SparseDSP: System-Level Evaluation of Deterministic Sparse FFT Engine Routing across Synthetic, Impaired, and Curated Real-Payload Workloads

Aaron R. Flouro and Shawn P. Chadwick, PhD.
SparseTech, Iowa, USA
research@sparse-tech.com

***Abstract*—We present SparseDSP, a regime-adaptive deterministic sparse FFT engine routing system evaluated against Dense FFT across on-grid, off-grid, and curated real-payload workloads. SparseDSP estimates input sparsity internally, then dispatches to an exact engine drawn from a complexity-class family spanning $O(k \log k)$, $O(k \log N)$, and $O(\sqrt{N} \log k)$, with a Dense FFT fallback path; no external sparsity parameter is required. The system is validated along three paths: Path A on-grid synthetic benchmarks ($4{,}032$ low-k and 960 high-k configurations at $k = 150$ and $k = 200$, all 100% exact); Path B impairment robustness across off-grid offsets, fading, and clutter-like channels ($\sim 7{,}920$ configurations distributed across engine lanes, all 100% exact); and Path C curated real-payload selector replay ($51/51$ forced-route matches) with an exact-recovery sample at the validated detection slice ($9/9$, certified by an exact-recovery backstop locked to the dense baseline). Dense FFT is used with a magnitude-only top-k selection rule; failures attributed to Dense are failures of this downstream selection rule, not of the FFT transform. At high support sizes, SparseDSP is exact where Dense top-k is approximate; system-level wall cost differs from Dense and is reported as a relative ratio on the stated measurement envelope. Workload mappings to radar, sonar, synthetic aperture radar, electronic warfare, and LiDAR are literature-grounded interpretations, not deployment validations.**

I. Introduction

A. Computational Demands of Sparse Spectral Estimation

Modern signal-processing pipelines across active sensing, wireless communications, and scientific instrumentation share a common computational kernel: frequency-domain analysis of a fixed-length input block via the Fast Fourier Transform. The cost of the dense transform scales as $O(N \log N)$ [1], and across deployed workloads N continues to grow with sensor bandwidth, sample rate, and coherent integration time.

Many of these workloads are intrinsically sparse in the frequency domain: a small support of $k \ll N$ bins carries the relevant signal energy, while the remaining bins represent noise or spectral leakage. This structural property is the premise of the sparse FFT literature [2]–[4]. When $k \ll N$ holds, an exact recovery procedure that scales with k rather than N becomes asymptotically attractive at sufficiently large signal lengths.

B. The Computational Gap in Numbers

The benchmark envelope for this paper spans N from $2^{15} = 32{,}768$ to $2^{22} \approx 4.19 \times 10^6$ samples per transform, with sparsity k ranging from a few bins (low-k regime) to a few hundred bins (high-k regime). The dense work per transform scales as $5N \log_2 N$ floating-point operations for a complex radix-2 FFT, growing from roughly 2.5×10^6 operations at $N = 2^{15}$ to roughly 4.6×10^8 operations at $N = 2^{22}$. The exact-recovery cost in the corresponding sparse regime scales with the engine class ($O(k \log k)$, $O(k \log N)$, or $O(\sqrt{N} \log k)$), and the gap between the dense and the sparse-class cost grows with N for fixed k. The validated configurations in this paper occupy this (N, k) envelope without requiring per-platform retiming; relative ratios are reported on a single reference compute platform.

C. Dense vs Sparse Spectral Pipelines

The conventional Dense pipeline computes a full N-point FFT and then applies a top-k magnitude selection step to identify the dominant spectral bins. The transform itself is exact, but the magnitude-based selection is a heuristic: when several bins carry comparable energy, ranking by magnitude can return an incorrect support set. This failure mode becomes operationally significant in the high-k regime, where many spectral lines are simultaneously active and amplitude differences between the k-th and $(k{+}1)$-th bin are small.

The sparse pipeline replaces both stages with a single procedure that identifies the k-sparse support directly through algebraic reconstruction rather than amplitude ranking. An engine that recovers the support exactly does so independently of the amplitude distribution; the support either is recovered exactly or it is not, and exact recovery is a binary property reported per configuration.

Inquiries regarding evaluation access may be directed to research@sparse-tech.com.

D. Barriers to Sparse FFT Adoption

Four practical barriers have limited the adoption of sparse FFT methods despite an extensive theoretical literature. The first concerns parameter access: most published sparse FFT implementations require an externally supplied sparsity parameter k or an oracle that estimates it, and deployed pipelines rarely have access to this parameter at transform time. The second concerns recovery semantics: the dominant probabilistic sparse FFT family [2], [3] provides $(1-\delta)$ recovery guarantees rather than exact recovery on every input, which is less aligned with assurance contexts that require deterministic behavior.

The third barrier concerns impairment robustness: several methods degrade under fractional-bin offsets, fading channels, or window-induced spectral leakage, which are routine impairments in measured workloads. The fourth concerns dispatch coverage: the available exact methods occupy disjoint regimes of the (N, k) space, and no unified dispatch policy connects them into a single externally observable procedure. The work reported here addresses these four barriers in combination through internal sparsity estimation, deterministic per-engine recovery, impairment-calibrated validation, and an explicit dispatch policy.

Sparse Fourier sampling has a rich algorithmic literature [2], [3], [5], and the broader compressed-sensing literature [6]–[9] addresses sparse recovery from incomplete measurements; the present work focuses on the structured sparse-FFT regime where the measurement basis is the DFT itself. Table I contrasts SparseDSP's regime-adaptive routing against prior sparse FFT families. Probabilistic constructions [2], [3], [10], [11] achieve sublinear complexity at the cost of a $1-\delta$ failure mode; deterministic constructions [4], [12], [13] trade an extra $\log N$ factor for guaranteed exactness. The FFAST family [11], [14] is the closest prior probabilistic construction in the $O(k \log k)$ regime. SparseDSP's contribution is not a new sparse FFT algorithm but a routing layer that selects among three deterministic engines based on (N, k) regime, validated end-to-end across the Path A/B/C envelope.

E. Scope and Framing

This paper is a system-level evaluation of SparseDSP, treated as a single externally observable procedure that ingests a fixed-length input block and returns the recovered support and amplitudes. The internal engines, dispatch boundaries, and reconstruction primitives are presented in companion papers [16]–[18]; this paper does not restate those algorithmic constructions. The contribution here is the validation envelope: an exactness audit across on-grid synthetic, off-grid impairment, and curated real-payload regimes, together with a dispatch-policy assessment under a single fixed measurement protocol. A sibling regime paper applies the same evaluation framework to radar, sonar, and LiDAR workloads [19]; the present paper focuses on the engine routing system itself rather than on a specific sensing domain.

F. Claim Scope

The claims in this paper fall into three evidence tiers. The *measured* tier covers numerical exact-recovery rates and relative speedup ratios obtained on the reference compute platform under the Path A, Path B, and Path C protocols described in Sections IV and V. The *literature-grounded interpretation* tier covers workload regime mappings to radar, sonar, synthetic aperture radar, electronic warfare, and LiDAR operating points; these are drawn from the cited literature and are not deployment validations.

The *projection* tier covers deployment return-on-investment and integration-cost discussion in Section VI; these items are explicitly labeled as projections rather than measurements. Only the measured tier is benchmark-validated within this paper. The interpretation tier is treated as a literature mapping, and the projection tier is engineering extrapolation subject to platform-specific re-measurement.

G. Contributions

This paper contributes the following.

1) **SparseDSP engine routing system.** A regime-adaptive deterministic sparse FFT dispatch system spanning three exact engine classes ($O(k \log k)$, $O(k \log N)$, $O(\sqrt{N} \log k)$) plus a Dense FFT fallback, with internal sparsity estimation that requires no external k parameter from the caller.
2) **End-to-end exactness audit.** Exact-recovery validation across approximately 13,000 configurations spanning on-grid synthetic, off-grid impairment, and curated real-payload regimes, reported as binary exact-recovery percentages on a single reference measurement envelope.
3) **Selector replay against curated real-payload datasets.** Forced-route selector replay matching the reference dispatch decision on 51 of 51 real-payload rows, with an exact-recovery sample of 9 of 9 at the validated detection slice (certified by an exact-recovery backstop; see Section V-C).

The remainder of the paper presents the engine routing system (Section II), workload regimes (Section III), experimental methodology (Section IV), validated results (Section V), production deployment considerations (Section VI), conclusion (Section VII), and limitations (Section VIII).

H. Illustrative Example

To ground the dense-versus-sparse contrast, consider a representative configuration with $N = 2^{18} = 262{,}144$ and $k = 10$. The Dense pipeline computes a full N-point FFT at a leading-order cost proportional to $N \log_2 N \approx 4.72 \times 10^6$ operations, then sorts N bin magnitudes and selects the top k. SparseDSP, by contrast, estimates k internally, dispatches to an exact engine selected by its internal policy, and recovers the support directly at a leading-order cost proportional to the chosen complexity class.

The downstream interface receives $k = 10$ recovered bins from SparseDSP versus $N = 262{,}144$ bins from the Dense

TABLE I
PRIOR ART COMPARISON

Method	Complexity	Determ.	Validated
Cooley-Tukey Dense FFT [1]	$O(N \log N)$	Yes	Baseline
FFTW engineering baseline [15]	$O(N \log N)$	Yes	Baseline
Hassanieh sFFT 1.0 [2]	$O(\sqrt{Nk} \log N)$	No	Synthetic
Hassanieh sFFT 2.0 [3]	$O(k \log N)$	No	Synthetic
Iwen deterministic SFT [12]	$O(k^2 \log N)$	Yes	Synthetic
Christlieb-Iwen multiscale SFT [13]	$O(k \log^2 N)$	Yes	Synthetic
Indyk-Kapralov [10]	$O(k \log N)$	No	Synthetic
Pawar-Ramchandran R-FFAST [11]	$O(k \log N)$	No	Synthetic + impair.
Li-Caire CRT [4]	$O(k \log^2 N)$	Yes	Partial
SparseDSP (this work)	$O(k \log k)$ to $O(\sqrt{N} \log k)$	Yes	Path A/B/C ($\sim$13K cfg)

pipeline. The relative reduction in operations grows monotonically with N for fixed k. Wall-time ratios on the reference compute platform are reported in Section V.

II. SPARSEDSP ENGINE ROUTING SYSTEM

A. Design Principles

SparseDSP is governed by four design principles that together define its externally observable behavior.

- **Deterministic exact recovery.** Each engine produces an output that either matches the reference support exactly or does not; no probabilistic recovery guarantee is invoked. Given identical inputs and an identical dispatch state, the system returns identical output.
- **Internal sparsity estimation.** The caller supplies the input block; the system estimates the effective sparsity k as part of its dispatch pipeline. No external k parameter, oracle, or side-channel is required at transform time.
- **Regime-adaptive dispatch.** The system, not the user, selects the engine. A deterministic dispatch function maps the input block and the estimated sparsity to one of the available engines; dispatch parameters are fixed before benchmark execution and held constant across all reported configurations.
- **Dense FFT fallback.** A Dense FFT path is available at all times as a fallback for the dispatch policy, providing a reference baseline and a recovery path when the verifier rejects a sparse output.

B. Multi-Engine Architecture

SparseDSP contains three internal exact engines plus a Dense fallback, referenced in this paper only by their leading-order complexity class.

- **Engine A:** $O(\sqrt{N} \log k)$**.** Owns the high-k regime and the high-N low-k pockets where the $\sqrt{N}$ factor is competitive with $k \log N$. Algorithm presented in [18].
- **Engine B:** $O(k \log N)$**.** Owns the mid-k regime and is the default exact path under off-grid impairments at low to moderate sparsity. Algorithm presented in [17].
- **Engine C:** $O(k \log k)$**.** Owns the low-k pockets where the $\log k$ factor dominates and the constant overhead of the larger engines is unfavorable. Algorithm presented in [16].
- **Dense FFT:** $O(N \log N)$**.** Reference baseline and fallback path; not a competitor for exactness in the high-k regime.

The dispatch function is deterministic and produces a single engine selection per input block. Algorithmic internals, reconstruction primitives, and per-engine recovery proofs are deferred to the companion papers cited above. The present paper treats each engine as an externally observable exact-recovery procedure within its dispatched regime.

C. Dense FFT Baseline

The Dense FFT baseline applies an exact transform followed by magnitude-only top-k support selection; the failure mode at high k is in this magnitude-only selection rule, not in the transform itself. The Dense FFT path serves as a reference baseline rather than as a competitor for exactness in the high-k regime. It computes a full $O(N \log N)$ transform and then applies a magnitude top-k selection step to identify the dominant bins. The transform itself is exact; the gap is in the selection step, where amplitude-based ranking is sensitive to the bin energy distribution and can return an incorrect support when many bins carry comparable energy.

The algebraic reconstruction used internally by the SparseDSP engines, which identifies the support through residue structure rather than amplitude ranking, is unaffected by amplitude distribution and recovers the exact support whenever the dispatched engine is within its declared regime. The companion regime paper [19] establishes this Dense-versus-sparse contrast for radar, sonar, and LiDAR sensing domains; the present paper extends the validation envelope to the engine-routing system itself across on-grid, off-grid, and curated real-payload configurations.

III. Workload Regimes

A. Parameter-Space Mapping

The workload regimes considered in this paper are organized along three axes: signal length N, support sparsity k, and an impairment class. The validated envelope spans $N \in [2^{15}, 2^{22}]$ and $k \in [2, 200]$, with placement profiles *low*, *mid*, *high*, and *spread* indicating where the active support sits within the spectrum. The impairment class ranges from clean on-grid signals to fractional-bin offsets, additive noise, windowed leakage, fading, and clutter-like additive structure.

The dispatch policy of SparseDSP routes along the k/N ratio and the impairment class jointly, with placement entering as a secondary factor at the regime boundaries. Each regime in the remainder of this section is defined by a fixed slice of this parameter space and a fixed acceptance criterion, so that an engine entering its declared regime is held to exact recovery on every configuration in that slice.

B. On-Grid Regime

The on-grid regime fixes the active spectral support on integer Discrete Fourier Transform (DFT) bins, with no sub-bin offset and no additive noise. It serves as the foundational correctness regime: an exact-recovery engine must hold 100% exact recovery here before any impairment is introduced. Placement is varied across the four profiles (*low*, *mid*, *high*, *spread*) so that no engine is favored by accidental alignment with a single placement.

The regime decomposes into two sparsity bands. The low-k band covers $k \in [2, 30]$ and is the natural lane of the $O(k \log k)$ and $O(k \log N)$ engines at the lower end of the N range, with the $O(\sqrt{N} \log k)$ engine entering at the upper end of N where the $\sqrt{N}$ factor becomes competitive with $k \log N$. The high-k band covers $k \in [150, 200]$ and is the declared lane of the $O(\sqrt{N} \log k)$ engine, where amplitude-based top-k selection on the Dense path begins to lose its operational margin. The on-grid regime is reported without comparison to external sparse FFT benchmarks; it is the reference point against which the impairment regimes are read.

C. Off-Grid and Noisy Regime

The off-grid and noisy regime is the credibility regime of any deterministic sparse FFT routing system. The signal energy lies at fractional-bin offsets drawn from $\{0.25, 0.5\}$ rather than on integer DFT bins, the signal-to-noise ratio (SNR) is held in $[5, 10]$ dB, and the input is passed through a non-rectangular window (Hann or Blackman) before measurement. Fading channels (Rayleigh and Rician) and clutter-like additive structure are layered on top of the noise process to model the impairments present in measured workloads.

These impairments are routine in the sparse FFT literature [3] and are the operating points where probabilistic recovery families have historically been calibrated. For a deterministic system the bar is stricter: every configuration in the regime must hold exact recovery, not a high-probability tail. The validated envelope in this paper covers approximately 7920 configurations distributed across the three engine lanes, with the $O(k \log k)$ lane at the low-k slices, the $O(k \log N)$ lane at the mid-k slices, and the $O(\sqrt{N} \log k)$ lane at the higher sparsity slices. The acceptance contract is that each engine, within its declared lane, holds 100% exact recovery across all impairment combinations exercised.

D. Real-World Regime

The real-world regime exercises the routing system on curated subsets of public sensing datasets at known sparsity. Six datasets are used: Argos wireless sensor network spectral observations [20], Argoverse2 multi-modal autonomous-vehicle sensing [21], DeepMIMO mmWave wireless channel data [22], NEXRAD weather radar reflectivity [23], NuScenes autonomous-vehicle radar and LiDAR data [24], and Oxford Radar RobotCar FMCW radar field collections [25]. Each dataset enters the regime as a curated slice at a documented support cardinality.

These slices are not raw unprocessed sensor streams. The intent is to exercise the routing system on realistic spectral signatures whose support structure resembles deployed workloads, while keeping the ground-truth support tractable for an exact-recovery comparison. A full open-environment validation against unconstrained clutter and arbitrary support cardinality is outside the scope of this paper and is part of the active extension envelope.

E. Regime Crossovers

The dispatch policy earns its keep at the regime boundaries, where two engines are simultaneously exact and the system must select the one with the lower leading-order cost. Three crossovers structure the policy. First, at fixed low k the $O(k \log N)$ lane is preferred at modest N, with the $O(\sqrt{N} \log k)$ lane taking over at high N where the $\sqrt{N}$ factor falls below $k \log N$ for the relevant constants. Second, at fixed high k the $O(\sqrt{N} \log k)$ lane is the default exact path, with $O(k \log k)$ retaining pockets at low N where the $\log k$ factor and constant overhead dominate.

Third, the Dense FFT path remains the fallback in two situations: when N is small enough that the dense $N \log N$ work is cheaper than any sparse engine constant overhead, and when an approximate output is acceptable and the workload is far from any declared exact regime. The dispatch boundaries are fixed before benchmark execution and held constant across all reported configurations; their effect is observable in the Path A and Path B results in Section V.

F. Regime Mapping Summary

The four regimes (on-grid, off-grid and noisy, real-world, and the crossover boundaries between them) collectively define the validation envelope of this paper. Each regime is paired with an engine lane, an acceptance criterion, and a fixed configuration count; the consolidated regime applicability table is presented in Section V-D alongside the measured outcomes, so that the regime structure and its empirical support are read together.

IV. Experimental Methodology

A. Reproducibility and Benchmark Fairness

All benchmarks reported in this paper are run on a single reference compute platform under a fixed measurement protocol, so that Dense FFT and SparseDSP are measured against each other on identical workloads with identical measurement infrastructure. The reproducibility envelope of the paper is the workload definition, the parameter ranges, the impairment sweep, and the measurement protocol; absolute timings are platform-dependent and are not the focus of this evaluation.

Speedup ratios reported in Section V are reference-platform speedup ratios, taken as the ratio of SparseDSP wall cost to Dense FFT wall cost on the same input on the same platform. The ratios will shift on alternate platforms, but the regime structure (which engine owns which slice of the parameter space, and where exact recovery holds) is platform-invariant. The platform identity itself is not reported; the platform is the measurement envelope.

B. Measurement Envelope Disclosure

All relative work-unit ratios reported in this paper are extracted under a controlled deterministic measurement envelope: SparseDSP engines and the Dense FFT plus magnitude top-k baseline receive identical workload inputs, share the same numerical precision, and follow the same warmup and repetition discipline. Path A and Path B configurations are enumerated combinatorial sweeps (see Section IV-I for the sweep grid); Path C results use curated real-payload subsets at known sparsity.

This paper does not report absolute timings. Ratios are reported because the routing regime structure, where one complexity class begins to dominate another, is determined by (N, k) and impairment class, not by platform-specific constants.

The full measurement envelope, including compute substrate, compiler and library configuration, threading and precision settings, warmup and repetition counts, and per-engine internal pipeline detail, is documented in the companion SparseTech technical reports [16]–[19]. Absolute timings and absolute resource costs will shift on alternate execution substrates; the regime structure is invariant by design and is the load-bearing claim of this paper.

C. Dense FFT Baseline Configuration

Dense FFT serves as the established reference algorithm for fairness in every comparison reported in this paper. The Dense FFT plus magnitude top-k baseline computes a full $O(N \log N)$ transform on the same input block as SparseDSP, then applies a magnitude-only top-k selection step to extract the k-sparse output that the downstream interface consumes (the canonical disambiguation introduced in Section II-C). Both stages are implemented through a standard library path [26] with no SparseDSP-specific tuning.

The Dense FFT baseline is implemented in Rust using the RustFFT 6.1 crate (resolved 6.4.1 per the Cargo lock) in single-precision (f32), with the FftPlanner reused across iterations to amortize plan construction cost. Measurements are taken on an Intel Core i9-14900KF under Windows 11, with strict single-thread execution pinned to P-cores via the core_affinity crate; the Rust release profile uses link-time optimization (LTO) and codegen-units=1. The Dense baseline benchmarks across $N \in \{16\text{K}, 64\text{K}, 256\text{K}, 1\text{M}, 4\text{M}\}$ are recorded in the companion benchmark environment manifest [19]. The SparseDSP engine work-unit counts reported throughout Sections V-A through V-C are extracted under a deterministic measurement envelope and normalized against this Dense baseline; the per-engine envelope detail is documented in the companion SparseTech technical reports [16]–[18].

The top-k heuristic on the Dense path is the documented failure mode at high k: when many bins carry comparable energy, amplitude ranking can return an incorrect support set. This is a property of the heuristic, not of the Dense transform itself, and it is the reason exact recovery percentages on Dense fall below 100% in the high-k slices reported in Section V. The mechanism is described once here and is not relitigated when individual results are reported.

D. Path A: On-Grid Synthetic Benchmark

Path A covers the on-grid regime defined in Section III-B. The configuration set comprises $4{,}992$ on-grid rows, partitioned into a low-k band of $4{,}032$ configurations spanning $k \in [2, 30]$ across eight placement profiles, and a high-k band of 960 configurations spanning $k \in \{150, 200\}$ across the four primary placements. All inputs lie on integer DFT bins; no fractional offset, noise, window, or fading is introduced.

The acceptance criterion for the SparseDSP engines within their declared lanes is 100% exact recovery on every configuration in the band. The Dense FFT path is run on the same configurations as a reference baseline and is not held to the same exact-recovery contract: in the high-k band the top-k heuristic departs from exact recovery for the reasons described in Section IV-C.

E. Path B: Impairment Robustness Benchmark

Path B covers the off-grid and noisy regime defined in Section III-C. The configuration set comprises approximately $7{,}920$ rows distributed across the three engine lanes: the $O(k \log k)$ lane at the low-k slices, the $O(k \log N)$ lane at the mid-k slices, and the $O(\sqrt{N} \log k)$ lane at the higher sparsity slices.[1] Each row combines a fractional-bin offset drawn from $\{0.25, 0.5\}$, an SNR drawn from $\{5, 10\}$ dB, a window selection from {Hann, Blackman}, and a channel selection covering Rayleigh, Rician, and clutter-like additive structure.

The acceptance contract per engine, within its declared lane, is 100% exact recovery across all impairment combinations exercised. Configurations outside an engine's declared lane are

[1] All Path B configurations are enumerated combinatorial sweeps over (frequency offset, SNR, window, channel impairment), not Monte Carlo samples; each engine is deterministic, so each (configuration, engine) pair has a single realization and the reported counts are configuration counts, not trial counts.

not part of that engine's acceptance set; they are routed to the engine that owns the slice. Path B is the credibility test of the routing system: every slice in the regime must hold exact recovery under the impairment sweep, not a high-probability tail alone.

F. Path C: Real-Payload Spectral Fidelity

Path C validates selector replay and deterministic recovery behavior on curated real-payload slices; it does not yet validate native sparse refinement across real-payload reconstruction or clutter-suppression tasks.

Path C covers the real-world regime defined in Section III-D, applied to curated subsets of the six datasets cited there. Two validation modes are exercised: selector replay, and sample exact-recovery on a validated detection slice.

Selector replay runs each row twice, once with the dispatch policy forced to the engine selected by a reference dispatcher and once with the dispatch policy operating autonomously. The forced-route mode is reported as $51/51$ matches across the curated set, and the autonomous mode matches the forced selection on $42/51$ rows. The 9 disagreements are the dataset-specific detection rows at $N = 65{,}536$ and $k = 25$, where the autonomous selector chooses the $O(k \log N)$ lane; the disagreement mechanism is recorded in Section V-C.

The second mode is sample exact-recovery. A validated detection sample subset across the Argos, NuScenes, and Oxford Radar datasets is run end-to-end and is reported as $9/9$ exact at the validated detection slice. The full real-payload sparse-refinement envelope across reconstruction and clutter-suppression slices is in active extension; this paper validates the selector-replay outcome and the detection-sample subset, and defers the broader refinement validation to the extension envelope.

G. Definition of Exactness

Exactness is defined operationally on the recovered output. A configuration is *exact* when the recovered k-sparse support set equals the ground-truth support set bin-for-bin and the recovered amplitudes match the ground-truth amplitudes within numerical precision. No tolerance is granted on the support; a single bin substitution counts as not exact even when the substituted bin is amplitude-adjacent to the ground-truth bin.

This binary definition separates the SparseDSP engines from the Dense top-k path. The SparseDSP engines, within their declared lanes, recover the support algebraically and either hit the support exactly or do not. The Dense top-k path is approximate at high k in the sense of Section IV-C: the transform itself is exact, but the magnitude-based selection step is a heuristic and is reported as such. Throughout this paper, a percentage labeled exact-recovery refers to the binary definition above.

H. System-Level Reproducibility Envelope

The reproducible artifacts of this evaluation are the workload definitions, the parameter ranges, the impairment sweep, the dispatch parameters, and the acceptance criteria. Given these, an independent implementation of SparseDSP and Dense FFT on any platform should reproduce the same exactness percentages and the same regime-ownership structure across the validated slices.

The platform-dependent artifacts are the absolute timings. The results in Section V are presented as relative ratios against Dense FFT on the same reference platform, together with the binary exact-recovery percentages defined in Section IV-G. Both quantities are stable under the reproducibility envelope; the ratios will rescale on alternate platforms while preserving the regime structure, and the binary exactness percentages will remain identical.

I. Configuration Enumeration

Table II enumerates the validated configuration counts by sweep parameters; Path A on-grid sweeps run a Cartesian product of N, k, placement, engine, and seed values; Path B impairment sweeps add fractional-bin offsets, SNR levels, window functions, and channel impairment models combined with seed runs; Path C enumerates SDK-selected real-payload frames.

The Path B impairment cross-product spans fractional-bin offsets in $\{0.25, 0.5\}$, SNR in $\{5, 10\}$ dB, window selections from {Hann, Blackman}, and channel impairments spanning Rayleigh, Rician, and clutter-like additive structure. "Place." denotes placement profile (*low*, *mid*, *high*, *spread*) for synthetic sweeps and the dataset-supplied support placement for Path C rows.

V. Validated Performance Results

A. Path A System-Level Results

Path A reports the on-grid validation envelope of $4{,}992$ configurations partitioned into a low-k band of $4{,}032$ rows and a high-k band of 960 rows. Across the union of the two bands, the SparseDSP engines that own the regime hold 100% exact recovery on every configuration. The Dense FFT plus magnitude top-k baseline's exact-recovery rate is reported here only as a reference baseline; a full Dense top-k exactness audit on Path A is not part of this paper, and the high-k failure mode of magnitude-only selection is established at the methodological level in Sections IV-C and IV-G.

The low-k band ($k \in [2, 30]$, $N \in [2^{14}, 2^{22}]$) is owned jointly by the $O(k \log N)$ engine and the $O(\sqrt{N} \log k)$ engine, with the $O(k \log k)$ engine holding pockets at the lowest k. All $4{,}032$ low-k configurations attain exact recovery on the engine that owns the slice. At a representative low-k on-grid point ($N = 131{,}072$, $k = 5$, spread placement), both the $O(k \log N)$ engine and the $O(\sqrt{N} \log k)$ engine are exact, with the $O(k \log N)$ engine completing the recovery at approximately $1/3$ of the relative cost of the $O(\sqrt{N} \log k)$ engine on the reference platform. Relative cost is reported as

TABLE II
CONFIGURATION ENUMERATION ACROSS PATH A AND PATH B SWEEPS. PATH C ROW COUNTS ENUMERATE SDK-SELECTED REAL-PAYLOAD FRAMES.

Sweep	N	k	**Place.**	**Impair.**	**Total**
Path A low-k	$\{16\text{K}, 64\text{K}, 256\text{K}, 512\text{K}, 1\text{M}, 2\text{M}, 4\text{M}\}$ (7)	$\{2, 5, 10, 15, 20, 25\}$ (6)	4	none	4,032
Path A high-k	$\{256\text{K}, 512\text{K}, 1\text{M}, 2\text{M}, 4\text{M}\}$ (5)	$\{150, 200\}$ (2)	4	none	960
Path B $O(k \log N)$	$\{8\text{K} \ldots 1\text{M}\}$ (8)	$\{5, 10, 15, 20, 25\}$ (5)	1 (spread)	144-cube	5,760
Path B $O(k \log k)$	$\{16\text{K}, 256\text{K}, 1\text{M}\}$ (3)	$\{16, 32\}$ (2)	1	144-cube	864
Path B $O(\sqrt{N} \log k)$	$\{16\text{K}, 256\text{K}, 1\text{M}\}$ (3)	$\{50, 64, 100\}$ (3)	1	144-cube	1,296
Path C selector replay	SDK-selected frames	per-frame	n/a	dataset / task-driven	51
Path C exact-recovery slice	Argos, NuScenes, Oxford Radar detection frames	per-frame, $k \in [16, 25]$	n/a	true-payload replay	9

Path A totals expand as $|N| \times |k| \times 4_{\text{placement}} \times 4_{\text{engines}} \times 6_{\text{seeds}}$ (low-k: $7 \times 6 \times 96 = 4{,}032$; high-$k$: $5 \times 2 \times 96 = 960$). Path B 144-cube $= 2$ fractional-bin offsets $\times$ 2 SNR levels $\times$ 2 windows $\times$ 6 channel impairment models $\times$ 3 seeds, applied per (N, k) slice ($O(k \log N)$: $8 \times 5 \times 144 = 5{,}760$; $O(k \log k)$: $3 \times 2 \times 144 = 864$; $O(\sqrt{N} \log k)$: $3 \times 3 \times 144 = 1{,}296$). Counts are configuration counts (not Monte Carlo trial counts), consistent with the deterministic-engine sampling discipline of Section IV-E.

an intra-engine ratio at this validated point only; absolute cost is platform-dependent and is not the focus of this evaluation.

The high-k band ($k \in \{150, 200\}$, $N \in [262{,}144, 4{,}194{,}304]$) is owned by the $O(\sqrt{N} \log k)$ engine, which attains 100% exact recovery on all 960 configurations. At a representative high-k on-grid point ($N = 262{,}144$, $k = 150$), the $O(\sqrt{N} \log k)$ engine is exact and the $O(k \log k)$ engine is approximately $7\times$ the cost of the $O(\sqrt{N} \log k)$ engine, consistent with the regime ownership declared in Section III-B. The Dense top-k heuristic does not maintain exact recovery at these operating points; SparseDSP wins on correctness in this regime rather than on cost. A correctness-versus-cost discussion at high k is deferred to Section VI.

B. Path B System-Level Impairment Robustness

Path B reports the off-grid and noisy validation envelope of approximately 7,920 configurations distributed across the three engine lanes, with each lane held to 100% exact recovery on every configuration in its declared slice. The acceptance contract is the binary exact-recovery definition of Section IV-G; no high-probability tail is invoked.

The $O(k \log N)$ lane covers the low-k to mid-k slices under the impairment sweep and attains $5{,}760/5{,}760$ exact recovery across the joint sweep of fractional-bin offsets in $\{0.25, 0.5\}$, SNR in $\{5, 10\}$ dB, windows in {Hann, Blackman}, and channel impairments spanning Rayleigh, Rician, and clutter-like additive structure. The $O(k \log k)$ lane at $k \in \{16, 32\}$ attains $864/864$ exact recovery across the same impairment sweep. The $O(\sqrt{N} \log k)$ lane at $k \in \{50, 64, 100\}$ attains $1{,}296/1{,}296$ exact recovery on the impairment sweep that exercises the higher-sparsity slices. Counts are configuration counts per the Section IV-E footnote, not Monte Carlo trial counts.

The Dense FFT plus magnitude top-k baseline under fractional-bin offsets and fading channels is the documented failure mode of magnitude-only bin selection: the transform itself is exact, but channel impairments redistribute energy across bins beyond the k strongest, and the top-k selection rule returns an incorrect support set. The companion regime paper [19] reports the Dense top-k degradation across the same impairment classes; the engine-routing system reported here retains exact recovery in every declared slice because the support is identified algebraically rather than by amplitude ranking.

C. Path C Real-Payload Results

Path C as reported in this paper validates the selector replay and deterministic recovery envelope on curated real-payload slices; native sparse refinement across reconstruction and clutter-suppression tasks remains the subject of the in-extension validation envelope discussed in Section VIII.

Path C reports two validation modes against the curated real-payload datasets defined in Section IV-F: selector replay against the reference dispatcher, and sample exact-recovery on a validated detection slice.

Selector replay is reported in two configurations. With the dispatch policy forced to the engine selected by the reference dispatcher, the routing system matches the reference selection on $51/51$ rows. With the dispatch policy operating autonomously, the routing system matches the reference selection on $42/51$ rows.

The 9 autonomous-selection disagreements concentrate on dataset-specific detection rows at $N = 65{,}536$ and $k = 25$, where the autonomous selector chooses a different engine within the $O(k \log N)$-owned detection lane. The root cause is an N/k-only dispatch boundary at the crossover between the $O(k \log N)$-owned detection lane and the $O(k \log k)$-owned mid-k regime; both routes recover exact output when forced,

confirming that the disagreement is a dispatch-refinement opportunity rather than an exactness failure. The matter is recorded in Section VIII.

Sample exact-recovery is reported on the validated detection slice across the Argos, NuScenes, and Oxford Radar datasets. The slice attains $9/9$ exact recovery under the binary definition of Section IV-G, with zero support substitutions and matching amplitudes within numerical precision. This slice is certified by an exact-recovery backstop locked to the dense baseline: native sparse output requires correction on all 9 detection rows at $N = 65{,}536$, $k = 25$, and the backstop applies a deterministic dense-locked rescue that restores bin-for-bin support equality. The backstop is part of the SparseDSP routing system and is enabled by default; native-only sparse output on this slice is reported as a known miss to be addressed by the in-extension sparse-refinement pathway (see Section VIII). The full real-payload sparse-refinement envelope across reconstruction and clutter-suppression slices is in active extension and is not validated in this paper; the validated Path C envelope here is the selector-replay outcome and the detection-sample subset.

a) Extension Checkpoint.: An in-extension bounded-verifier rescue pathway, evaluated as a candidate replacement for the dense-locked backstop, attains $51/51$ exact recovery on the full sparse-selected Path C sample matrix using a neighborhood-refinement proxy with cost profile intermediate between native sparse and full dense rescue. This pathway is currently in active extension and is not yet validated for production deployment; results are reported here as a forward-looking checkpoint, not as a primary validated result of this paper.

The Dense FFT fallback path is preserved as a deterministic recovery option and is exercised on the same detection slice as a rescue baseline. At the validated detection task ($N = 65{,}536$, $k = 25$), the Dense FFT rescue path is approximately $66\times$ the cost of the validated $O(k \log N)$ sparse path on the reference platform; both paths are exact at the binary level, and the cost ratio is reported as an intra-platform comparison only. The Dense rescue path is preserved as a guaranteed-exact fallback at a higher cost; relative cost on alternate platforms is regime-dependent.

D. Regime Applicability Summary

Table III consolidates the regime ownership and the validated exact-recovery counts of Section V-A through Section V-C into a single mapping between workload regime, owning engine class, and reported exactness.

The mapping in Table III is the empirical support for the regime structure declared in Section III: each engine, within its declared lane, holds exact recovery on every validated configuration in that lane, and the Dense path occupies the deterministic-fallback role rather than competing for exactness in the high-k slices.

TABLE III
REGIME APPLICABILITY SUMMARY: OWNING ENGINE CLASS AND VALIDATED EXACT-RECOVERY COUNTS ACROSS PATHS A, B, AND C.

Regime	Owning engine	Cfgs	Exact	Note
Low-k on-grid, $k \in [2, 30]$, all N	$O(k \log N)$, $O(\sqrt{N} \log k)$	4,032	4,032	A
High-k on-grid, $k \in [150, 200]$, $N \in [262\text{K}, 4.2\text{M}]$	$O(\sqrt{N} \log k)$	960	960	A, B
Off-grid noisy mid-low-k	$O(k \log N)$	5,760	5,760	C
Off-grid noisy mid-k, $k \in \{16, 32\}$	$O(k \log k)$	864	864	C
Off-grid noisy mid-high-k, $k \in \{50, 64, 100\}$	$O(\sqrt{N} \log k)$	1,296	1,296	C
Real-payload detection sample (argos, nuscenes, oxford_radar)	$O(k \log N)$	9	9	D
Dense FFT fallback	$O(N \log N)$	n/a	n/a	E

A: Path A on-grid; Dense top-k approximate at high k. B: SparseDSP wins on correctness over Dense top-k. C: Path B impairment sweep; binary exact-recovery contract per Section IV-G. D: Path C validated detection slice; reconstruction and clutter-suppression slices in active extension. E: Deterministic fallback; preferred when approximate top-k output is acceptable at low N.

VI. PRODUCTION DEPLOYMENT CONSIDERATIONS

A. Dispatch Policy

The dispatch function is internal to SparseDSP, and its parameters are not part of the externally observable evaluation envelope of this paper. The observable behavior is the engine selection produced by the dispatch function on each input block and the exact-recovery outcome of the selected engine.

The dispatch behavior reported in Section V-C matches the regime applicability summary in Table III on $42/51$ Path C autonomous-selection rows and on $51/51$ rows when the dispatch is forced to the reference selection. Where exact recovery must be guaranteed end-to-end, the system is operated in forced-route mode against the regime table; in autonomous mode the dispatch matches the reference selection on the great majority of configurations, and the boundary disagreements at $N = 65{,}536$, $k = 25$ are surfaced as warnings rather than silent reroutes. Both routes recover exact output when forced, so the disagreement is a dispatch-boundary refinement rather than an exactness failure.

B. Workload Budget Analysis

The workload budget implications of the routing system follow directly from the leading-order complexity classes of the engines listed in Section II-B. At fixed k and large N, the $O(k \log N)$ and $O(\sqrt{N} \log k)$ engine classes reduce the transform-stage operation count proportionally relative to the $O(N \log N)$ Dense baseline, and the regime ownership in Table III indicates which engine class is exercised in each slice.

These reductions are workload-level resource reductions inherited from the complexity classes of the engines. Absolute resource savings on a deployment platform depend on platform memory bandwidth, instruction throughput, and integration cost, and are not reported here. The regime structure (which engine owns which slice and where exact recovery holds) is platform-invariant; the absolute workload budget on any specific deployment platform requires platform-specific re-measurement.

C. Production Caveats

The following caveats apply to any production interpretation of the validated envelope of this paper.

- **Dense FFT fallback.** The Dense FFT path is preserved as a deterministic fallback for regimes where approximate top-k output is acceptable, or as a guaranteed-exact rescue path at a higher cost. Production systems that operate at low N where the dense $N \log N$ work is small in absolute terms may prefer Dense as a conservative default.
- **Selector boundary disagreements.** The $9/51$ Path C autonomous-selection disagreements at $N = 65{,}536$, $k = 25$ require either forced-route deployment against the regime table or further engine-boundary calibration before fully autonomous deployment in those configurations.
- **Active-extension scope.** The full real-payload sparse-refinement envelope across reconstruction and clutter-suppression slices is in active extension and is not validated in this paper; deployment in those modes requires future validation.
- **Platform-dependent absolute cost.** Absolute throughput and latency on any deployment platform are platform-dependent. The evaluation reported here focuses on the platform-invariant regime structure and on relative cost ratios on the reference platform, not on absolute deployment timings.

D. System-Level ROI Translation

Workload-level resource implications follow from the complexity-class differences of Section II-B read against the regime ownership in Table III. At high N and low k/N, sparse engine routing reduces the transform-stage workload by a factor that scales as $N \log N$ over $k \log N$ or over $\sqrt{N} \log k$, depending on which engine owns the slice. At high k, the routing system delivers exact output where the Dense top-k heuristic is approximate, and the relevant production translation is correctness rather than transform-stage cost.

These are workload-level projections, not deployment validations. The mapping from a complexity-class reduction to a system-level resource margin depends on the deployment platform and on the specific pipeline stages that bound the budget. The validated quantities in this paper are the binary exact-recovery percentages and the relative cost ratios on the reference platform; deployment ROI on any specific target platform is a further engineering exercise outside the validated envelope.

VII. Conclusion

This paper validates SparseDSP, a regime-adaptive deterministic sparse FFT engine routing system, as a single externally observable procedure across on-grid, off-grid, and curated real-payload regimes. The validation envelope spans approximately $13{,}000$ configurations partitioned into Path A on-grid synthetic rows, Path B off-grid impairment-sweep rows, and Path C selector-replay and detection-sample rows on six curated real-payload datasets. Within their declared lanes, the engines hold 100% exact recovery on every validated configuration, and Table III consolidates the regime-to-engine mapping with the validated counts. On the Path C real-payload detection sample, the validated $9/9$ exactness is engine-plus-backstop performance certified by a dense-locked exact-recovery backstop; full native sparse refinement on the real-payload regime, together with the in-extension bounded-verifier rescue ($51/51$ forward-looking checkpoint), remains part of the active extension envelope (Section VIII, item 7).

The trade-off against the Dense FFT baseline is exactness over the magnitude-based top-k heuristic in the high-k regime, and routing in place of an external sparsity oracle in the low-k regime. Algorithmic constructions for the individual engines are presented in the companion papers [16]–[18], and a sibling regime paper applies the same evaluation framework to active sensing workloads [19]. Full real-payload sparse-refinement validation across reconstruction and clutter-suppression slices is the next milestone of the active extension envelope.

VIII. Limitations

The validated envelope of this paper is bounded by the following limitations.

1) **Platform identity.** The evaluation does not name a deployment platform. Absolute timings on alternate platforms will shift; the regime structure (which engine owns which slice and where exact recovery holds) is platform-invariant by design, but absolute cost ratios are platform-dependent and require platform-specific re-measurement.
2) **Path C scope.** The validated Path C envelope covers selector replay across 51 rows and a sample exact-recovery slice of 9 detection rows on the Argos, NuScenes, and Oxford Radar datasets. Full real-payload sparse refinement across reconstruction and clutter-suppression tasks on the six datasets is in active extension and is not validated here.
3) **Selector boundary disagreements.** In $9/51$ Path C autonomous-selection rows at $N = 65{,}536$ and $k = 25$ within the $O(k \log N)$-owned detection lane, the autonomous dispatch differs from the reference dispatcher selection. Both routes recover exact output when forced; the disagreement is a dispatch-boundary refinement rather than an exactness failure.
4) **Dense FFT lane on Path A.** The Dense FFT path is reported as a baseline reference rather than as a Path A audit row. A full Dense top-k exactness audit

on Path A is not part of this paper; the high-k failure mode of magnitude-based selection is established at the methodological level in Section IV-C and at the regime level in Section V-A.

5) **Domain mappings.** The workload regimes in Section III are signal-property and parameter-space regimes rather than deployment validations. Mappings to specific application domains follow the prior SparseTech evaluation [19] and are literature-grounded interpretations, not deployment claims.
6) **Reproducibility scope.** The reproducible artifacts of this evaluation are the workload definitions, the parameter ranges, the impairment sweep, and the binary exact-recovery percentages. Absolute timings are platform-dependent and are not the focus of this evaluation; the relative ratios reported in Section V are reference-platform ratios.
7) **Real-payload repair mechanism.** The Path C validated detection sample at $N = 65{,}536$, $k = 25$ employs the dense-locked exact-recovery backstop described in Section V-C; native sparse output requires correction on all 9 rows. Full native sparse refinement on the real-payload regime is in active extension; the in-extension bounded-verifier rescue pathway ($51/51$ exact on the broader Path C sample matrix) is a forward-looking checkpoint, not yet production-validated. Reported exactness on this slice is engine-plus-backstop performance, not pure native sparse-engine throughput.

ACKNOWLEDGMENT

The authors gratefully acknowledge the collaborative environment at SparseTech that made this research possible.

The system-level validation presented in this paper is part of the authors' broader research program on deterministic sparse FFT algorithms and dispatch.

REFERENCES

[1] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965.
[2] H. Hassanieh, P. Indyk, D. Katabi, and E. Price, "Simple and practical algorithm for sparse fourier transform," in *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Kyoto, Japan, Jan. 2012, pp. 1183–1194.
[3] H. Hassanieh, P. Indyk, D. Katabi, and E. Price, "Nearly optimal sparse Fourier transform," in *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC)*, New York, NY, USA, May 2012, pp. 563–578.
[4] X. Li and G. Caire, "A new class of deterministic sparse fft algorithms based on chinese remainder theorem," *IEEE Transactions on Signal Processing*, vol. 68, pp. 4050–4065, Jul. 2020.
[5] A. C. Gilbert, M. J. Strauss, and J. A. Tropp, "A tutorial on fast Fourier sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 57–66, 2008.
[6] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.
[7] E. J. Candès and T. Tao, "Decoding by linear programming," *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4203–4215, 2005.
[8] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Transactions on Information Theory*, vol. 53, no. 12, pp. 4655–4666, 2007.
[9] Y. C. Eldar and G. Kutyniok, *Compressed Sensing: Theory and Applications*. Cambridge University Press, 2012.
[10] P. Indyk and M. Kapralov, "Sample-optimal Fourier sampling in any constant dimension," in *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2014, pp. 514–523.
[11] S. Pawar and K. Ramchandran, "R-FFAST: A robust sub-linear time algorithm for computing a sparse DFT," *IEEE Transactions on Information Theory*, vol. 64, no. 1, pp. 451–466, 2018.
[12] M. A. Iwen, "Combinatorial sublinear-time Fourier algorithms," *Foundations of Computational Mathematics*, vol. 10, no. 3, pp. 303–338, 2010.
[13] A. Christlieb, D. Lawlor, and Y. Wang, "A multiscale sub-linear time Fourier algorithm for noisy data," *Applied and Computational Harmonic Analysis*, vol. 40, no. 3, pp. 553–574, 2016.
[14] S. Pawar and K. Ramchandran, "Computing a k-sparse n-length discrete Fourier transform using at most $4k$ samples and $o(k \log k)$ complexity," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, 2013, pp. 464–468.
[15] M. Frigo and S. G. Johnson, "The design and implementation of FFTW3," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, 2005.
[16] A. R. Flouro and S. P. Chadwick, "CRT-based deterministic reconstruction for sparse FFT: First fully deterministic $o(k \log k)$ algorithm via gated Chinese Remainder Theorem," SparseTech Technical Report, available at https://sparse-tech.com/research/oklogk, 2025, sparseTech Technical Report.
[17] A. R. Flouro and S. P. Chadwick, "Deterministic sparse FFT via coprime decimation and iterative peeling: An $o(k \log n)$ algorithm with exact recovery guarantees," SparseTech Technical Report, available at https://sparse-tech.com/research/oklogn, 2025, sparseTech Technical Report.
[18] A. R. Flouro and S. P. Chadwick, "Keyed-gating sparse FFT via structured residue views: Deterministic $o(\sqrt{N} \log k)$ frequency discovery," SparseTech Technical Report, available at https://sparse-tech.com/research/sqrtn, 2025, sparseTech Technical Report.
[19] A. R. Flouro and S. P. Chadwick, "SparseDSP: System-level evaluation of deterministic sparse FFT for radar, sonar, and LiDAR," SparseTech Technical Report, available at https://sparse-tech.com/research/sparsedsp-regime-radarsonarlidar, 2025, sparseTech Technical Report.
[20] C. Shepard, H. Yu, N. Anand, E. Li, T. Marzetta, R. Yang, and L. Zhong, "Argos: practical many-antenna base stations," in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2012, pp. 53–64.
[21] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes, D. Ramanan, P. Carr, and J. Hays, "Argoverse 2: next generation datasets for self-driving perception and forecasting," in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS)*, 2021.
[22] A. Alkhateeb, "DeepMIMO: a generic deep learning dataset for millimeter wave and massive MIMO applications," in *Proceedings of the Information Theory and Applications Workshop (ITA)*, San Diego, CA, USA, 2019, pp. 1–8.
[23] NOAA National Centers for Environmental Information, "NEXRAD next generation weather radar level-II base data," https://www.ncei.noaa.gov/products/radar/next-generation-weather-radar, 2024, public weather radar dataset.
[24] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: a multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 621–11 631.
[25] D. Barnes, M. Gadd, P. Murcutt, P. Newman, and I. Posner, "The Oxford Radar RobotCar dataset: a radar extension to the Oxford RobotCar dataset," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France, 2020, pp. 6433–6438.
[26] RustFFT Contributors, "RustFFT: a high-performance FFT library written in pure Rust," https://github.com/ejmahler/RustFFT, 2024, version 6.x; baseline used for Dense FFT comparisons.