Iterative Architectural Refinement of U-Net Models for Urban Radio Coverage Prediction

Youfu Fan
M.S. in Telecommunications
University of Maryland, College Park
College Park, MD, USA
Email: ykf5030@umd.edu
https://www.youfufan.top/radio-coverage/

***Abstract*—Predicting how a base station's signal spreads across a city is a prerequisite for cellular network planning. Classical empirical formulas are fast but ignore building geometry, while deterministic ray-tracing simulations are accurate but slow. Learning-based radio map estimation seeks a third option: a neural network that approximates the simulator in milliseconds. This paper presents an empirical case study that builds such a model from scratch and refines it through four successive versions on the public RadioMapSeer dataset (5.9 GHz urban propagation). Rather than reporting a single trained model, we treat each version as a diagnose-and-fix iteration. A baseline U-Net (v1) exhibits a far-field "square cutoff" that we trace to a limited receptive field; a distance-to-transmitter input channel (v2) removes it but exposes checkerboard artifacts from transposed-convolution upsampling; resize-convolution upsampling (v3) removes the checkerboard but leaves a residual fading of long unobstructed corridors; and a cascaded two-stage network (v4, in the spirit of RadioUNet) with deep supervision and stage-wise warm-starting resolves the fade and roughly halves the error of v3. The final model reaches a root-mean-square error of about 1.7 dB on held-out cities. We accompany the quantitative results with a receptive-field analysis that explains, quantitatively, why each failure mode appears and why each fix works, and we deploy the final model for client-side inference in the browser via ONNX Runtime Web. The contribution is methodological: a transparent, reproducible diagnose-then-fix workflow that runs end to end on a single consumer GPU.**

***Index Terms*—Radio map estimation, path loss prediction, U-Net, convolutional neural networks, receptive field, deep learning, wireless network planning.**

I. Introduction

Before deploying a base station, a mobile operator must estimate where its signal will be strong and where buildings will cast "shadows" that block it. Two families of tools dominate practice. The first comprises *empirical* models such as the Hata model [1], which express path loss as a closed-form function of distance and a few environment parameters. They are extremely fast but blind to the specific building layout, so they cannot capture shadowing or street-canyon effects. The second comprises *deterministic* solvers, principally ray tracing, which simulate electromagnetic propagation through a 3-D scene. They are accurate but computationally expensive, often requiring minutes per map, which makes large-scale or interactive use impractical.

Learning-based radio map estimation offers a third path. If a convolutional neural network (CNN) can be trained on the input–output pairs produced by a slow simulator, it may reproduce the simulator's accuracy at a small fraction of the runtime cost. RadioUNet [3] demonstrated this convincingly, framing coverage prediction as an image-to-image regression problem and solving it with U-Net–style networks trained on the RadioMapSeer dataset [4].

This paper does not propose a new architecture. Instead, it reports an honest, reproducible *engineering case study*: we build a coverage predictor from scratch on a single consumer GPU and improve it through four versions, where each version is motivated by a concrete, observable failure of its predecessor. We argue that for practitioners learning to apply deep learning to a domain problem, the *path* a model takes is at least as instructive as its final score. A model that simply reports "1.7 dB RMSE" conceals the reasoning that produced it; a model whose every change is tied to a diagnosed failure and a literature-grounded remedy is reproducible and teachable. Accordingly, our contributions are:

- A four-stage refinement of a U-Net coverage predictor (v1–v4), in which each architectural change targets one specific, diagnosed failure mode rather than chasing a metric blindly.
- A receptive-field analysis that explains, with explicit layer-by-layer arithmetic, why a single-pixel transmitter cannot influence the whole map, and how the distance channel and the cascaded network circumvent that limit.
- An ablation, induced naturally by the version sequence, isolating the contribution of the distance channel, resize-convolution upsampling, and the cascade with deep supervision.
- A fully reproducible pipeline—training, evaluation, and ONNX export on a single 8 GB GPU—together with a client-side, in-browser demonstration that runs the final model with no server backend.

The remainder of the paper is organized as follows. Section II reviews propagation models and learning-based estimation. Section III formalizes the task and Section IV defines the key concepts. Section V describes the dataset, baseline, and metrics. Section VI presents the four-stage refinement, which is the core of the paper, and Section VII gives implementation details. Section VIII reports results, Section IX describes deployment, and Sections X and XI discuss limitations and conclude.

II. Background and Related Work

A. Empirical and Deterministic Propagation Models

Empirical models fit measured path loss to simple functions of distance and frequency. The Hata model [1] and the log-distance model are canonical examples [2]. Because they depend only on distance (and coarse environment categories), they are fast and robust but cannot represent the effect of an individual building: two transmitters at the same distance but on opposite sides of a tower are predicted identically. Importantly, the Hata model is calibrated only for carrier frequencies below roughly 1.5 GHz, which makes it inapplicable at the 5.9 GHz carrier of the dataset used here; we therefore adopt a fitted log-distance model as the classical baseline. At the other extreme, deterministic solvers such as ray tracing and the dominant-path model (DPM) trace propagation paths through the scene geometry and produce accurate maps, at a cost that scales poorly with scene size and resolution.

B. Learning-Based Radio Map Estimation

Casting coverage prediction as image-to-image regression connects it to a large body of dense-prediction work, including conditional image translation [13]. RadioUNet [3] is the most directly relevant: it showed that U-Net–style networks can approximate ray-traced maps with single-digit-decibel error, and that a *cascade* of two U-Nets—the second refining the first—improves accuracy further. The accompanying RadioMapSeer dataset [4] provides tens of thousands of simulated maps over real city layouts and has become a standard benchmark. Our final model (v4) adopts the cascaded design of RadioUNet; the present work's emphasis is the staged path that leads there and the analysis that justifies each intermediate step.

C. Relevant Deep-Learning Components

The U-Net [5] is an encoder–decoder with skip connections, originally introduced for biomedical segmentation and now ubiquitous for dense prediction; we adapt it for regression by using a sigmoid output paired with mean squared error and batch normalization [12] for stable training. Transposed convolutions used for upsampling can introduce periodic "checkerboard" artifacts, an effect analyzed by Odena *et al.* [6]; replacing them with resize-then-convolve operations removes the artifact. The notion of the *effective* receptive field—typically far smaller than the theoretical one—was characterized by Luo *et al.* [7] and is central to our analysis. Deep supervision [8] attaches auxiliary losses to intermediate outputs and stabilizes the training of deeper or cascaded models. Approaches to enlarging the receptive field include deeper residual stacks [15] and dilated convolutions [14].

III. Problem Formulation

We treat coverage prediction as image-to-image regression. Let $B \in [0,1]^{H\times W}$ be a binary building map (1 inside buildings) and $T \in [0,1]^{H\times W}$ a transmitter map that is 1 at the transmitter pixel and 0 elsewhere, with $H = W = 256$. The target $Y \in [0,1]^{H\times W}$ is the simulated coverage (path-loss) map, where bright denotes strong signal. We seek a network f_θ minimizing the pixel-wise mean squared error (MSE)

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \big\| f_\theta(X_i) - Y_i \big\|_2^2, \tag{1}$$

where X_i stacks the input channels and N is the number of pixels in a batch. The input is $X = [B, T]$ for the two-channel model and $X = [B, T, D]$ for the distance-augmented models, with D defined in Section VI.

The choice of MSE is natural for a continuous, per-pixel target and pairs cleanly with a sigmoid output. It carries one consequence worth stating explicitly: where the model is uncertain, the loss-minimizing prediction is the conditional mean, which tends to be smoother and dimmer than any single plausible map. This "regression to the mean" reappears as a concrete failure mode in Section VI.

a) Decibel conversion: RadioMapSeer encodes a path-loss dynamic range of approximately 80 dB into the $[0,1]$ gray scale. Following the dataset convention [3], [4], the gray-level RMSE is a linear rescaling of the path-loss RMSE in decibels, so we report

$$\mathrm{RMSE}_{\mathrm{dB}} \approx 80 \times \mathrm{RMSE}_{\mathrm{gray}}. \tag{2}$$

b) Generalization protocol: To measure generalization honestly we split *by city map*, not by random image: the 700 maps are partitioned into disjoint training, validation, and test cities (Section V). A model is therefore always evaluated on building layouts it has never seen during training, which prevents the optimistic bias that arises when different transmitters of the same city appear in both training and test sets.

IV. Preliminaries

We briefly define the concepts the paper relies on.

a) Coverage / path-loss map: A per-pixel estimate of received signal strength (equivalently, path loss) for a fixed transmitter. We normalize it to $[0,1]$ and recover decibels via (2).

b) Receptive field: The receptive field (RF) of an output pixel is the set of input pixels that can influence it. For a stack of layers with kernel sizes k_ℓ and strides s_ℓ, the RF grows as

$$r_\ell = r_{\ell-1} + (k_\ell - 1)\prod_{j<\ell} s_j, \tag{3}$$

with $r_0 = 1$. The *theoretical* RF computed from (3) is an upper bound; the *effective* RF—the region that meaningfully affects the output—is typically much smaller and decays from the center, often growing only as the square root of the number of layers [7]. A worked derivation for our encoder appears in Appendix A.

c) U-Net: An encoder that repeatedly halves resolution while increasing channels, a bottleneck, and a decoder that restores resolution. Skip connections copy encoder features to the matching decoder level so that fine spatial detail is not lost in compression [5].

TABLE I
MAP-DISJOINT DATASET SPLIT (BY CITY).

Split	Map IDs	# Maps	# Samples
Train	0–499	500	∼40,000
Val	500–599	100	∼8,000
Test	600–699	100	∼8,000

d) Transposed vs. resize convolution: Transposed convolution upsamples by inserting learned values; when the kernel size is not divisible by the stride, adjacent outputs receive uneven numbers of contributions, stamping a periodic grid (the checkerboard artifact) [6]. Resize-convolution instead upsamples by a fixed interpolation (here, bilinear) and then applies an ordinary convolution, which is spatially uniform and artifact-free, and also exports cleanly to inference runtimes.

e) Deep supervision: Attaching an auxiliary loss to an intermediate prediction so that earlier stages receive a direct training signal [8], which both stabilizes optimization and forces the intermediate output to be meaningful in its own right.

f) Cascaded U-Net (WNet): Two U-Nets in series: the first produces a coarse estimate, the second refines it. Because the second network's input already contains a global estimate, each of its pixels effectively "sees" the whole map, which enlarges the effective receptive field without requiring a single enormous network.

V. DATASET, BASELINE, AND METRICS

A. Dataset

We use RadioMapSeer [4]: 700 real city maps, each with 80 transmitter positions, yielding more than 56,000 ray-traced coverage maps at 256×256 resolution and a 5.9 GHz carrier. Three image folders supply the model's tensors: a building map shared across a city's 80 transmitters, a per-transmitter location map, and the per-transmitter gain map used as the target. We train against the dominant-path-model (DPM) gain maps, which are cheaper to generate than the full intelligent-ray-tracing (IRT) variants and are sufficient to study the architectural questions of interest. All images are loaded as grayscale and normalized to $[0, 1]$. Because the building map is reused across 80 transmitters, we cache it in memory to reduce I/O. Table I lists the map-disjoint split.

B. Classical Baseline

As a fair classical reference we fit a distance-only log-distance model $g(d) = a + b \log_{10}(d)$ by least squares on training samples, where d is the Euclidean distance from each pixel to the transmitter. By construction this baseline cannot see buildings; the gap between it and the learned models quantifies the value the network adds. We use the log-distance form rather than Hata because Hata is not valid at 5.9 GHz.

TABLE II
PER-VERSION CONFIGURATION. "DECONV" = TRANSPOSED CONVOLUTION; "RESIZE" = BILINEAR UPSAMPLE + CONVOLUTION.

Ver.	Input ch.	Upsampling	Stages	Params
v1	2 (B, T)	deconv	1 U-Net	∼12.8M
v2	3 (B, T, D)	deconv	1 U-Net	∼12.8M
v3	3 (B, T, D)	resize	1 U-Net	12.77M
v4	3 (B, T, D)	resize	2 U-Nets	25.55M

C. Metrics

We report RMSE in normalized gray levels and in decibels via (2), plus the relative improvement over the baseline. Cross-version comparisons (v1–v4) are reported on the validation split, which is held out by city, for a strictly apples-to-apples comparison; the learned-vs-classical comparison is reported on the test split.

VI. METHOD: FOUR-STAGE ITERATIVE REFINEMENT

This section is the core of the paper. Each version makes one architectural change motivated by a diagnosed failure of its predecessor. Figure 1 summarizes the four architectures, Table II lists their configurations, and Fig. 2 compares their predictions on a single held-out scene.

A. v1: Baseline U-Net

Design. v1 is a standard U-Net with a two-channel input $[B, T]$. The encoder applies a double 3×3 convolution block (each followed by batch normalization [12] and ReLU) and four max-pooling downsampling stages ($256 \to 128 \to 64 \to 32 \to 16$), doubling channels from a base of 64 until the bottleneck. The decoder mirrors the encoder with transposed-convolution upsampling and skip connections, and a final 1×1 convolution with a sigmoid maps to the $[0, 1]$ coverage map. The model has about 12.8M parameters.

Result. On the test cities v1 attains 4.11 dB RMSE versus 14.30 dB for the log-distance baseline—a 71.3% reduction in error—confirming that the network learns building shadowing the classical model cannot represent.

Failure. Predictions exhibit a far-field "square cutoff": coverage is reasonable near the transmitter but truncates into a roughly square region, beyond which the far field is blank.

Diagnosis. Three observations localize the cause. (i) The training and validation loss curves plateau early with a small, stable gap, indicating a representational ceiling rather than under-training or over-fitting. (ii) Probing encoder feature maps shows the transmitter registers as a *local* bright patch at every level, including the bottleneck; it never spreads spatially. (iii) The receptive-field arithmetic of (3), tabulated in Table III, shows the bottleneck RF is only about 140 px and the output RF about 200 px, both smaller than the 256 px image width and far smaller than its 362 px diagonal. Because the effective RF is smaller still [7], a single transmitter pixel simply *cannot* influence distant outputs. The square is the footprint of the effective receptive field.

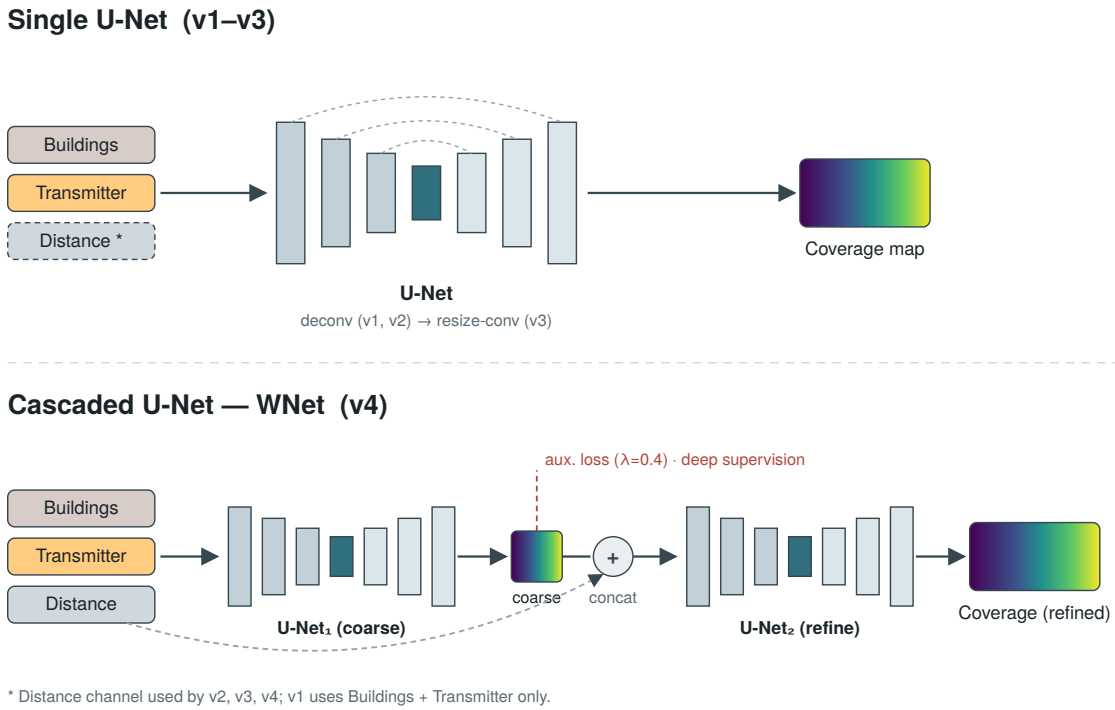

Fig. 1. The four model versions. v1–v3 share a single U-Net, differing only in input channels and upsampling; v4 cascades two U-Nets with deep supervision, the refiner (U_2) taking the input concatenated with the coarse map. The distance channel is used by v2–v4.

TABLE III
THEORETICAL RECEPTIVE-FIELD GROWTH THROUGH THE ENCODER (KERNEL 3, POOLING 2). DERIVATION IN APPENDIX A.

Stage	Output size	Receptive field (px)
input	256	1
inc	256	5
down1	128	14
down2	64	32
down3	32	68
down4 (bottleneck)	16	140
decoder → output	256	∼200

B. v2: Distance-to-Transmitter Channel

Design. Rather than forcing the network to propagate the transmitter's influence across many layers, we hand every pixel its distance to the transmitter directly. We add a third input channel

$$D(y,x) = \frac{\sqrt{(y-r)^2 + (x-c)^2}}{\sqrt{2}\,H}, \tag{4}$$

where (r, c) is the transmitter location, normalized to approximately $[0, 1]$. The input becomes $[B, T, D]$. The channel is computed on the fly and adds negligible parameters and compute.

Result. The square cutoff disappears: coverage now spans the whole map, with far-field rays and distant building shadows. Validation RMSE improves from 0.052 to 0.034 gray (≈ 2.7 dB).

Failure. With a detailed, smooth far field now being predicted, a faint periodic "checkerboard" texture becomes visible in smooth regions.

Diagnosis. The texture is the transposed-convolution checkerboard artifact [6]. It was present in v1 as well, but v1's blank far field offered no smooth gradient to reveal it; fixing the cutoff merely *exposed* a pre-existing artifact. This is a useful reminder that solving one failure can unmask another that was always present.

C. v3: Resize-Convolution Upsampling

Design. We replace each transposed convolution in the decoder with a bilinear upsample followed by a 1×1 convolution (resize-convolution). The upsampling becomes spatially uniform, so no uneven overlap and no grid can form. The change is parameterized by an upsampling-mode switch so that v1/v2 checkpoints remain loadable under the same code base, which also keeps the cross-version comparison controlled.

Result. The checkerboard is removed at the source; smooth regions become clean. Validation RMSE improves to 0.030 gray (≈ 2.4 dB). Resize-convolution mainly changes texture rather than gross accuracy, so the improvement is modest but the qualitative gain is clear.

Failure. A subtler effect remains: along a long, unobstructed corridor the far-field beam, which stays bright to the map edge in the ground truth, fades before reaching the corner in the prediction.

Diagnosis. This is again a receptive-field limitation, now of a finer kind. The distance channel tells each pixel *how far*

it is from the transmitter, but predicting that a long corridor remains bright requires integrating building occlusion along a path longer than the effective RF. Where the model cannot confirm the path is clear, the MSE objective hedges toward a dimmer mean (Section III), so the beam fades.

D. v4: Cascaded U-Net (WNet)

Design. Following RadioUNet [3], v4 places two U-Nets in series. The first, U_1, maps the input to a coarse coverage estimate $C = U_1(X)$. The second, U_2, takes the input concatenated with the coarse map and produces the refined output $\hat{Y} = U_2([X, C])$. Because U_2's input already contains a global coverage estimate, every pixel has long-range context, enlarging the effective receptive field without a single very deep network. We train end-to-end with deep supervision [8],

$$\mathcal{L} = \|\hat{Y} - Y\|_2^2 + \lambda \|C - Y\|_2^2, \quad \lambda = 0.4, \tag{5}$$

so that the coarse stage is itself encouraged to predict coverage rather than arbitrary features. To accelerate convergence we *warm-start* U_1 from the trained v3 weights, since U_1 performs exactly the task v3 already solves; U_2 is trained from scratch. Both stages use the resize-convolution decoder of v3. The model has about 25.5M parameters.

Result. The fading corridor is resolved: coverage stays bright to the map edge, matching the ground truth. Validation MSE drops to 0.00044 (RMSE 0.021 gray, ≈ 1.7 dB), roughly half the error of v3 and approaching the published RadioUNet range. Training shows the expected behavior of a larger, warm-started model: a widening train–validation gap as it approaches the validation minimum, at which point we select the best checkpoint.

VII. Implementation and Training Details

All models are implemented in PyTorch [11] and trained on a single NVIDIA RTX 2080 (8 GB). We use the Adam optimizer [9] with a constant learning rate of 10^{-3}, MSE loss, automatic mixed precision [10], and cuDNN autotuning of convolution algorithms (the input size is fixed). Single U-Nets (v1–v3) train at batch size 32; the cascaded v4 trains at batch size 12 to fit within 8 GB. We checkpoint the model with the lowest validation loss and stop near the validation plateau. The building map is cached in memory to reduce data-loading overhead, which is the main throughput bottleneck on a single consumer GPU.

For deployment, each trained model is exported to ONNX and verified against the PyTorch outputs to within a maximum absolute difference of about 10^{-6}, confirming the exported graph is faithful. Full hyperparameters are listed in Appendix B.

VIII. Experiments and Results

A. Quantitative Results and Ablation

Table IV reports validation RMSE for v1–v4 and the learned-vs-classical comparison. Because each version adds exactly one component to its predecessor, the table doubles as an ablation: the distance channel accounts for the largest single drop (v1→v2), resize-convolution yields a smaller accuracy change but removes the checkerboard texture (v2→v3), and the cascade with deep supervision provides the second-largest drop while fixing the corridor fade (v3→v4). Every version improves monotonically on the previous one, with v4 achieving the lowest error.

TABLE IV
Validation RMSE across versions (held-out cities); decibels via (2). The bottom block gives the learned-vs-classical comparison on the test split. Each row adds one component, so the table also serves as an ablation.

Version	Change	RMSE (gray)	RMSE (dB)
v1	baseline (2-ch)	0.052	∼4.2
v2	+ distance channel	0.034	∼2.7
v3	+ resize-conv	0.030	∼2.4
v4	+ cascade (WNet)	**0.021**	∼**1.7**
Log-distance baseline (test)		0.179	14.30
v1 (test)		0.051	4.11

TABLE V
Computational cost. Peak VRAM is the training peak at the listed batch size on an RTX 2080 (8 GB).

Version	Params	Batch	Peak VRAM
v1–v3 (single U-Net)	∼12.8M	32	∼6.8 GB
v4 (cascade)	25.5M	12	∼4.8 GB

Note. Cross-version numbers use the validation split for a strictly controlled comparison; the released code reproduces matched test-split figures for every version.

B. Computational Cost

Table V reports the practical cost of each model on the single 8 GB GPU. The single U-Nets share essentially the same footprint; the cascade doubles the parameters and, even at the smaller batch size, remains comfortably within memory. Inference of any version, once exported to ONNX, completes in well under a second in a browser (Section IX), versus minutes for the ray-tracing simulator the network approximates.

C. Qualitative Results

Figure 2 compares the four predictions and their absolute-error maps on a held-out test scene. The transmitter sits near the centre with a strong main lobe, and signal propagates through the surrounding streets. The square cutoff is visible in v1; across v1→v4 the absolute error, brightest in the far field for the early versions, grows progressively darker until v4's error map is nearly uniform, consistent with its much lower per-sample RMSE (0.021 versus 0.065 for v1). The progression is evident to the eye even where the gray-level RMSE differences between adjacent versions are modest, underscoring that a single scalar metric does not fully capture perceptual quality.

D. Diagnostic Evidence

Figures 3 and 4 collect the diagnostic artifacts that drove the design: the v1 training/validation loss curves indicate a repre-

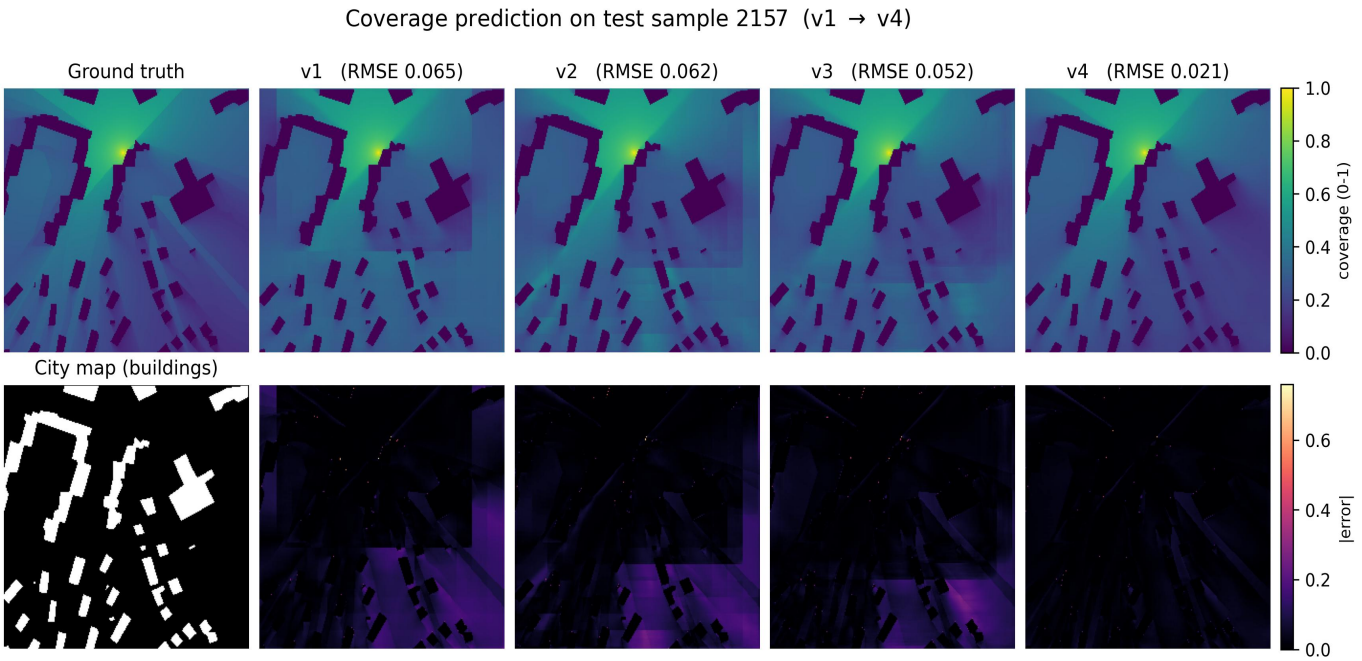

Fig. 2. Qualitative comparison on a held-out test scene (sample 2157). Top: ground truth and the v1–v4 predicted coverage maps. Bottom: the city (building) map and the corresponding absolute-error maps (shared scale). Per-sample RMSE is annotated; these per-scene values differ from the dataset-wide figures of Table IV. The square cutoff is visible in v1, and the far-field error shrinks across versions until v4's error map is nearly uniform.

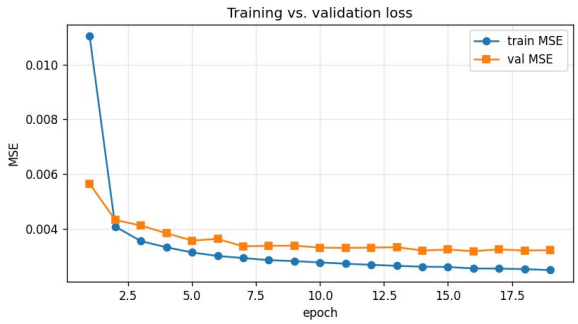

Fig. 3. Training and validation MSE for the baseline (v1). The curves plateau early and track each other with a small gap, indicating a representational ceiling rather than under-training or over-fitting.

sentational ceiling, while the encoder feature maps show the transmitter remaining a local activation through the bottleneck. Together with Table III, they provide a quantitative account of the failure modes rather than a post-hoc rationalization, which is the methodological point of the paper: the fixes were chosen *because* of these observations, not justified after the fact.

IX. Deployment

To show the approach is practical we export each model to ONNX and run inference *client-side* in the browser using ONNX Runtime Web [16]. A bilingual interactive page lets a visitor pick a city layout, click to place a transmitter, and switch between versions v1–v4 to see the prediction update in well under a second, with no server backend. The distance channel is recomputed in the browser from the click location, and the viridis-colored coverage map is rendered on a canvas. Models are loaded lazily—only the selected version is downloaded—which matters because the cascaded model is the largest. Because the entire inference runs locally, the demonstration is private by construction and scales trivially: the marginal server cost of an additional visitor is the static transfer of the model file, which can be cached. For bandwidth-constrained settings, fp16 quantization of the exported graph would roughly halve the download.

X. Discussion, Limitations, and Future Work

Several limitations bound the claims. First, training uses only the DPM targets of a single dataset; transfer to other simulators, frequencies, or measured data is untested, and the gap between simulated and measured propagation is itself an open problem. Second, the cross-version comparison is on the validation split; matched test-split numbers are reproducible from the released code but are not the headline figures here. Third, hardware constraints (a single 8 GB GPU) capped the batch size and model size, and v4 was stopped near a validation plateau rather than trained to exhaustion, so its reported error is a conservative estimate of what the architecture can reach. Fourth, the in-browser model is large ($\sim$100 MB for the cascaded network); quantization or pruning would be needed for truly lightweight deployment.

The analysis points to clear next steps. Growing the effective receptive field further—via additional downsampling levels, dilated convolutions [14], or residual and attention blocks [15]—may close the remaining gap to the published

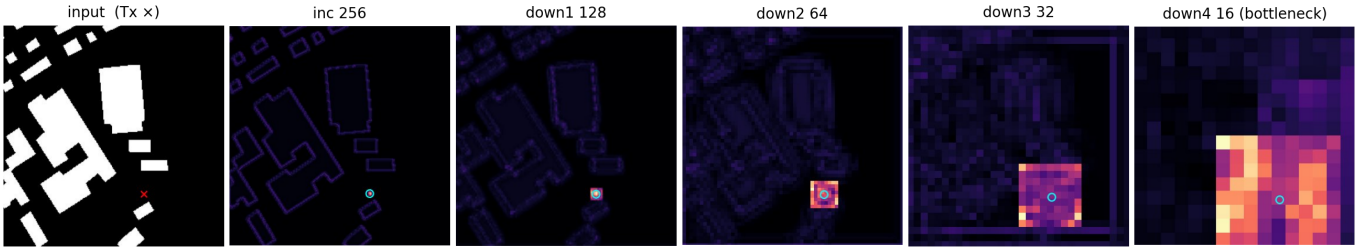

Fig. 4. Encoder feature maps (channel-wise maximum) of the baseline U-Net for one transmitter (marked). From the input through the encoder (inc $256 \rightarrow$ down4 16), the transmitter remains a *local* activation that grows only slowly with depth and never spans the map—visualizing the receptive-field limitation quantified in Table III.

$\sim$1 dB results without resorting to the cascade's parameter doubling. Training against the more accurate IRT targets, and ultimately validating against measurements, would test real-world utility. The regression-to-the-mean behavior we observed suggests that predicting a calibrated *distribution* over coverage, rather than a single MSE-optimal map, could both improve sharpness and provide useful uncertainty estimates for planning. Finally, conditioning on richer inputs (antenna pattern, transmit power, terrain height) would move the model closer to a drop-in replacement for a deterministic solver.

XI. Conclusion

We presented a transparent, reproducible refinement of a U-Net radio coverage predictor across four versions. Each step fixed one diagnosed failure—the far-field square cutoff (via a distance channel), the checkerboard artifact (via resize-convolution), and the fading corridor (via a cascaded, deeply supervised network)—and each fix was justified by an explicit receptive-field or upsampling argument rather than by metric-chasing. The final model reaches about 1.7 dB RMSE on held-out cities and runs in the browser on commodity hardware. We hope the diagnose-then-fix workflow, and the receptive-field analysis that underpins it, are useful to others building image-to-image regressors under modest resource constraints.

Appendix A
Receptive-Field Derivation

Applying (3) layer by layer through the encoder, with all convolutions 3×3 (stride 1) and all pooling 2×2 (stride 2), and tracking the cumulative stride ("jump") J: the two convolutions of the input block give $r = 1+2+2 = 5$ at $J = 1$; the first pooling and its two convolutions give $r = 5+1+4+4 = 14$ at $J = 2$; and continuing in the same manner yields $r = 32$ (J=4), $r = 68$ (J=8), and $r = 140$ at the 16×16 bottleneck (J=16), matching Table III. Carrying the computation through the decoder's convolutions raises the output RF to roughly 200 px. Since the image is 256 px wide and 362 px along the diagonal, even the theoretical RF cannot span the map from a single pixel; reaching the full diagonal from one source pixel would require roughly two additional downsampling levels (a 4×4 bottleneck, RF ≈ 570 px). This quantifies the v1 limitation and motivates both the distance channel (which removes the need to propagate the source) and the cascade (which provides global context to the refiner).

Appendix B
Hyperparameters

Optimizer: Adam, learning rate 10^{-3}, default β's [9]. Loss: MSE; for v4 the deep-supervision weight is $\lambda = 0.4$ (5). Precision: automatic mixed precision (fp16/fp32) [10]. Base channel width: 64. Image size: 256×256. Batch size: 32 (v1–v3), 12 (v4). Distance-channel normalization: $\sqrt{2}\,H$ (4). Upsampling: transposed convolution (v1–v2), bilinear-resize then 1×1 convolution (v3–v4). v4 warm-starts U_1 from the trained v3 checkpoint.

References

[1] M. Hata, "Empirical formula for propagation loss in land mobile radio services," *IEEE Trans. Veh. Technol.*, vol. 29, no. 3, pp. 317–325, 1980.
[2] T. S. Rappaport, *Wireless Communications: Principles and Practice*, 2nd ed. Prentice Hall, 2002.
[3] R. Levie, Ç. Yapar, G. Kutyniok, and G. Caire, "RadioUNet: Fast radio map estimation with convolutional neural networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 6, pp. 4001–4015, 2021.
[4] Ç. Yapar, R. Levie, G. Kutyniok, and G. Caire, "Dataset of pathloss radio maps for urban environments (RadioMapSeer)," 2022.
[5] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. MICCAI*, 2015, pp. 234–241.
[6] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and checkerboard artifacts," *Distill*, 2016.
[7] W. Luo, Y. Li, R. Urtasun, and R. Zemel, "Understanding the effective receptive field in deep convolutional neural networks," in *Proc. NeurIPS*, 2016.
[8] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Proc. AISTATS*, 2015, pp. 562–570.
[9] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.
[10] P. Micikevicius *et al.*, "Mixed precision training," in *Proc. ICLR*, 2018.
[11] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. NeurIPS*, 2019.
[12] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. ICML*, 2015, pp. 448–456.
[13] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. CVPR*, 2017, pp. 1125–1134.

[14] F. Yu and V. Koltun, “Multi-scale context aggregation by dilated convolutions,” in *Proc. ICLR*, 2016.
[15] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. CVPR*, 2016, pp. 770–778.
[16] ONNX Runtime developers, “ONNX Runtime,” https://onnxruntime.ai, 2021.