# Quantization-Aware Neural Architecture Search: Towards Efficient Semantic Segmentation on Edge Devices


Hiram Rayo Torres Rodriguez✉, Sebastian Vogel, Nick van de Waterlaat, Willem Sanberg, Gerardo Daalderop

NXP Semiconductors, Eindhoven, The Netherlands.

✉hiram.rayotorresrodriguez@nxp.com


## Problem Context and Solution Proposal

- **Enabling efficient semantic segmentation on the edge is relevant** for enabling various applications (e.g., self-driving vehicles) **but designing optimal Deep Neural Networks** (DNN) is hard and **requires specialized skills.**
- For successful edge deployment, **hardware properties need to be considered during the design cycle** increasing the complexity of the design space.
- **Quantized model deployment** can enable real-time performance on the edge but **degrades task performance.**

**Method:**

- "AI designing AI": Multi-objective hardware-aware optimization via **Quantization-Aware Neural Architecture Search (QA-NAS)** for optimal semantic segmentation on the edge.

## Background and Motivation

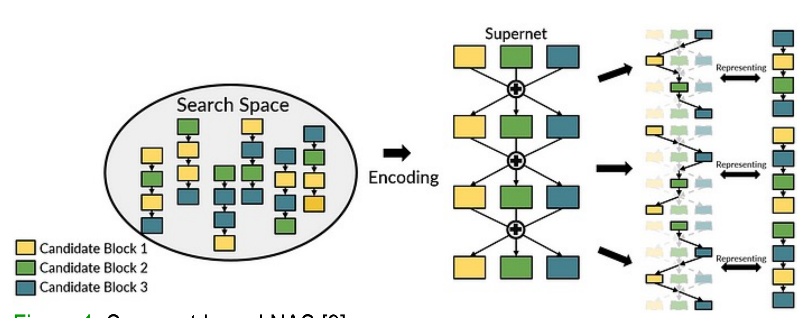


Figure 1: Supernet-based NAS [9]

- Neural Architecture Search (NAS) [1] can automatically design neural networks for multiple objectives but can be expensive.
- Weight-sharing NAS [2] speeds up NAS by using a supernet (Figure 1) but recent work:
  - Mostly focuses in improving supernet training.
  - Tackles low-scale Computer Vision (CV) tasks.
  - Disregards HW-awareness (quantization).

**Our contributions:**

- Extend a weight-sharing approach (FairNAS [3]) towards optimal semantic segmentation on Cityscapes [4] and MS-COCO [5].
- Introduce quantization-awareness into weight-sharing NAS to search for efficient **edge-ready** models.

## Search Space Design

- Searchable MobileNetV2 [6] encoder for DeeplabV3 [7]
  - Kernel sizes, expansion ratios, dilation rates, layers.
- Designed for high task performance (mIoU) and low latency

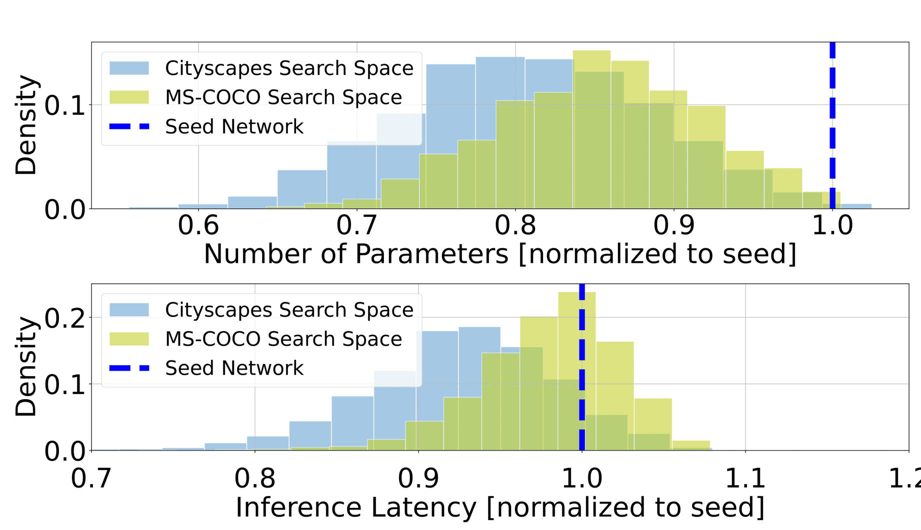


Figure 2: Number of parameters and inference latency distribution across the Cityscapes and MS-COCO search spaces.

## Quantization-Aware NAS Method

**Overview:**

1. **Supernet Training**
   - Initialize supernet with ImageNet weights [8].
   - Train supernet following FairNAS [3].
2. **Search**
   - Randomly sample subnets from supernet.
   - Evaluate quantized task performance.
   - Profile networks on the target hardware.
   - Select the Pareto optimal solutions.

Train the found solutions to full convergence.

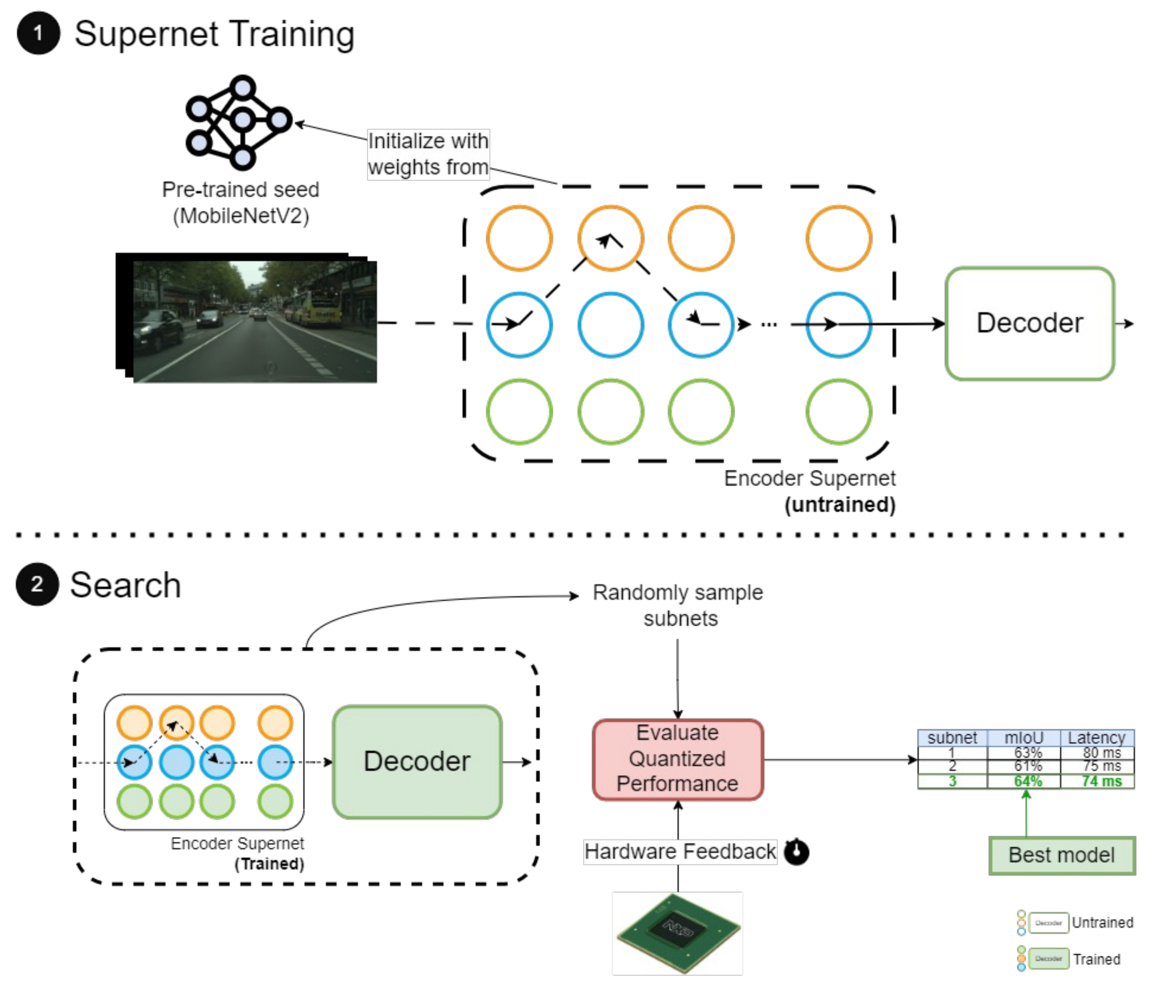


Figure 3: Quantization-Aware NAS for highly efficient semantic segmentation networks on NXP hardware.

## Results

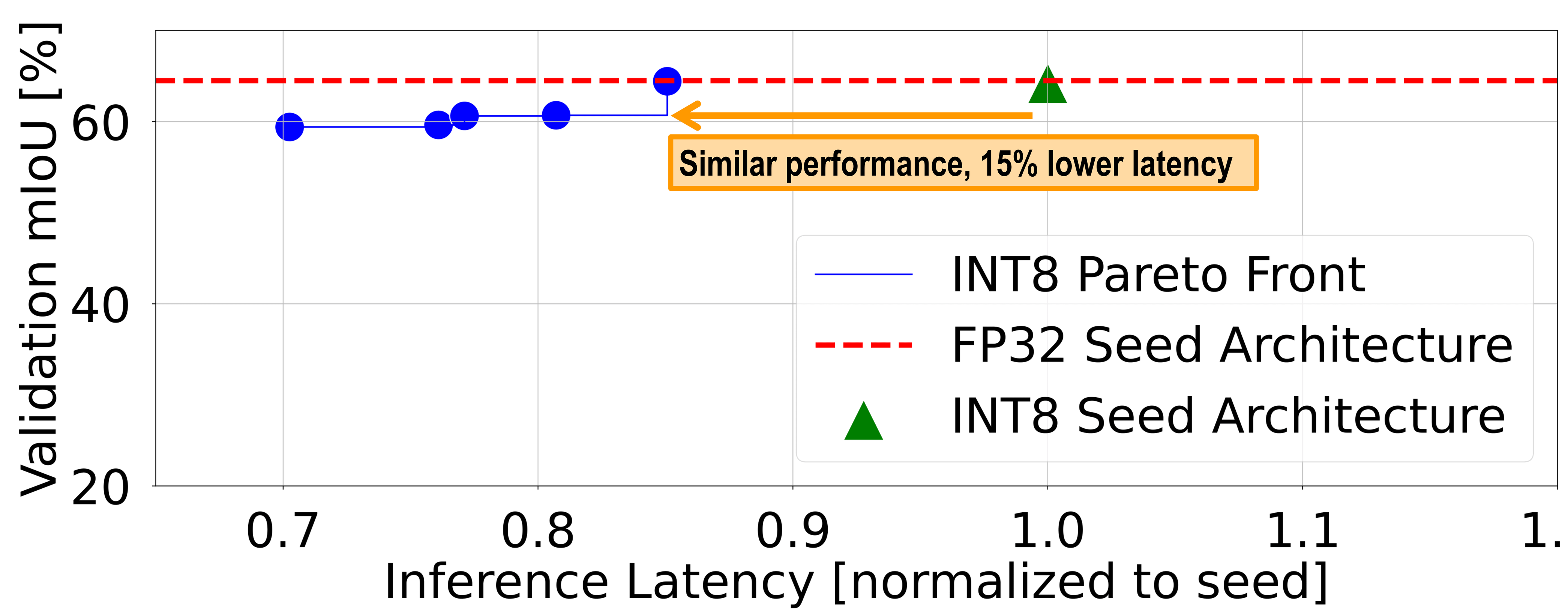


Figure 4: Performance of the found Pareto-optimal solutions on the **Cityscapes dataset.**

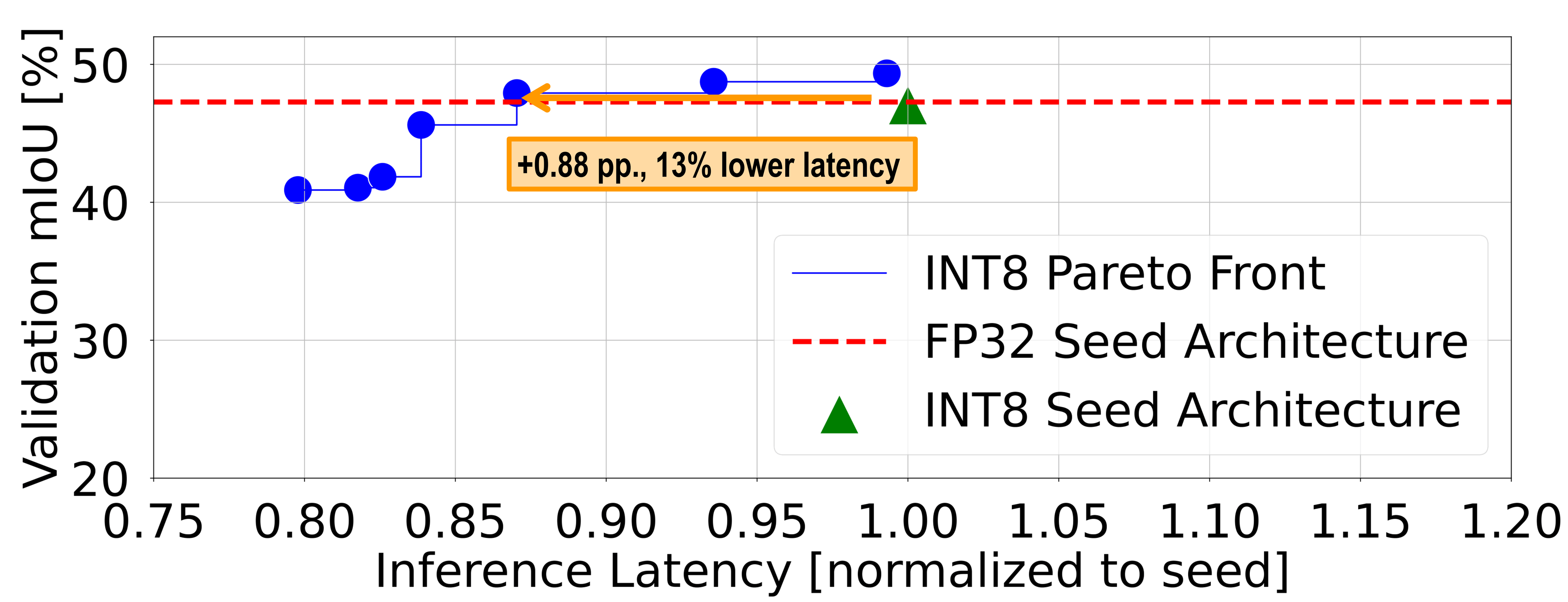


Figure 5: Performance of the found Pareto-optimal solutions on the **MS-COCO dataset.**

## Ranking correlation analysis

**For both datasets, we evaluate the ranking correlation between subnet performance and stand-alone** performance for 25 architectures, including:

- 12 Pareto optimal networks.
- 12 Pareto worse networks.
- The seed network.

**We find a significant correlation between subnet and stand-alone performance, which encourages us to use this ranking mechanism in our experiments.**

Table 1: Ranking correlation between subnet performance and stand-alone performance.

| | Cityscapes | MS-COCO |
|---|---|---|
| Kendall $\tau$ | 0.6000 | 0.7045 |
| Spearman $\rho$ | 0.7777 | 0.8554 |

## Future Work

**Investigate:**

- Extendibility of this method towards more diverse search spaces.
- Applicability towards further industrial applications.

## Executive Summary

**We extend existing work [3]** on weight-sharing NAS **towards searching for optimal quantized semantic segmentation networks:** Quantization-Aware Neural Architecture Search.

**Via Quantization-Aware Neural Architecture Search we:**

- Derive a trade-off between multiple-objectives.
- Reduce latency by up to 15% without compromising task performance on the Cityscapes dataset.
- Reduce inference latency by up to 13% while improving mIoU by 0.88 pp.

Figure 6: Optimized model running on NXP hardware.

## References


[1] T. Elsken et al., JMLR '19
[2] L. Xie et al., ACM Surveys, 54-9, '21
[3] X. Chu et al., ICCV '21
[4] M. Cordts et al., CVPR '15
[5] T. Lin et al., ECCV '14
[6] M. Sandler et al., CVPR '18
[7] L. Chen et al., arXiv:1706.05587
[8] J. Fang et al., ICLR '20
[9] https://tinyurl.com/5xrnkht5