Leveraging Neural Architecture Search for Efficient Computer Vision at the Edge

Hiram Rayo Torres Rodriguez

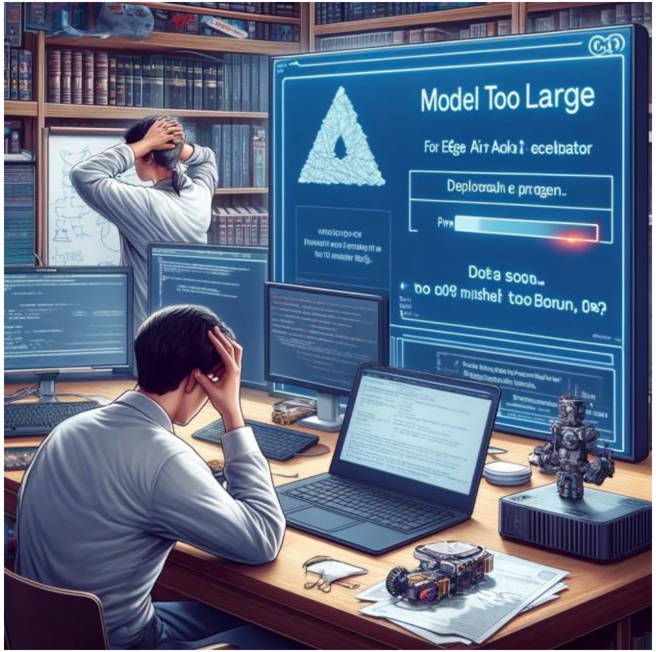
Senior Embedded AI Research Engineer

NXP Semiconductors

Efficiently deploying AI models on embedded devices can be challenging

- Hardware deployment is typically not considered when designing AI models
- **Common problems:**
  - Sub-optimal real-time performance and/or model does not fit on device
    - **Even after applying common NN optimizations (e.g., quantization and/or pruning)**

What to do when common NN optimizations (e.g., quantization and/or pruning) are not sufficient?

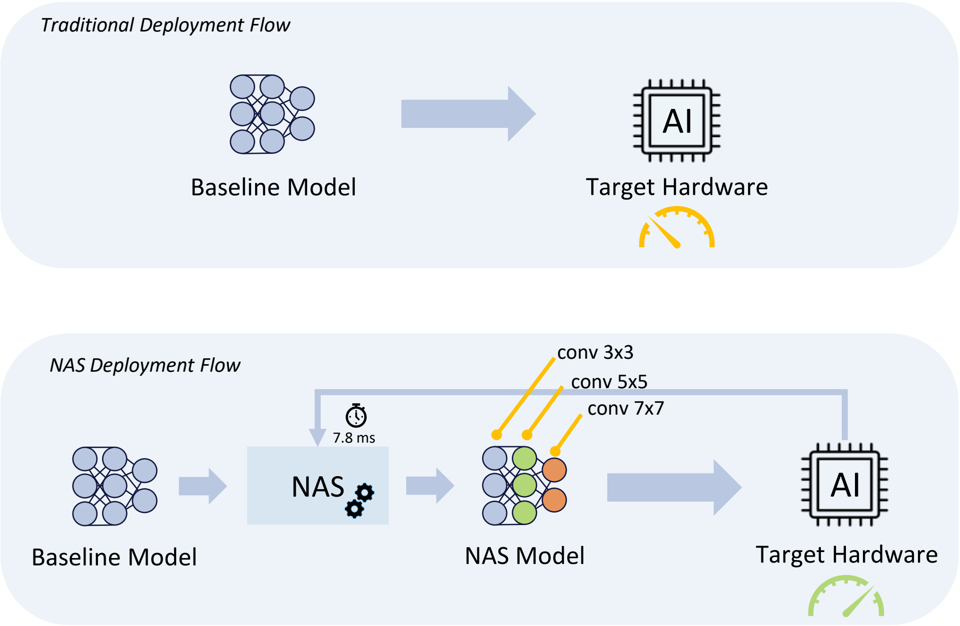


*

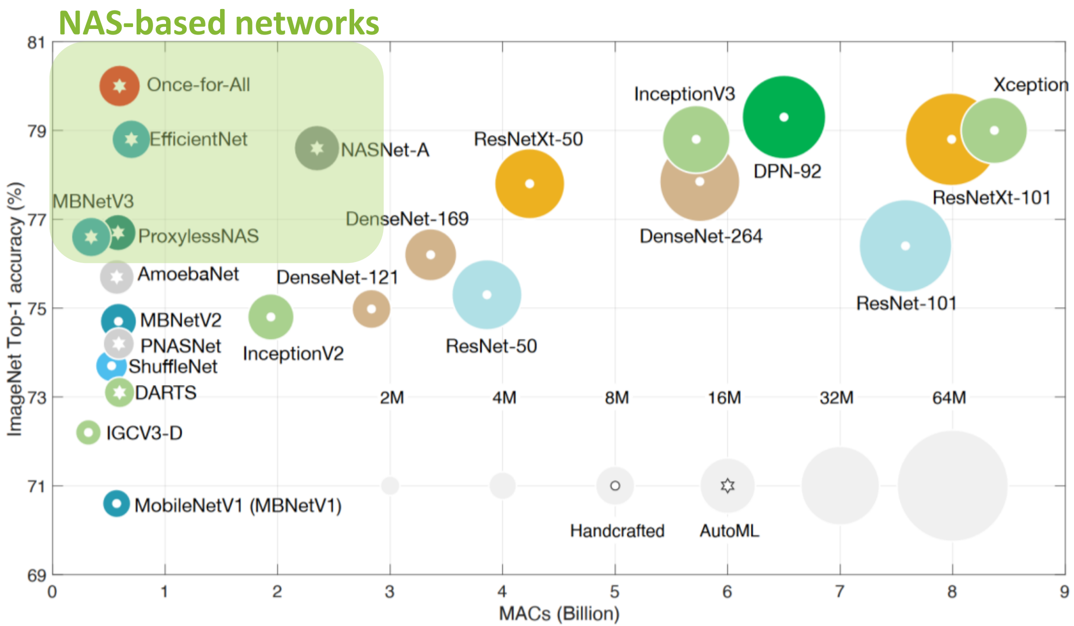
Figure generated by DALL-E3

Neural Architecture Search (NAS) can derive edge-ready models automatically

- Neural Architecture Search (NAS) can derive highly efficient edge-ready models automatically:
  - **Optimized for multiple objectives** (e.g., task performance, and hardware-related metrics)
  - **Considering deployment aspects** during the search process (e.g., efficiency of quantized operators)



NAS outperforms manually designed NN architectures

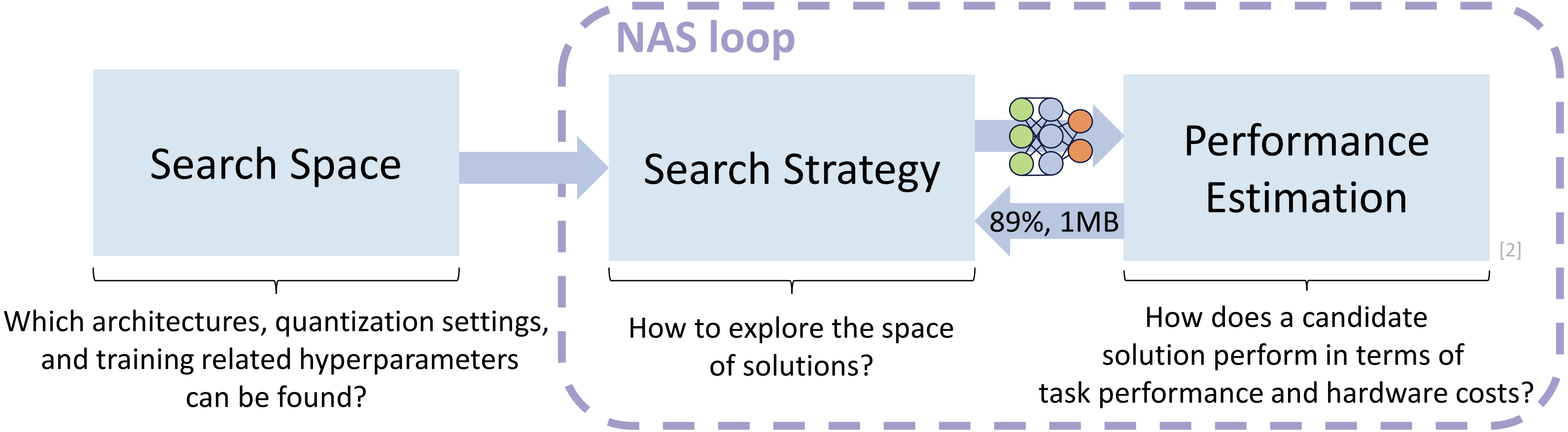


[1]

NAS has become the de facto approach for NN design, as it can find NN architectures that outperform manual designs in an automated manner

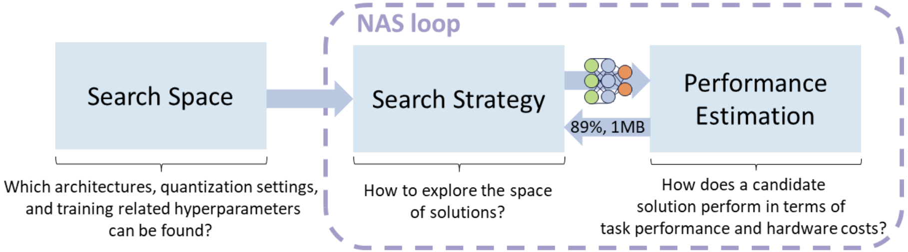
How does NAS work?

- NAS is coarsely defined by three aspects:
    1. Search space
    2. Search strategy
    3. Performance estimation



How does NAS work?

- NAS is coarsely defined by three aspects:
    1. Search space
    2. Search strategy
    3. Performance estimation



[2]

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

Search Space

Smaller space

- ➢ Multi-branch networks
- ➢ Chain structured DNNs
- ➢ Cell-based

Search Strategy

Faster Convergence

- ➢ Random search
- ➢ Evolutionary Algorithms
- ➢ Bayesian optimization

Performance Estimation

Faster Estimation

- ➢ Task performance and hardware-related cost:
    - ➢ Full training
    - ➢ HIL*
    - ➢ Zero cost proxies
    - ➢ Surrogate models

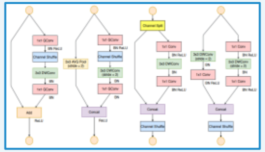
*Hardware-in-the-loop

NAS can be computationally expensive, so how to approach NAS in a scalable manner?

How to approach NAS in a scalable manner?
Demo application of NAS: real-time person detection

Given a CNN for **real-time person detection**[3]

PASCAL2 Pattern Analysis, Statistical Modelling and Computational Learning [4]


[5]

Person Detector (ShuffleNetV2[5] based)

Reduce inference latency when deployed on edge hardware **without degrading performance**

PASCAL2 Pattern Analysis, Statistical Modelling and Computational Learning [4]

Person Detector (INT8)
Baseline: 5.95 FPS

NAS

Person Detector (INT8)
Target: 10.0 FPS

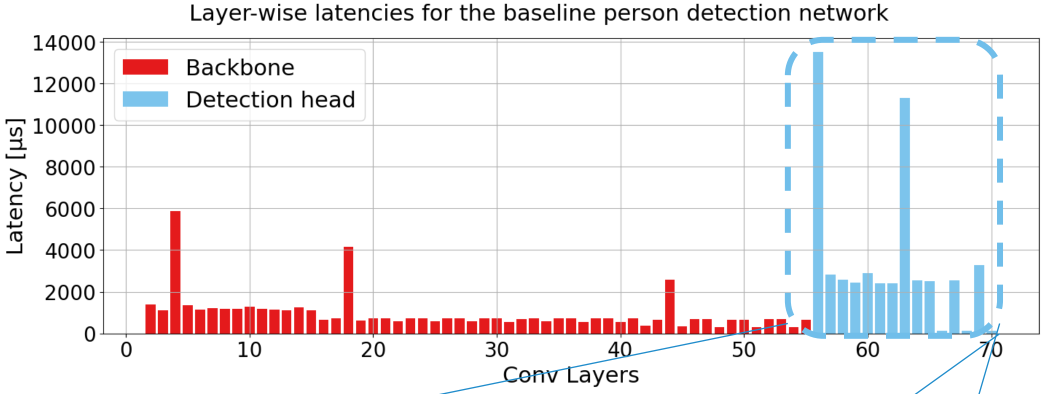
AI
Target Hardware

How to approach NAS in a scalable manner?
Design the search space looking at layer-wise statistics

Search Space

Which parts of the network contribute the most towards latency?

- **Idea:** Focus first on optimizing performance bottlenecks



Parameter	Baseline	Options
Kernel size	5	{3, 5}
# Groups	1	{1, 2, 4, 8}
# Channels	96	[24, 96]
Img. width	320	[220, 320]

How to approach NAS in a scalable manner?
Select the search strategy based on the search space size

embedded VISION SUMMIT

Search Space

Which parts of the network contribute the most towards latency?

- **Idea:** focus first on optimizing performance bottlenecks

- **Detection head search space:**

Parameter	Baseline	Options
Kernel size	5	{3, 5}
# Groups	1	{1, 2, 4, 8}
# Channels	96	[24, 96]
Img. width	320	[220, 320]

Search Strategy

Which search strategy can adequately explore the search space?

- **Idea:** Select based on the size of the search space
  - Given the relatively large space, rely on a more "sophisticated" approach: **Bayesian optimization**

How to approach NAS in a scalable manner?
Select perf. estimation based on the search compute budget

embedded VISION SUMMIT

Search Space

Which parts of the network contribute the most towards latency?

- **Idea:** focus first on optimizing performance bottlenecks

- **Detection head search space:**

Parameter	Baseline	Options
Kernel size	5	{3, 5}
# Groups	1	{1, 2, 4, 8}
# Channels	96	[24, 96]
Img. width	320	[220, 320]

Search Strategy

Which search strategy can adequately explore the search space?

- **Idea:** select based on the size of the search space
  - Given the relatively large space, rely on a more "sophisticated" approach: **Bayesian optimization**

Performance Estimation

Which strategy can address my compute budget?

- **Idea:** Use the time it takes to train a single network as a reference to estimate the search time for *N* trials and select based on this.

- **Example for demo application:**
  - One network → ~12 min.
  - 100 trials → ~2.5 GPU days‡
  - If 2.5 days is within compute budget, **full training can be a good solution.**
  - **Hardware-related cost:** Inference latency via HIL*

‡ GPU day = # GPUs x Wall clock days

*Hardware-in-the-loop

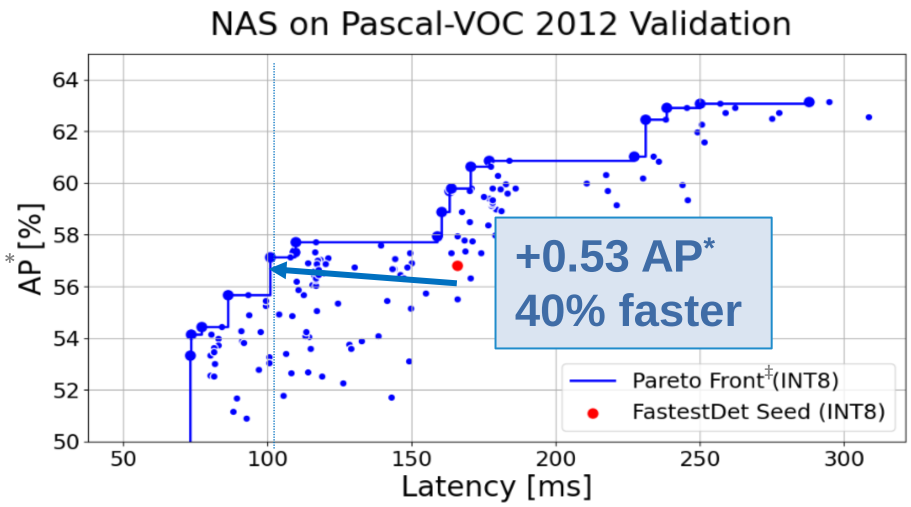
NAS can achieve substantial efficiency improvements without compromising task performance

NAS tool:

- Optuna[6]

Search time:

- ~2.5 GPU days‡ (100 trials)



NAS reduces inference latency by 40% while keeping similar task performance compared to the baseline seed network

‡ GPU day = # GPUs x Wall clock days

‡ Pareto front: best trade-off between conflicting objectives

*AP: Average Precision [@0.5 IoU]

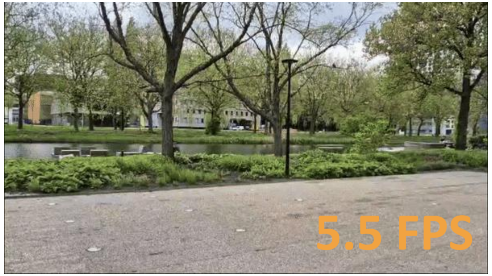
NAS can achieve substantial efficiency improvements without compromising task performance

NAS tool:

- Optuna[6]

Search time:

- ~2.5 GPU days‡ (100 trials)



Baseline Model

NAS Model

NAS reduces inference latency by 40% while keeping similar task performance compared to the baseline seed network

‡ GPU day = # GPUs x Wall clock days

How to approach NAS in a scalable manner?
Select perf. estimation based on the search compute budget

embedded VISION SUMMIT

Search Space

Which part of the network to optimize to reduce the inferen

- **Idea:** bottle
  - #FL (hig
- **Detec space:**

Parameter	Baseline	Options
Kernel size	5	{3, 5}
# Groups	1	{1, 2, 4, 8}
# Channels	96	[24, 96]
Img. width	320	[220, 320]

Search Strategy

Which search strategy can adequately explore the

ed on the ch space

ce, rely on a ated"

i-objective Tree Parzen Estimation

- For smaller search spaces, random search may be sufficient!

What if I don't have this compute budget, or baseline training is substantially higher for my use case?

Performance Estimation

Which strategy can address my compute budget?

- **Idea:** Use the time it takes to train a single network as a reference to estimate the search time for *N* trials and select based on this.
- **Example for demo application:**
  - One network → ~12 min.
  - 100 trials → ~2.5 GPU days‡
  - If 2.5 days is within compute budget, **full training can be a good solution.**
  - **Hardware-related cost:** Inference latency via HIL*

How to approach NAS in a scalable manner?
Improving NAS scalability via efficient perf. estimation

embedded VISION SUMMIT

Performance Estimation

Which strategy can address my compute budget?

If little compute budget is available:

Idea: Rely on **low-fidelity estimates**[7]

- **Challenge:** How to select one?

Low-fidelity estimates

Learning-curve Methods
(e.g., early stopping)

- Can be sensitive to # epochs

Model-based Predictors
(e.g., XGBoost)

- May require many training samples

Zero-Cost Proxies
(e.g., # FLOPs#)

- Many to pick from + wildly different correlations depending on the task[7]

FLOP: Floating Point Operation

How to approach NAS in a scalable manner?
Improving NAS scalability via efficient perf. estimation

embedded VISION SUMMIT

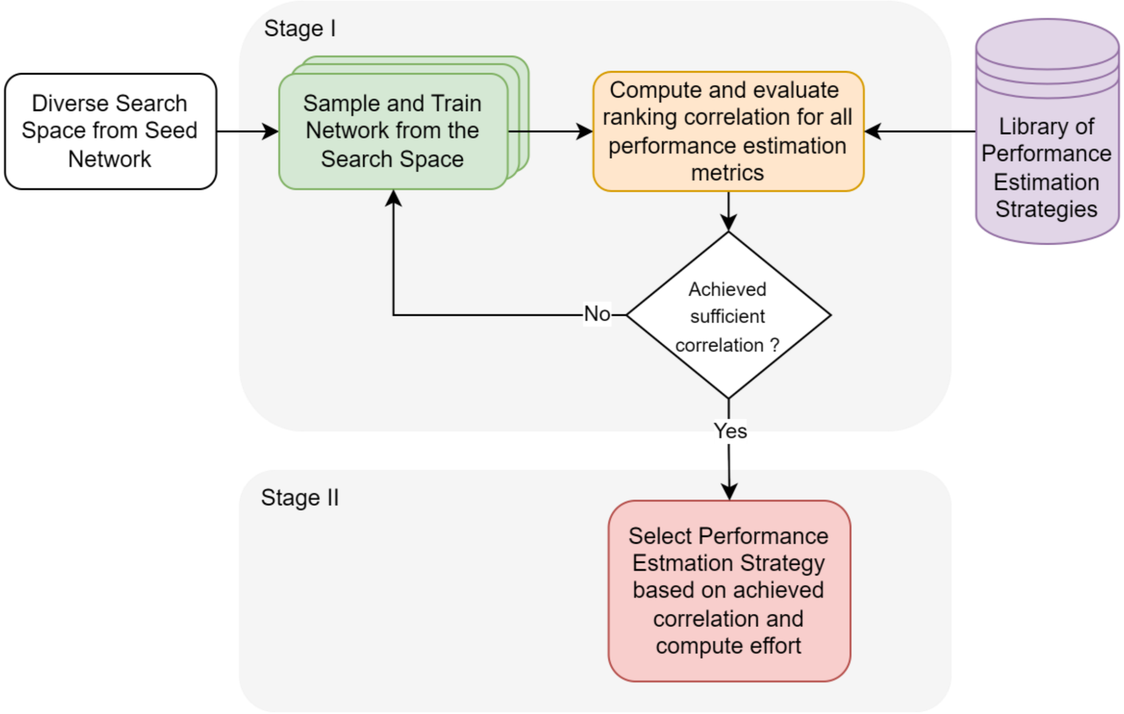
Performance Estimation

Which strategy can address my compute budget?

If little compute budget is available:

Idea: Rely on **low-fidelity estimates**[7]

- **Challenge:**
  How to select one?
- **Solution:**
  Two-stage approach for performance estimation strategy selection



Two-stage approach for performance estimation strategy selection

NXP

Efficient performance estimation can substantially improve NAS scalability

embedded VISION SUMMIT

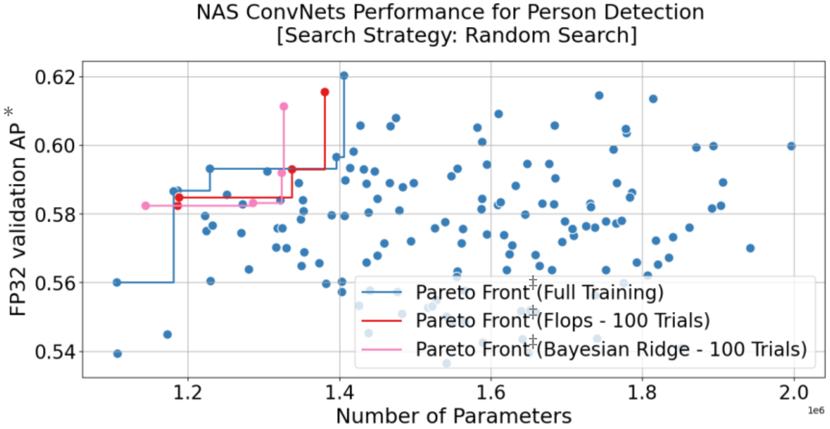
Performance Estimation

Out of **24 estimation strategies, Bayesian Ridge ($\tau = 0.52$) and # FLOPs# ($\tau = 0.5$) achieve the highest correlation on this use case**

- **~40 training samples are sufficient** to make an informed selection

Performing NAS using the above strategies, we achieve **competitive performance compared to full training while substantially speeding up search time**

- Note that the reported speedup already considers the time required to select the perf. estimation strategies



	Search Time (Seconds)	Search Speedup	Post Search Training Time (Seconds)	Total Search Time (seconds)	Overall Speedup
Full training	96,000	1.0	N/A	96,000 (26.6 Hours)	1.0
Bayesian Ridge	30	**x3,200**	6,400	6,430 (1.78 Hours)	**x14.93**
FLOPs#	6	**x16,000**	9,600	9,606 (2.66 Hours)	**x10**

FLOP: Floating Point Operation

‡ Pareto front: best trade-off between conflicting objectives

*AP: Average Precision [@0.5 IoU]

Let’s wrap-up: Some insights and takeaways

embedded VISION SUMMIT

Search Space Design

- Focus first on the performance bottlenecks:
  - Focused searches can be a way to leverage the power of NAS while keeping compute tractable

Search Strategy Selection

- Consider the search space size:
  - Large search spaces can benefit from “sophisticated” approaches. However, random search may be sufficient for small ones

Performance Estimation Strategy Selection

- Consider the time it takes to train the baseline network:
  - Depending on your compute budget, there may be no need for “sophisticated” performance estimation techniques if training a single network is cheap
  - Efficient performance estimation can unlock substantial speedups when compute budget is limited

NXP

Resources

NXP @ 2024 Embedded Vision Summit

Enabling Technologies Session:

- Efficiency Unleashed: The Next-Gen NXP i.MX 95 Applications Processor for Embedded Vision (Thursday, May 23rd – 12:00 PM)

See us at the NXP booth (503)

- i.MX95 Quad Camera Object Detection Demo
- Mobile Robot Buggy Demo
- i.MX93 Smart Fitness
- and more!

References:

- [1] H. Cai, et al., "Once-for-all: Train One Network and Specialize it for Efficient Deployment", ICLR '20
- [2] T. Elsken, et al., "Neural Architecture Search: A Survey", JMLR '19
- [3] https://github.com/dog-qiuqiu/FastestDet
- [4] M. Everingham, et al., "The PASCAL Visual Object Classes (VOC) Challenge", IJCV '10
- [5] N. Ma, et al., "Shufflenet v2: Practical guidelines for efficient cnn architecture design", ECCV '18
- [6] T. Akiba, et al., "Optuna: A next-generation hyperparameter optimization framework," KDD '19
- [7] C. White, et al., "How Powerful are Performance Predictors in Neural Architecture Search", NIPS '21

NXP Semiconductors AI/ML:

- NXP Semiconductors Edge AI Portfolio
- NXP eIQ ML Software Development Environment