

Article

Annotix: An Integrated Desktop Platform for Multi-Modal Data Annotation, Collaborative Labeling, and End-to-End Machine Learning Training

Nicolás Baier Quezada ¹, Vanessa Uribe Hernández ¹, Haydée Barrientos Toledo ¹, Cristina Vargas Bustamante ¹, Martín Arrigo Figueroa ², Aaron Mancilla Leiva ², Felipe Brana Peña ² and Fernanda López-Moncada ^{1,*}

¹ Escuela de Tecnología Médica, Sede Puerto Montt, Universidad Austral de Chile, Puerto Montt 5504335, Chile; david.avila@uach.cl (N.B.Q.); vanessa.uribe@uach.cl (V.U.H.); haydee.barrientos@uach.cl (H.B.T.); cristina.vargas@uach.cl (C.V.B.)

² Escuela de Ingeniería Civil en Informática, Universidad Austral de Chile, Valdivia 5111187, Chile; martin.arrigo@alumnos.uach.cl (M.A.F.); aaron.mancilla@alumnos.uach.cl (A.M.L.); felipe.brana@alumnos.uach.cl (F.B.P.)

* Correspondence: fernanda.lopez@uach.cl

Abstract

Annotated dataset preparation remains a critical bottleneck in machine learning (ML) pipelines. Existing annotation tools—cloud-hosted services, self-hosted web applications, and lightweight desktop editors—each cover part of this workflow, but few combine broad annotation-type support with offline operation, integrated training, and serverless collaboration. We present Annotix, an open-source, cross-platform desktop application that brings the complete ML data preparation workflow—annotation, training, model-assisted labeling, and review—into a single privacy-preserving environment. It is implemented with a Rust/Tauri 2 backend and a React 19 frontend, and runs locally without server infrastructure or cloud services. Annotix was evaluated through three complementary studies. First, a comparative feature analysis positioned the platform against established annotation tools. Second, a controlled efficiency experiment compared Annotix, CVAT, and Label Studio, in which three evaluators annotated 60 synthetic images across bounding box and mask tasks, analyzed with Kruskal–Wallis and Dunn–Bonferroni post hoc tests. Third, a heuristic usability evaluation assessed standardized tasks on real medical images (retinal fundus and otoscopic images). Results indicate that Annotix achieves annotation efficiency competitive with established tools while offering broader integrated coverage, including end-to-end model training and serverless peer-to-peer collaboration. Annotix is freely available under the MIT license and targets privacy-sensitive domains such as medical imaging and ecological monitoring.



Academic Editors: Waddah Saeed and AbdulRahman Alsewari

Received: 10 April 2026

Revised: 25 May 2026

Accepted: 28 May 2026

Published: 3 August 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Keywords: data annotation; multi-modal annotation; offline-first ML; human-in-the-loop; machine learning pipeline

1. Introduction

Supervised machine learning requires large volumes of accurately labeled data to achieve robust generalization. In computer vision alone, tasks such as object detection [1], semantic segmentation [2], instance segmentation [3], pose estimation [4], and oriented object detection [5] each demand distinct annotation modalities, including bounding boxes, pixel-level masks, keypoint skeletons, and rotated rectangles. Beyond vision, time-series

classification [6], anomaly detection [7], and forecasting [8] introduce additional labeling paradigms that few existing tools address. This heterogeneity forces practitioners to adopt multiple disconnected tools, leading to fragmented workflows, format conversion errors, and significant overhead in dataset management.

Current annotation solutions fall into three broad categories: (i) cloud-hosted services (Roboflow [9], V7 [10], Supervisely [11]), which offer rich functionality but require uploading data to third-party servers—a constraint incompatible with privacy-sensitive domains such as clinical research and defense; (ii) self-hosted web applications (CVAT [12], Label Studio [13]), which provide flexibility but demand non-trivial server infrastructure and DevOps expertise; and (iii) lightweight desktop tools (LabelImg [14], LabelMe [15], MakeSense [16]), which operate offline but support limited annotation types and lack integration with training pipelines. No single category simultaneously addresses the requirements of diverse annotation modalities, offline-first data sovereignty, integrated multi-framework training, model-assisted labeling, and real-time collaboration without centralized servers. This fragmentation is a particular concern in regulated domains such as medical imaging, where data sovereignty is a hard constraint and annotation tooling has been identified as a critical bottleneck in the ML pipeline [17]. Prior work has further shown that tool design directly influences annotator behavior and inter-annotator variability [18], highlighting the importance of usability alongside functional coverage.

Beyond operational inefficiency, this fragmentation constitutes a structural barrier to entry: converting annotated data into a trained, deployable model requires Python expertise, GPU infrastructure setup, and familiarity with training frameworks—knowledge that most domain professionals lack, including clinical researchers and field biologists. The design premise of Annotix is that domain knowledge, not technical knowledge, should be the entry requirement for building a computer vision model.

We present Annotix, an open-source desktop application designed to unify the complete ML data preparation workflow within a single, privacy-preserving environment. The key contributions of this work are: (1) a unified offline annotation environment spanning seven image primitives, temporal video annotation, nine time-series paradigms, and tabular data labeling; (2) end-to-end training integration with 19 ML backends via automated environment provisioning; (3) model-assisted labeling through a built-in ONNX (Open Neural Network Exchange) Runtime inference engine; (4) serverless peer-to-peer collaboration over the QUIC (Quick UDP Internet Connections) protocol without cloud infrastructure; and (5) comprehensive interoperability with eleven export and eight import formats.

To position and evaluate Annotix, this work combines a comparative analysis with empirical evaluation. Section 2 positions the platform against existing annotation tools through a comparative feature analysis. The platform is then evaluated through two empirical studies: a controlled annotation efficiency experiment comparing Annotix against CVAT and Label Studio on standardized tasks (Section 4.2), and a heuristic usability evaluation conducted on real medical imaging data (Section 4.3). An end-to-end case study illustrating the complete workflow on a real medical imaging task is additionally presented in Section 4.4.

The remainder of this paper is organized as follows. Section 2 reviews related annotation tools and positions Annotix among them. Section 3 describes the experimental methodology. Section 4 presents the Annotix platform and the empirical results. Section 5 discusses the findings and their applicability. Section 6 concludes.

2. Related Work

Building on the categories of annotation tools outlined in Section 1, this section positions Annotix relative to existing platforms. Section 2.1 presents a comparative feature

analysis across representative tools, derived from a structured review of their documented capabilities; Section 2.2 identifies the two tools selected as reference comparators for the empirical evaluation reported in Section 4.

2.1. Comparative Feature Analysis

The comparative feature matrix (Table 1) was constructed through systematic review of the official documentation of each tool, accessed in March 2026. Features were selected according to a workflow coverage criterion: we included dimensions that directly determine the completeness of the ML data preparation pipeline, specifically annotation modality breadth, data type support beyond images, training pipeline integration, model-assisted labeling, collaboration model, format interoperability, deployment model (offline vs. server-dependent), and localization. Feature states were verified against documentation for all tools; direct testing was performed only for the tools involved in the experimental evaluation (Annotix, CVAT, Label Studio).

Table 1. Comparative analysis of annotation tools. Annotation types for images include: bounding box, oriented bounding box, polygon, mask, keypoints, landmarks, and classification.

Feature	LabelImg	LabelMe	CVAT	Label Studio	Roboflow	Annotix
Offline operation	✓	✓	×	×	×	✓
Annotation types	1	2	5	4	3	7
Video annotation	×	×	✓	×	×	✓
Time-series support	×	×	×	✓	×	✓
Tabular data	×	×	×	×	×	✓
Integrated training	×	×	×	×	✓ ¹	✓
Training backends	0	0	0	0	1	19
Model-assisted labeling	×	×	✓	✓	✓	✓
P2P collaboration	×	×	×	×	×	✓
Export formats	2	1	6	5	8	11
Import formats	0	0	4	3	5	8
Localization	1	1	2	1	1	10
Data sovereignty	✓	✓	Partial	Partial	×	✓

¹ Roboflow offers cloud-based training with limited model selection. ✓ = feature present; × = feature absent.

Table 1 summarizes the positioning of Annotix against representative tools across key dimensions.

This table characterizes feature breadth—presence and variant counts—rather than the depth, implementation maturity, or usability of individual capabilities. CVAT and Label Studio are mature platforms whose implementations of overlapping features (e.g., model-assisted labeling, format export, video annotation) may be more refined than those of Annotix, reflecting their longer development histories and larger user bases. Tools optimized for different use cases—such as collaborative cloud-based labeling at scale—may likewise exhibit strengths not captured by a feature count. Table 1 therefore serves to map the design space and locate the integrative niche that Annotix addresses.

2.2. Reference Comparators

Among the tools identified in the comparative analysis (Table 1), CVAT and Label Studio were selected as reference comparators for the empirical evaluation. This selection is justified on several grounds. First, both tools consistently appear as the primary benchmarks in the annotation tool literature and are widely adopted in academic and applied ML workflows [12,13], making them the closest approximation to a gold standard among open-source alternatives. Second, both are freely available, which ensures reproducibility of the comparison without licensing barriers—a prerequisite for scientific validity.

Third, both tools exemplify the paradigm against which Annotix's core contribution is most clearly evaluated: as annotation-only platforms that require server deployment, they represent the architectural model—and the pipeline boundary—that Annotix is designed to transcend. Where CVAT and Label Studio end at labeled data export, Annotix continues through training, inference, and model distribution within a single offline application, making this comparison the most informative test of whether integrating the full pipeline yields practical benefits in efficiency and usability. In our laboratory environment, Label Studio collaborative sessions are accessible only within the institutional network, and CVAT restricts certain functionalities in its free tier. These operational constraints—absent in Annotix—are precisely the conditions under which an efficiency and usability comparison is most informative for prospective users in resource-limited or privacy-sensitive settings. CVAT v2.61.1 and Label Studio v1.23.0 were deployed on the institutional laboratory server using their default configurations. During experimental sessions, each evaluator accessed these tools from their own workstation via the institutional network; Annotix ran locally on the same workstations without network dependency.

3. Materials and Methods

Annotix is an open-source desktop application built on a Rust/Tauri 2 backend and a React 19 frontend; the version described in this work is publicly available at <https://github.com/Debaq/Annotix> (accessed on 13 March 2026) under the MIT license, and its feature set (Section 4.1) was characterized directly from the software. The platform was evaluated through two empirical studies—an annotation efficiency experiment on synthetic stimuli and a heuristic usability evaluation on real medical images—complemented by an end-to-end case study on a real medical imaging task. Section 3.1 describes the within-subjects, Latin-square counterbalanced design shared by both evaluator-based experiments; Sections 3.2 and 3.3 describe the protocols of the efficiency and heuristic studies; Section 3.4 describes the case study protocol; Section 3.5 discloses the use of generative AI tools during manuscript preparation.

3.1. Study Design and Participants

Three evaluators (AE-1, AE-2, AE-3), members of the research laboratory with a background in Medical Technology and intermediate-to-advanced computer skills, participated in the annotation efficiency experiment (Section 3.2). For the heuristic usability evaluation (Section 3.3), six additional evaluators (AE-4 through AE-9) with the same Medical Technology background and equivalent computer proficiency were recruited, yielding a total of nine evaluators for the heuristic study. None of the nine had prior experience with any annotation platform. This profile reflects the typical onboarding scenario for new members of a research or clinical team and avoids the confound of tool-specific expertise, ensuring that observed differences in efficiency are attributable to intrinsic workflow characteristics of each tool rather than to prior skill differences.

Before the first session with each tool, evaluators received a standardized 10-min demonstration conducted by the principal investigator (PI), covering the core annotation workflow required for that session's tasks. During all timed sessions, the PI did not provide usage assistance; only protocol-related questions (what to annotate, how many structures) were answered.

For the heuristic usability evaluation (Section 3.3), all nine evaluators (AE-1 through AE-9) participated. AE-1, AE-2, and AE-3 provided their ratings after completing the efficiency sessions, having had direct hands-on experience with all three tools. The remaining six evaluators (AE-4 through AE-9) received an equivalent 10-min demonstration of each

tool plus a brief familiarization period prior to the heuristic task, ensuring all evaluators had the informed basis required to provide severity ratings.

A within-subjects design was employed across both experiments: each evaluator used all three tools across all tasks. Tool order was counterbalanced using a Latin square design—a balanced arrangement ensuring each tool appeared in each ordinal position exactly once per replicate. For the efficiency experiment, the three evaluators (AE-1, AE-2, AE-3) corresponded to a single Latin square replicate. For the heuristic evaluation, the design was extended to three complete replicates (AE-1 through AE-9), so that each tool appeared in each position exactly three times across the full sample, providing robust control for learning and fatigue effects.

AE-1, AE-4, AE-7: Annotix → CVAT → Label Studio;

AE-2, AE-5, AE-8: CVAT → Label Studio → Annotix;

AE-3, AE-6, AE-9: Label Studio → Annotix → CVAT.

3.2. Annotation Efficiency Evaluation of Synthetic Images

3.2.1. Stimulus Materials

Two sets of 30 synthetic images (400×400 px, JPG) were generated: SYN-BBOX-001–030 for the bounding box task and SYN-MASK-001–030 for the mask task. Each image depicted a circular field of view (black background, white-textured circular region) containing standardized geometric shapes. The SYN-BBOX set contained a red circle, a blue circle, a yellow star, an orange rectangle, and a green triangle; the SYN-MASK set contained a bicolor circle (half red/half blue), a thick yellow worm, a thin green worm, and an orange star. The synthetic stimulus images are available in the Supplementary Materials. Synthetic rather than real images were used to ensure that all evaluators encountered identical visual content across tools and sessions, eliminating any variability attributable to image complexity or anatomical knowledge. The same 30 images were annotated by all three evaluators in all three tools, yielding 90 time measurements per tool per task.

3.2.2. Procedure

Each evaluator performed a standardized five-step protocol (CMRD: Create, Move, Resize, Delete) per image, designed to exercise the complete lifecycle of an annotation object. For the bounding box task: (1) create a bounding box over the red circle; (2) create a bounding box over the blue circle; (3) create a bounding box over the yellow star and move it to enclose the orange rectangle; (4) create a bounding box over the apex of the green triangle and resize it to enclose all three vertices; (5) delete the bounding box of the blue circle. For the mask task: (1) create a complete mask of the bicolor circle (half red/half blue); (2) create a mask of the thick worm (yellow); (3) reduce brush diameter and create a mask of the thin worm (green); (4) erase the red half of the bicolor circle, retaining only the blue half; (5) delete the complete class of the thin worm. Time per image (in seconds) was recorded from first interaction to task completion. Images within each set were presented in the same shuffled order across evaluators and tools.

3.2.3. Statistical Analysis

Normality within each group (tool $\times$ task) was assessed using the Shapiro–Wilk test. For the mask task, CVAT did not meet the normality assumption ($p < 0.05$), while Annotix and Label Studio did ($p > 0.05$). Following the convention that non-normality in any group within a comparison warrants a non-parametric approach for the full set, Kruskal–Wallis tests ($\alpha = 0.05$) were applied to both the bounding box and mask datasets. Pairwise post hoc comparisons (Annotix vs. CVAT, Annotix vs. Label Studio, CVAT vs. Label Studio) were performed using Dunn’s test with Bonferroni correction. All analyses were conducted in GraphPad Prism v10.6.0; results are presented as box-and-whisker plots (Tukey style).

3.3. Heuristic Usability Evaluation of Medical Images

3.3.1. Method

Heuristic usability was evaluated using the 10 heuristics of Nielsen and Molich [19], each rated on a severity scale of 0 (no usability problem) to 4 (usability catastrophe) per heuristic per tool. This evaluation was conducted by nine evaluators using real medical images to contextualize usability within clinically relevant annotation tasks—the primary application domain of the research team—and to simultaneously demonstrate the platform’s applicability to medical imaging workflows. Two independent evaluation blocks were defined, one per annotation modality.

3.3.2. Task Protocol

Block A—Retinal fundus images (bounding box): Evaluators (1) created a project with two classes (fovea, optic_disc); (2) loaded 5 real retinal fundus images; (3) annotated each image with one bounding box per class (fovea and optic disc); and (4) exported in COCO JSON format. Block B—Otoscopic images (mask): Evaluators (1) created a project with class hammer_umbo; (2) loaded 5 real tympanic membrane images; (3) annotated each image with a mask covering the malleus handle and umbo as a continuous region; and (4) exported in COCO JSON format. Anatomical structures were selected based on having visually well-defined borders, minimizing variability attributable to anatomical ambiguity (i.e., uncertainty in identifying structure boundaries due to natural biological variation) rather than tool behavior. Each block was completed independently in all three tools in counterbalanced order (same Latin square as Section 3.1). Heuristic severity scores were recorded on the evaluation worksheet immediately after completing each block in each tool, while the experience was still fresh.

3.3.3. Analysis

Mean severity scores were computed for each heuristic and tool combination. Inter-rater reliability for each heuristic was quantified using the Intraclass Correlation Coefficient (ICC; two-way mixed model, absolute agreement, single measures), with the three tools as targets and the nine evaluators as raters. $ICC \geq 0.70$ was set as the acceptable reliability threshold [20]; values below this threshold are interpreted descriptively, given the exploratory nature of heuristic evaluation [19].

3.4. End-to-End Case Study Protocol

To complement the comparative experiments, an end-to-end case study was conducted to illustrate the full annotate–train–deploy workflow within a single environment (results reported in Section 4.4). A 150-image subset of the Messidor diabetic retinopathy dataset was obtained from Kaggle and imported into Annotix; the platform’s built-in randomization and split functionality was used to partition the set into 100 training and 50 validation images. Bounding-box annotations were produced by a single trained operator and reviewed by a domain expert prior to training. Training was executed locally through the integrated YOLOv11 detection backend at 640×640 input resolution; the resulting PyTorch (v2.6.0) checkpoint was converted to ONNX through the integrated export module and deployed as a static client-side web demonstration for in-browser inference. Reported metrics correspond to the validation set. The case study is reported as a qualitative demonstration of pipeline integration on real domain data, not as a controlled experimental comparison or a multi-backend benchmark. Full protocol details—class definitions, annotation guidelines, training hyperparameters, per-class metrics, and deployment configuration—are reported in <https://nexam.org/dird/en.html#modelo> (accessed on 27 May 2026).

3.5. Use of Generative AI Tools

During the preparation of this manuscript, generative AI assistance (Claude Sonnet 4.6, Anthropic) was used for the following purposes: (1) manuscript drafting and revision support; (2) generation of SVG flow diagrams presented in Figures 1, 4 and 7; (3) generation of the radar chart presented in Figure 6, based on the heuristic severity data reported in Section 4.3; and (4) code debugging support. No AI tool was used to generate, modify, or interpret experimental data. All AI-assisted content was reviewed and edited by the authors, who take full responsibility for the accuracy and integrity of this publication.

4. Results

Annotix is characterized first as a system—its architecture and its annotation, training, collaboration, and interoperability subsystems (Section 4.1). The empirical findings then address how the platform performs in practice: annotation efficiency on synthetic stimuli (Section 4.2) and heuristic usability on real medical images (Section 4.3). Section 4.4 closes with an end-to-end case study illustrating the full pipeline on a real medical imaging task.

4.1. The Annotix Platform

4.1.1. Architectural Overview

Annotix follows a layered architecture consisting of three principal tiers: a presentation layer (React 19 + TypeScript); a bridge layer (Tauri 2 IPC, Inter-Process Communication); and a core layer (Rust). The frontend communicates with the backend through 132 registered IPC commands. This architecture is illustrated in Figure 1.

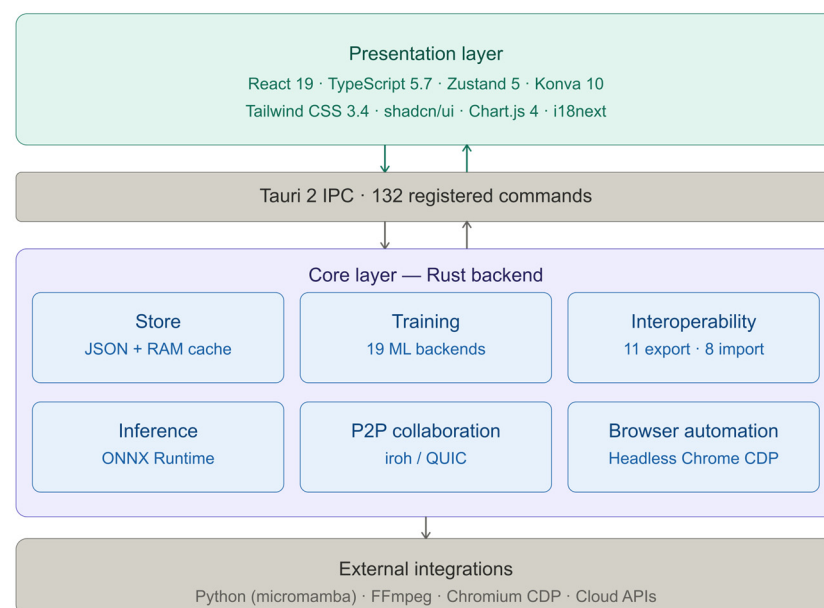


Figure 1. Layered architecture of Annotix. The platform is organized in three tiers: a presentation layer (React 19 + TypeScript 5.7) handling the annotation canvas and user interface via Konva 10 and Zustand 5; a bridge layer implementing 132 Tauri 2 IPC commands for typed communication between frontend and backend; and a core Rust layer hosting the project store (in-memory cache with atomic JSON persistence), the training subsystem (19 ML backends), the export/import engine (11 and 8 formats respectively), the ONNX Runtime inference engine, the peer-to-peer collaboration system (iroh/QUIC), and the browser automation module (headless Chrome via CDP). External dependencies (Python v3.12 /micromamba, FFmpeg, Chromium) are managed automatically by the core layer.

The Rust backend employs an in-memory cache with atomic persistence. Each project is represented as a single JSON file containing all metadata, class definitions, image entries, annotations, video tracks, training jobs, and inference models.

Two accessor patterns govern all project interactions: read operations are served directly from memory, while write operations trigger an atomic flush via a temporary-file-then-rename strategy, ensuring crash consistency through filesystem-level atomicity guarantees of the `rename(2)` system call [21]. This design eliminates the need for an embedded database, reducing dependency complexity while maintaining data integrity.

The annotation canvas is implemented using Konva 10, providing hardware-accelerated rendering through the HTML5 Canvas API. All entities use Universally Unique Identifier (UUID) v4 identifiers; serialization is handled by `serde/serde_json` on the Rust side and native JSON on the TypeScript side.

4.1.2. Annotation Capabilities

Annotix provides seven image annotation primitives: (1) bounding boxes (axis-aligned rectangles for object detection); (2) oriented bounding boxes (rotated rectangles parameterized by center, dimensions, and angle $\theta \in [0^\circ, 360^\circ)$, essential for aerial imagery [5]); (3) polygons (ordered vertex sequences for instance segmentation); (4) segmentation masks (raster-based brush tool with configurable radius and eraser mode for pixel-level annotation); (5) keypoints with skeleton (anatomical or structural points connected by predefined graphs, with built-in presets for COCO 17-point human pose [4], face 68-point, hand 21-joint, and MediaPipe 33-point full body [22]); (6) landmarks (named reference points with individual text labels); and (7) classification labels (single-label and multi-label image-level classification). Figure 2 illustrates the annotation canvas during a bounding box session on retinal fundus images, showing the image gallery, active annotations, class panel, and integrated export and training controls.

The annotation canvas supports zoom, pan, 90° image rotation, real-time grid overlay, and a complete undo/redo system. Keyboard shortcuts enable rapid tool switching and class selection. A non-destructive image adjustment panel provides real-time control over brightness, contrast, color temperature, sharpness, and CLAHE (Contrast Limited Adaptive Histogram Equalization), operating exclusively on the canvas rendering layer without modifying source files on disk. This is particularly relevant for low-contrast medical imagery where anatomical boundaries may be difficult to delineate at default settings. Figure 3 illustrates the effect of CLAHE on an otoscopic image, where the borders of the pars tensa and the umbo become substantially more defined prior to mask annotation.

Video annotation follows a track-based paradigm: FFmpeg extracts frames at configurable FPS; tracked objects receive class labels and persistent identifiers; keyframes are manually placed and intermediate frames receive linearly interpolated bounding boxes; a bake operation materializes interpolated annotations for export. This reduces annotation effort approximately proportionally to the keyframe interval. Additionally, Annotix supports nine time-series annotation paradigms (classification, forecasting, anomaly detection, segmentation, pattern recognition, event detection, regression, clustering, imputation) from CSV files, and an integrated tabular data editor for classical ML tasks.

4.1.3. Training and Inference

Annotix integrates 19 ML training backends organized by task (Table 2). Training environments are provisioned automatically using micromamba, a lightweight C++-based conda-compatible package manager that installs backend dependencies in isolated environments. GPU detection is automatic: NVIDIA GPUs via CUDA and Apple Silicon via Metal Performance Shaders. Four execution modes are supported: local (real-time metric

streaming), download package (self-contained ZIP), cloud (Vertex AI, Kaggle, Lightning AI, HuggingFace Spaces, Saturn Cloud), and browser automation (headless Chrome via Chrome DevTools Protocol for Google Colab GPU sessions). Six domain-specific training presets (small_objects, industrial, traffic, edge_mobile, medical, aerial) provide optimized hyperparameter configurations. Training procedures—including dataset partitioning into training and validation splits, batch size, and learning rate schedules—are configurable through each backend’s native parameters and are not imposed by the platform.

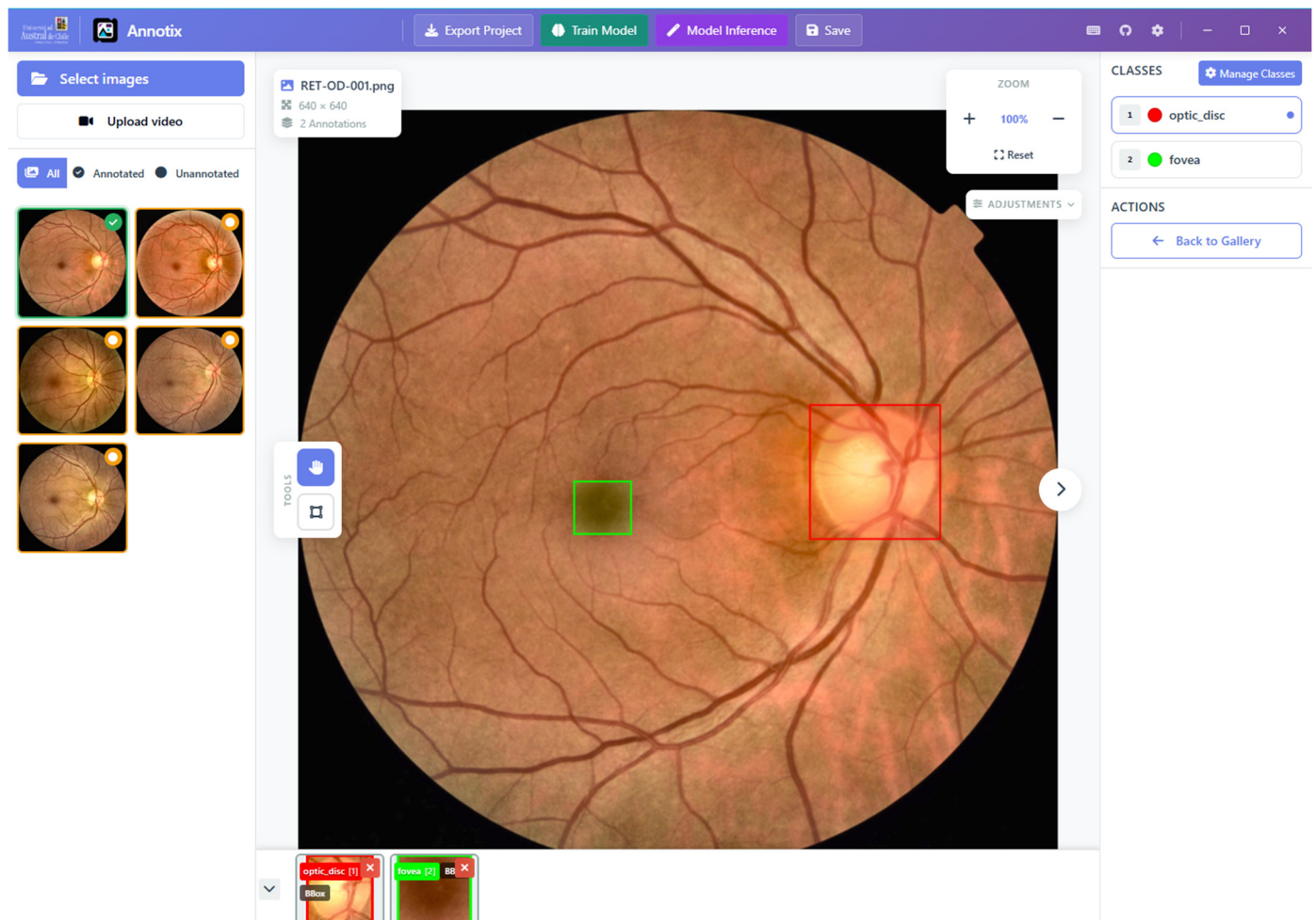


Figure 2. Annotation canvas of Annotix during a bounding box annotation session on retinal fundus images. The interface shows: **(left)** the image gallery panel with annotation status indicators (green checkmark = annotated, orange dot = pending); **(center)** the main canvas displaying a retinographic image with two active bounding box annotations—fovea (green, selected, with resize handles visible) and optic disc (red); **(right)** the class panel listing the two defined classes with their associated colors and visibility controls. The bottom strip displays per-annotation thumbnails with class label, instance count, and quick-delete controls. The top toolbar provides one-click access to project export, model training, and model inference workflows. The tools panel (vertical, center-left) shows the pan and bounding box tools; the zoom and image adjustment controls are accessible from the top-right of the canvas. This view corresponds to Block A of the heuristic usability evaluation (Section 3.3.2), in which evaluators annotated fovea and optic disc structures across five retinographies per tool.

The inference subsystem enables model-assisted labeling through a built-in ONNX Runtime integration [23]. Users upload ONNX or PyTorch models; the system performs automatic metadata extraction, flexible class mapping to project definitions, batch inference with configurable confidence thresholds, and per-prediction accept/reject controls in a human-in-the-loop workflow. Together, these five stages—annotate, train, export to

ONNX, batch inference, and human review—form a closed iterative cycle that runs entirely on the user’s machine without cloud dependency, as illustrated in Figure 4. A video demonstration of this end-to-end workflow is available as Video S1 in the Supplementary Materials. Trained models can be exported to PyTorch (.pt), ONNX, TorchScript, TFLite, CoreML, and TensorRT formats.

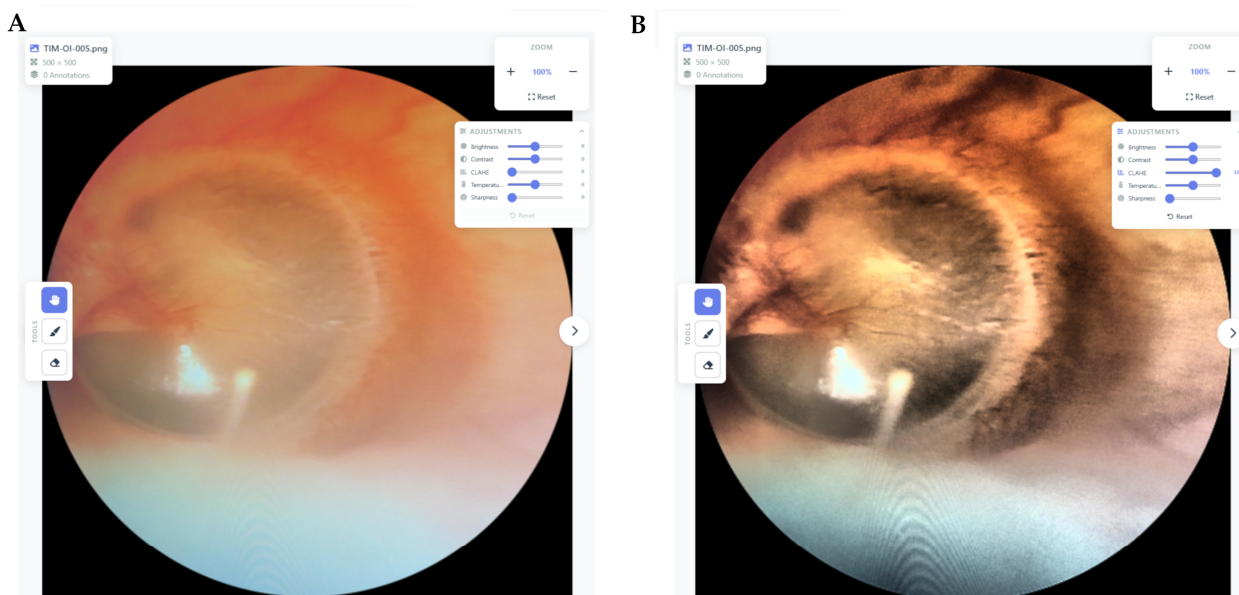


Figure 3. Image adjustment capabilities of Annotix applied to an otoscopic image. (A) Original image at default adjustment settings; the tympanic membrane structures (pars tensa, malleus handle) are partially obscured by low contrast and uneven illumination. (B) Same image after applying CLAHE (Contrast Limited Adaptive Histogram Equalization) via the built-in Adjustments panel, alongside increases in brightness and contrast. The anatomical borders of the pars tensa and the light reflex of the umbo become substantially more defined, facilitating precise mask annotation without modifying the source file on disk. Adjustment controls (brightness, contrast, CLAHE, color temperature, sharpness) operate exclusively on the canvas rendering layer and are non-destructive.

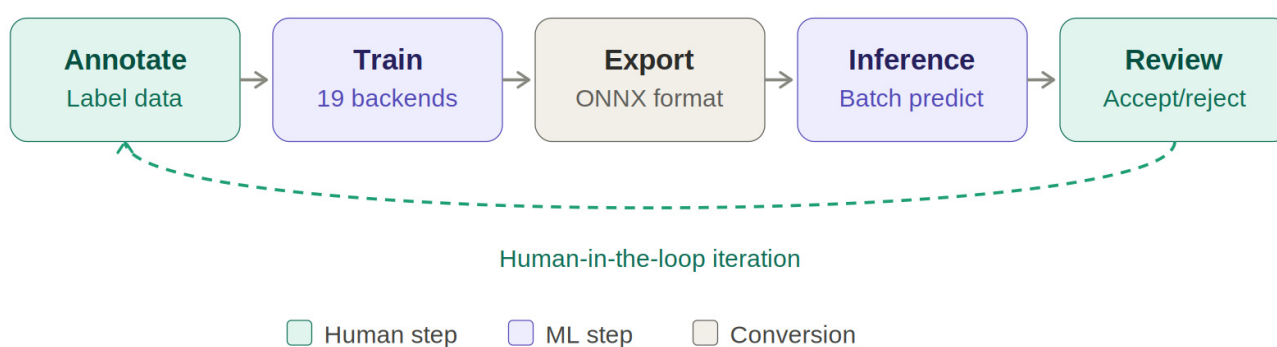


Figure 4. Human-in-the-loop annotation pipeline implemented in Annotix. The cycle consists of five stages: (1) Annotate—manual labeling of the dataset using any supported annotation primitive; (2) Train—model training via any of the 19 integrated ML backends, executed locally without cloud infrastructure; (3) Export—conversion of the trained model to ONNX format for deployment within the platform; (4) Inference—batch prediction over unlabeled images using the built-in ONNX Runtime engine; (5) Review—human acceptance or rejection of machine-generated predictions, which are converted to editable annotations upon acceptance. The dashed arc represents the iterative feedback loop: accepted predictions augment the labeled dataset, triggering a new training cycle with improved supervision. Teal boxes indicate human-driven steps; purple boxes indicate automated ML computation; gray indicates format conversion.

Table 2. Training backends integrated in Annotix.

Task	Backend	Architectures	References
Object Detection	Ultralytics	YOLOv8, v9, v10, v11, v12	Jocher et al. [24]
Object Detection	Ultralytics	RT-DETR-l, RT-DETR-x	Jocher et al. [24]; Zhao et al. [25]
Object Detection	Roboflow	RF-DETR-base, RF-DETR-large	Robinson et al. [26]
Object Detection	MMDetection	30+ architectures	Chen et al. [27]
Semantic Segmentation	SMP	U-Net, DeepLabV3+, FPN, PSPNet, MAAnet, LinkNet, PAN	Iakubovskii [28]; Ronneberger et al. [29]; Chen et al. [30]
Semantic Segmentation	HuggingFace	SegFormer, Mask2Former	Xie et al. [31]; Cheng et al. [32]
Semantic Segmentation	MMSegmentation	Full OpenMMLab catalog	MMSegmentation Contributors [33]
Instance Segmentation	Detectron2	Mask R-CNN, Cascade Mask R-CNN	He et al. [3]; Wu et al. [34]
Pose Estimation	MMPose	HRNet, ViTPose, RTMPose	MMPose Contributors [35]; Sun et al. [36]; Jiang et al. [37]
Oriented Detection	MMRotate	Oriented R-CNN, RoI Transformer	Xie et al. [5]; Zhou et al. [38]
Image Classification	timm	700+ architectures	Wightman [39]; Wightman et al. [40]
Image Classification	HuggingFace	ViT, BEiT, DeiT, Swin Transformer	Dosovitskiy et al. [41]; Bao et al. [42]; Touvron et al. [43]; Liu et al. [44]
Time-Series (DL)	tsai	InceptionTime, LSTM-FCN, TSTPlus	Oguiza [45]
Time-Series (DL)	PyTorch Forecasting	TFT, N-BEATS, DeepAR	Beitner [46]; Lim et al. [47]
Anomaly Detection	PyOD	Isolation Forest, LOF, AutoEncoder	Zhao et al. [48]
Clustering	tslearn	k-Shape, DTW Barycenter	Tavenard et al. [49]
Imputation	PyPOTS	SAITS, Transformer-based	Du [50]
Pattern Recognition	STUMPY	Matrix Profile	Law [51]
Tabular ML	scikit-learn	RandomForest, SVM, kNN, GBM	Pedregosa et al. [52]

4.1.4. Peer-to-Peer Collaboration

Annotix implements serverless real-time collaboration using the iroh protocol [53] over QUIC (RFC 9000 [54]). Three iroh components underpin the system: iroh-docs (replicated key-value document synchronization for annotation state), iroh-blobs (content-addressed transfer for image distribution), and iroh-gossip (pub-sub messaging for real-time coordination). A host creates a session and distributes a unique session code; collaborators join via direct encrypted connections without intermediary servers. Configurable permissions govern collaborator rights (image upload, class editing, annotation deletion, export). Image-level locking prevents conflicting edits: collaborators acquire exclusive write access per image; batch assignment distributes separate subsets; automatic lock expiration prevents deadlocks from disconnected peers.

4.1.5. Interoperability

Annotix supports 11 export formats: YOLO Detection, YOLO Segmentation, COCO JSON [4], Pascal VOC [55], CSV Detection, CSV Classification, CSV Keypoints, CSV Landmarks, Folders by Class, U-Net Masks, and TIX (native project archive). Eight import formats are supported with automatic format detection based on file structure and content patterns: YOLO (detection and segmentation), COCO JSON, Pascal VOC, CSV (four variants), U-Net Masks, Folders by Class, and TIX. The platform is localized in 10 languages.

4.2. Annotation Efficiency Evaluation

4.2.1. Bounding Box Task

Shapiro–Wilk tests indicated that none of the three groups met the normality assumption for the bounding box task (Annotix: $W = 0.897$, $p = 0.007$; CVAT: $W = 0.634$, $p < 0.0001$; Label Studio: $W = 0.802$, $p < 0.0001$). The Kruskal–Wallis test revealed a statistically significant difference in annotation time across tools ($H(2) = 52.41$, $p < 0.0001$). Median annotation times per image were: Annotix 26.50 s (IQR 25.25–30.08), CVAT 41.67 s (IQR 38.00–46.75), and Label Studio 41.50 s (IQR 37.92–46.58). Dunn–Bonferroni post hoc comparisons showed that Annotix was significantly faster than both CVAT (mean rank difference = 43.15, $Z = 6.40$, $p < 0.0001$) and Label Studio (mean rank difference = 41.35, $Z = 6.13$, $p < 0.0001$), while no significant difference was found between CVAT and Label Studio (mean rank difference = 1.80, $Z = 0.27$, $p > 0.9999$). Results are presented in Figure 5C.

4.2.2. Mask Task

Shapiro–Wilk tests for the mask task indicated that Annotix ($W = 0.959$, $p = 0.286$) and Label Studio ($W = 0.939$, $p = 0.083$) met the normality assumption, while CVAT did not ($W = 0.846$, $p = 0.0005$). Because at least one group violated normality, Kruskal–Wallis was applied to all three groups. The test revealed a statistically significant difference in annotation time ($H(2) = 65.00$, $p < 0.0001$). Median annotation times were: Annotix 31.83 s (IQR 28.58–34.00), Label Studio 52.17 s (IQR 45.92–61.17), and CVAT 62.00 s (IQR 58.58–72.00). Dunn–Bonferroni post hoc tests revealed significant differences across all three pairs: Annotix vs. CVAT (mean rank difference = 53.27, $Z = 7.90$, $p < 0.0001$), Annotix vs. Label Studio (mean rank difference = 36.08, $Z = 5.35$, $p < 0.0001$), and CVAT vs. Label Studio (mean rank difference = 17.18, $Z = 2.55$, $p = 0.033$), indicating that all three tools differed significantly from one another for the mask task. Results are presented in Figure 5F.

4.3. Heuristic Usability Evaluation

In the heuristic usability evaluation (0–4 scale, $n = 9$ evaluators), Annotix achieved the lowest mean total severity across all heuristics (0.29), followed by Label Studio (0.82) and CVAT (1.39). No tool reached a group mean severity of ≥ 2.5 for any heuristic, indicating an absence of critical usability failures across all three platforms. Individual ratings of 4 (usability catastrophe) appeared sporadically in Label Studio (H3, H9, H10) and CVAT (H2, H7, H10), but not in Annotix; these reflect specific frustrations of single evaluators rather than systematic usability defects.

CVAT showed the highest severity on H7 (Flexibility and efficiency of use, mean = 2.22), H1 (Visibility of system status, mean = 1.89), and H10 (Help and documentation, mean = 1.67). These findings suggest that CVAT's interface, while functionally comprehensive, imposes a steeper learning curve: workflow acceleration features such as keyboard shortcuts and annotation templates were perceived as non-discoverable, and documentation entry points were not readily accessible during task execution. Label Studio's highest severities were H1 (Visibility of system status) and H5 (Error prevention), both at mean = 1.33, reflecting uncertainty about the current state of the platform during multi-step

operations such as project configuration and annotation modification. Annotix showed the lowest individual heuristic values, with no heuristic exceeding 0.67 (H5: Error prevention). Inter-rater agreement varied across heuristics. The highest agreement was observed for H7 (Flexibility and efficiency of use; ICC = 0.676), the dimension with the most differentiated severity profile across tools, indicating that the nine evaluators consistently identified workflow efficiency as the most discriminating factor between platforms. Agreement on the remaining heuristics was lower (ICC range: -0.07 to 0.45), consistent with the exploratory nature of heuristic evaluation, in which inter-evaluator variability is methodologically expected [19]. Severity scores are accordingly interpreted descriptively. Figure 6 presents the heuristic severity profiles of all three tools as a radar chart.

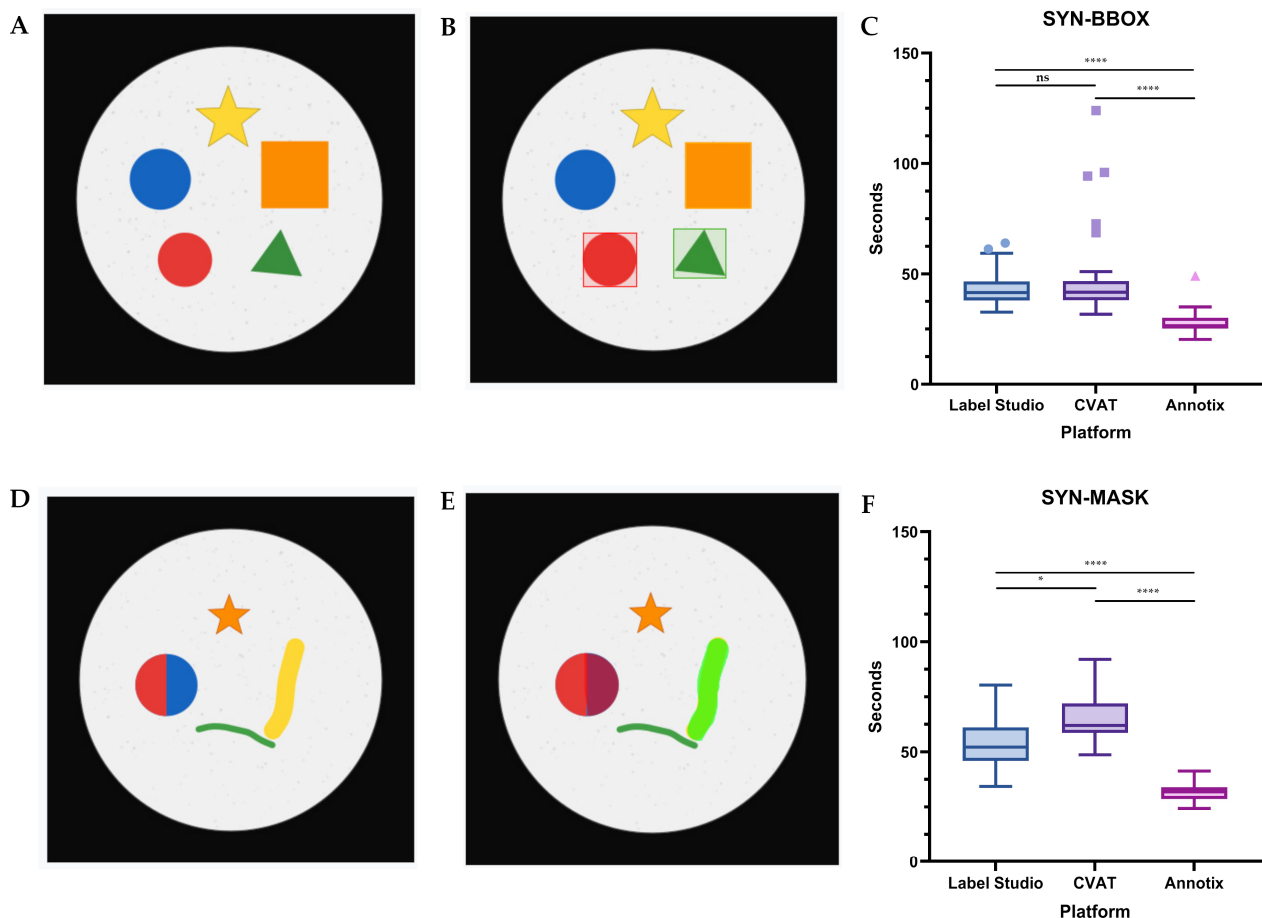


Figure 5. Synthetic stimuli and annotation efficiency results for the bounding box (top row) and mask (bottom row) tasks. (A) Representative unannotated synthetic image from the SYN-BBOX set (400 × 400 px), depicting a circular field of view containing standardized geometric shapes (red circle, blue circle, yellow star, orange rectangle, green triangle) on a black background. (B) Same image after completing the five-step bounding box CMRD protocol in Annotix: bounding boxes shown for the red circle and green triangle following steps 1 and 4 of the protocol. Annotations from Annotix are shown as representative; visual rendering of bounding boxes and masks is equivalent across all three platforms. (C) Box-and-whisker plots (Tukey style) of annotation time per image (seconds) for the SYN-BBOX task; $n = 90$ observations per platform (30 images × 3 evaluators). (D) Representative unannotated synthetic image from the SYN-MASK set, containing a bicolor circle (half red/half blue), a thick yellow worm, a thin green worm, and an orange star. (E) Same image after completing the five-step mask CMRD protocol in Annotix: masks shown for the bicolor circle and the thick worm (green). (F) Box-and-whisker plots (Tukey style) of annotation time per image (seconds) for the SYN-MASK task; $n = 90$ per platform. In both (C,F), horizontal brackets indicate pairwise comparisons: ns = not significant; * $p < 0.05$; **** $p < 0.0001$ (Dunn's test with Bonferroni correction, $\alpha = 0.05$). AN = Annotix; CV = CVAT; LS = Label Studio.

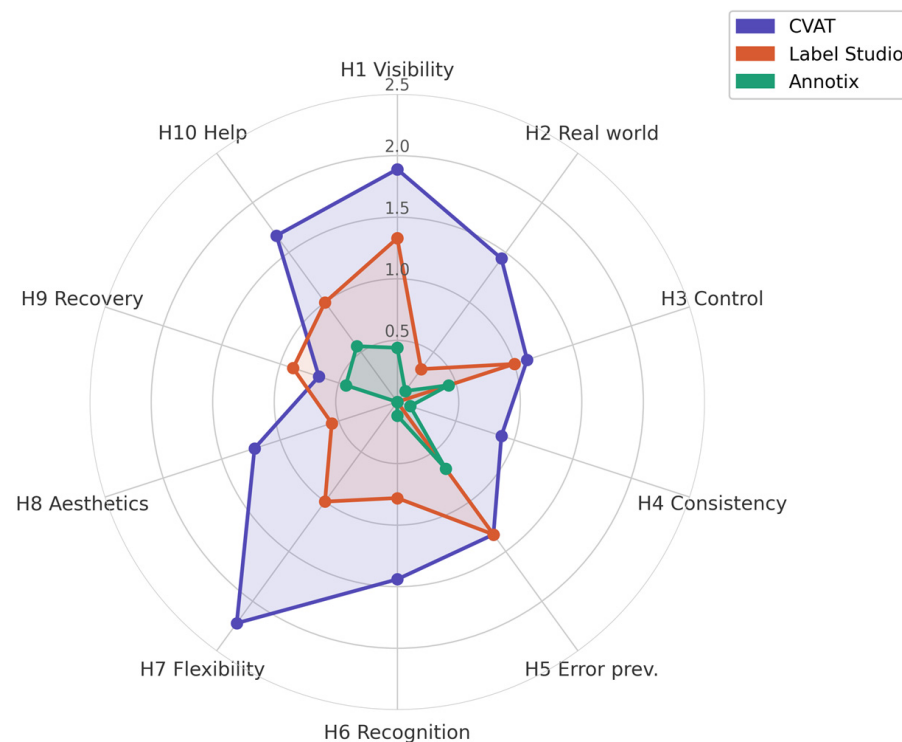


Figure 6. Heuristic severity profiles of the three annotation platforms across the ten Nielsen–Molich heuristics, averaged across nine evaluators. The scale ranges from 0 = no usability problem to 4 = usability catastrophe. Each axis represents one heuristic; the area enclosed by each polygon reflects the overall severity burden of that platform. Annotix is shown in green, CVAT in purple, and Label Studio in orange. Numeric values and interpretation are provided in Section 4.3. H1 Visibility of system status; H2 Match between system and real world; H3 User control and freedom; H4 Consistency and standards; H5 Error prevention; H6 Recognition rather than recall; H7 Flexibility and efficiency of use; H8 Aesthetic and minimalist design; H9 Help users recognize, diagnose, and recover from errors; H10 Help and documentation.

4.4. End-to-End Case Study: Diabetic Retinopathy Lesion Detection

Finally, to illustrate the complete annotate–train–deploy workflow within a single environment, Annotix was used to build DIRD v1, a diabetic retinopathy lesion detector trained on a 150-image subset of the Messidor dataset obtained from Kaggle. This case study is intended as a qualitative demonstration of pipeline integration on real domain data, not as a controlled experimental comparison or a multi-backend benchmark.

The dataset was randomized and split into 100 training and 50 validation images directly within the platform. A single trained operator produced 1016 bounding-box annotations across seven retinal classes in approximately four hours, with expert review. Training was performed locally through the integrated YOLOv11 backend at 640×640 input resolution, yielding $mAP@50 = 0.826$, precision = 0.82, and recall = 0.79 on the validation set. The trained model was exported from PyTorch to ONNX through the integrated export module and deployed as a static client-side web demonstration at <https://nexam.org/dird> (accessed on 27 May 2026), performing real-time inference in the browser without server round-trips. Stages 1–4 of the pipeline (Figure 7) were performed entirely inside Annotix on a single workstation; only the final web hosting runs on a static external host. Additional implementation details, dataset characterization, the trained ONNX model, and a live preview of the deployed system for the DIRD v1 case study are publicly available at <https://nexam.org/dird/en.html#modelo> (accessed on 27 May 2026).

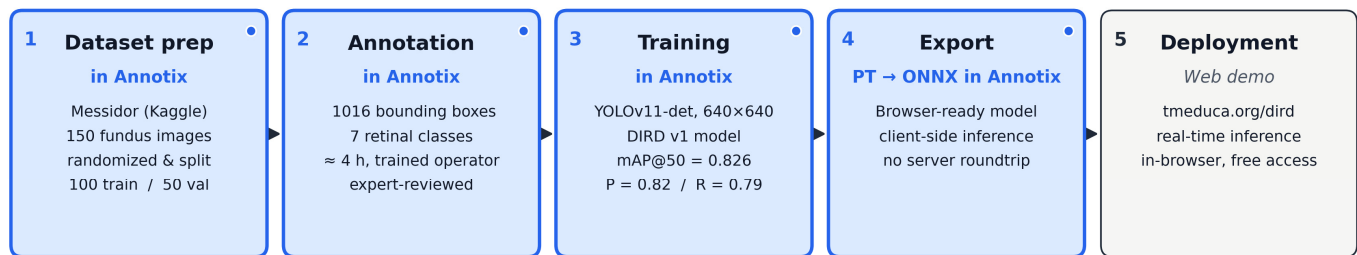


Figure 7. End-to-end case study: Annotix workflow for the DIRD v1 (Diabetic Retinopathy lesion Detector). Five-stage pipeline executed on a 150-image subset of the Messidor dataset (Kaggle): dataset preparation (100 train/50 validation), annotation of 1016 bounding boxes across seven retinal classes (≈ 4 h, single trained operator, expert-reviewed), training of a YOLOv11 detection model at 640×640 , PyTorch $\rightarrow$ ONNX export, and browser-based deployment for client-side real-time inference at <https://nexam.org/dird> (accessed on 27 May 2026). Reported metrics on the validation set: $\text{mAP@50} = 0.826$, precision = 0.82, recall = 0.79. Stages 1–4 were performed entirely inside Annotix; only the final web hosting runs on a static external host.

5. Discussion

Three observations emerge from the empirical evaluation: Annotix is competitive in annotation efficiency relative to established open-source tools, shows the lowest heuristic severity among the three platforms tested, and integrates the full ML data preparation cycle within a single offline environment. The discussion opens with Annotix’s positioning as an integration-focused contribution (Section 5.1), interprets the efficiency (Section 5.2) and usability (Section 5.3) results, examines applicability to medical imaging workflows (Section 5.4), and closes with limitations and future directions (Section 5.5).

5.1. Annotix as a Tool Paper Contribution

The central contribution of this work is the integration of the full ML data preparation cycle within a single offline environment. As established in Section 1, several mature platforms partially address segments of this pipeline: CVAT [12] and Label Studio [13] for annotation, Roboflow [9] for cloud-based training, and lightweight desktop tools for offline labeling. The medical-imaging annotation tool survey of Aljabri et al. [17] confirms that no current open-source solution closes annotation, training, and model distribution within a single locally deployable environment. Annotix is positioned as an integration-focused engineering contribution targeting this specific gap.

What defines Annotix’s design space is not any individual feature but the co-occurrence of offline data sovereignty, multi-modal annotation, 19 training backends with automatic environment provisioning, ONNX-assisted labeling, and serverless P2P collaboration within a single low-friction environment, as summarized in Table 1. The efficiency and usability results reported in Sections 3.2 and 3.3 indicate that this combined environment is practically viable for first-time users.

5.2. Annotation Efficiency

The efficiency results reveal consistent patterns across both modalities. For a user approaching the ML pipeline without prior technical training, annotation efficiency matters not only as a performance metric but as a barrier to completion: each additional point of friction in the workflow represents an opportunity to abandon the process before producing a usable model. For the bounding box task, CVAT and Label Studio were statistically indistinguishable from each other ($p > 0.9999$), while both differed substantially from Annotix ($p < 0.0001$ in both cases). This convergence between the two web-based tools suggests that the performance difference reflects the browser-mediated interaction model—including per-request network latency, page rendering overhead, and multi-step

project navigation—rather than tool-specific design choices. This interpretation is consistent with Human–Computer Interaction (HCI) evidence showing that even modest interactive latency on the order of hundreds of milliseconds substantially degrades user efficiency in structured exploratory tasks [56], and with recent empirical findings that browser-based applications systematically demand higher CPU and memory resources than their native counterparts for equivalent functionality [57]—an architectural overhead intrinsic to browser-mediated request–render cycles. This architectural distinction is part of each tool’s design rather than a separable parameter: CVAT and Label Studio are not distributed as native desktop applications, and Annotix’s offline-first operation is not a configurable switch.

The mask task showed more pronounced differences: all three tools differed significantly, with the ordering Annotix < Label Studio < CVAT. The additional gap between CVAT and Label Studio for mask annotation ($p = 0.033$) likely reflects structural differences in how the two web tools implement brush-based segmentation. CVAT requires more initialization steps and tool-mode switching; Label Studio provides a more streamlined brush interface. Annotix’s advantage in both tasks is attributable to its native canvas rendering pipeline and a workflow design in which all annotation operations are accessible within a single screen context without server round-trips.

These results should be interpreted in context. The study used evaluators with no prior annotation platform experience, which means the observed differences reflect initial learnability following minimal training—exactly the scenario of a user approaching the ML pipeline for the first time. As a benchmark, annotation times documented in large-scale crowd-sourced pipelines have been reported at approximately 42 s per bounding box [4], a figure consistent with the medians observed for CVAT (41.67 s) and Label Studio (41.50 s), and notably higher than the 26.50 s obtained by Annotix under equivalent conditions. For teams that have already invested in CVAT or Label Studio infrastructure, the efficiency differences observed here do not by themselves constitute a reason to migrate. Rather, the results demonstrate that a user adopting Annotix for the first time can achieve annotation throughput competitive with established tools, while completing the training and inference stages within the same environment without additional technical knowledge.

5.3. Heuristic Usability

The heuristic evaluation characterizes usability for first-time users, for whom the initial learning curve is a relevant dimension of practical adoption. CVAT’s elevated scores on H7 (Flexibility and efficiency of use, mean = 2.22), H1 (Visibility of system status, mean = 1.89), and H10 (Help and documentation, mean = 1.67) are consistent with a design oriented toward the expert annotator, which maximizes efficiency for familiar users at the cost of higher initial cognitive load and reduced support during early-stage learning. This pattern reflects the well-documented trade-off in interface design between learnability and efficiency of use: platforms that expose a large feature surface to maximize expert throughput tend to impose higher initial cognitive load on first-time users, as workflow accelerators such as keyboard shortcuts require prior knowledge to be discovered [58].

Label Studio’s elevated scores on H1 (Visibility of system status) and H5 (Error prevention), both at mean = 1.33, reflect uncertainty about platform state during multi-step operations such as project setup and class reassignment. Its template-driven project model offers flexibility in task configuration but constrains mid-session operations—a behavior that, while reasonable in structured labeling campaigns with defined schemas, is less suited to exploratory annotation in which data categories are defined during the task itself.

Annotix’s low severity scores across all heuristics are consistent with a design that favors ease of initial use over feature breadth: direct manipulation on a native canvas,

minimal navigation between screens, and annotation state that is immediately visible and reversible. The heuristic on which Annotix showed its highest severity (H5: Error prevention, mean = 0.67) points to a specific area for improvement in the training and export stages, which are the longest operations in the workflow and where progress feedback is most useful to the user.

The ICC pattern reinforces this interpretation: H7 (Flexibility and efficiency of use) showed the highest inter-rater agreement (ICC = 0.676), indicating that workflow efficiency was the dimension on which the nine evaluators most consistently agreed when discriminating between platforms. Although this value falls just below the 0.70 threshold pre-specified in Section 3.3.3, it is the closest among all heuristics; lower agreement on the remaining heuristics is methodologically expected for heuristic evaluation, which is designed for individual problem identification rather than consensus [19], and severity scores are therefore interpreted descriptively, with the ordering most reliable for H7.

The evaluation was conducted by nine evaluators, exceeding the 3–5 range recommended by Nielsen for heuristic evaluation [19], which provides a robust descriptive characterization of the three platforms. Generalizability remains bounded by the evaluator profile: all participants shared a Medical Technology background, and studies with users spanning the full range of the target audience—clinical researchers, biologists, students from non-computing disciplines—would further strengthen these findings. Inter-annotator variability in annotation quality—a distinct but complementary dimension to the usability evaluated here—represents an additional challenge documented in the medical image segmentation literature [18], and future studies should incorporate annotation quality metrics alongside heuristic severity scores to provide a more complete characterization of performance in clinical annotation workflows.

5.4. Applicability to Medical Imaging

The medical imaging domain represents the prototypical use case for Annotix's target user: a professional with deep clinical knowledge—capable of identifying the fovea, the optic disc, the malleus handle, or the umbo with precision—but facing two simultaneous barriers to building ML models. The first is regulatory: privacy requirements established by HIPAA (Health Insurance Portability and Accountability Act) and GDPR (General Data Protection Regulation) preclude the use of cloud-based annotation services for sensitive clinical data. A review of annotation and curation pipelines for medical images [59] documents that both regulatory frameworks impose informed consent, data encryption, access control, and defined retention periods as preconditions for any workflow involving identifiable patient images—requirements structurally incompatible with third-party cloud services. The second barrier is technical: even with the regulatory issue resolved through local deployment of CVAT or Label Studio, the path from annotated data to a trained and deployable model requires technical expertise that most clinical professionals do not possess. Aljabri et al. [17] identify data sovereignty as one of the central unresolved requirements in existing medical annotation tools, but do not address this second barrier—the training pipeline. Annotix resolves both simultaneously: data never leaves the local machine, and the path from raw image to exportable model requires no programming knowledge.

The use of real retinal and tympanic membrane images in the heuristic evaluation (Section 4.3) demonstrates that the platform is functional and usable for these tasks. The non-destructive CLAHE image adjustment (Figure 3) directly addresses the low-contrast conditions common in otoscopic and retinal fundus images, enabling more precise anatomical boundary delineation without modifying source files—a design detail that reflects an understanding of the actual working conditions of the target clinical user.

5.5. Limitations and Future Work

Several limitations of the current study should be explicitly acknowledged. (i) The annotation efficiency experiment used a sample of three evaluators ($n = 3$); a larger sample would strengthen the generalizability of the efficiency results. (ii) Across both experiments, evaluators shared a Medical Technology background, and studies spanning the full range of the target audience—clinical researchers, biologists, ecologists, students from non-computing disciplines—are needed to establish broader generalizability. (iii) The efficiency protocol used synthetic images that, while methodologically appropriate for comparative tool benchmarking (controlled visual content eliminates image-complexity confounds across tools), may not capture the complexity of annotation tasks on real-domain images with ambiguous or variable content. (iv) As discussed in Section 5.2, the deployment model (desktop vs. browser-mediated) is part of each tool’s architectural identity and is not a separable parameter; the relative contributions of interface design and deployment to the observed efficiency differences therefore cannot be cleanly partitioned within this design. (v) The efficiency metric measures annotation time and not annotation quality, error rate, or inter-annotator agreement on label content; quality-oriented metrics are methodologically distinct from, and complementary to, the tool-comparison metric used here, and we have framed this dimension explicitly as a future research direction in Section 5.3. We note, however, that for the controlled, well-defined targets used in our protocols (CMRD operations on standardized synthetic shapes, and high-contrast anatomical structures with well-defined borders), the annotation task is unambiguous by design and quality variation is not the primary differentiator between tools. (vi) The work also characterizes the integration of 19 training backends but does not benchmark their predictive performance on domain datasets; the underlying performance of each backend is documented in the respective original publications. While Section 4.4 reports an end-to-end demonstration with a single backend (YOLOv11) on a representative medical imaging dataset, systematic multi-backend benchmarking on domain datasets remains a target for future studies. Finally, to our knowledge, no prior empirical study has evaluated mask annotation performance between CVAT and Label Studio under controlled conditions equivalent to those used here; this absence represents a gap in the literature that future work should address.

At the platform level, current limitations include: (i) P2P collaboration requires direct peer connectivity—NAT traversal via iroh may fail in restrictive corporate networks; (ii) the browser automation approach for Colab training is sensitive to UI changes in the target platform; (iii) time-series annotation lacks waveform-specific assistance tools such as peak detection; and (iv) 3D annotation over point clouds and volumetric data is not yet supported. The long-term development direction extends this accessibility principle toward both ends of the pipeline: toward the input end, through integration of foundation models such as SAM 2 [60] and Grounding DINO [61] to reduce the volume of manual annotation required; and toward the output end, through model packaging and distribution mechanisms that allow the user to deliver a functional detector to a colleague or clinical system without additional configuration steps. Federated learning across P2P peers and a plugin architecture for community-contributed backends are also planned to extend the platform’s reach and adaptability.

6. Conclusions

We have presented Annotix, an open-source desktop platform that addresses the fragmentation of the ML data preparation pipeline by integrating annotation, training, inference-assisted labeling, and serverless collaboration in a single offline-first application. The platform’s design is motivated by a gap that existing tools—each excellent within its

respective paradigm—do not close: the need for a complete, privacy-preserving annotation and training environment that requires no cloud infrastructure, no server deployment, and no DevOps expertise.

Empirical evaluation on synthetic and medical imaging tasks indicates that Annotix performs competitively against CVAT and Label Studio—the de facto open-source standards—with annotation throughput measurably higher in first-use scenarios and heuristic usability severity consistently lower across the ten Nielsen–Molich heuristics. These results provide empirical evidence of the platform’s practical viability. An end-to-end case study on diabetic retinopathy lesion detection further illustrates the complete pipeline, from raw fundus images to a deployed browser-based model, executed entirely within the platform.

With seven image annotation primitives, 19 ML training backends, ONNX-based model-assisted labeling, serverless P2P collaboration, and support for video, time-series, and tabular data, Annotix provides a comprehensive toolset for ML workflows in regulated and resource-limited settings, including medical imaging, ecological monitoring, remote sensing, and industrial inspection. The platform is freely available at <https://github.com/Debaq/Annotix> (accessed on 13 March 2026) under the MIT license.

Supplementary Materials: The following supporting information can be downloaded at: <https://www.mdpi.com/article/10.3390/computers15080496/s1>, complete source code and dataset images used in this work; And Video S1, a demonstration showcasing the end-to-end Annotix workflow—from project creation through annotation, training, model export, and ONNX-assisted inference.

Author Contributions: Conceptualization, N.B.Q., V.U.H. and F.L.-M.; Methodology, N.B.Q., F.L.-M., V.U.H. and H.B.T.; Software, N.B.Q., V.U.H., M.A.F., A.M.L. and F.B.P.; Validation, N.B.Q., F.L.-M., V.U.H., H.B.T. and C.V.B.; Formal Analysis, N.B.Q., F.L.-M. and V.U.H.; Investigation, N.B.Q., F.L.-M., V.U.H., H.B.T., C.V.B., M.A.F., A.M.L. and F.B.P.; Writing—Original Draft Preparation, N.B.Q., V.U.H. and F.L.-M.; Writing—Review and Editing, N.B.Q., F.L.-M., V.U.H., H.B.T. and C.V.B.; Visualization, N.B.Q., F.L.-M. and V.U.H.; Supervision, N.B.Q., V.U.H. and F.L.-M.; Project Administration, N.B.Q., V.U.H. and F.L.-M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Annotix is freely available at <https://github.com/Debaq/Annotix> (accessed on 13 March 2026) under the MIT license. Supplementary Materials for the DIRD v1 case study—including dataset details, the trained model artifact, and an interactive demonstration of the deployed system—are maintained by the authors and publicly accessible at <https://nexam.org/dird/en.html#modelo> (accessed on 27 May 2026).

Acknowledgments: The authors thank Ercilia Águila for her administrative support throughout this project. During the preparation of this manuscript, the authors used Claude Sonnet 4.6 (Anthropic) for the following purposes: assistance in manuscript draft and revision, generation of the radar chart presented in Figure 6, generation of SVG flow diagrams presented in Figures 1, 4 and 7; and code debugging support. The authors have reviewed and edited all AI-assisted content and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AE	Annotator-Evaluator
API	Application Programming Interface
CDP	Chrome DevTools Protocol
CLAHE	Contrast Limited Adaptive Histogram Equalization
COCO	Common Objects in Context

CMRD	Create, Move, Resize, Delete
CUDA	Compute Unified Device Architecture
CVAT	Computer Vision Annotation Tool
FPS	Frames Per Second
GDPR	General Data Protection Regulation
GPU	Graphics Processing Unit
HIPAA	Health Insurance Portability and Accountability Act
ICC	Intraclass Correlation Coefficient
IPC	Inter-Process Communication
IQR	Interquartile Range
JSON	JavaScript Object Notation
ML	Machine Learning
NAT	Network Address Translation
ONNX	Open Neural Network Exchange
P2P	Peer-to-Peer
QUIC	Quick UDP Internet Connections (IETF RFC 9000)
RFC	Request for Comments
SAM	Segment Anything Model
SYN-BBOX	Synthetic Bounding Box stimulus dataset
SYN-MASK	Synthetic Mask stimulus dataset
UUID	Universally Unique Identifier
VOC	Visual Object Classes (Pascal VOC)

References

- Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [\[CrossRef\]](#)
- Shelhamer, E.; Long, J.; Darrell, T. Fully Convolutional Networks for Semantic Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 640–651. [\[CrossRef\]](#) [\[PubMed\]](#)
- He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2980–2988. [\[CrossRef\]](#)
- Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common Objects in Context. In *Computer Vision—ECCV 2014*; Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T., Eds.; Springer: Cham, Switzerland, 2014; pp. 740–755. [\[CrossRef\]](#)
- Xie, X.; Cheng, G.; Wang, J.; Yao, X.; Han, J. Oriented R-CNN for Object Detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 11–17 October 2021; pp. 3500–3509. [\[CrossRef\]](#)
- Ismail Fawaz, H.; Forestier, G.; Weber, J.; Idoumghar, L.; Muller, P.A. Deep Learning for Time Series Classification: A Review. *Data Min. Knowl. Discov.* **2019**, *33*, 917–963. [\[CrossRef\]](#)
- Chalapathy, R.; Chawla, S. Deep Learning for Anomaly Detection: A Survey. *arXiv* **2019**, arXiv:1901.03407. [\[CrossRef\]](#)
- Lim, B.; Zohren, S. Time-Series Forecasting with Deep Learning: A Survey. *Philos. Trans. R. Soc. A* **2021**, *379*, 20200209. [\[CrossRef\]](#)
- Roboflow. Available online: <https://roboflow.com> (accessed on 13 March 2026).
- V7 Labs. Available online: <https://www.v7labs.com> (accessed on 13 March 2026).
- Supervisely. Available online: <https://supervisely.com> (accessed on 13 March 2026).
- Sekachev, B.; Manovich, N.; Zhavoronkov, A. Computer Vision Annotation Tool (CVAT). Zenodo. 2019. Available online: <https://zenodo.org/records/3497106> (accessed on 27 May 2026).
- Tkachenko, M.; Malyuk, M.; Holmanyuk, A.; Liubimov, N. Label Studio: Data Labeling Software. 2020. Available online: <https://github.com/heartexlabs/label-studio> (accessed on 13 March 2026).
- Tzutalin. LabelImg. 2015. Available online: <https://github.com/tzutalin/labelImg> (accessed on 13 March 2026).
- Russell, B.C.; Torralba, A.; Murphy, K.P.; Freeman, W.T. LabelMe: A Database and Web-Based Tool for Image Annotation. *Int. J. Comput. Vis.* **2008**, *77*, 157–173. [\[CrossRef\]](#)
- Makesense.ai. Available online: <https://www.makesense.ai> (accessed on 13 March 2026).
- Aljabri, M.; AlAmir, M.; AlGhamdi, M.; Abdel-Mottaleb, M.; Collado-Mesa, F. Towards a Better Understanding of Annotation Tools for Medical Imaging: A Survey. *Multimed. Tools Appl.* **2022**, *81*, 25877–25911. [\[CrossRef\]](#)
- Yang, F.; Zamzmi, G.; Angara, S.; Rajaraman, S.; Aquilina, A.; Xue, Z.; Jaeger, S.; Papagiannakis, E.; Antani, S.K. Assessing Inter-Annotator Agreement for Medical Image Segmentation. *IEEE Access* **2023**, *11*, 21300–21312. [\[CrossRef\]](#) [\[PubMed\]](#)

19. Nielsen, J.; Molich, R. Heuristic Evaluation of User Interfaces. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '90), Seattle, WA, USA, 1–5 April 1990; pp. 249–256. [\[CrossRef\]](#)
20. Landis, J.R.; Koch, G.G. The Measurement of Observer Agreement for Categorical Data. *Biometrics* **1977**, *33*, 159–174. [\[CrossRef\]](#) [\[PubMed\]](#)
21. Pillai, T.S.; Chidambaram, V.; Alagappan, R.; Al-Kiswany, S.; Arpaci-Dusseau, A.C.; Arpaci-Dusseau, R.H. All File Systems Are Not Created Equal: On the Complexity of Crafting Crash-Consistent Applications. In Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI '14), Broomfield, CO, USA, 6–8 October 2014; pp. 433–448.
22. Lugaresi, C.; Tang, J.; Nash, H.; McClanahan, C.; Uboweja, E.; Hays, M.; Zhang, F.; Chang, C.L.; Yong, M.; Lee, J.; et al. MediaPipe: A Framework for Building Perception Pipelines. *arXiv* **2019**, arXiv:1906.08172. [\[CrossRef\]](#)
23. ONNX Runtime. Available online: <https://onnxruntime.ai> (accessed on 13 March 2026).
24. Jocher, G.; Chaurasia, A.; Qiu, J. Ultralytics YOLO (Version 8.0.0). 2023. Available online: <https://github.com/ultralytics/ultralytics> (accessed on 5 May 2026).
25. Zhao, Y.; Lv, W.; Xu, S.; Wei, J.; Wang, G.; Dang, Q.; Liu, Y.; Chen, J. DETRs Beat YOLOs on Real-Time Object Detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 17–21 June 2024; pp. 16965–16974.
26. Robinson, I.; Robicheaux, P.; Popov, M.; Ramanan, D.; Peri, N. RF-DETR: Neural Architecture Search for Real-Time Detection Transformers. *arXiv* **2025**, arXiv:2511.09554.
27. Chen, K.; Wang, J.; Pang, J.; Cao, Y.; Xiong, Y.; Li, X.; Sun, S.; Feng, W.; Liu, Z.; Xu, J.; et al. MMDetection: Open MMLab Detection Toolbox and Benchmark. *arXiv* **2019**, arXiv:1906.07155. [\[CrossRef\]](#)
28. Iakubovskii, P. Segmentation Models PyTorch. 2019. Available online: https://github.com/qubvel-org/segmentation_models.pytorch (accessed on 5 May 2026).
29. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015; Lecture Notes in Computer Science*; Springer: Cham, Switzerland, 2015; Volume 9351, pp. 234–241. [\[CrossRef\]](#)
30. Chen, L.-C.; Zhu, Y.; Papandreou, G.; Schroff, F.; Adam, H. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 801–818.
31. Xie, E.; Wang, W.; Yu, Z.; Anandkumar, A.; Alvarez, J.M.; Luo, P. SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 12077–12090.
32. Cheng, B.; Misra, I.; Schwing, A.G.; Kirillov, A.; Girdhar, R. Masked-Attention Mask Transformer for Universal Image Segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 19–24 June 2022; pp. 1290–1299.
33. MMSegmentation Contributors. MMSegmentation: OpenMMLab Semantic Segmentation Toolbox and Benchmark. 2020. Available online: <https://github.com/open-mmlab/mms Segmentation> (accessed on 13 March 2026).
34. Wu, Y.; Kirillov, A.; Massa, F.; Lo, W.-Y.; Girshick, R. Detectron2. 2019. Available online: <https://github.com/facebookresearch/detectron2> (accessed on 5 May 2026).
35. MMPose Contributors. OpenMMLab Pose Estimation Toolbox and Benchmark. 2020. Available online: <https://github.com/open-mmlab/mmpose> (accessed on 5 May 2026).
36. Sun, K.; Xiao, B.; Liu, D.; Wang, J. Deep High-Resolution Representation Learning for Human Pose Estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 5693–5703.
37. Jiang, T.; Lu, P.; Zhang, L.; Ma, N.; Han, R.; Lyu, C.; Li, Y.; Chen, K. RTMPose: Real-Time Multi-Person Pose Estimation Based on MMPose. *arXiv* **2023**, arXiv:2303.07399.
38. Zhou, Y.; Yang, X.; Zhang, G.; Wang, J.; Liu, Y.; Hou, L.; Jiang, X.; Liu, X.; Yan, J.; Lyu, C.; et al. MMRotate: A Rotated Object Detection Benchmark Using PyTorch. In Proceedings of the 30th ACM International Conference on Multimedia, Lisbon, Portugal, 10–14 October 2022; pp. 7331–7334.
39. Wightman, R. PyTorch Image Models (timm). 2019. Available online: <https://github.com/huggingface/pytorch-image-models> (accessed on 6 May 2026).
40. Wightman, R.; Touvron, H.; Jégou, H. ResNet Strikes Back: An Improved Training Procedure in timm. In Proceedings of the NeurIPS 2021 Workshop on ImageNet: Past, Present, and Future, Online, 13 December 2021. [\[CrossRef\]](#)
41. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An Image Is Worth 16 × 16 Words: Transformers for Image Recognition at Scale. In Proceedings of the International Conference on Learning Representations (ICLR), Online, 3–7 May 2021.
42. Bao, H.; Dong, L.; Piao, S.; Wei, F. BEiT: BERT Pre-Training of Image Transformers. In Proceedings of the International Conference on Learning Representations (ICLR), Online, 25–29 April 2022.

43. Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; Jégou, H. Training Data-Efficient Image Transformers and Distillation through Attention. In Proceedings of the 38th International Conference on Machine Learning (ICML), Online, 18–24 July 2021; Volume 139, pp. 10347–10357.
44. Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; Guo, B. Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Online, 11–17 October 2021; pp. 10012–10022.
45. Oguiza, I. tsai—A State-of-the-Art Deep Learning Library for Time Series and Sequential Data. 2023. Available online: <https://github.com/timeseriesAI/tsai> (accessed on 5 May 2026).
46. Beitner, J. PyTorch Forecasting. 2020. Available online: <https://github.com/sktime/pytorch-forecasting> (accessed on 5 May 2026).
47. Lim, B.; Arık, S.Ö.; Loeff, N.; Pfister, T. Temporal Fusion Transformers for Interpretable Multi-Horizon Time Series Forecasting. *Int. J. Forecast.* **2021**, *37*, 1748–1764. [[CrossRef](#)]
48. Zhao, Y.; Nasrullah, Z.; Li, Z. PyOD: A Python Toolbox for Scalable Outlier Detection. *J. Mach. Learn. Res.* **2019**, *20*, 1–7.
49. Tavenard, R.; Faouzi, J.; Vandewiele, G.; Divo, F.; Androz, G.; Holtz, C.; Payne, M.; Yurchak, R.; Rußwurm, M.; Kolar, K.; et al. Tslearn, A Machine Learning Toolkit for Time Series Data. *J. Mach. Learn. Res.* **2020**, *21*, 1–6.
50. Du, W. PyPOTS: A Python Toolkit for Machine Learning on Partially-Observed Time Series. *arXiv* **2023**, arXiv:2305.18811.
51. Law, S.M. STUMPY: A Powerful and Scalable Python Library for Time Series Data Mining. *J. Open Source Softw.* **2019**, *4*, 1504. [[CrossRef](#)]
52. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-Learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
53. n0 Computer. iroh: Efficient QUIC-based Data Transfer. Available online: <https://iroh.computer> (accessed on 13 March 2026).
54. Iyengar, J.; Thomson, M. *QUIC: A UDP-Based Multiplexed and Secure Transport*; RFC 9000; Internet Engineering Task Force: Fremont, CA, USA, 2021; pp. 1–151. [[CrossRef](#)]
55. Everingham, M.; Van Gool, L.; Williams, C.K.I.; Winn, J.; Zisserman, A. The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338. [[CrossRef](#)]
56. Liu, Z.; Heer, J. The Effects of Interactive Latency on Exploratory Visual Analysis. *IEEE Trans. Vis. Comput. Graph.* **2014**, *20*, 2122–2131. [[CrossRef](#)]
57. Horn, R.; Lahnaoui, A.; Reinoso, E.; Peng, S.; Isakov, V.; Islam, T.; Malavolta, I. Native vs Web Apps: Comparing the Energy Consumption and Performance of Android Apps and their Web Counterparts. In Proceedings of the 2023 IEEE/ACM 10th International Conference on Mobile Software Engineering and Systems (MOBILESoft), Melbourne, Australia, 14–15 May 2023; pp. 44–54. [[CrossRef](#)]
58. Nielsen, J. Enhancing the Explanatory Power of Usability Heuristics. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems CHI '94, Boston, MA, USA, 24–28 April 1994; pp. 152–158. [[CrossRef](#)]
59. Lutz de Araujo, A.; Wu, J.; Harvey, H.; Lungren, M.P.; Graham, M.; Leiner, T.; Willemink, M.J. Medical Imaging Data Calls for a Thoughtful and Collaborative Approach to Data Governance. *PLoS Digit. Health* **2025**, *4*, e0001046. [[CrossRef](#)] [[PubMed](#)]
60. Ravi, N.; Gabeur, V.; Hu, Y.T.; Hu, R.; Ryali, C.; Ma, T.; Khedr, H.; Rädle, R.; Rolland, C.; Gustafson, L.; et al. SAM 2: Segment Anything in Images and Videos. *arXiv* **2024**, arXiv:2408.00714. [[CrossRef](#)]
61. Liu, S.; Zeng, Z.; Ren, T.; Li, F.; Zhang, H.; Yang, J.; Jiang, Q.; Li, C.; Yang, J.; Su, H.; et al. Grounding DINO: Marrying DINO with Grounded Pre-training for Open-Set Object Detection. In *Computer Vision—ECCV 2024*; Leonardi, A., Ricci, E., Roth, S., Russakovsky, O., Sattler, T., Varol, G., Eds.; Springer: Cham, Switzerland, 2024; pp. 38–55. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.