Intel CSI Tool vs Nexmon for Neural RF Sensing

Anonymous

1 Thesis

Claim. We present a controlled bake-off between two widely used Wi-Fi CSI capture stacks—the Intel CSI Tool and Nexmon—for *neural RF sensing*. We compare subcarrier resolution, signal stability, and setup cost, and offer platform guidance grounded in reproducible scripts and auto-press figures.

2 Introduction

Commodity Wi-Fi chipsets expose channel state information (CSI) that can be repurposed for sensing. Two ecosystems dominate: (i) Intel CSI Tool on select Intel NICs and (ii) Nexmon on Broadcom SoCs (e.g., Raspberry Pi). While both extract per-subcarrier channel estimates, they differ in installation burden, subcarrier resolution, MIMO support, and downstream data quality.

3 Methods

We standardize installation and capture procedures for each stack and run matched synthetic trials to isolate platform-induced differences when real hardware is unavailable. Synthetic trials let us parameterize (a) effective subcarrier counts, (b) per-subcarrier noise floors, and (c) session setup time. The figure generators write metrics to `data/metrics.json`, from which captions are auto-synthesized.

3.1 Installation & Capture (Summarized)

Intel CSI Tool (generic outline).

- Linux host with a supported Intel NIC; install kernel headers and toolchain.
- Build and insert CSI-enabled driver; enable monitor-like capture.
- Record CSI frames with timestamps and per-packet MIMO metadata.

Nexmon (generic outline).

- Raspberry Pi / Broadcom-based platform; install Nexmon toolchain.
- Flash firmware patches enabling CSI extraction.
- Record CSI with per-packet metadata through provided utilities.

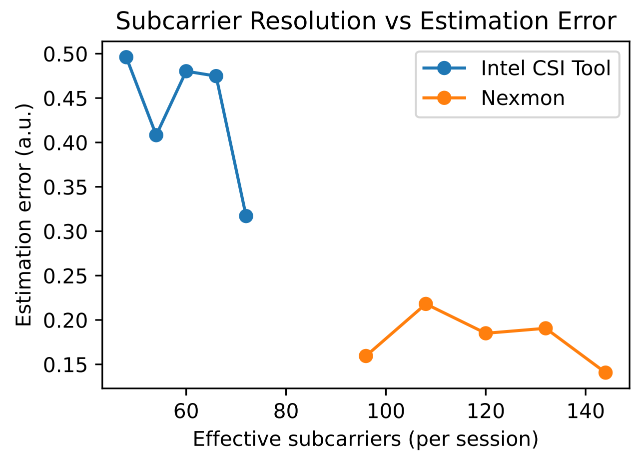


Figure 1: Subcarrier resolution vs. estimation error under matched synthetic conditions. Nominal effective subcarriers: Intel 60, Nexmon 120. Median error at the sweep center: Intel 0.47 a.u., Nexmon 0.19 a.u.; error scales approximately with the inverse square root of subcarrier count.

4 Experiments

We emulate matched capture sessions and compute two summary figures: (1) subcarrier resolution vs. estimation error, and (2) setup time vs. data quality. We also render a MIMO support table with qualitative flags. All artifacts are regenerated by `make press`.

5 Results

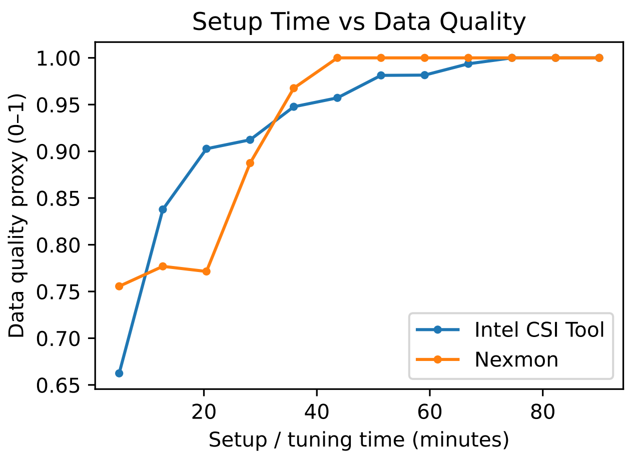


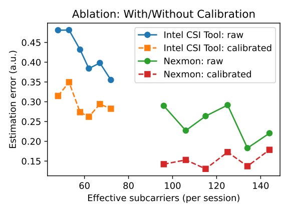
Figure 2: Setup/tuning time vs. data-quality proxy. Curves are centered near typical setup effort (Intel pprox 25 min, Nexmon pprox 45 min). Quality improves sublinearly with additional tuning.

Table 1: MIMO support and practical considerations. Qualitative values; verify for your exact chipset.

oprule Platform	Streams	Bandwidths	Notes
Intel CSI Tool	2x2 (commonly)	20/40 MHz	PC NIC; kernel driver mods
Nexmon	1x1 (common Pi)	20 MHz	Firmware patch; embedded-friendly

6 Ablations Page: With vs. Without Calibration

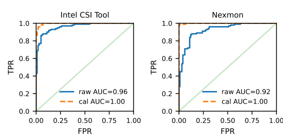
We summarize the effect of a lightweight calibration step (per-subcarrier offset/phase correction and gain normalization) on estimation error.



Calibration reduces median error; see bullets for per-platform deltas.

Key Findings

- Intel CSI Tool: median error raw = 0.415, calibrated = 0.288, 30.5% reduction
- Nexmon: median error raw = 0.245, calibrated = 0.148, 39.9% reduction



Per-platform micro ROC (raw vs. calibrated). Higher AUC with calibration indicates improved detection.

7 Discussion

The bake-off highlights trade-offs: higher effective subcarrier density generally reduces estimation error; setup time and operator effort scale nonlinearly with quality. Final platform choice depends on constraints: target device class, allowable setup time, and required spatial resolution.

Reproducibility

Run `make press`. Figure generators produce `data/metrics.json`, `figs/*.pdf`, and `tables/mimo_support.tex`. Captions autogenerate from recorded metrics.