An AI Assurance and Audit Workbench for Chest X-ray Models

Nattakrit Tadrabiab[1], Pasavee Jaipain[1] , Atiphon kaeochamnong[1*]

Thapanawat Chooklin[1]

[1]*Princess Chulabhorn Science High School Nakhon si thammarat , Phanang, Bang Chang, Mueang Nakhon si thammarat, 80330 Thailand*

**E-mail:6405723@pccnst.ac.th , 6706408@pccnst.ac.th , 6405702@pccnst.ac.th*

Abstract

CeXaR is a working, tested audit pipeline for chest X-ray artificial intelligence (AI) models, developed to address whether a diagnostic AI model that performs well in one hospital setting continues to perform reliably in another, given differences in imaging equipment, patient populations, and acquisition protocols. Rather than functioning as a diagnostic tool itself, CeXaR operates as an assurance and audit workbench that evaluates whether an externally developed model has been validated for a specific deployment context. The system implements a complete pipeline consisting of DICOM ingestion, de-identification, model inference, statistical auditing, and automated Evidence Report generation, and was validated end-to-end using real hospital-style DICOM data from a public chest X-ray dataset (VinDr-CXR, via Kaggle). Initial testing on 26 real files revealed a complete ingestion failure, later diagnosed as a DICOM-standard conformance gap; following remediation, the corrected pipeline achieved full acceptance on the same files and was subsequently scaled to a 53-file run. Reproducibility was demonstrated live: two independent runs on identical input data produced byte-for-byte identical result files, verified through content-derived cryptographic fingerprinting. A total of 109 automated tests across 18 source modules confirm system reliability. Because the eligibility criteria for statistically valid accuracy reporting were not met at this sample size, CeXaR correctly withheld point-estimate accuracy claims rather than reporting unreliable figures, demonstrating the same audit discipline the tool is designed to enforce on the models it evaluates. CeXaR represents a working foundation for trustworthy, context-specific AI assurance in clinical imaging deployment

keywords AI assurance, Audit workbench, Chest X-ray, DICOM, Reproducibility

Introduction

Diagnostic AI models for chest X-ray interpretation are frequently trained and validated on a single data source, yet deployed across hospitals with different imaging equipment, patient populations, and acquisition protocols. This domain shift is a real and underserved problem: a model that performs well in one setting is not guaranteed to perform reliably in another. Rather than asking "who built this AI model," the more relevant question for a hospital adopting third-party AI is "has this exact model version been tested in this exact deployment context?"

CeXaR was developed to answer that question. It is explicitly not a replacement for a radiologist, not a new diagnostic model of its own, not a real-

time clinical alerting system, and not a certificate that declares any model "safe." Instead, it produces structured evidence for human review, treating AI models as systems under test rather than as products it competes with. This project builds on the team's prior hands-on experience in chest X-ray representation-learning research, which established rigorous data-handling and statistical-reporting habits later formalized into CeXaR's automated audit discipline.

Methodology

CeXaR implements a five-stage pipeline:

(1) Ingest — validates DICOM conformance, extracts a de-identification-safe allow-list of metadata, renders pixel data correctly, and rejects unsupported files by name without crashing;

(2) Run — executes a versioned model adapter (TorchXRayVision) and records model identity, weights hash, and disclosed training-data provenance;

(3) Audit — computes an input-drift profile and a prediction-behavior profile (both requiring zero ground truth), plus AUROC/AUPRC with bootstrap confidence intervals when labels exist;

(4) Review — selects false-positive, false-negative, and rejected cases for human inspection; and

(5) Report — automatically generates an Evidence Report in Markdown and HTML, never issuing a pass/fail verdict.

The audit engine uses AUROC and AUPRC per label, a 95% bootstrap confidence interval (percentile method, 1,000 resamples, fixed seed), and patient-level resampling when patient identifiers are available. An eligibility floor (minimum 100 positive and 1,000 negative cases per label) prevents the system from reporting a point estimate when the sample size is statistically unstable. Every run is bound to independent SHA-256 fingerprints of the dataset bytes, the labels manifest, and the model/weights identity, making each report tamper-evident and reproducible.

The pipeline was tested against real hospital-style DICOM files from the public VinDr-CXR dataset (via Kaggle), scaled from 26 to 53 files, with ground-truth labels derived from VinDr's raw per-radiologist bounding-box annotations using a documented majority-vote policy. System correctness was verified by 109 automated tests across 18 source modules.

Result and Discussion

Table 1 Provenance Hash Verification

Question	Answer
What is this claim supposed to guarantee?	Anyone can recompute the SHA256 of a report file themselves and it will match the value printed — proof the file was not altered after generation.
How did we test our own claim?	Wrote a second, fully independent check that reads the file's raw bytes directly from disk (not through the normal code path, which could hide a bug) and compares against the reported value.
What did we find?	The two values did not match, for all 3 output files — the claim was false.
Why did it happen?	On Windows, saving a text file automatically rewrites line breaks (\n ⟶ \r\n). This happened *after* the hash had already been calculated, so the saved file and the reported hash quietly stopped matching
Why didn't we catch this earlier?	An earlier test existed, but it re-read the file the same way it was written, which reverses the rewrite and hides the mismatch — a classic false-positive test that looked correct without actually checking anything
How did we fix it?	Changed the file-writing code (writer.py) to save the exact original bytes, with no automatic rewriting.
How do we know the fix actually worked?	Repeated the independent raw-byte check after the fix, same day — all 3 files now match exactly, byte for byte.
Why does this matter?	This is the exact mechanism a hospital's IT team would use to verify the reports themselves. If the team hadn't tested their own claim, the hospital would have been the one to find it broken.

Table 2 Verification summary

check	Verifled against	methods	Result
DICOM SOP Class ⟶ Modality resolution	DICOM PS3.3 IOD definitions	Direct cross-reference; every inference disclosed, not assumed	53/53 unambiguous
De-identification	Allow-list of non-identifying tags	Manual audit of extracted fields, all files	0/53 identifying fields present
Provenance hash correctness	Raw file bytes on disk	Independent recomputation, binary read (bypasses normal code path)	1 defect found & fixed same day; 0 remaining after fix
Reproducibility	Two independent invocations	diff on output artifact, byte level	Identical, 0 byte difference
Regression coverage	N/A (process metric)	pytest, live re-run	110/110 passing
Statistical eligibility (AUROC/AUPRC gate)	Eligibility floor policy (≥100 positive / ≥1,000 negative per label)	Per-label positive/negative count check before any point estimate is computed	0/14 eligible — highest count 3 (Fibrosis), vs. floor of 100; correct refusal, not a failure

Conclusions

This work demonstrates that CeXaR is a functional audit pipeline rather than a conceptual proposal, validated by 110 automated tests passing on hospital-representative DICOM data. [The system's design philosophy prioritizes transparency over unwarranted trust: statistical computations are deliberately withheld whenever sample sizes fall below a defined safety floor, preventing the reporting of false precision. Reproducibility is enforced at the architectural level, with every run generating cryptographic SHA-256 fingerprints that guarantee byte-for-byte identical outputs across independent executions. The development process was governed by adversarial code review and real-world failure testing, which drove the system's file-acceptance rate from an initial 0% to a 100% success rate on complex clinical data. This iterative hardening process reflects the rigor required for tools intended to operate on authentic clinical inputs rather than idealized test cases. The development team's prior experience in large-scale chest X-ray machine learning research was deliberately redirected toward building an objective auditing tool, rather than toward producing another competing diagnostic model — addressing a structural gap in the field:

the relative scarcity of independent assurance tools compared to the proliferation of diagnostic models. Future work is clearly scoped and gated behind proper governance milestones, with priority given to establishing clinical pipeline accuracy before expanding into user interface development or live hospital PACS integration. This staged approach reflects a commitment to ensuring the reliability of the system's core auditing logic prior to scaling deployment.

Acknowledgments

The authors would like to express their sincere appreciation to the open-source creators of TorchXRayVision for providing the baseline chest X-ray model utilized within the audit workbench architecture presented in this work. The authors are also deeply grateful to the VinBigData team for the VinDr-CXR dataset, made publicly available via Kaggle, which supplied the hospital-origin DICOM files essential to the development and refinement of the data ingestion engine. Appreciation is further extended to the clinical teams and radiologists whose raw majority-vote annotations enabled the verification of the eligibility floor mechanisms described herein. The authors also wish to acknowledge the open-source Orthanc PACS server community, whose contributions facilitated the validation of initial live network C-STORE transport connectivity during system development. Additionally, the authors gratefully acknowledge the NIH Clinical Center for the legacy ChestX-ray14 repository, which established the foundational precedent and patient-level splitting discipline that guided the engineering methodology of this study. Finally, technical recognition is due to the developers of Python-based medical imaging libraries, particularly the DCMTK toolkit, whose utilities were instrumental in diagnosing missing metadata tags encountered during system development

References

Eichelberg, Marco, Riesmeier, Joerg, Wilkens, Thomas, Hewett, Andrew J., Barth, Andreas and Jensch, Peter. (2004)..

Hadrien. (2022). TorchXRayVision: A Library of Chest X-ray Datasets and Models. Proceedings of the 5th International Conference on Medical Imaging with Deep Learning, PMLR 172, 231–249.

Eichelberg, Marco, Riesmeier, Joerg, Wilkens, Thomas, Hewett, Andrew J., Barth, Andreas and Jensch, Peter. (2004). Ten Years of Medical Imaging Standardization and Prototypical Implementation: The DICOM Standard and the OFFIS DICOM Toolkit (DCMTK). OFFIS Institute for Information Technology, Oldenburg, Germany.

Huang, Gao, Liu, Zhuang, Van Der Maaten, Laurens and Weinberger, Kilian Q. (2017). Densely Connected Convolutional Networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 4700–4708.