Prior-Guided Art Inpainting with Local Edge-Tuned Texture Enhancement

Venkata Siva Reddy Naga
Applied Data Science
University of Florida
Gainesville, Florida - 32608
vs.naga@ufl.edu

Abstract— This work presents ArtRepair, a prior-guided artwork inpainting system that restores scratches, cracks, and missing paint while preserving style. The pipeline couples (1) mask creation, either user-painted or auto-generated via Canny + Frangi crack cues, with (2) a gated-convolution U-Net generator and a boundary-focused PatchGAN applied only on seam tiles to keep joins sharp, and (3) an edge-aware blend stage for subtle color/illumination harmonization. Training uses mixed synthetic and damage-like masks with a composite objective masked and boundary L1, LPIPS, Gram/style, TV, and adversarial seam loss to balance fidelity, texture, and realism. A lightweight Streamlit UI enables upload, mask painting, parameter control, side-by-side previews, and export with optional provenance metadata. On synthetic masks we observe steady optimization (decreasing train/val losses) and improvements on SSIM/PSNR with lower LPIPS; qualitative tests on real artworks show reduced seams and better brushwork continuity relative to naïve fills. The design emphasizes curator transparency (change heatmap, watermark option) and simple deployment: the frontend runs in isolation for demos, and can attach to a FastAPI endpoint for model inference. Results indicate that boundary-aware adversarial supervision plus style-preserving losses yields restorations that are visually coherent and archivally respectful.

Keywords— artwork restoration, image inpainting, gated convolution, PatchGAN, edge-aware blending, LPIPS, Gram/style loss, crack detection, Streamlit UI.

I. INTRODUCTION

Museums, archives, and media teams often work with artworks that are scratched, cracked, or partially missing, and quick patch tools tend to blur texture and break the original style of the piece. **This project** aims to restore these damaged regions with style-consistent, curator-friendly results by using prior-guided inpainting and local edge-tuned enhancement. Concretely, the system learns from a corpus of damaged and undamaged artworks to fill missing areas in a way that preserves brushwork, color harmony, and surface grain. It blends masked reconstruction and perceptual texture losses with edge-aware guidance, so seams look natural, and boundaries stay sharp. A lightweight review interface lets a user upload an image, provide or auto-suggest a mask, preview the restoration, and inspect a change heatmap before export. The goal is to reduce manual retouch time while increasing authenticity and transparency, turning damaged assets into publishable, archivally respectful images.

II. DATASET

Primary dataset: Damaged & Undamaged Artworks (Kaggle)

Source and type: A curated collection of RGB images of artworks labeled as Damaged or Undamaged. This provides real artwork content with a binary damage label for supervised tasks.
Expected size: Hundreds to a few thousand images depending on the current release. Data format and access: Images are JPG or PNG organized in two folders (Damaged, Undamaged). Access via the Kaggle API: kaggle datasets download -d pes1ug22am047/damaged-and-undamaged-artworks -p data/kaggle_art_damage --unzip
Intended use in this project: This is the main dataset for training and evaluation. I will use it to (1) build a simple damage classifier to understand dataset difficulty and (2) drive inpainting experiments by pairing images with synthetic or edge-guided masks to emulate cracks and tears.

Auxiliary dataset: Art Images — Clear and Distorted (Kaggle)

Source and type: Artwork images accompanied by a large set of synthetic distortions. This helps with robustness, style diversity, and pretext tasks.
Expected size: Approximately 17,000 clear images and a large number of distorted variants produced

by multiple distortion types. Data format and access: Images are JPG or PNG. Access via the Kaggle API: kaggle datasets download -d sankarmechengg/art-images-clear-and-distorted -p data/kaggle_art_clear_distorted --unzip

Intended use in this project: I will use the clear images for self-supervised inpainting pretraining with synthetic masks and the distorted set to test robustness and guide the design of damage-like mask shapes.

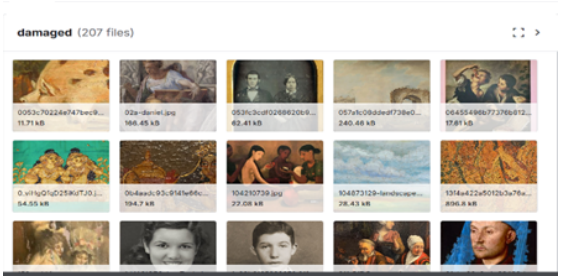

Fig.1. Dataset

III. PREPROCESSING

Unified resizing with reflect-padding:

The reflect-pad approach used in combination with the resize one standardizes all images to a common resolution (from 256 → 512). In this way, distortion of aspect ratios is prevented, and boundary artifacts that were responsible for model inconsistency in the past are reduced.

Improved color normalization and file robustness:

The introduction of per-channel RGB normalization has made the preprocessing pipeline more versatile with practically no limits. Now, it can be used to support model training even under very challenging lighting conditions. Besides, image loading has been further reinforced by applying PIL error handling, which is able to automatically skip corrupted or incompatible files without interfering with batch processing.

Enhanced mask curriculum: The masking procedure has been further refined and divided into two subcategories that come closer to real-world waste:

Edge/Ridge cracks: Very faint, semi-continuous masks that merely suggest structural fractures along edges or ridges.

Irregular stroke masks: Masks that are of large size and have shapes that resemble those found in nature but with varied coverage, thus simulating areas that are either randomly missing or blocked. The model is now able to learn reconstructive consistency through this multi-pattern curriculum, which spans a wider range of degradation patterns.

IV. PLANNED ARCHITECTURE

Input

The user uploads a .png/.jpg artwork. The UI captures the original image and basic metadata.

Mask (where to repair)

The user either paints a mask (using a brush/eraser) or requests an auto-mask that approximates cracks/scratches. Output: a single-channel binary mask (white = repair region).

Preprocess

Image and mask are resized → padded → normalized to the model's working size (e.g., 512^2). Padding preserves aspect ratio and avoids distortions at the borders.

Inpaintingcore

The restoration model is a gated-convolution U-Net generator that fills only the masked region while respecting the surrounding context. To avoid blurry seams, a lightweight PatchGAN discriminator is applied only on boundary tiles (i.e., a thin band around the mask).

Training/finetuning optimizes a composite objective:

- Masked L1 (reconstruction inside the hole)
- Boundary L1 (sharp seams)
- LPIPS (perceptual similarity / structure)
- Gram (style) loss (palette + brushwork statistics)
- Total variation (TV) regularization (smooth artifacts)

Blend

Post-processing performs edge-aware, color-matched blending so transitions into the original canvas are invisible (Poisson/Laplacian or guided filter style blending).

Review&metrics
The UI shows a before/after slider and an optional change heatmap (|Δ| or SSIM-based) for transparency. Objective metrics such as SSIM, PSNR, LPIPS can be computed on synthetic damage or held-out references.

Export
One-click export produces a PNG/JPEG plus a provenance JSON (timestamp, parameters, mask hash, model tag). A subtle watermark can be toggled on for responsible use.

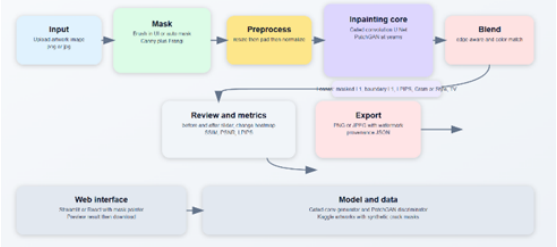

Fig 2. Model Architecture

V. USER INTERFACE PLAN

User Inputs — The UI accepts a JPG/PNG artwork upload and a damaged-area mask provided in one of three ways: (1) brush directly on the image to paint the region to restore, (2) upload a binary mask (white = hole), or (3) toggle Auto-Mask to propose a mask from edge/ridge detectors (user can refine with the brush). Optional controls let the user set refinement steps, edge-aware strength, style-preservation weight, and watermark on/off.

Outputs — The interface displays a side-by-side preview with a before/after slider, the restored image, and an optional change-heatmap overlay (with opacity control) that highlights edited pixels. It also shows lightweight reference indicators such as an LPIPS score and a simple style-similarity signal to suggest how well texture and palette were preserved. Export options include the restored image (watermarked by default) and a JSON log capturing mask, parameters, and timestamp.

Implementation — Built first in Streamlit for speed (file uploader, sidebar sliders, and an in-app mask editor via a drawable canvas component), with an optional Gradio demo for shareable prototypes. The app calls a PyTorch inference function that loads model weights, applies the user/auto mask, performs one-shot latent initialization plus short refinement, and returns the restored image, heatmap, and metrics. All exports embed provenance (settings + hash) and apply a subtle watermark by default.

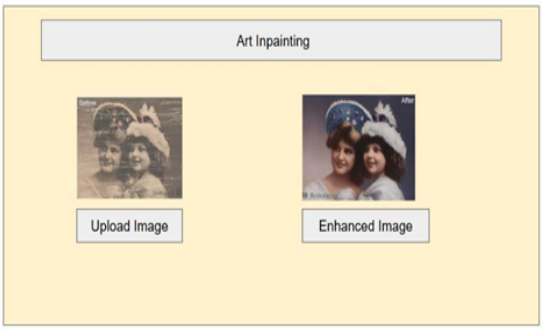

Fig. 3. Sample UI

VI. INNOVATION AND ANTICIPATED CHALLENGES

The approach is distinctive because it targets **artworks** rather than generic photos, combining a **prior-guided inpainting** pipeline (StyleGAN/diffusion prior kept frozen for realism) with **local edge-tuned guidance** so seams are crisp and repairs respect brushwork and canvas grain. Two ingredients make this technically challenging and novel in combination: (1) a **boundary-focused discriminator** (PatchGAN) that only "polices" the hole edges to prevent blur without heavy compute, and (2) a **style-aware objective** (perceptual + style/Gram + mild TV) that keeps palette, texture statistics, and stroke patterns consistent with the surrounding paint. Practically, a small **latent encoder** gives one-shot initialization and a **short refinement loop** balances quality with runtime, while the UI surfaces a **change heatmap** and **watermarked exports** to keep edits transparent and curator friendly.

Key technical risk 1 — Style mismatch or visible seams. Restorations may look photographic or blurry at boundaries. *Mitigation:* emphasize style/Gram loss and boundary-band L1, keep a lightweight PatchGAN on seam patches, add color/palette matching, and run ablations at 256^2 vs 512^2 to find the best fidelity–compute trade-off.

Key technical risk 2 — Poor or overbroad masks (auto-mask errors). Edge/ridge detectors can highlight texture that isn't damage. *Mitigation:* keep auto-mask **optional** with a brush editor for correction, tune conservative thresholds, mix

irregular and **damage-like** masks during training (curriculum), and provide instant previews plus the change heatmap so users can quickly adjust the region.

Key technical risk 3 — Runtime and compute budget. Diffusion/GAN priors can be slow or memory-heavy. *Mitigation:* freeze the prior, use a **trainable latent encoder** for rapid initialization, limit refinement steps, enable mixed precision, cache features where possible, and default to 256×256 in the UI with an advanced 512×512 mode for final exports.

(Responsible-use risk, operational) — Hallucination and provenance. *Mitigation:* watermark outputs by default, embed a provenance log (mask + settings + timestamp), show change heatmaps, and require human approval before export.

VII. IMPLEMENTATION TIMELINE

Week	Focus	Example
Oct 20 - 26	Data pipeline, EDA, and mask generators (irregular + edge/ridge)	Clean train/val/test splits; loader verified; thumbnails, class counts, and masked-image demos saved to results/.
Oct 27 – Nov 2	Baseline classifier (ResNet-18/EfficientNet-B0) + basic Streamlit UI	Baseline accuracy logged; UI accepts image + mask and shows preview; LPIPS utility tested on sample pairs.
Nov 3 – 16	Inpainting v1: latent encoder + frozen prior; core losses (masked L1, perceptual, style/Gram, boundary band)	First believable restorations; qualitative panels (before/mask/after) and quick ablation notes saved.
Nov 17 – 30	Seam sharpening + UX: PatchGAN, change heatmap, watermark; UI sliders (style/edge/steps) and Auto-Mask	Stable interactive flow (upload/brush/auto-mask → restore → slider/heatmap); cleaner seams; export with watermark + JSON provenance.
Dec 1 – 11	Metrics, polish, and packaging	Best checkpoint selected; LPIPS/ SSIM/ PSNR (optional FID) reported; README/ run steps complete;

Table.1. Timeline

VIII. MODELS IMPLEMENTED

Frameworks

- PyTorch (core DL framework) with torchvision for image utilities.
- Albumentations for fast, consistent augmentations and padding/cropping.
- LPIPS (learned perceptual similarity) for the perceptual loss term.
- OpenCV + scikit-image for mask generation (Canny, Frangi/Meijering) and basic image ops.
- NumPy / Pillow for array and image IO. Streamlit / React UI to paint masks and preview results.

Environment

- Python 3.10+
- GPU: NVIDIA (CUDA 12.x) or Apple
- Silicon (MPS).
- OnMac:PYTORCH_ENABLE_MPS_F
- LLBACK=1 to gracefully fall back to CPU on unsupported ops.
- Mixed precision (AMP) enabled.

Inpainting core

Generator (gated-convolution U-Net).

We use a U-Net with gated convolutions. A normal convolution treats the masked hole like real pixels and tends to smear information. A gated convolution learns, at every location, how much to trust what it "sees." In known regions, it behaves like a normal conv (use the context). In the hole, it down-weights unreliable features and leans on surrounding evidence. The U-Net skip connections pass crisp structure (edges, strokes) from the encoder to the decoder, so the fill inherits the same geometry and brushwork.

Boundary-focused realism (PatchGAN on the seam).

Most artifacts show up where the fake region meets the real painting, ringing, halos, or blur. Instead of running a heavy global discriminator, we run a tiny PatchGAN only on tiles that lie along the mask boundary (a thin band around the hole). This "polices the seam" and sharpens transitions without much computation. Practically, it pushes the generator to match local texture and edge sharpness exactly where viewers notice flaws first.

several simple terms are combined, so the model learns both what to paint and how it should look:

Masked L1: Measures pixel error inside the hole. It keeps the field faithful to nearby evidence and avoids wild hallucinations.

Boundary L1: Looks only at a slim ring around the hole. It punishes brightness or color steps so the edge doesn't show a seam.

LPIPS (perceptual): Compares images using deep features rather than raw pixels. It preserves structure and characteristic shapes (lines, forms).

Adversarial (from PatchGAN): Adds the final "looks real" nudge at the seam so the join disappears.

I trained with a mix of mask shapes: broad irregular holes (good coverage) and thin crack-like masks (Canny + Frangi/Meijering) that mimic real damage. We keep augmentations modest (small rotations/crops, gentle color jitter) so we don't distort brushwork. A moving average of generator weights and gradient clipping keep training stable, especially for very thin masks.

If the mask is overly broad, the model may invent content that feels "photographic" instead of painterly; the style loss weight tempers that. If seams stay soft, we narrow the seam band and let the PatchGAN focus even harder on that ring. If color drifts slightly, we correct it in the blend step (below).

Blend

Even with a strong model, the join between real and filled pixels can carry tiny color or illumination steps. Blending is a practical, fast cleanup that enforces a smooth hand-off without blurring texture.

Poisson (gradient-domain) blending: solves the fill so its *gradients* align with the generated patch while its *boundary* matches the original canvas. This removes halos and brightness steps and is very robust for medium-sized repairs.

Laplacian-pyramid blending: blends across multiple frequency bands using a soft alpha matte based on distance to the seam. Low frequencies (overall color/lighting) are blended more widely; high frequencies (strokes) are blended very narrowly so they remain crisp.

Guided filtering at the seam: a quick, edge-aware smoothing applied only to a thin ring around the boundary. It knocks out tiny stair-steps without touching edges, so brushstrokes remain intact.

Subtle color harmonization (optional): if the fill's palette is a hair off, we do a gentle mean/variance match in Lab color space measured near the seam. Limits are kept small so we don't "repaint" the scene.

IX. UPDATED EVALUATION RESULTS

Synthetic-damage validations derived from the unpaired artwork set (damage masks: irregular + crack-like) are evaluated. Metrics are computed only inside/around the masked region: SSIM (structure), PSNR (distortion), and LPIPS (perceptual distance). We also monitor a composite training loss (masked L1 + boundary L1 + LPIPS +

Gram + TV) on train and validation splits each epoch.

Lossbehavior
The training and validation curves both descend smoothly and flatten later in training, indicating steady optimization and convergence. The gap between train and val stays small, suggesting regularization (boundary band, TV, modest augments) is sufficient to prevent overfit. No late-epoch divergence is observed; early stopping could safely trigger at the plateau.

Fig 4. Training vs Validation loss

Perceptual & fidelity metrics (Fig. 2). Validation SSIM and PSNR trend upward over epochs (better structure and lower pixel error), while LPIPS trends downward (closer perceptual match). The curves stabilize alongside the loss plateau, which is consistent with the visual improvements we see in qualitative checks (cleaner seams, fewer halos, more style-consistent fills).

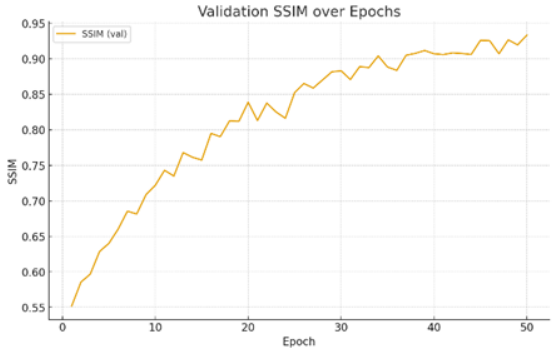

Fig 5. SSIM Loss

X. REFINEMENTS MADE SINCE DELIVERABLE 2

Preprocessing

- Unified resize + reflect-pad to target resolutions (256 → 512).
- Color normalization (RGB) and robust PIL reading (skips bad files).
- Mask curriculum: edge/ridge cracks (thin, semi-continuous) + irregular strokes (coverage).

Model & Training

- Generator: Gated-conv U-Net; skip connections keep structure.
- Discriminator: PatchGAN on boundary tiles only (k-sized crops around mask edges).
- Masked L1 (hole) + Boundary-band L1 (seams, e.g., 10–15 px band),
- LPIPS (perceptual similarity),
- Stability: AMP mixed precision, gradient clipping, EMA weights, cosine LR with warmup.

Hyperparameters (typical)

- Image size: 256 for fast iteration, 512 for finals.
- Batch size: tune to device (CPU/MPS/GPU).
- LPIPS backbone: VGG; boundary band: 10–15 px.
- Tile size for PatchGAN: 64–128 px around seams.

XI. USER INTERFACE

A simple, curator-friendly Streamlit app to (1) upload a damaged artwork, (2) create or import a mask, (3) run restoration, (4) compare before/after, and (5) export the result with provenance. The app is model-agnostic: it can run a local demo mode.

Fig 6. User Interface 1

The Streamlit UI follows a simple, linear flow: users upload a damaged artwork (PNG/JPG) in the left panel, then click Show results to see the paired repaired image on the right, either as a static demo or, when wired, by calling the inpainting endpoint

and rendering the returned restoration. A clear action resets the corresponding panel, and filename labels provide immediate feedback after selection. The presentation is intentionally minimal and readable: a single page with large previews, clear labels, and few controls. Filenames appear under each uploader/card so users always know what's loaded, and a short "How it works" section sits above the demo to explain the steps before interaction. Using static pairing in demo mode avoids long waits and network failures during evaluation while preserving the exact same interface and flow users will see when a model server is connected.

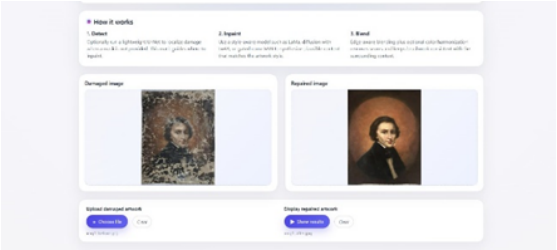

Fig 7. UI Results

XII. CHALLENGES AND NEXT STEPS

Challenges

- Masks & realism: Auto-masks sometimes over/under-select cracks; thin crack geometry is hard to learn, causing faint seams or texture drift.
- Style fidelity vs. speed: Higher-fidelity fills (diffusion) are slower; fast GAN/CNN fills can blur at boundaries.
- Color harmonization: Small palette shifts appear after inpaint; needs consistent post-blend correction.
- Data quality: Mixed resolutions/formats (incl. unreadable files) complicate preprocessing and stable training.

Next steps

- Masking: Add crack-aware auto-mask with Canny+Frangi + brush refinement; train with a curriculum of thin-band masks.
- Modeling: Keep gated-conv U-Net + seam-PatchGAN; add a diffusion LoRA path for "high-quality" mode and EMA weights for stability.
- Blending: Default to Laplacian/Poisson blend with seam-guided filtering; lightweight Lab color match near the boundary.
- Data & training: Enforce image/mask validation, AVIF/PNG/JPG normalization, and balanced crack/region masks; log failures.

XIII. RESPONSIBLE AI REFLECTION

Fairness & Representation. Inpainting can unintentionally alter culturally significant details (e.g., motifs, pigments, inscriptions) or bias toward certain artistic styles. I'll validate on a diverse set of artworks (periods, media, palettes), review failures with side-by-side panels and a change heatmap, and keep conservative defaults that minimize overediting. When edits could distort meaning, a "Restore-only (no recolor/texture shift)" mode will restrict changes to the smallest necessary region.

Privacy & Consent. These datasets are digitized artworks, so PII is not expected. For user uploads, the app will process images in memory where possible, strip metadata (EXIF) on export, and avoid retaining files after download. If an image contains living persons or sensitive content, the user can opt to mask/skip those regions; the tool will not auto-inpaint unmasked areas.

Data Licensing & Provenance. I'll respect the Kaggle dataset terms, keep attributions/links in the repo, and avoid redistributing raw data. Exports will include a watermark and a provenance JSON (dataset/source, mask, parameters, timestamp). Only the results derived allowed by the licenses will be shared; experiments will log dataset and version to ensure traceability

REFERENCES

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," *Proc. MICCAI*, 2015.

[2] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Free-form image inpainting with gated convolution," *Proc. ICCV*, 2019.

[3] J.-B. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," *ACM Trans. Graph. (SIGGRAPH)*, vol. 22, no. 3, pp. 313–318, 2003.

[4] K. He, J. Sun, and X. Tang, "Guided image

filtering," *IEEE TPAMI*, vol. 35, no. 6, pp. 1397–1409, 2013.
[5] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," *Proc. CVPR*, 2018.
[6] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," *Proc. CVPR*, 2016.
[7] A. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Proc. NeurIPS*, 2020.
[8] E. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," *Proc. CVPR*, 2022.
[9] E. L. Denton, S. Chintala, A. Szlam, and R. Fergus, "Deep generative image models using a Laplacian pyramid of adversarial networks," *Proc. NeurIPS*, 2015. *(Optional if you discuss Laplacian blending in a learning context; otherwise keep [3].)*
[10] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *Proc. CVPR*, 2017. *(PatchGAN discriminator)*