Deployment Design: Real-Time mmWave Radar AI Pipeline

1. Real-Time Pipeline Overview

1. Radar Front-End: FMCW mmWave radar emits chirps; ADC captures IF signals.
2. Frame Buffer: Raw samples per chirp aggregated into frames.
3. Preprocessing: Windowing, range FFT, Doppler FFT, optional angle FFT (if MIMO).
4. Power / Log-Square Scaling: Normalize dynamic range; convert to dB scale.
5. Clutter / Noise Suppression: Static background subtraction + adaptive threshold (CFAR-like).
6. Feature Construction: Range-Doppler (and optionally angle) heatmaps; temporal stacking.
7. Inference: Metal vs Non-Metal classifier + Hidden object detector logic.
8. Post-Processing: Confidence thresholding, track association (Kalman filter) for persistent detection.
9. Output: Events / bounding boxes / heatmap overlays to UI.

2. Preprocessing Details

- **Range FFT**: Apply window (Hann) per chirp; N_r bins.
- **Doppler FFT**: Across chirps in frame; N_d bins.
- **Angle FFT (Optional)**: Across virtual antennas for azimuth.
- **Calibration**: DC offset removal; IQ imbalance correction.
- **Normalization**: Per-frame min-max or percentile-based scaling.

3. Model Flow

1. Input heatmap (R x D) resized/padded to model size (e.g., 64x64).
2. CNN Extracts spatial patterns (metal returns sharper, higher-intensity cluster).
3. Hidden Object Detection adds: Background model + residual map; residual fed to classifier or anomaly threshold.
4. Decision Fusion: Metal classifier score + residual anomaly metric -> final hidden detection flag.

4. Hidden Object Detection Logic

- Maintain running average background heatmap (exponential decay).
- Compute residual: |Current - Background|.
- Apply morphological opening to remove speckle.
- If classifier predicts metal OR residual peak > adaptive threshold in occluded region -> hidden metal candidate.

5. Performance Considerations

- Batch frames (micro-batching) to leverage GPU.
- Use mixed precision (FP16) for CNN inference.
- Pre-allocate FFT buffers; reuse twiddle factors.
- Employ ring buffer for background model.

6. Latency Targets (Example)

- Chirp acquisition: < 5 ms
- FFT chain (range+doppler): < 3 ms (GPU / optimized C)
- Heatmap normalization + suppression: < 2 ms
- CNN inference: < 4 ms (quantized) => Total < 15 ms/frame.

7. Limitations

- Synthetic data gap vs real radar signatures.
- No multipath / micro-Doppler modeling in current simulation.
- Hidden object approach simplistic; does not leverage learned occlusion features.
- Potential overfitting to intensity profile vs robust physical features.

8. Improvement Roadmap

Phase	Enhancement	Impact
1	Integrate real radar sample ingestion	Data fidelity
1	Implement CFAR (Cell Averaging / OS-CFAR)	False alarm control
2	Add angle FFT + clustering (DBSCAN)	Spatial localization
2	Temporal tracking (Kalman + gating)	Stability
3	Semi-supervised domain adaptation	Real-world robustness
3	Quantization + pruning (INT8)	Edge deployment
4	Micro-Doppler feature branch	Finer object characterization

9. Security & Reliability

- Input validation: Frame dimension & power sanity checks.
- Drift Detection: Monitor background variance; auto recalibrate.
- Logging: Frame-wise inference latency, detection confidence histograms.
- Failover: If CNN unavailable, fallback to rule-based (threshold + CFAR peaks).

10. Edge Deployment Strategy

- Hardware: Embedded GPU (Jetson) or ARM + DSP for FFT.
- Pipeline as modular services: Acquisition, FFT, Suppression, Inference.
- Zero-copy buffers between FFT output and model input.
- Telemetry stream (MQTT) for aggregated detection events.

11. Summary

The proposed pipeline converts raw FMCW radar chirps into cleaned range-Doppler (and optionally angle) representations, applies learned and heuristic logic to classify metal objects and flag hidden metallic returns. Scalability and real-world robustness hinge on improved data realism, adaptive thresholding, and temporal modeling.