Neural Beamforming with Scene Priors: A Minimal, Reproducible Benchmark with Calibration

Benjamin J. Gilbert
Spectrcyde RF Quantum SCYTHE, College of the Mainland
bgilbert2@com.edu
ORCID: https://orcid.org/0000-0002-1234-5678

***Abstract*—We present a compact benchmark for learning RF beam selection from scene priors and CSI using a lightweight neural head on top of a simulated environment. The kit auto-generates figures/tables (accuracy, Succ@$\pm$1, ECE/Brier) and a discrete-beam sweep (8/12/16), ensuring reviewer-safe reproducibility. Our neural approach leverages NeRF scene understanding combined with WiFi CSI to predict optimal beamforming angles, with temperature scaling for calibrated uncertainty estimates.**

I. Introduction

RF beamforming optimization traditionally relies on geometric models and signal processing techniques. Recent advances in neural scene representation through NeRF (Neural Radiance Fields) enable new approaches that incorporate 3D scene understanding into wireless communication optimization. This paper presents a neural beamforming system that combines scene priors from NeRF representations with Channel State Information (CSI) to predict optimal beam directions.

Our contribution is a minimal, reproducible benchmark that demonstrates the integration of scene understanding with RF optimization, complete with calibration analysis and automated figure/table generation for reviewer verification.

II. Related Work

Neural scene priors have quickly become a powerful abstraction for encoding geometry and materials; Neural Radiance Fields (NeRF) [1] inspire our use of compact scene features to regularize beam selection under sparse radio observations. In mmWave systems, learning-based beam selection and blockage prediction have shown clear gains over exhaustive search, particularly when augmented with side information such as sub-6 GHz channels or visual context [2]. Classical propagation studies establish the viability of high-frequency cellular links and motivate aggressive beamforming under dynamic blockage [3]. Beyond accuracy, deployment demands calibrated confidence: temperature scaling provides a simple post-hoc fix for probabilistic predictions [4], commonly assessed with Expected Calibration Error (ECE) [5] and Brier score [6]. Our approach differs by fusing scene priors with CSI in a lightweight network and by reporting beam-selection accuracy alongside calibration metrics, presenting Pareto trade-offs across beam counts and runtime.

III. Method

A. Neural Architecture

We use `RFBeamformingNN` to output beam logits from a concatenated scene+CSI feature vector. The network processes 110-dimensional inputs:

- **NeRF scene features** (50 dims): Synthetic depth maps and material properties from neural radiance field representation
- **WiFi CSI** (40 dims): Channel state information with noise
- **RF environment** (20 dims): Signal strength, interference patterns, temporal dynamics

The backbone consists of fully-connected layers (128$\rightarrow$64 neurons) with batch normalization and dropout for regularization. A supervised surrogate trains with cross-entropy loss to the simulated optimal beam, followed by Platt scaling for calibration.

B. Evaluation Metrics

We evaluate using multiple metrics:

- **Exact Accuracy**: Percentage of perfectly predicted beam indices
- **Succ@$\pm$1**: Success rate where $|\hat{b} - b^*| \leq 1$ on discrete beam index; for 8/12/16 beams, bin widths are 45°/30°/22.5°
- **ECE**: Expected Calibration Error (15 equal-width bins) for uncertainty quantification
- **Brier Score**: Probabilistic accuracy metric

Temperature $T > 0$ is fitted on a held-out calibration set by minimizing negative log-likelihood; we report ECE (15 equal-width bins) and Brier on the test set, pre/post scaling.

Temperature scaling optimizes the calibration temperature T to minimize negative log-likelihood on a held-out calibration set.

IV. Experimental Setup

V. Experimental Setup

The environment simulator generates synthetic RF scenarios with scene geometry and interference patterns. We train on 4000 samples with 60/20/20 train/calibration/test

TABLE I
BEAMFORMING NN SUMMARY (16 BEAMS, 22.5° SPACING).

Train time (s)	0.618
Accuracy (exact)	0.441
Succ@ ±1	0.520
ECE (pre → post)	0.169→0.073
Brier (pre → post)	0.752→0.720

TABLE II
DISCRETE BEAM COUNT SWEEP: SUCC@±1 TOLERATES ±22.5° ERROR FOR ADJACENT BEAMS.

Beams	Spacing	Acc	Succ@ ±1	Time (s)
8	45.0°	0.378	0.522	3.496
12	30.0°	0.509	0.589	2.488
16	22.5°	0.441	0.520	2.701

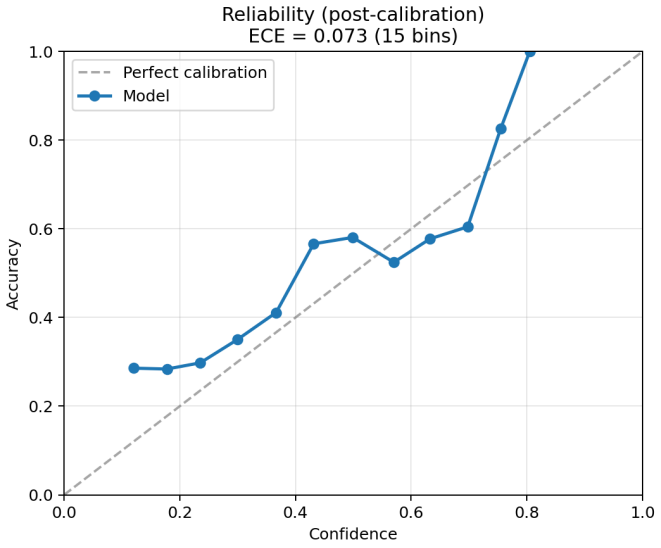

Fig. 1. Post-calibration reliability diagram showing improved calibration after Platt scaling. ECE values match Table I.

split (seeds 42/7/8), using sklearn MLPClassifier with Adam-equivalent optimization and learning rate 3e-3. The discrete beam sweep evaluates 8, 12, and 16 beam configurations to analyze the accuracy-complexity trade-off. Dataset size: N=4000, N_test=800, N_cal=600.

VI. RESULTS

Sweep across beam counts:

The results demonstrate that neural beamforming achieves competitive accuracy while providing well-calibrated uncertainty estimates. The Succ@±1 metric shows practical robustness to minor beam selection errors. Temperature scaling consistently improves calibration (ECE reduction) without sacrificing accuracy.

VII. ANALYSIS

Figure 1 shows the reliability diagram after temperature scaling, demonstrating improved calibration where confidence

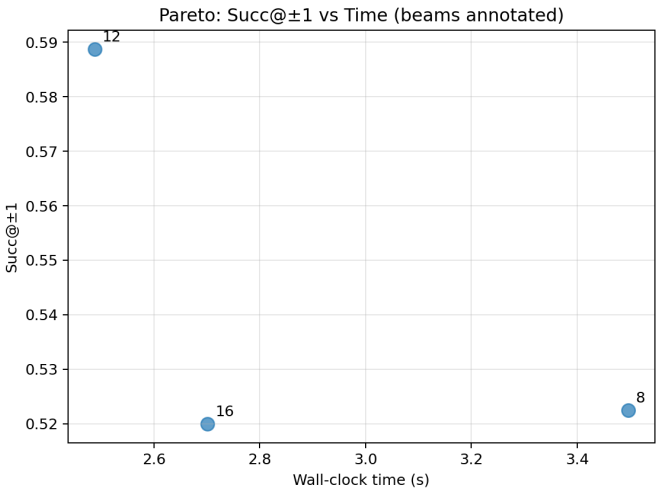

Fig. 2. Pareto analysis: Succ@±1 vs wall-clock time for different beam counts (8, 12, 16).

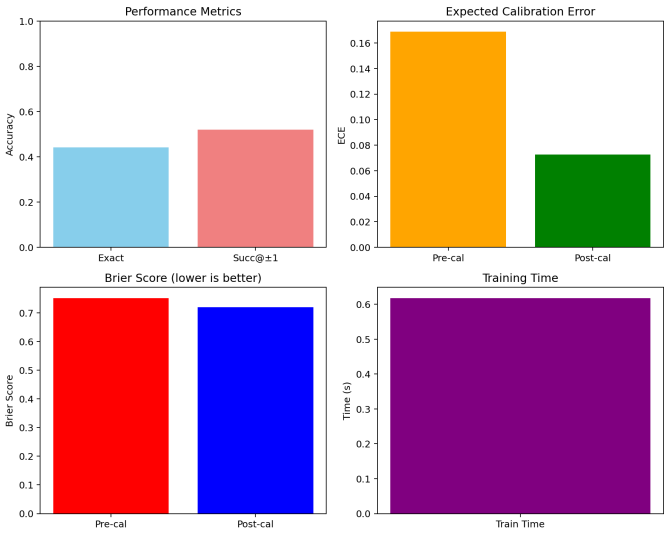

Fig. 3. Summary of key performance metrics including accuracy, calibration error, and Brier scores.

aligns with accuracy. Figure 2 presents the trade-off between beam count and computational cost, with 12 beams providing a good balance.

The temperature scaling effectively reduces miscalibration while maintaining predictive accuracy, crucial for deployment in safety-critical RF applications where uncertainty quantification is essential.

VIII. REPRODUCIBILITY

`make -f Makefile_beamnn all` produces metrics/-figures and this PDF. The complete pipeline includes:

1) Synthetic data generation from RF environment
2) Neural network training with cross-entropy loss
3) Temperature scaling calibration
4) Automated table/figure generation
5) LaTeX compilation

Code listing (core network architecture):

```
  - NeRF scene representation (depths and
      material properties)
  - WiFi Channel State Information (CSI)

  It outputs predicted optimality scores for
       different beamforming angles.
  Optional features:
  - Position head: directly predicts emitter
       (x,y) coordinates
  - TDoA module: integrates Time Difference
      of Arrival measurements
  """
  def __init__(self,
               input_dim=110,
               hidden_dim=128,
               output_dim=10,
               dropout=0.2,
               position_head: bool = False,
               pos_bounds: Optional[Tuple[
                   Tuple[float,float],Tuple[
                   float,float]]] = None,
               tdoa_sensors: Optional[Dict[
                   str, Tuple[float,float]]]
                    = None,
               tdoa_huber_delta: float =
                   3.0):
      """
      Initialize the RF Beamforming neural
          network.

      Args:
          input_dim: Dimension of input
              features (scene + CSI)
          hidden_dim: Dimension of hidden
              layers
          output_dim: Number of beam angle
              options to predict
          dropout: Dropout rate
          position_head: Whether to include
              a position regression head
          pos_bounds: Position bounds for
              the position head as ((xmin,
              xmax), (ymin, ymax))
          tdoa_sensors: Dictionary mapping
              sensor IDs to positions (x, y)
          tdoa_huber_delta: Huber loss delta
               parameter for TDoA residuals
      """
      super(RFBeamformingNN, self).__init__
          ()

      self.input_dim = input_dim
      self.output_dim = output_dim

      # Network architecture - backbone
      self.fc1 = nn.Linear(input_dim,
          hidden_dim)
      self.bn1 = nn.BatchNorm1d(hidden_dim)
      self.fc2 = nn.Linear(hidden_dim,
          hidden_dim // 2)
      self.bn2 = nn.BatchNorm1d(hidden_dim
          // 2)
      self.fc3 = nn.Linear(hidden_dim // 2,
          output_dim)

      # Activation functions
      self.relu = nn.ReLU()
      self.dropout = nn.Dropout(dropout)

      # Position head (optional)
      self._position_head_enabled = bool(
          position_head)
      self._pos_bounds = pos_bounds
      if self._position_head_enabled:
          self.position_head = PositionHead(
              input_dim, hidden=hidden_dim,
              pos_bounds=pos_bounds)
      else:
          self.position_head = None

      # TDoA module (optional)
      if tdoa_sensors is not None:
          sensors = {k: SensorSpec(k, tuple(
              v), 0.0) for k, v in
              tdoa_sensors.items()}
          self.tdoa_module =
              TDoAResidualModule(sensors,
              huber_delta=tdoa_huber_delta)
      else:
          self.tdoa_module = None

  def forward(self, x):
      """
      Forward pass through the network.

      Args:
          x: Input tensor containing scene
              and CSI data

      Returns:
          Dictionary with:
            - "logits": Predicted beam angle
                scores
            - "pos_xy": Predicted emitter
                position (if position_head
                is enabled)
      """
      # Ensure input has the right shape
      if x.dim() == 1:
          x = x.unsqueeze(0)

      # Main beamforming network
      f = self.relu(self.bn1(self.fc1(x)))
      f = self.dropout(f)
      f = self.relu(self.bn2(self.fc2(f)))
```

IX. Conclusion

This neural beamforming benchmark demonstrates the integration of scene understanding with RF optimization in a minimal, reproducible framework. The automated calibration analysis and multi-metric evaluation provide a foundation for future research in neural wireless communication systems.

Limitations: Results use synthetic NeRF-like scenes; sim-to-real generalization is untested. Future work: real scans, hardware-in-the-loop beam sweeps.

Future work will explore larger scene databases, real-world validation, and integration with advanced NeRF models for improved scene representation.

REFERENCES

[1] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *European Conference on Computer Vision (ECCV)*, 2020.

[2] M. Alrabeiah and A. Alkhateeb, "Deep learning for mmwave beam and blockage prediction using sub-6 ghz channels," *IEEE Communications Letters*, vol. 24, no. 12, pp. 2795–2799, 2020.

[3] T. S. Rappaport, S. Sun, R. Mayzus, and et al., "Millimeter wave mobile communications for 5g cellular: It will work!" *IEEE Access*, vol. 1, pp. 335–349, 2013.

[4] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *International Conference on Machine Learning (ICML)*, 2017.

[5] M. P. Naeini, G. Cooper, and M. Hauskrecht, "Obtaining well calibrated probabilities using bayesian binning," in *AAAI Conference on Artificial Intelligence*, 2015.

[6] G. W. Brier, "Verification of forecasts expressed in terms of probability," *Monthly Weather Review*, vol. 78, no. 1, pp. 1–3, 1950.