UMR - GAN: Early Implementation and Evaluation on 4-Class Brain MRI Restoration

Aslesha Sanjana Kodavali
University of Florida
Gainesville, FL, USA
akodavali@ufl.edu

***Abstract*—We present the first end-to-end prototype of *UMR-GAN*, a mask-conditioned conditional GAN (cGAN) for brain MRI restoration. The system ingests four-class MRI data (glioma, meningioma, pituitary, no-tumor), applies standardized preprocessing, and restores slices degraded by noise or local masking. A Colab/Gradio interface enables interactive inference for denoising and inpainting. On a held-out set, UMR-GAN improves PSNR by 4.04 dB and SSIM by 0.169 over a supervised baseline, and qualitative triplets show sharper tissue boundaries with reduced speckle. This preliminary report documents the architecture, implementation, UI, evaluation protocol, challenges, and Responsible-AI considerations.**

***Index Terms*—GAN, MRI restoration, denoising, inpainting, PatchGAN, PyTorch, Gradio, medical imaging**

I. Project Summary

Purpose. UMR–GAN aims to restore brain MRI slices degraded by noise and missing regions so downstream review and classical pipelines (e.g., segmentation, radiomics) receive higher-quality inputs.

Progress since Deliverable 1. We: (i) upgraded to a *mask-conditioned cGAN* with a U-Net generator (residual blocks + light attention) and a multi-scale PatchGAN discriminator; (ii) implemented a reproducible data pipeline with normalization, augmentation, and synthetic degradations (Gaussian noise and square masks); (iii) scripted training and evaluation with fixed seeds, config logs, and figure exporters; and (iv) built a Colab/Gradio interface supporting upload, denoise/inpaint toggle, preview, and download.

Current capability. The model restores noisy inputs and inpaints contiguous masked patches. Early results indicate consistent fidelity gains (Sec. V), especially around tissue boundaries.

Remaining gaps. Planned work includes ablations for each conditioning signal (mask, σ map, 2.5D neighbors, class one-hot), hyperparameter sweeps, LPIPS and clinician preference studies, DICOM I/O with window/level controls, and uncertainty estimates (MC-dropout/ensembles).

II. System Architecture and Pipeline

A. Data $\rightarrow$ Preprocessing $\rightarrow$ Model $\rightarrow$ Interface

The pipeline follows: **Data** $\rightarrow$ **Preprocessing** $\rightarrow$ **Model** $\rightarrow$ **Interface**. We operate on axial 2D slices; intensity values are scaled to $[0, 1]$ using robust min–max on nonzero voxels. We harmonize sizes by center-crop/resize and apply training-time degradations to create paired examples (x, y) and a binary mask M (known vs. missing).

Degradation models.: (1) *Denoise:* add zero-mean Gaussian noise with $\sigma \in [0.02, 0.08]$. (2) *Inpaint:* erase a square (or random) region; $M = 0$ marks missing pixels. Optionally we provide a per-pixel σ map, two 2.5D neighbors ($t\pm1$), and a one-hot class vector.

B. Mask-Conditioned cGAN

Conditioning.: We concatenate channels: $z = [x \parallel M \parallel \sigma\text{-map} \parallel x_{t-1} \parallel x_{t+1} \parallel \text{class-1h}]$. If an optional signal is unavailable, we input a zero channel to keep tensor shapes stable.

Generator G.: A U-Net encoder–decoder with (i) strided conv downsampling to $1/64$ scale, (ii) residual blocks in the bottleneck to preserve content, (iii) lightweight attention (channel–spatial gating) at two decoder stages for edge emphasis, and (iv) skip connections from encoder to decoder to retain high-frequency detail. We use LeakyReLU in the encoder, ReLU in the decoder, instance norm in deeper layers, and a final $\tanh$ mapped back to $[0, 1]$.

Discriminator D.: A multi-scale PatchGAN that scores overlapping $N \times N$ patches at two receptive-field scales. D is conditioned on $[x, \hat{y}]$ (fake) or $[x, y]$ (real). Spectral normalization on D improves stability; TTUR (different learning rates for G and D) is enabled in ablations.

Objective.: Let $\hat{y} = G(z)$, then

$$L_{\text{cGAN}} = \mathbb{E}_{x,y}\big[\log D(x, y)\big] + \mathbb{E}_{x}\big[\log(1 - D(x, \hat{y}))\big], \tag{1}$$

$$L_1^M = \frac{1}{|\Omega|}\sum_{p\in\Omega}\alpha(1-M_p)\,|\,y_p-\hat{y}_p\,| + (1-\alpha)M_p\,|\,y_p-\hat{y}_p\,|, \tag{2}$$

$$L_{\text{perc}} = \sum_{\ell}\big\|\phi_\ell(y) - \phi_\ell(\hat{y})\big\|_1, \quad L_{\text{ssim}} = 1 - \text{SSIM}(y, \hat{y}), \tag{3}$$

and the total generator loss is

$$L_G = \lambda_1 L_1^M + \lambda_{\text{gan}} L_{\text{cGAN}} + \lambda_{\text{perc}} L_{\text{perc}} + \lambda_{\text{ssim}} L_{\text{ssim}}. \tag{4}$$

We use $(\lambda_1, \lambda_{\text{gan}}, \lambda_{\text{perc}}) = (100, 1, 0.1)$ and $\lambda_{\text{ssim}} \in [0, 1]$. The mask weight $\alpha \in [0.3, 0.7]$ reduces domination by large holes while still supervising context.

C. Training Loop

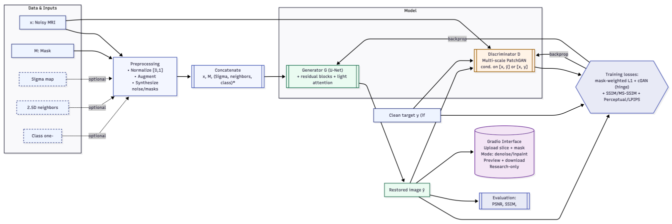

Fig. 1. UMR-GAN v2: mask-conditioned cGAN with U-Net generator and multi-scale PatchGAN discriminator; losses mix L_1, cGAN, SSIM/MS-SSIM, and perceptual/LPIPS.

Algorithm 1 UMR-GAN Training (per mini-batch)

Sample (x, y, M, opts); form $z = [x \| M \| \text{opts}]$.
$\hat{y} \leftarrow G(z)$.
$D_{\text{real}} \leftarrow D(x, y)$; $\quad D_{\text{fake}} \leftarrow D(x, \hat{y}$ (stopgrad)).
$L_D \leftarrow -\log D_{\text{real}} - \log(1 - D_{\text{fake}})$; $\quad$ update D.
Recompute $D_{\text{fake}} \leftarrow D(x, \hat{y})$.
$L_G \leftarrow \lambda_1 L_1^M + \lambda_{\text{gan}}(-\log D_{\text{fake}}) + \lambda_{\text{perc}} L_{\text{perc}} + \lambda_{\text{ssim}} L_{\text{ssim}}$.
Update G; apply EMA to G weights (eval only).

III. Model Implementation Details

A. Frameworks and Libraries

PyTorch/torchvision for modeling; NumPy and scikit-image for array ops/metrics; Gradio for the Colab interface. Code is modular under `src/`: `datasets/`, `models/`, `train.py`, `metrics.py`, `ui/`.

B. Training Setup

Single-GPU Colab runtime (T4/A100). Adam with lr $= 2{\times}10^{-4}$, $\beta_1{=}0.5$, $\beta_2{=}0.999$. Batch 8–16; 50–100 epochs. Augmentations: flips and mild intensity jitter (0.9 to 1.1 scale). Mixed precision (AMP) reduces memory/time; gradient clipping at 1.0 prevents spikes. Early stopping monitors SSIM on a validation fold.

C. Losses and Mask Weighting

$L = 100L_1^M + 1L_{\text{cGAN}} + 0.1L_{\text{perc}} + \lambda_{\text{ssim}} L_{\text{ssim}}$, with $\lambda_{\text{ssim}} \in [0, 1]$. We anneal λ_{ssim} from 0 to its target over the first epochs to stabilize adversarial learning.

D. Initialization and Reproducibility

He init for conv layers; spectral norm on D. Seeds are fixed across PyTorch/NumPy; `torch.backends.cudnn.deterministic=True`. Every run writes a JSON config, a model hash, and metric CSVs; figure exporters regenerate qualitative triplets programmatically to avoid manual cherry-picking.

IV. Interface Prototype

A. Design and Workflow

The Gradio app presents: (1) image upload or sample picker; (2) mode toggle (denoise/inpaint); (3) *Restore*; and (4) side-by-side output with download. Inpainting uses a default square mask when none is provided; denoising assumes a zero mask. Outputs include provenance (model hash, config) in the console for traceability.

TABLE I
Key training configuration (early runs).

Generator / Discriminator	U-Net (residual+attn) / multi-scale PatchGAN
Optimizer / LR	Adam / $2{\times}10^{-4}$
Batch / Epochs	8–16 / 50–100
Loss weights	L_1 : 100, cGAN:1, perc:0.1, SSIM:$[0, 1]$
Stability	AMP, grad clip 1.0, spectral norm (D)
Early stopping	best val. SSIM

B. Usability, Latency, and Limits

End-to-end latency in Colab (T4) is typically $<$400 ms for 256^2 slices (excluding upload). Current limits: single-slice PNG/JPEG inputs, fixed window/level, and no DICOM metadata. Planned: batch uploads, DICOM read with adjustable window/level, and a "compare with target" pane when ground truth is present.

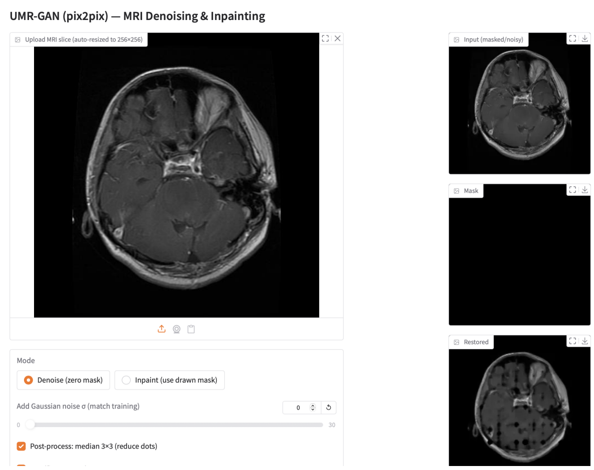

Fig. 2. Gradio UI: upload and task selection (denoise/inpaint).

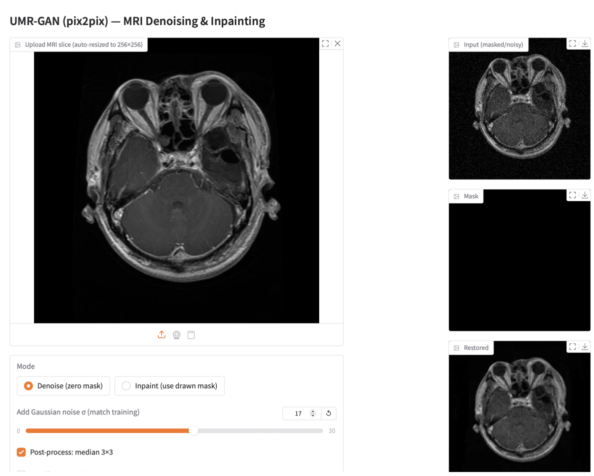

Fig. 3. Gradio UI: restored output preview and download.

V. Early Evaluation and Results

A. Protocol and Metrics

We stratify validation folds by class to avoid leakage. Quantitative metrics include PSNR and SSIM computed on $[0, 1]$ images:

$$\mathrm{PSNR}(y, \hat{y}) = 10 \log_{10} \frac{1}{\mathrm{MSE}(y, \hat{y})}, \ \mathrm{MSE} = \frac{1}{|\Omega|} \sum_{p} (y_p - \hat{y}_p)^2, \tag{5}$$

$$\mathrm{SSIM}(y, \hat{y}) = \frac{(2\mu_y \mu_{\hat{y}} + C_1)(2\sigma_{y\hat{y}} + C_2)}{(\mu_y^2 + \mu_{\hat{y}}^2 + C_1)(\sigma_y^2 + \sigma_{\hat{y}}^2 + C_2)}. \tag{6}$$

We also inspect qualitative triplets for anatomical plausibility (sharp edges without ringing, no invented structures). For future work we will report LPIPS and 1 000x bootstrap CIs.

TABLE II
QUANTITATIVE RESULTS (MEAN).

Method	PSNR (dB)	SSIM
Baseline (Noisy→Clean)	30.191	0.749
UMR-GAN	34.231	0.918
Δ (Gain)	4.04	0.169

Fig. 4. Denoising example: noisy input (left), restored (middle), clean target (right).

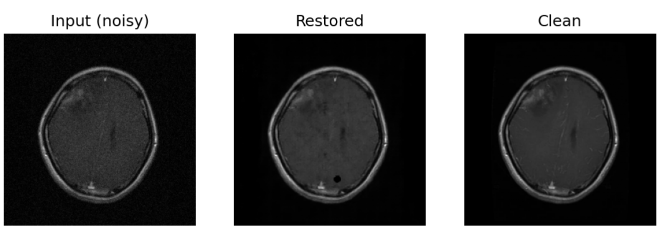

Fig. 5. Second denoising example.

B. Interpretation and Failure Modes

PSNR/SSIM gains reflect improved fidelity and structure. Visuals show reduced speckle and sharper cortical boundaries. Residual issues: (i) over-smoothing at very low SNR, (ii) texture incoherence inside large holes, and (iii) faint haloing at mask borders. We hypothesize these stem from L_1/SSIM trade-offs and limited texture priors; planned ablations target these (Tab. III).

Fig. 6. Single-sample output with input ID displayed on the noisy slice.

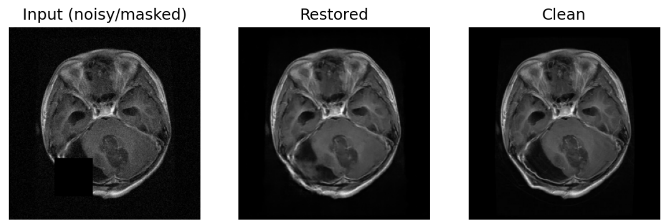

Fig. 7. Inpainting example: masked input (left), restored (middle), clean target (right).

C. Planned Ablations

VI. Challenges and Next Steps

Acquisition variability. Heterogeneous scanners/contrasts motivate stronger normalization and domain randomization; we will also test histogram matching.
Adversarial stability. We balance losses and use early stopping; next we will test TTUR, R_1 regularization, and discriminator spectral norm schedules.
Coverage of rare anatomies. Limited examples can bias restoration; we will stratify metrics, add class-aware sampling, and curate hard cases.
Roadmap (D3). (1) Full ablation study (Tab. III); (2) LPIPS

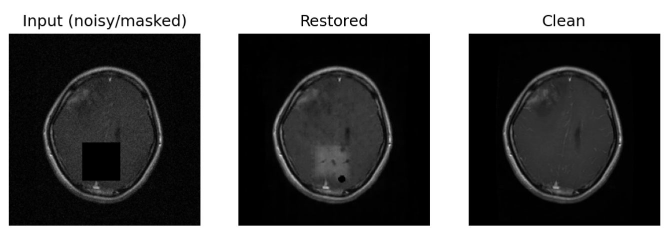

Fig. 8. Second inpainting example.

TABLE III
ABLATION PLAN AND EXPECTED IMPACT.

Factor	Variants	Hypothesis
Mask use	with / without	Improves inpaint SSIM
σ map	on / off	Aids denoise at high noise
2.5D neighbors	none / ± 1	Boosts structure recovery
Perceptual loss	0 / 0.1	Better texture (LPIPS)
SSIM weight	0 / 0.2 / 0.5	Edge vs. texture balance
TTUR / SpecNorm	off / on	Stability ↑

+ clinician preferences; (3) DICOM I/O + window/level; (4) uncertainty maps (MC-dropout) in UI; (5) export packs (restored PNG + JSON provenance).

VII. Responsible AI Reflection

Privacy & security. Training uses anonymized slices; no PHI persists. We log configs, seeds, and model hashes for auditability; outputs never overwrite source data.

Fairness & performance parity. We will report class- and scanner-stratified metrics with CIs and investigate gaps. If gaps exceed a pre-set threshold, we will retrain with targeted augmentation and adjust loss weights.

Clinical safety & uncertainty. The UI displays a research-only disclaimer. We plan to surface uncertainty heatmaps (e.g., MC-dropout variance) to discourage over-reliance on hallucinated details.

Transparency. A one-click export will bundle restored images, metrics, and provenance (model hash, config), functioning as a lightweight model card for each run.

Acknowledgment

We thank the course staff and peers for helpful feedback; compute support via Google Colab.

References

[1] I. Goodfellow *et al.*, "Generative Adversarial Nets," in *NeurIPS*, 2014.

[2] P. Isola *et al.*, "Image-to-Image Translation with Conditional Adversarial Networks," in *CVPR*, 2017.

[3] O. Ronneberger *et al.*, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *MICCAI*, 2015.

[4] Z. Wang *et al.*, "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE T-IP*, 2004.

[5] R. Zhang *et al.*, "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric," in *CVPR*, 2018.