# CISC 473 Midterm Report — Group 7

Monica Stef, Aidan Sibley

October 2025

## PIPELINE ROADMAP — GFPGAN VS CODEFORMER (WORKING DRAFT)

Monica Stef Aidan Sibley

---

### S0 OVERVIEW

- PURPOSE: Comparative study between GFPGAN (TencentARC) and CodeFormer (Zhou et al.) for face restoration
- DATASET: CelebA aligned (single source, no historicals)
- APPROACH: Synthetic degradation → pretrained inference → metric + qualitative comparison
- FRAMEWORK: PyTorch 2.x, conda env, deterministic seeds
- GOAL: Understand trade-offs in fidelity, perceptual realism, and identity preservation

[ADD: simple diagram later showing linear pipeline S1→S7]

---

### S1 DATA INGESTION

- INPUT: ./data/img_align_celeba + metadata CSVs
- PURPOSE: Hold immutable aligned faces for all downstream stages

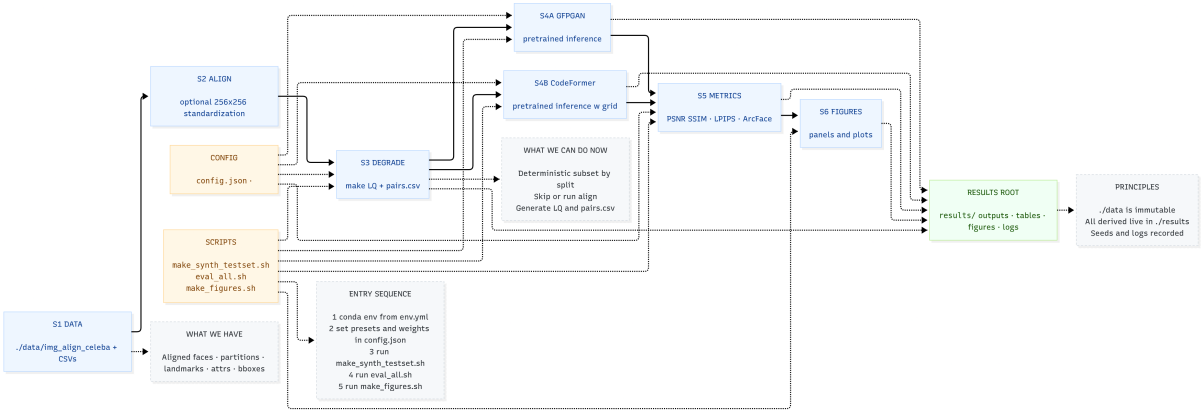


Figure 1: **S0** Overall Pipeline Architecture - Data flow from source to metrics and visualization.

- REASON: Simplify setup, avoid training data noise from multiple sources
- MODEL: none
- OUTPUT: no file output, read-only source
- FILES: list_attr_celeba.csv (attributes) list_bbox_celeba.csv (bounding boxes) list_landmarks_align_celeba.csv (5 facial keypoints) list_eval_partition.csv (split 0/1/2)

[SUGGESTION: add checksum or manifest log for reproducibility]

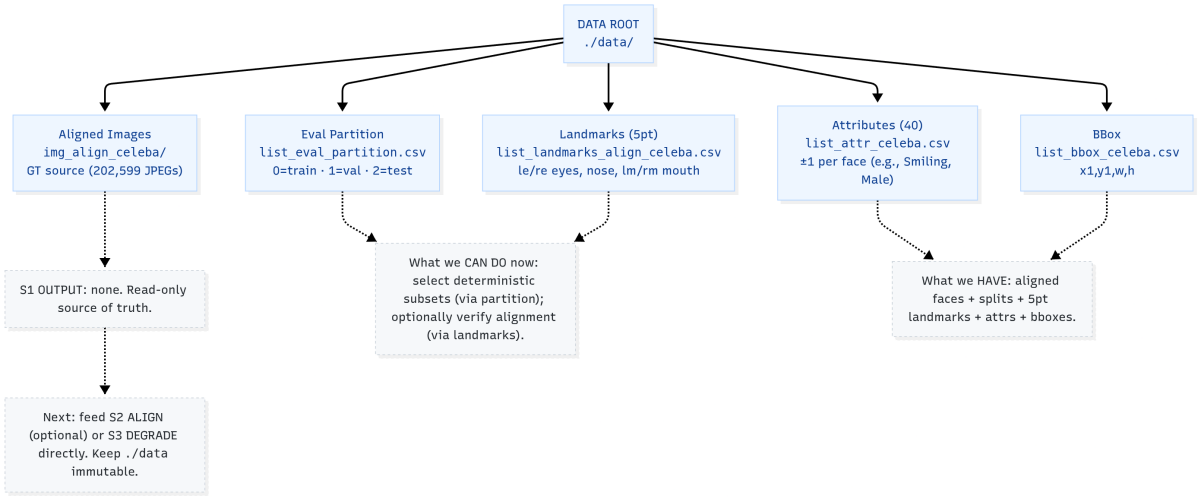


Figure 2: **S1** Data Stage - Structure of CelebA dataset and annotation CSVs.

---

## S2 FACE ALIGNMENT / STANDARDIZATION

- INPUT: images + landmark CSV

- PURPOSE: ensure all faces are spatially consistent (256×256 crops)
- MODEL: facexlib detector (if alignment verified, skip re-run)
- SCRIPT: src/detect_align.py
- OUTPUT: aligned images (temporary or stored in ./results/logs if needed)

[SUGGESTION: skip for midterm if existing alignment sufficient] [SUGGESTION: if re-align, add deterministic seed + log file]

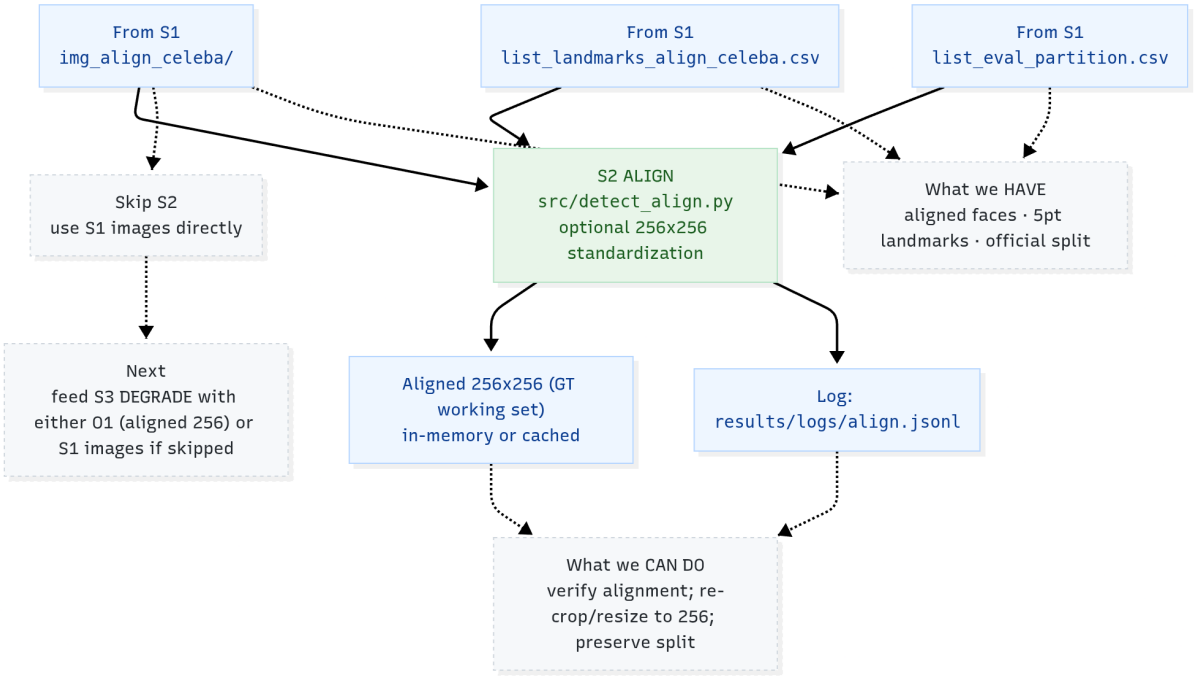


Figure 3: **S2** Alignment Stage - Optional re-alignment and normalization of face crops.

---

## S3 SYNTHETIC DEGRADATION

- INPUT: aligned faces from S2
- PURPOSE: create low-quality (LQ) versions for restoration input
- MODEL: none (procedural operations: blur, noise, jpeg compression)
- SCRIPT: src/degrade.py
- OUTPUT: LQ images in ./results/inputs_lq/_/imgs mapping pairs.csv (LQ↔GT) manifest JSONL (parameters + seed)

[SUGGESTION: start with one preset (blur+jpeg) before adding more] [SUGGESTION: record seed and RNG for exact reproducibility]

---

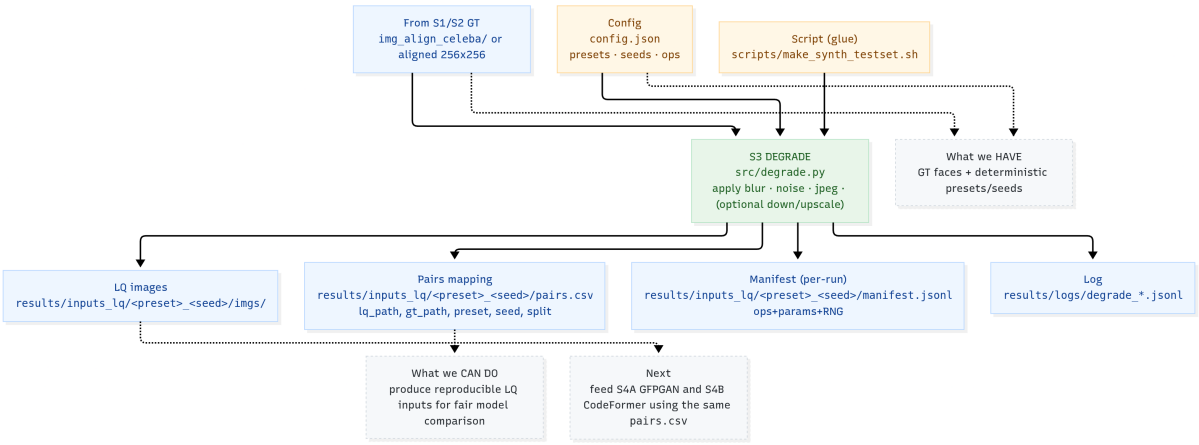


Figure 4: **S3** Degradation Stage - Synthetic generation of low-quality inputs with controlled

## S4A INFERENCE — GFPGAN

- MODEL: TencentARC/GFPGANv1 (official pretrained, CVPR 2021)
- INPUT: LQ images (from S3)
- PURPOSE: perform blind face restoration using StyleGAN2-based prior
- SCRIPT: src/run_gfpgan.py
- OUTPUT: restored faces in ./results/outputs/gfpgan//
- LOGS: ./results/logs/gfpgan_.txt

[NOTES: download GFPGANv1.pth manually or via wget command from repo] [SUGGESTION: add option for upscale factor = 2] [SUGGESTION: optional parameter config via config.json]

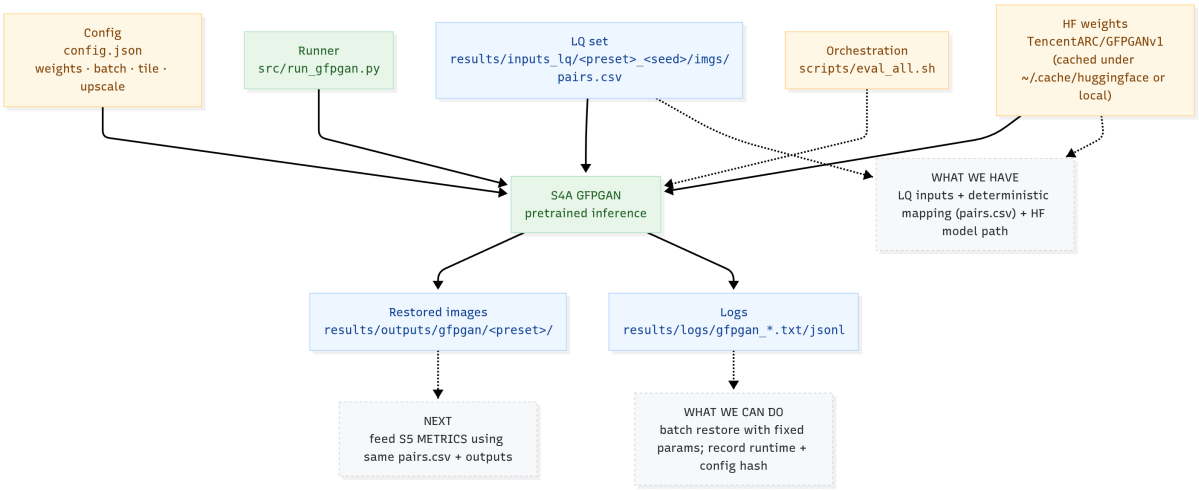


Figure 5: **S4A** GFPGAN Restoration - Inference using pretrained TencentARC/GFPGANv1.

## S4B INFERENCE — CODEFORMER

- MODEL: sczhou/CodeFormer (transformer architecture with fidelity knob w)
- INPUT: same LQ dataset from S3
- PURPOSE: produce restored outputs at varying fidelity-quality balance ($0 \leq w \leq 1$)
- SCRIPT: src/run_codeformer.py
- OUTPUT: ./results/outputs/codeformer///

[SUGGESTION: test w values {0.3, 0.5, 0.7, 1.0}] [SUGGESTION: align naming with GFPGAN outputs for easy comparison]

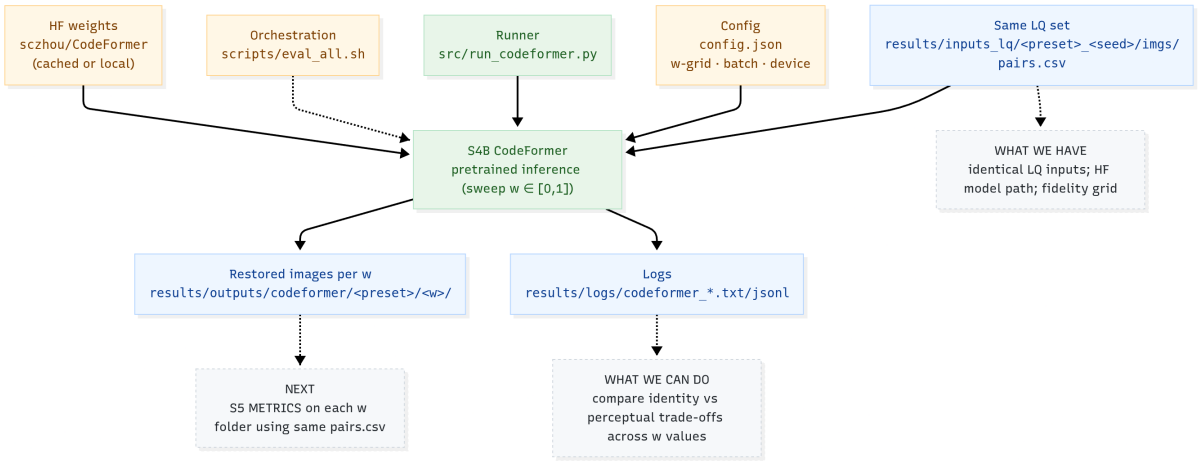


Figure 6: **S4B** CodeFormer Restoration - Inference using pretrained sczhou/CodeFormer with fidelity sweep.

---

## S5 METRICS EVALUATION

- PURPOSE: quantify differences between model outputs and ground truth
- INPUT: GT from S2, outputs from S4A/B, mapping pairs.csv
- MODELS USED: LPIPS backbone (AlexNet or VGG) ArcFace (identity metric)
- SCRIPTS: src/metrics_pixel.py (PSNR/SSIM) src/metrics_lpips.py (LPIPS) src/metrics_identity.py (ArcFace cosine similarity)
- OUTPUT: CSVs under ./results/tables/metrics__.csv

[SUGGESTION: keep aggregate means + stddev for each preset/model] [SUGGESTION: identity metric can also confirm if model hallucinated features]

---

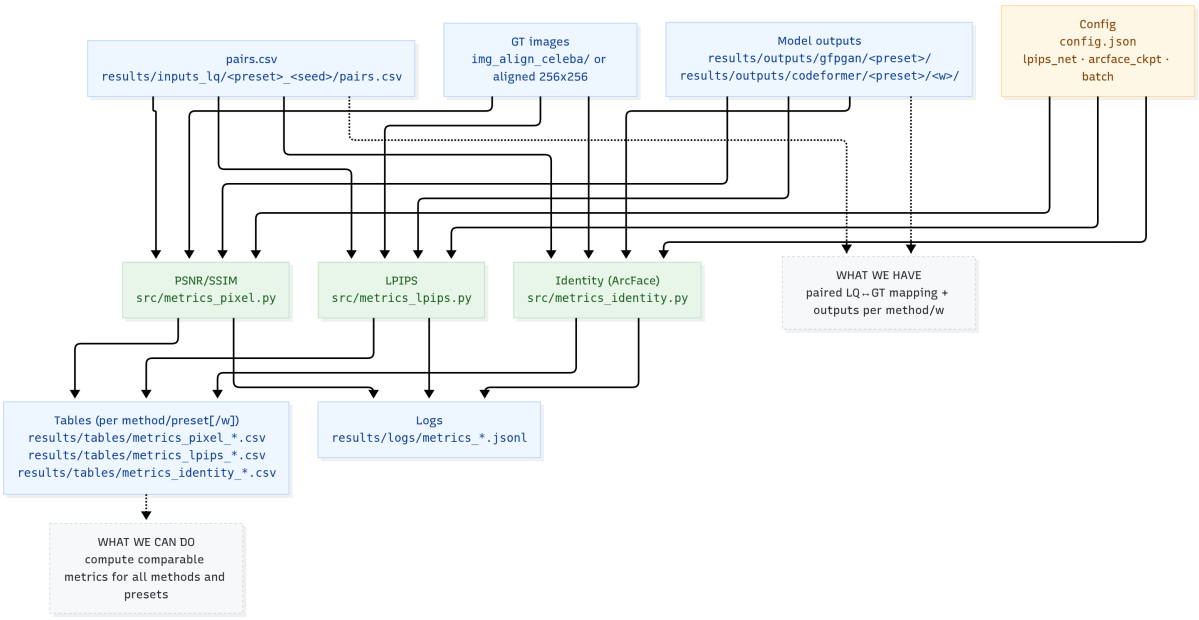


Figure 7: **S5** Metrics Evaluation - Computation of pixel, perceptual, and identity metrics.

## S6 VISUALIZATION AND FIGURE GENERATION

- PURPOSE: produce side-by-side panels and metric plots
- INPUT: selected outputs + GT + tables
- SCRIPT: scripts/make_figures.sh
- OUTPUT: ./results/figures/*.png
- USE: for midterm report visual presentation

[SUGGESTION: start with one panel (GT, LQ, GFPGAN, CodeFormer w=0.5)] [SUGGESTION: add bar plots for average LPIPS and ArcFace cosine]

---

## S7 LOGGING AND CONFIGURATION

- PURPOSE: maintain deterministic reproducibility
- FILES: config.json (seeds, paths, presets, model paths) env.yml (PyTorch version, dependencies) results/logs/ (timestamped run logs)

[SUGGESTION: include git commit hash + timestamp at each run] [SUGGESTION: add checksum of pretrained weights if time allows]

---

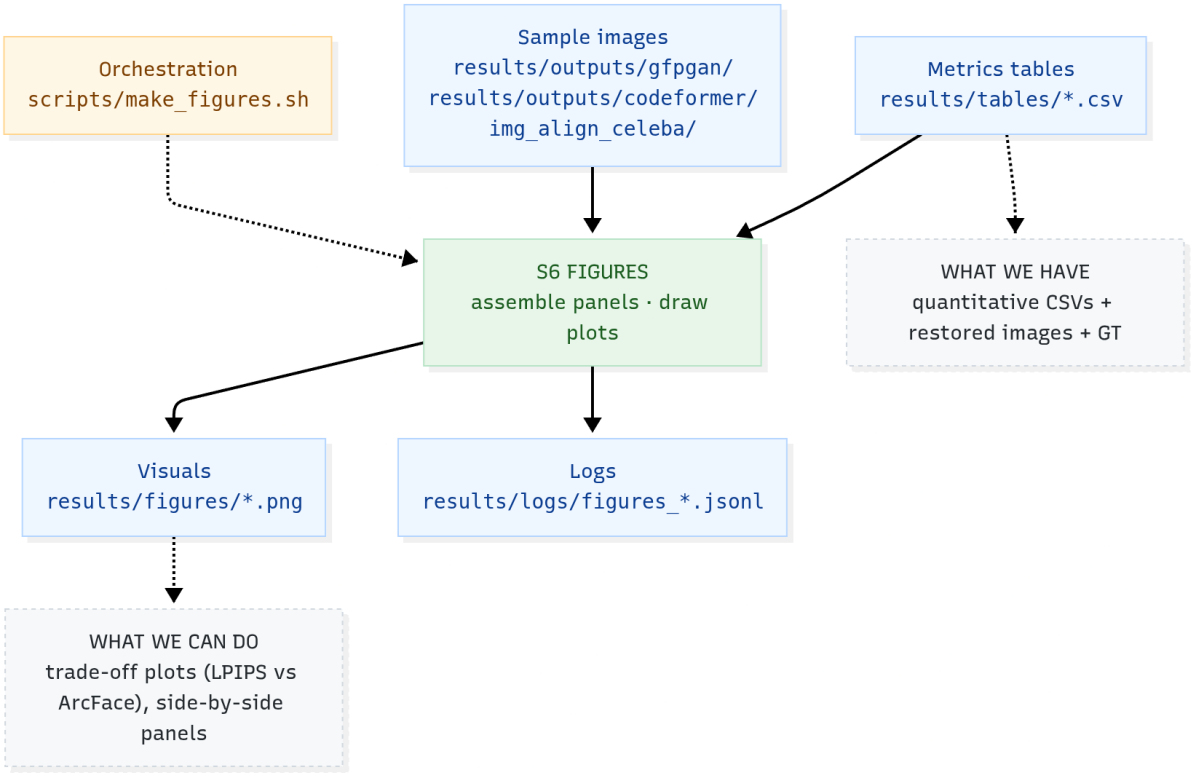


Figure 8: **S6** Visualization Stage - Generation of comparative panels and metric plots.

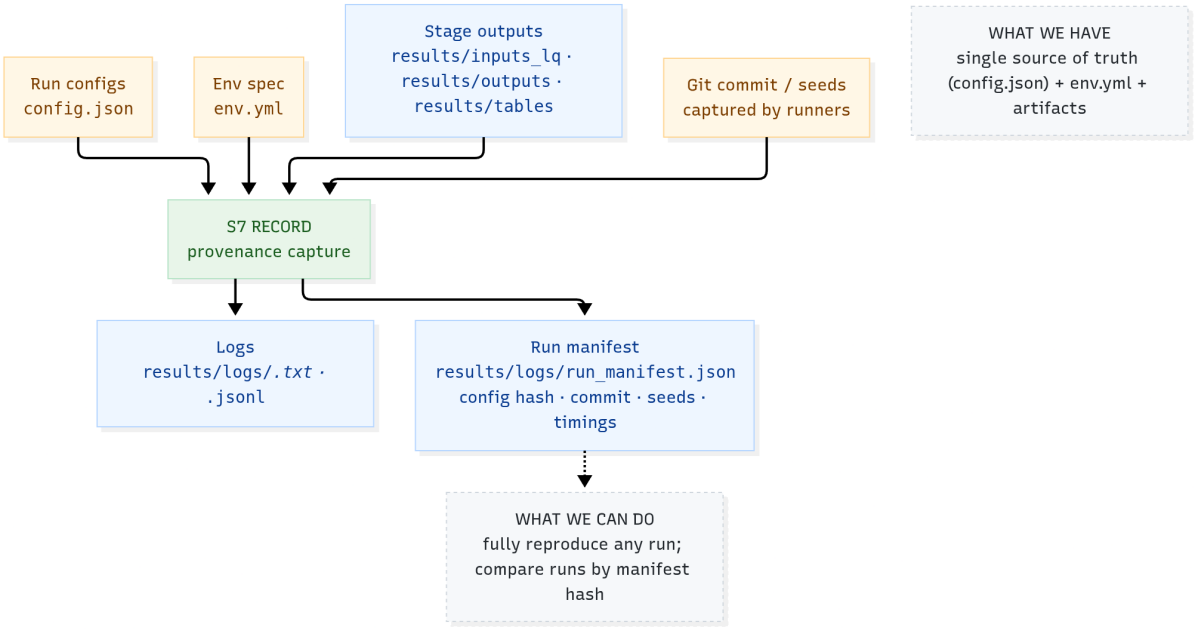


Figure 9: **S7** Logging and Reproducibility - Provenance, configuration capture, and run manifests.

NOTES ON MODEL CHOICE

- GFPGANv1 (TencentARC) → generative prior from StyleGAN2, excels at realistic detail
- CodeFormer (Zhou et al.) → transformer with fidelity-quality tradeoff
- Combined comparison exposes trade-offs between GAN-based realism and transformer-based fidelity

[SUGGESTION: capture both quantitative and perceptual contrasts, not just scores]

---

NEXT STEPS

- Download pretrained GFPGANv1.pth and verify inference
- Generate first LQ set using one degradation preset
- Run both models on same LQ inputs
- Compute PSNR, LPIPS, ArcFace similarity
- Create single visual panel for midterm submission

[ADD: short paragraph summary of observations once first batch complete] [KEEP: this document as a running plan → update stage status as completed]