HIL – hardware in the loop

[1] Neural Architecture Search: A Survey, JMLR 2019

Slide credit: Sebastian Vogel (NXP)

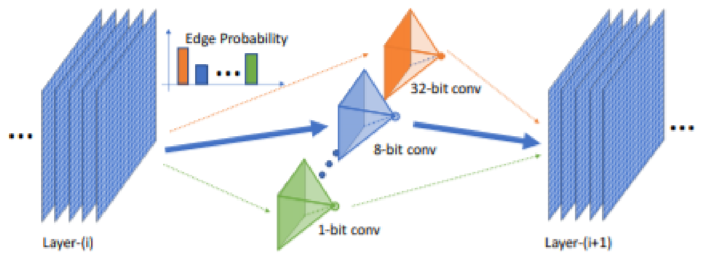
THE FIELD OF NAS IS EXPLODING



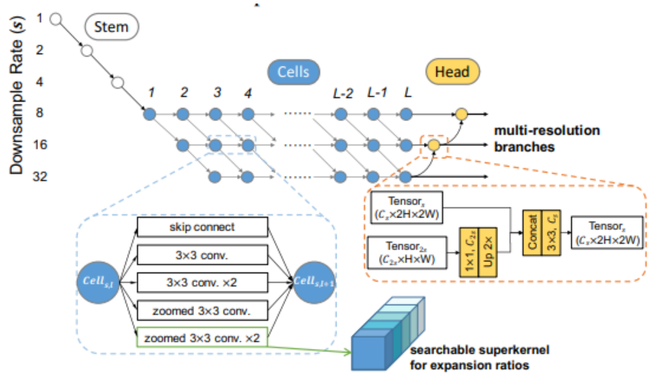


Real et al., Large-scale evolution of Image Classifiers (**2017**)

- Increasing number of commercial services around AutoML & NAS available
  - Microsoft NNI & ArchAI, Google AutoML, etc.



Wu et al., Mixed Precision Quantization in NAS (**2018**)



Chen et al., FasterSeg (**2020**)

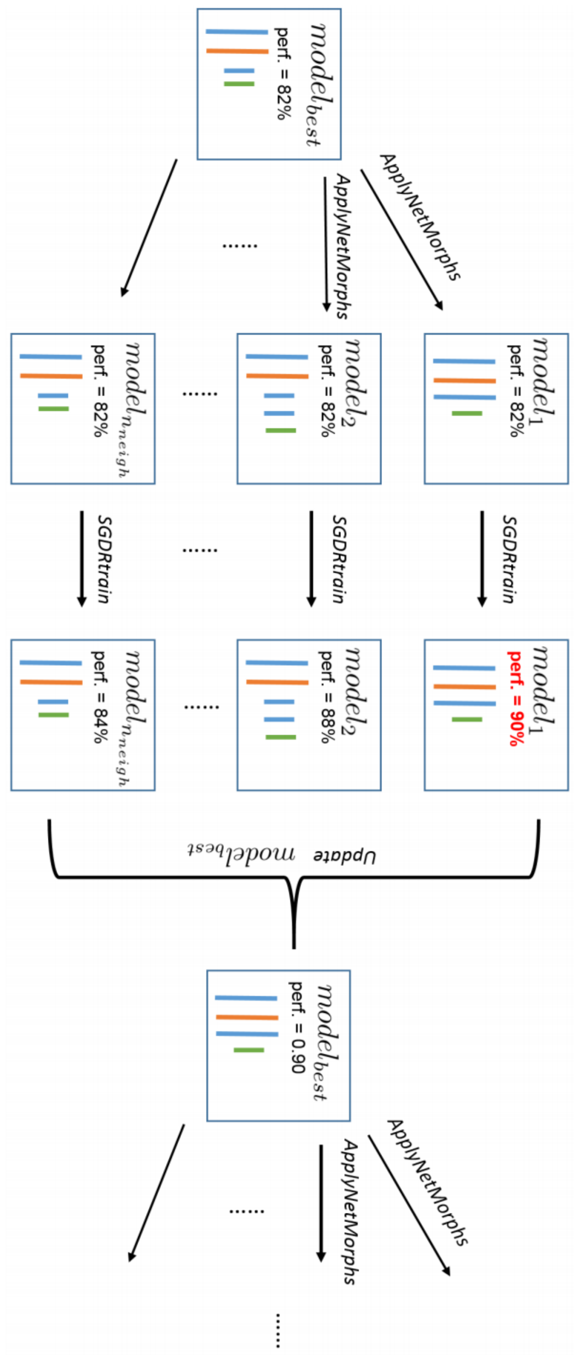
*no data on HA-NAS for 2021
Source on indicative numbers in graph above :
- HA-NAS from Benmeziane et al., 2021, A Comprehensive Survey on Hardware-Aware Neural Architecture Search
- NAS from http://www.ml4aad.org/automl/literature-on-neural-architecture-search/ (which is constantly being updated)

NASH: NAS BY HILL CLIMBING (CONCEPT)

- NAS with Evolutionary search
- Search guided via selection of parents
  - Select most successful child after quick training

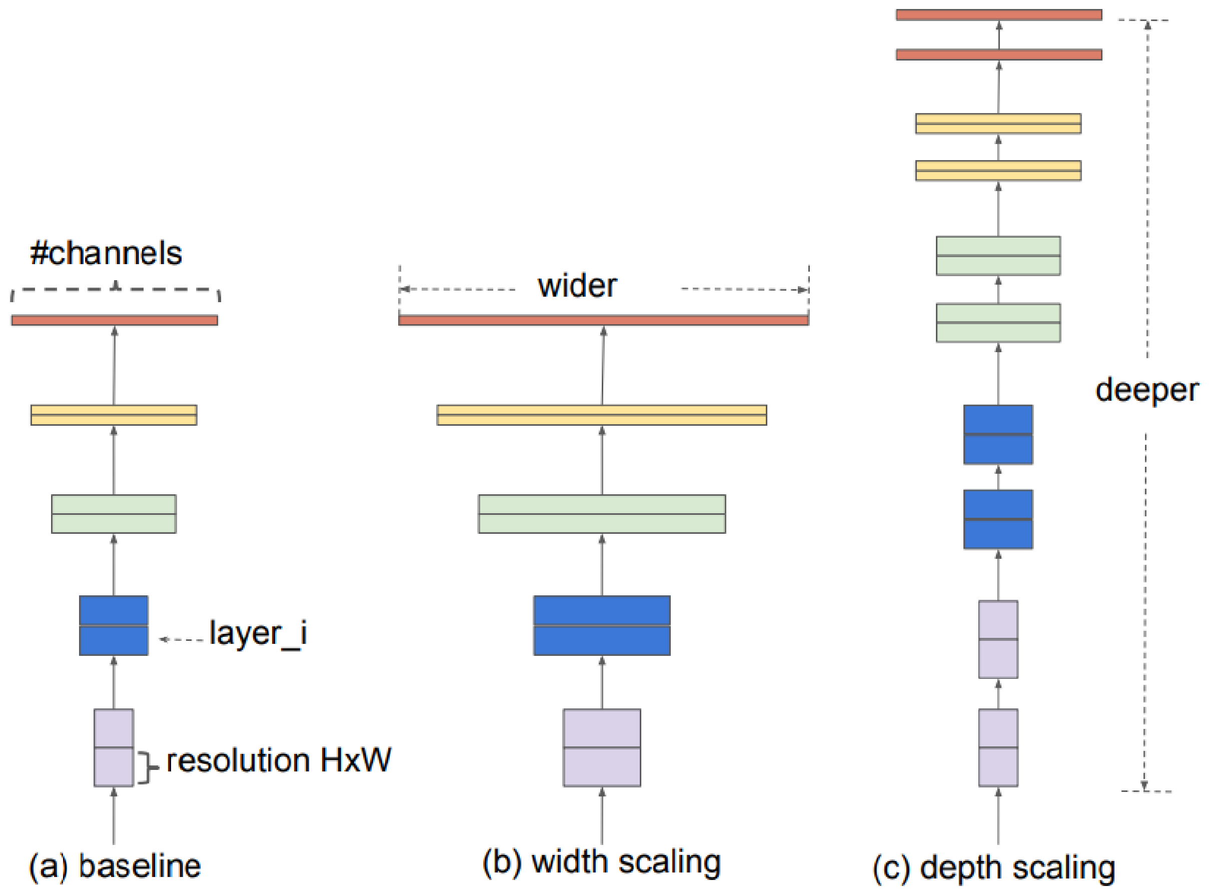
Algorithm 1 Network architecture search by hill climbing

```
1: function NASH( $model_0, n_s, n_n, n_{NM}, e_{neigh}, e_{final}, \lambda_a, \lambda_s$ )
2:     $model_{best} \leftarrow model_0$
3:     for $i \leftarrow 1, \ldots, n_s$ do
4:         for $j \leftarrow 1, \ldots, n_n$ do
5:             $model_j \leftarrow ApplyNetMorphs(model_{best}, n_{NM})$
6:             $model_j \leftarrow \text{SGDRtrain}(model_j, e_{neigh}, \lambda_s, \lambda_a)$
7:         $model_{best} \leftarrow \arg\max_{j=1,\ldots,n_n}\{performance_{vali}(model_j)\}$
8:     $model_{best} \leftarrow \text{SGDRtrain}(model_{best}, e_{final}, \lambda_s, \lambda_a)$
9:     return $model_{best}$
```



NASH: NAS BY HILL CLIMBING (CONCEPT)

- Three morphisms (sampled uniformly):
  1. **Make deeper** (add [Conv, BN, Relu] block)
     - Sample uniformly:
       position, kernel size {3,5};
     - #channels from predecessor
  2. **Make wider** (increase #channels)
     - Sample uniformly:
       - Target conv layer
       - Widening factor {2,4}
  3. **Add bypass/skip connection**
     - Sample uniformly:
       - Target layers to connect
       - Concatenation or addition



EfficientNet, Tan & Le, 2018

NASH: NAS BY HILL CLIMBING (RESULTS)

- Three morphisms (sampled uniformly):
  1. **Make deeper** (add [Conv, BN, Relu] block)
     - Sample uniformly:
       position, kernel size {3,5};
     - #channels from predecessor
  2. **Make wider** (increase #channels)
     - Sample uniformly:
       - Target conv layer
       - Widening factor {2,4}
  3. **Add bypass/skip connection**
     - Sample uniformly:
       - Target layers to connect
       - Concatenation or addition

- Results:
  1. Similar performance as other NAS methods
  2. …. with way faster convergence
  3. … but still outperformed by some handcrafted methods

	Resources	CIFAR-10 error
Real et al. (2017)	250 GPUs, 10 days	5.4
NASH (2017)	1 GPU, 12 hrs 1 GPU, 48 hrs	5.7 4.7
handcrafted (Gastaldi 2017)	2 GPUs, 2 days	2.9

NASH: NAS BY HILL CLIMBING (IMPROVEMENTS)

- Three morphisms (sampled uniformly):

1. **Make deeper** (add [Conv, BN, Relu] block)
   - Sample uniformly:
     position, kernel size {3,5};
   - #channels from predecessor
2. **Make wider** (increase #channels)
   - Sample uniformly:
     - Target conv layer
     - Widening factor {2,4}
3. **Add bypass/skip connection**
   - Sample uniformly:
     - Target layers to connect
     - Concatenation or addition

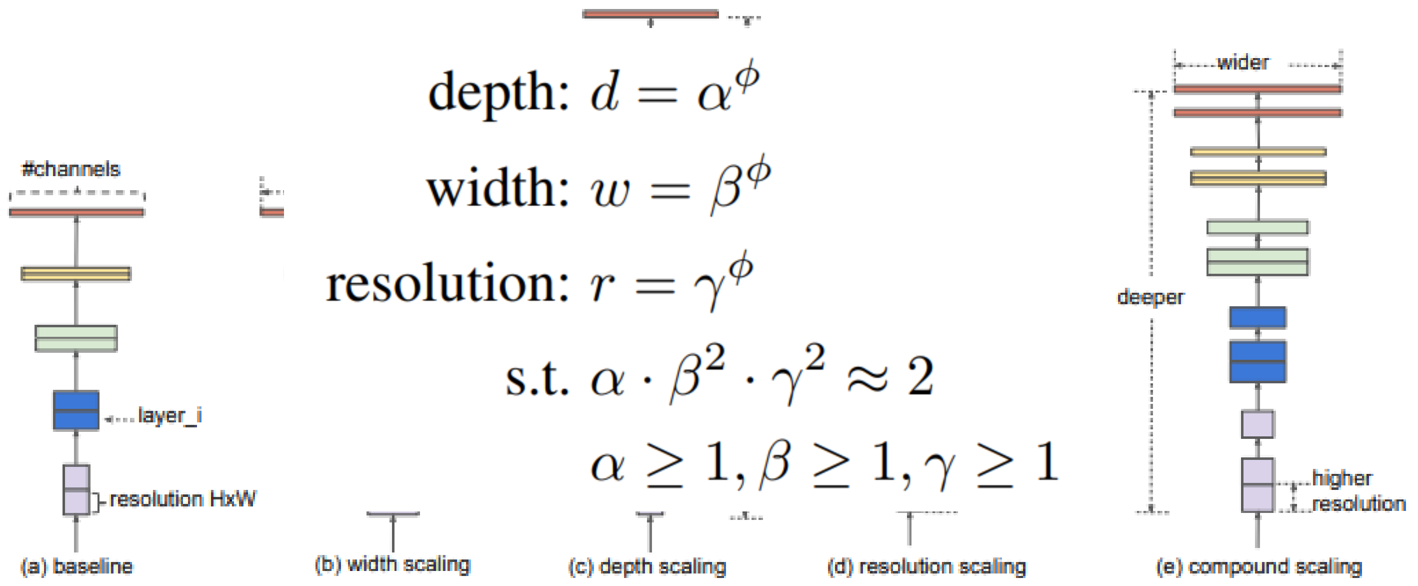
- Where to improve?

1. Better search guidance:

   Bayesian Optimization instead of uniform sampling (e.g. Auto-Keras)

2. Consider morphisms jointly

   E.g. compound scaling in EfficientNet



$$\text{depth: } d = \alpha^{\phi}$$
$$\text{width: } w = \beta^{\phi}$$
$$\text{resolution: } r = \gamma^{\phi}$$
$$\text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$
$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

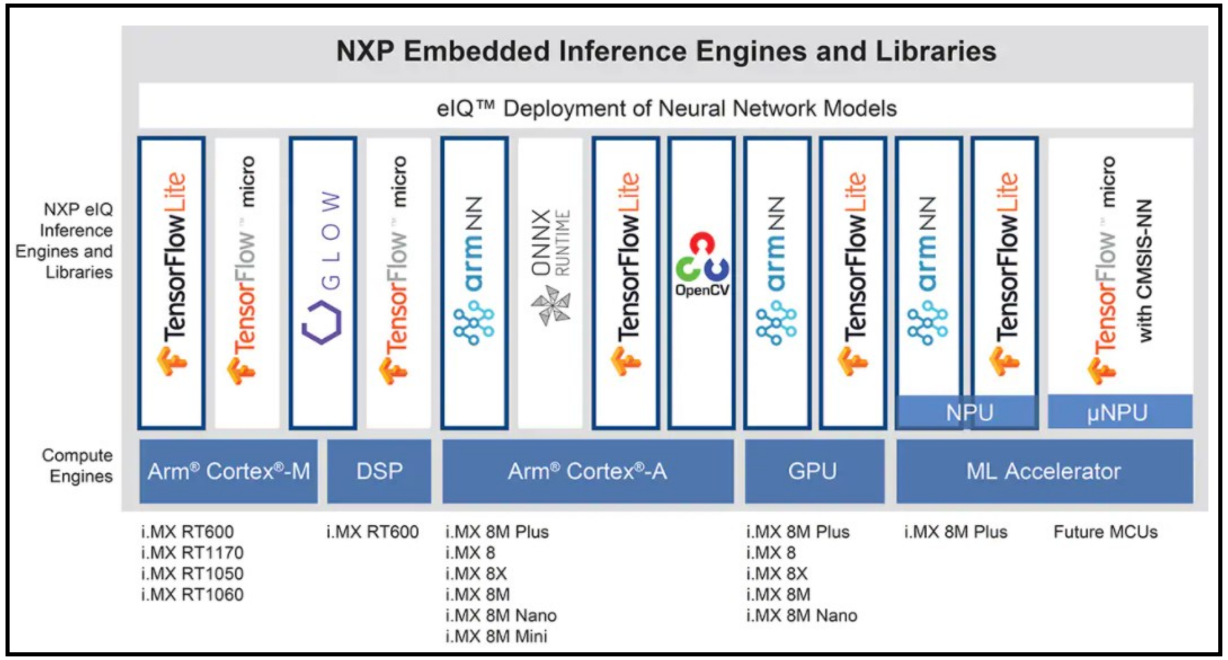
Auto-Keras: An Efficient Neural Architecture Search System, Jin et al., 2019
EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, Tan & Le, 20120

NAS RECAP

- NAS aims at
  - automatically finding ‘better’ solutions in a scalable way
  - to deal with design variety and constraints that are intractable for manual design
    - building blocks, topology; accuracy, speed, energy, memory…

- Embedded deployment gives
  - additional complexity, since
  - each platform different strengths & weaknesses

- Hardware-aware NAS aims at
  - jointly optimizing accuracy and efficiency

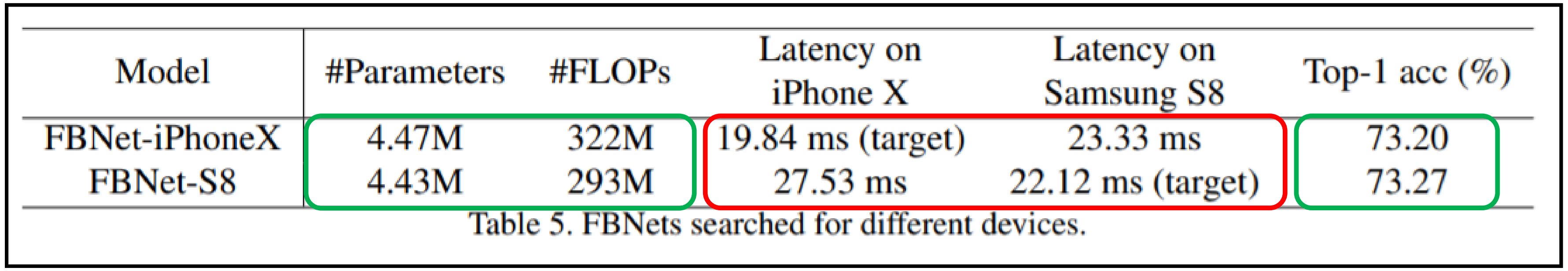


Hardware-aware Neural Architecture Search: motivation

SECURE CONNECTIONS
FOR A SMARTER WORLD

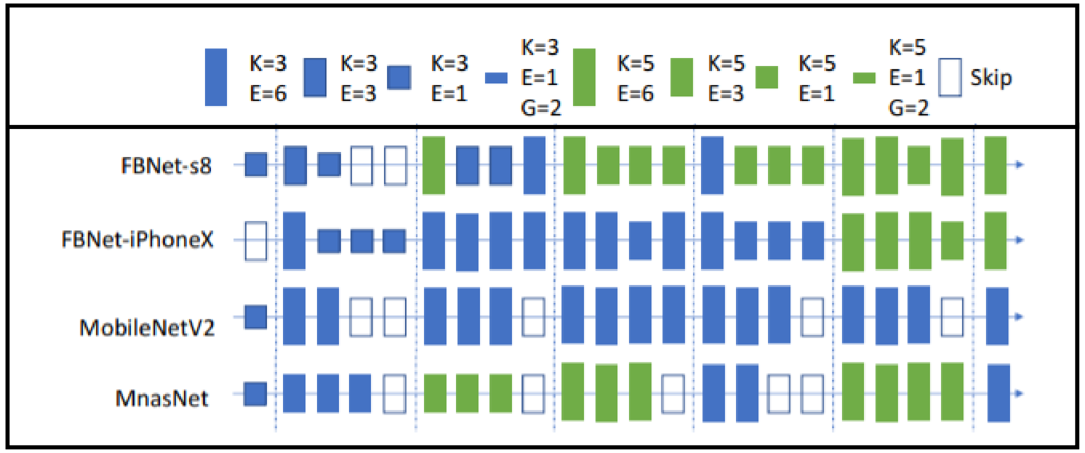
NETWORKS ON DIFFERENT EMBEDDED DEVICES

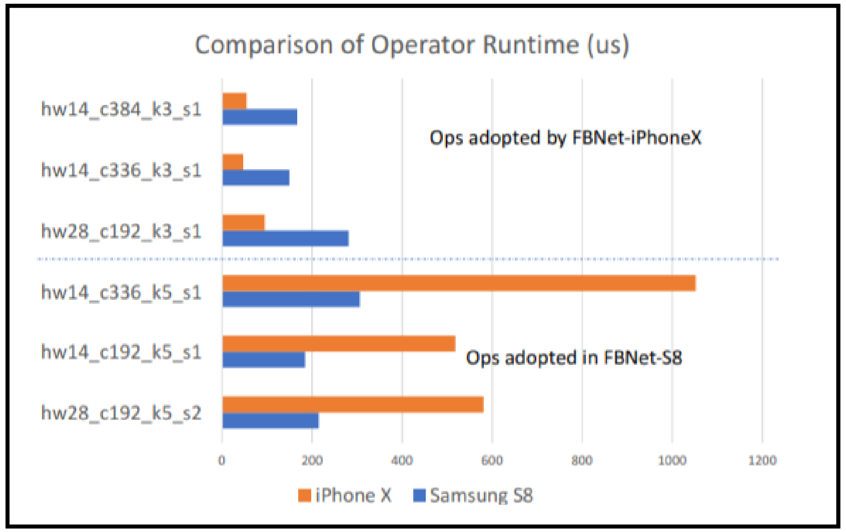
- FBNet example: NN for iPhoneX & NN for Samsung S8
  - Similar #params & accuracy
  - Different latency when swapping platforms



Model	#Parameters	#FLOPs	Latency on iPhone X	Latency on Samsung S8	Top-1 acc (%)
FBNet-iPhoneX	4.47M	322M	19.84 ms (target)	23.33 ms	73.20
FBNet-S8	4.43M	293M	27.53 ms	22.12 ms (target)	73.27

Table 5. FBNets searched for different devices.

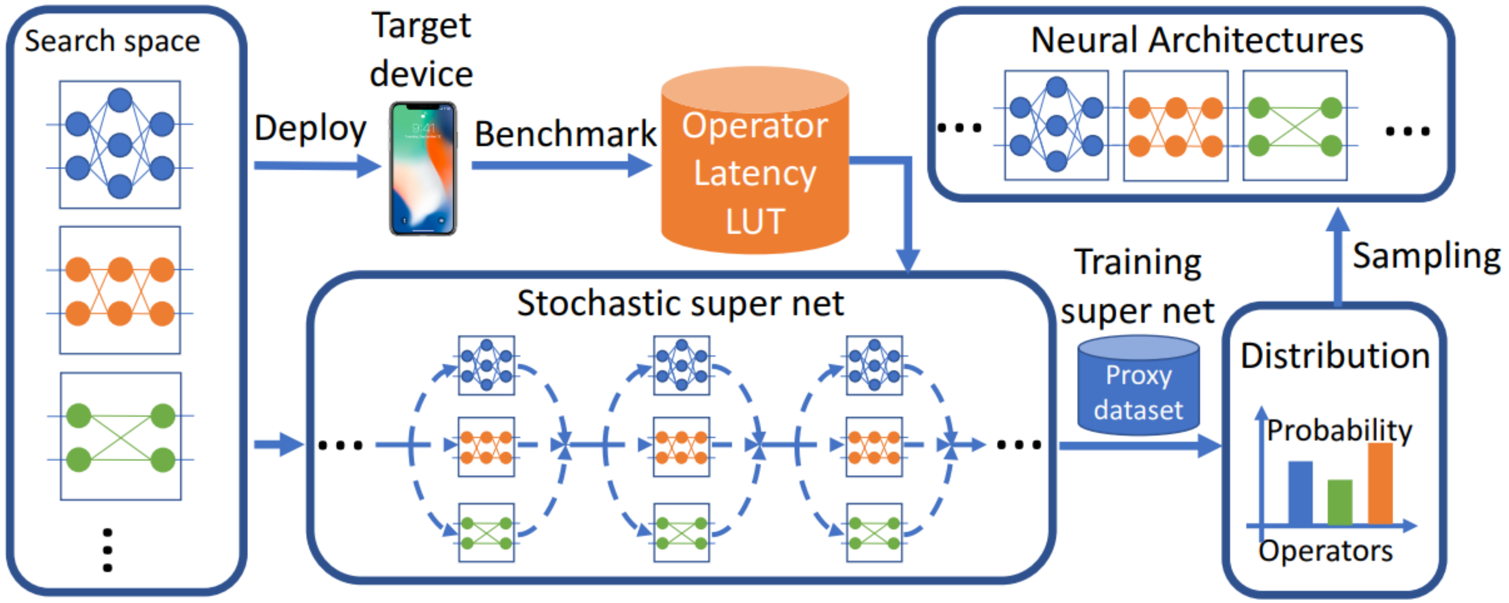




Wu et al., 2019, FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search

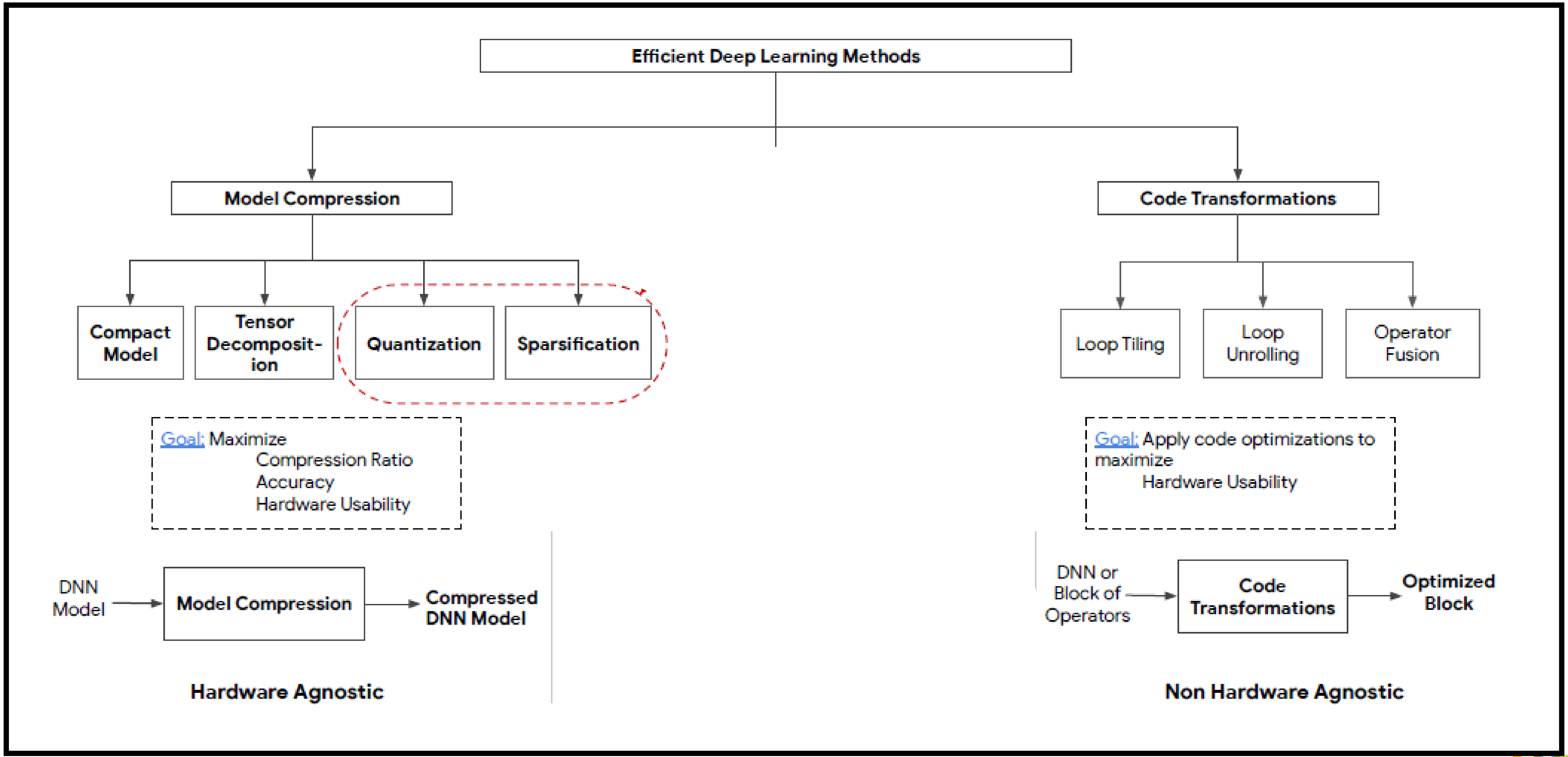
HARDWARE-AWARE NEURAL ARCHITECTURE SEARCH

- Incorporate platform-specific deployment cost (latency) in architecture design
- Exploit unique strengths of embedded platform
- Example of FBNet:
  - Measure latency of building blocks on target platform
  - Do multi-objective NAS: optimize both task-accuracy & platform latency

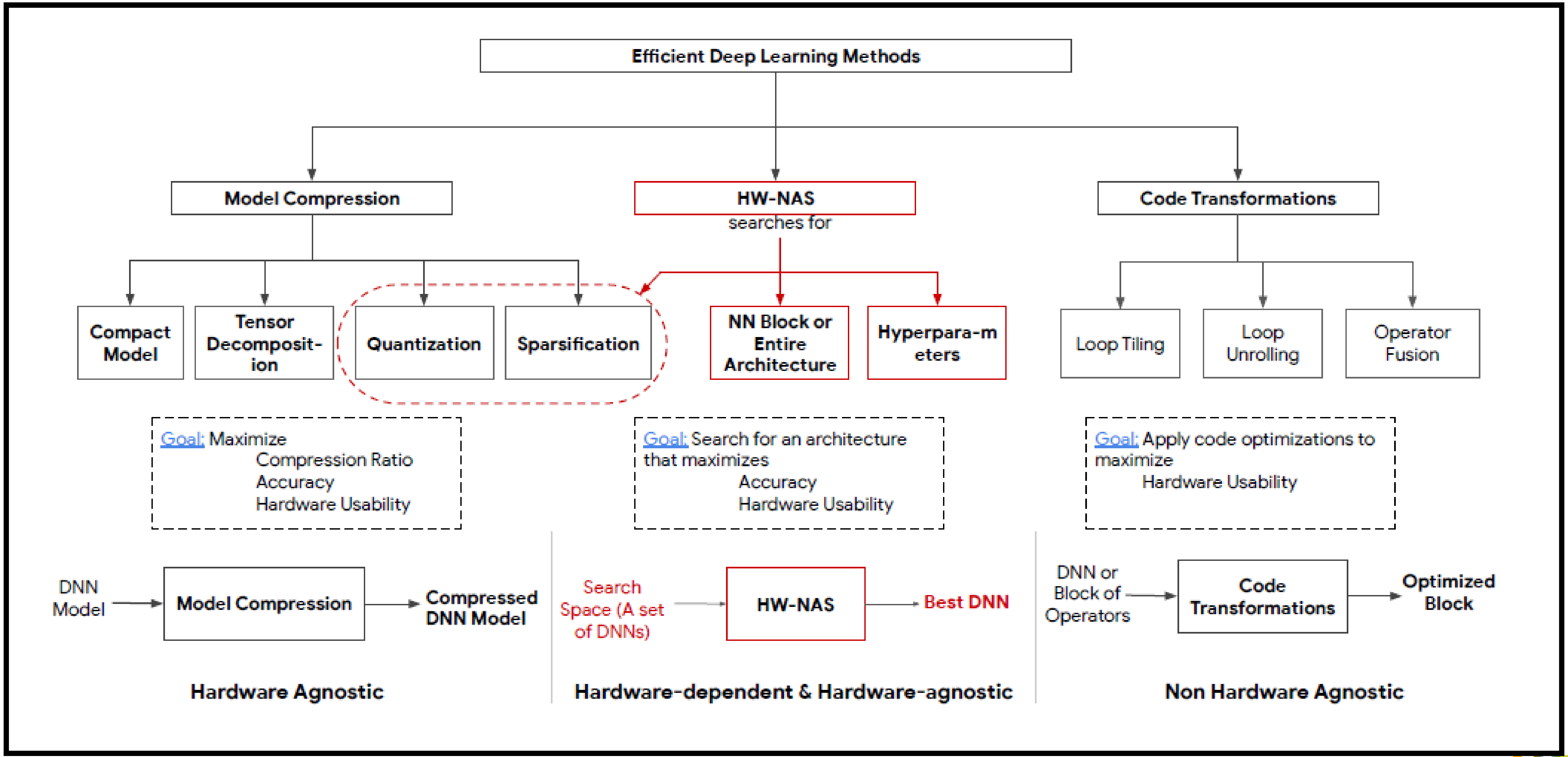


Wu et al., 2019, FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search

HARDWARE-AWARE NAS IS A METHOD FOR EFFICIENT DEEP LEARNING



HARDWARE-AWARE NAS IS A METHOD FOR EFFICIENT DEEP LEARNING

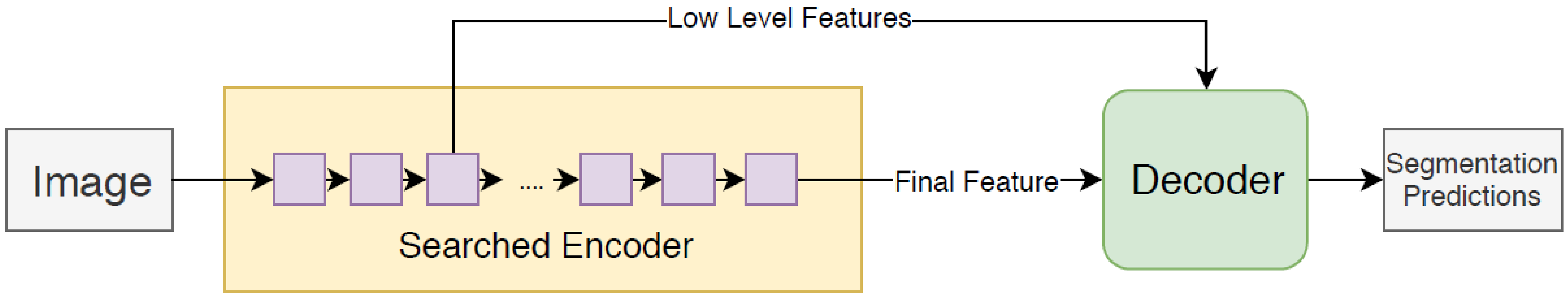

Efficient Deep Learning Methods
Model Compression
HW-NAS
searches for
Code Transformations
Compact Model
Tensor Decomposit-ion
Quantization
Sparsification
NN Block or Entire Architecture
Hyperpara-meters
Loop Tiling
Loop Unrolling
Operator Fusion
Goal: Maximize Compression Ratio Accuracy Hardware Usability
Goal: Search for an architecture that maximizes Accuracy Hardware Usability
Goal: Apply code optimizations to maximize Hardware Usability
DNN Model
Model Compression
Compressed DNN Model
Search Space (A set of DNNs)
HW-NAS
Best DNN
DNN or Block of Operators
Code Transformations
Optimized Block
Hardware Agnostic
Hardware-dependent & Hardware-agnostic
Non Hardware Agnostic

Hardware-aware Neural Architecture Search: example

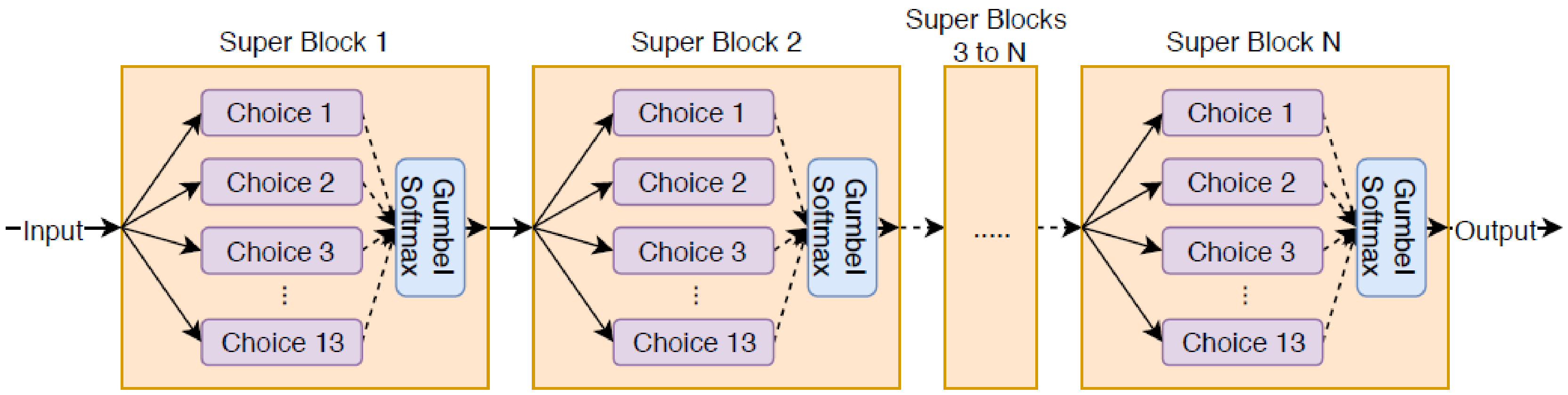
SECURE CONNECTIONS
FOR A SMARTER WORLD

EXAMPLE: STRATEGY FROM SQUEEZENAS (CONCEPT)

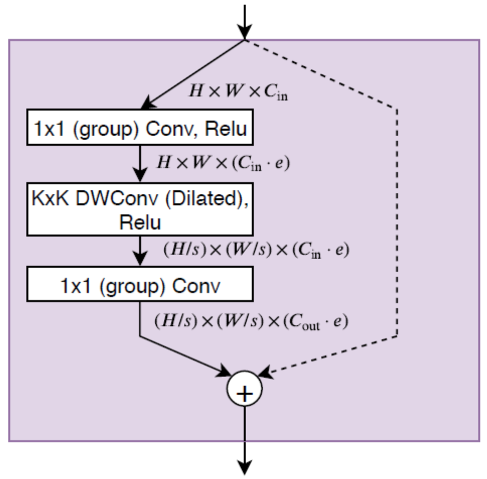
1. Concept



2. Search space

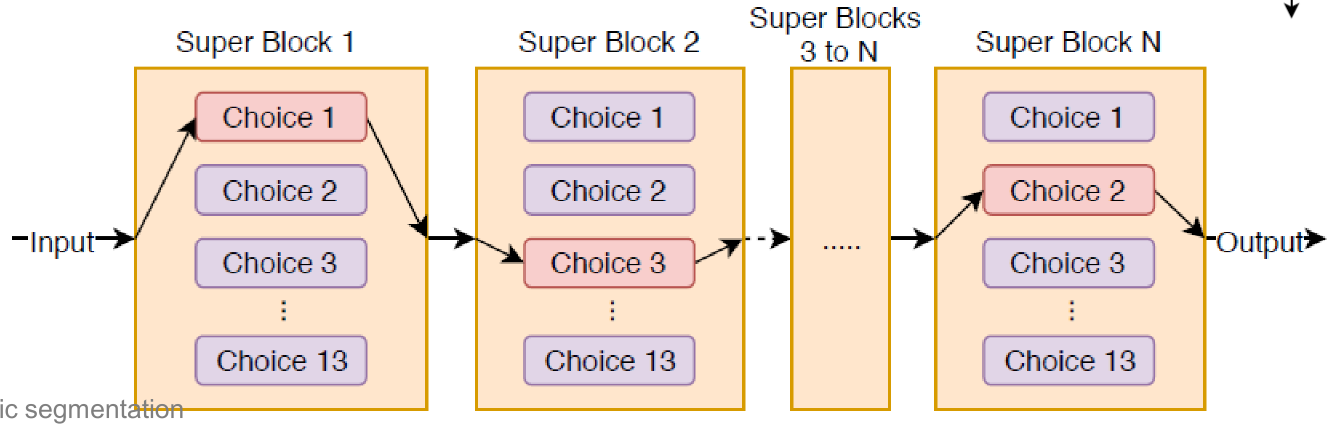


Choice 1…13 are different versions of a parametrized module:



$H \times W \times C_{in}$
1x1 (group) Conv, Relu
$H \times W \times (C_{in} \cdot e)$
KxK DWConv (Dilated), Relu
$(H/s) \times (W/s) \times (C_{in} \cdot e)$
1x1 (group) Conv
$(H/s) \times (W/s) \times (C_{out} \cdot e)$
+

3. Sample optimal realization architecture for inference



EXAMPLE: STRATEGY FROM SQUEEZENAS (METHOD)

- More formally:
  - Cast NAS as a path-selection problem within a stochastic supernetwork
  - Optimize the loss

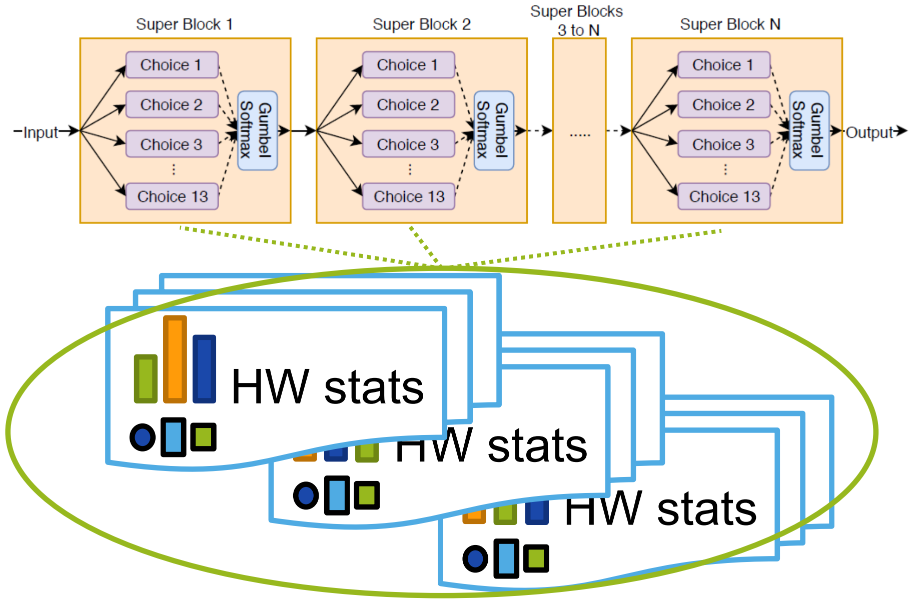
$$L(\theta, w) = L_P(\theta, w) + \alpha * L_E(\theta)$$

Architecture parameters

Conv. weights

Problem-specific loss

(Weighted) Resource-specific loss



  - Where

$$L_E(\theta) = \sum_{j}^{N} \sum_{i}^{13} p(i,j|\theta_i) C(i,j)$$

  - And C(i,j) contains the <u>cost-of-interest</u> to use candidate module 'i' as block 'j' in the network
  - Cost-of-interest?
    - MACs, latency (estimated or measured from specialized profiling effort), memory, etc.
    - Can be independent, weighted, correlated, etc.
    - *In SqueezeNAS: assumed that blocks are executed independently & consecutively*

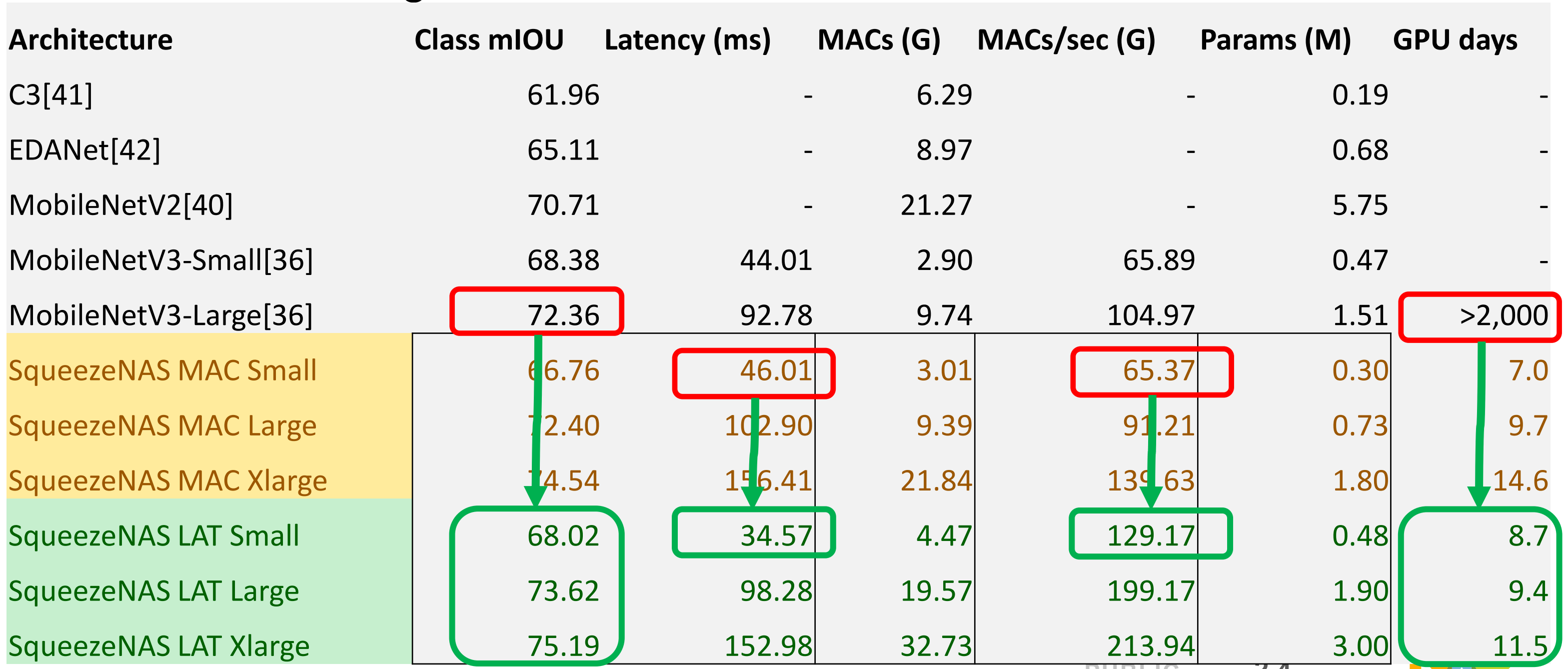
EXAMPLE: STRATEGY FROM SQUEEZENAS (KEY FINDINGS)

– Similar or better accuracy

- Orders of magnitude faster convergence
  (7-15 GPU days versus 1000s for MobileNetv3)

– HW-aware Latency-optimized networks are **faster architectures**, even **while they have more MACs** than MAC-optimized networks

- Reducing MACs is just a proxy for efficiency
- HA-NAS can develop richer models that better utilize target HW

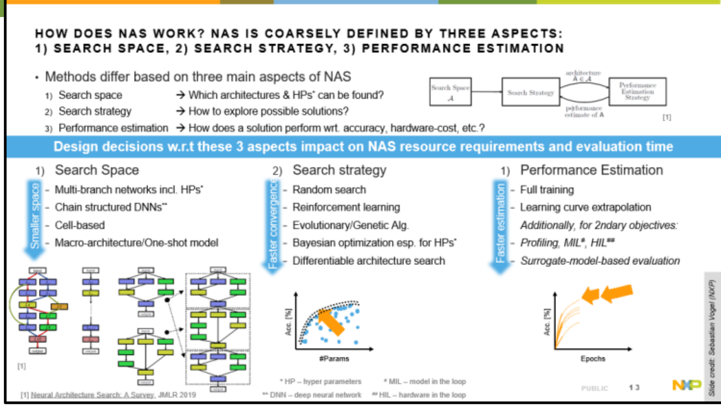


Architecture	Class mIOU	Latency (ms)	MACs (G)	MACs/sec (G)	Params (M)	GPU days
C3[41]	61.96	-	6.29	-	0.19	-
EDANet[42]	65.11	-	8.97	-	0.68	-
MobileNetV2[40]	70.71	-	21.27	-	5.75	-
MobileNetV3-Small[36]	68.38	44.01	2.90	65.89	0.47	-
MobileNetV3-Large[36]	72.36	92.78	9.74	104.97	1.51	>2,000
SqueezeNAS MAC Small	66.76	46.01	3.01	65.37	0.30	7.0
SqueezeNAS MAC Large	72.40	102.90	9.39	91.21	0.73	9.7
SqueezeNAS MAC Xlarge	74.54	156.41	21.84	139.63	1.80	14.6
SqueezeNAS LAT Small	68.02	34.57	4.47	129.17	0.48	8.7
SqueezeNAS LAT Large	73.62	98.28	19.57	199.17	1.90	9.4
SqueezeNAS LAT Xlarge	75.19	152.98	32.73	213.94	3.00	11.5

Hardware-aware Neural Architecture Search: there is way more…

SECURE CONNECTIONS
FOR A SMARTER WORLD

HARDWARE-AWARENESS ADDS MORE COMPLEXITY TO NAS


HOW DOES NAS WORK? NAS IS COARSELY DEFINED BY THREE ASPECTS:
1) SEARCH SPACE, 2) SEARCH STRATEGY, 3) PERFORMANCE ESTIMATION

- Methods differ based on three main aspects of NAS
  1) Search space → Which architectures & HPs* can be found?
  2) Search strategy → How to explore possible solutions?
  3) Performance estimation → How does a solution perform wrt. accuracy, hardware-cost, etc.?

Design decisions w.r.t these 3 aspects impact on NAS resource requirements and evaluation time

1) Search Space
- Multi-branch networks incl. HPs*
- Chain structured DNNs**
- Cell-based
- Macro-architecture/One-shot model

2) Search strategy
- Random search
- Reinforcement learning
- Evolutionary/Genetic Alg.
- Bayesian optimization esp. for HPs*
- Differentiable architecture search

1) Performance Estimation
- Full training
- Learning curve extrapolation

Additionally, for 2ndary objectives:
- *Profiling, MIL#, HIL##*
- *Surrogate-model-based evaluation*

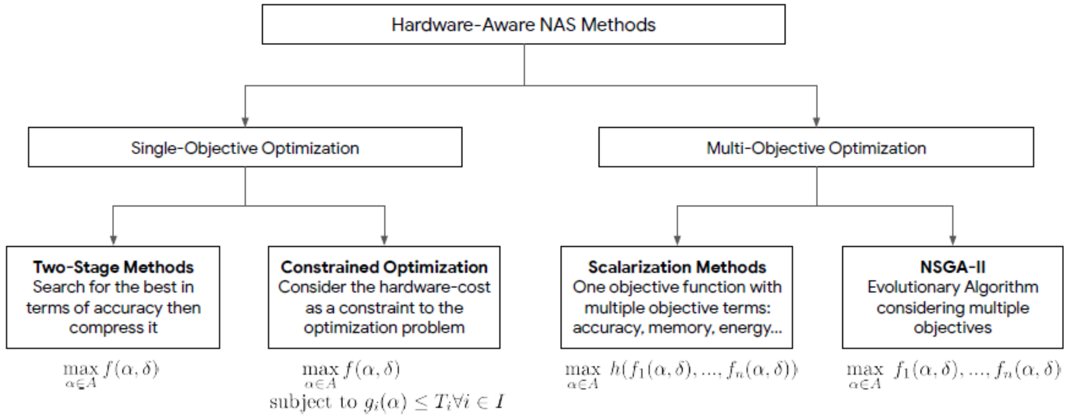
* HP – hyper parameters
** DNN – deep neural network
# MIL – model in the loop
## HIL – hardware in the loop

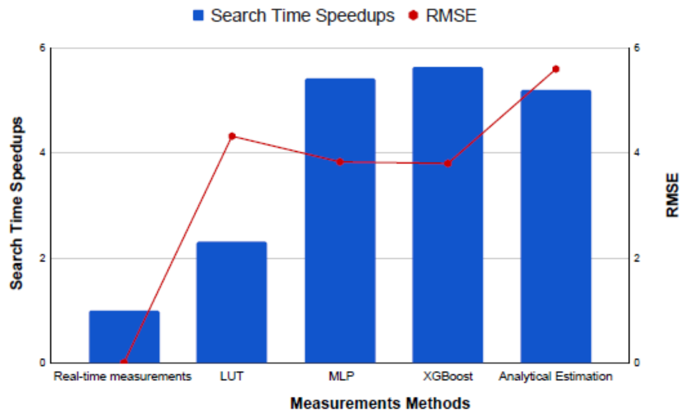
[1] Neural Architecture Search: A Survey, JMLR 2019

PUBLIC 13

Slide credit: Sebastian Vogel (NXP)

- Key impact of hardware-awareness on the three elements of NAS
  - Search space
    - Which graph structures and which building blocks are feasible and supported on HW?
  - Search Strategy
    - How to guide search by the additional deployment cost, which is a conflicting 'landscape' with task-accuracy?
  - Performance Estimation
    - How to assess deployment cost in a scalable manner?
      - HiL: accurate, but slow
      - LUT: fast and accurate, but constrains search to a fixed set of options
      - Model: fast and 'unconstrained', but additional problem to solve.
- And… is this enough?
  - How about optimizing also optimizing
    - Inference engine & memory scheduling (e.g. MCUNet)
    - Operation definitions (e.g. XD-operations)


Hardware-Aware NAS Methods
Single-Objective Optimization
Multi-Objective Optimization
Two-Stage Methods
Search for the best in terms of accuracy then compress it
Constrained Optimization
Consider the hardware-cost as a constraint to the optimization problem
Scalarization Methods
One objective function with multiple objective terms: accuracy, memory, energy...
NSGA-II
Evolutionary Algorithm considering multiple objectives
$\max_{\alpha \in A} f(\alpha, \delta)$
$\max_{\alpha \in A} f(\alpha, \delta)$ subject to $g_i(\alpha) \leq T_i \forall i \in I$
$\max_{\alpha \in A} h(f_1(\alpha, \delta), \ldots, f_n(\alpha, \delta))$
$\max_{\alpha \in A} f_1(\alpha, \delta), \ldots, f_n(\alpha, \delta)$



Lin et al., 2020, MCUnet: Tiny Deep Learning on IoT Devices

Roberts et al., 2021, 'Rethinking Neural Operations for Diverse Tasks'

HARDWARE-AWARE NEURAL ARCHITECTURE SEARCH – TAKE HOME MESSAGES

- Automatically design NN architectures that **jointly optimize**
  - **Accuracy** on application task (e.g. image classification, object detection)
  - and **hardware utilization** of the target inference platform
  - by taking HW and SW constraints of the latter into account in the training process.

- Three key strengths of HA-NAS as a tool for efficient deep learning:
  1. Extend design space coverage for neural networks beyond bias from human expert
  2. Provide optimal performance on a case-by-case basis (task, data, HW)
  3. Mitigate desired deployment bottlenecks directly (speed, energy, memory, compatibility, etc.)

- … however, it requires
  - Compute resources
  - Research into search space design, cost functions, modeling strategies, training re-use,…
  - … a lot to do for new researchers!

AVAILABLE STUDENT PROJECT POSITIONS (INTERNSHIP & GRADUATION PROJECTS)

- **Automatic neural network quantization and deployment optimization**
  - Optimizing quantization and pruning of neural networks and deployment SW parameterization via NAS
- **Hardware-aware NAS for next generation radar-based ADAS**
  - Improving object classification and tailoring NAS (search space, strategy, operations, evaluation) for Radar data
- **Transferring existing NAS methodologies to challenging embedded system tasks**
  - E.g. targeting audio processing, battery management, predictive maintenance, etc.
- **Intelligent automated design & configuration of next generation DL-HW-accelerators**
  - Co-optimizing HW accelerators and neural architectures (i.e. for quantizartion and sparsity).
- **Hardware-aware NAS for next generation hardware and software**
  - Enabling hardware targets, NAS fraemworks, benchmarking training modalities
- Feel free to reach out! (note: limited spots available…)
  willem.sanberg@nxp.com / sebastian.vogel@nxp.com

AVAILABLE STUDENT PROJECT POSITIONS (INTERNSHIP & GRADUATION PROJECTS)

- **Automatic neural network quantization and deployment optimization**
  - optimizing neural networks through quantization and pruning
  - taking multiple optimization criteria into account
  - investigating options to learn how to quantize/prune neural networks
  - automatically determining optimal SW deployment parameterizations for embedded devices

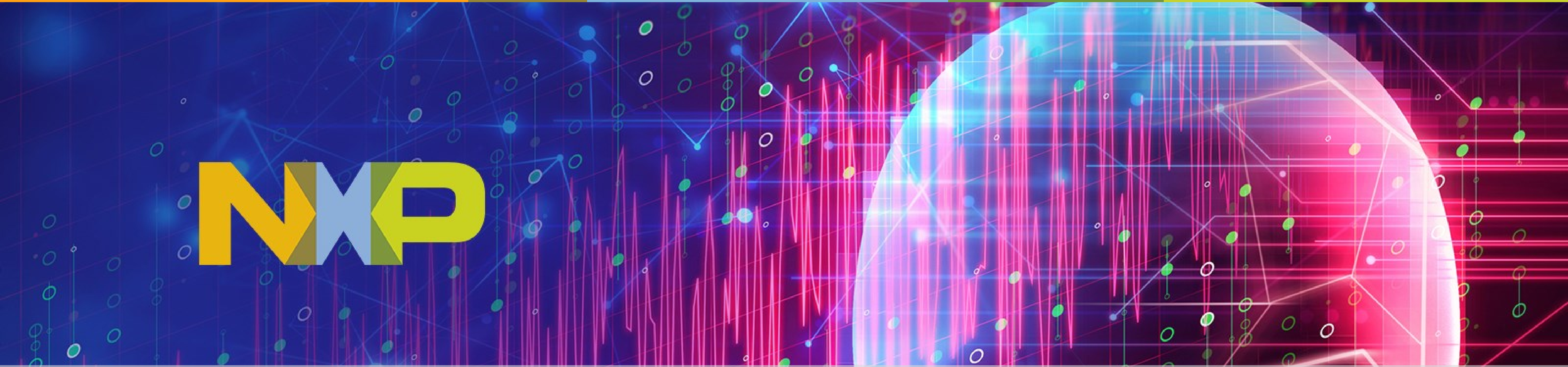
- **Hardware-aware NAS for next generation radar-based ADAS**
  - improving state of the art approaches on object classification with DNNs
  - leveraging ML and NN-design know-how from other domains for Radar signal processing
  - exploring NN designs that exploit Radar spectrum data, Radar target lists or a fusion of both
  - optimizing simultaneously the deployment properties on target hardware and the task accuracy

AVAILABLE STUDENT PROJECT POSITIONS (INTERNSHIP & GRADUATION PROJECTS)

- **Transferring existing NAS methodologies to challenging embedded system tasks**
  - audio processing (noise cancelation, keyword spotting, etc.)
  - battery management and battery health estimation
  - predictive maintenance (e.g., anomaly detection)
  - with the goal to derive insights on the trade-off between system requirements and task accuracy

- **Intelligent automated design & configuration of next generation DL-HW-accelerators**
  - automatically optimizing configurable HW accelerators and co-adapting neural architectures
  - especially focusing on quantization and sparsity features of HW-accelerators

- **Hardware-aware NAS for next generation hardware and software**
  - extending available hardware-aware NAS frameworks to new hardware targets;
  - integrating said NAS frameworks with one of our existing training modalities;
  - conducting extensive experiments in our training modalities.



References

- Benmeziane et al., A Comprehensive Survey on Hardware-Aware Neural Architecture Search, ACM J. Comp. Surv. 2021
- Chen et al., FasterSeg: Searching for Faster Real-time Semantic Segmentation, ICLR 2020
- Cozma, et al., DeepHybrid: Deep Learning on Automotive Radar Spectra and Reflections for Object Classification, ITSC 2021
- Elsken et al., Simple and Efficient Architecture Search for CNNs, ICLRws 2018
- He et al., Deep Residual Learning for Image Recognition, CVPR 2016
- Jin et al., Auto-Keras: An Efficient Neural Architecture Search System, ACM SIGKDD 2019
- Lin et al., MCUnet: Tiny Deep Learning on IoT Devices, NIPS 2020
- Ouaknine et al., Multi-View Radar Semantic Segmentation, ICCV 2021
- Real et al., Large-scale evolution of Image Classifiers, ICML 2017
- Roberts et al., Rethinking Neural Operations for Diverse Tasks, NIPS 2021
- Shaw et al., SqueezeNAS: Fast neural architecture search for faster semantic segmentation, CVPRws 2019
- Simonyan et al., Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2014
- Tan & Le, EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, ICML 2020
- Wu et al., FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search, CVPR 2019
- Wu et al., Mixed Precision Quantization in Differentiable NAS, arXiv 2018
- Wang, et al. HAQ: Hardware-Aware Automated Quantization with Mixed Precision, CVPR 2019
- Xie et al., Exploring Randomly Wired Neural Networks for Image Recognition, ICCV 2019
- Zoph et al., Learning Transferable Architectures for Scalable Image Recognition, CVPR 2018
- http://www.ml4aad.org/automl/literature-on-neural-architecture-search/

SECURE CONNECTIONS
FOR A SMARTER WORLD