

Image Annotation Tool

Pallavi Sable¹, Rutuja Bhore², Vipassana Shirsath³

¹ Dept. of Electronics & Telecommunication, PES's Modern College of Engineering, Pune, India

² Dept. of Electronics & Telecommunication, PES's Modern College of Engineering, Pune, India

³ Dept. of Electronics & Telecommunication, PES's Modern College of Engineering, Pune, India

Corresponding Author Email: pallavisable@gmail.com

Under the Guidance of Mrs. S. S. Bhosale, Dept. of Electronics & Telecommunication Engineering

Abstract — Preparing labelled image datasets for AI and computer-vision systems is one of those tasks that sounds straightforward until you actually try to do it at scale. This paper describes a web-based Image Annotation Tool. The platform lets users draw bounding-box annotations on images and export the resulting labels directly in YOLO or COCO format for use in model training. It is built on a three-layer stack — a React.js front end, a Django REST Framework back end, and a PostgreSQL database — and ships with JWT-based authentication, four-tier role-based access control, and the ability to ingest GeoTIFF files and automatically extract their embedded geographic metadata. A working set of 20–50 images was used during testing, and annotating a batch of ten images from start to export took roughly fifteen minutes. The tool is designed to operate on an intranet without requiring an internet connection, which gives organisations tighter control over sensitive image data while still providing a structured, team-friendly annotation workflow.

Keywords: Image Annotation, Computer Vision, Object Detection, Django REST Framework, React.js, GeoTIFF, YOLO, COCO

I. INTRODUCTION

Image data has quietly become one of the most valuable assets an organisation can hold — it feeds decision-making pipelines, drives automation, and underpins almost every modern computer-vision application. As AI and computer-vision adoption has grown, so has the appetite for large volumes of carefully labelled images, since these labels are what allow a model to actually learn to detect, segment, or recognise patterns.

The catch is that labelling images by hand is slow, tedious, and easy to get wrong, and many of the available tools were not built with strict data-handling rules in mind. Platforms such as Roboflow [1] work well but assume an always-on internet connection, while lighter alternatives like MakeSense.ai [2] and LabelImg [3] tend to fall short on either confidentiality controls or support for geo-referenced formats like GeoTIFF.

To close that gap, this paper introduces the Image Annotation Tool, developed under a sponsored collaboration with Vasundhara Geo Technologies Pvt. Ltd. The goal was straightforward: build something that works like a modern web app but can run entirely on an intranet, with real access controls and support for the GeoTIFF format that geospatial teams actually work with. The stack brings together a React.js front end, a Django REST Framework back end, and a PostgreSQL database.

The finished build was exercised on a pool of 20–50 images spanning several annotation scenarios, and a batch of 10 images could be fully annotated in about fifteen minutes — a clear improvement over doing the same work without a structured workflow. Export support for YOLO and COCO keeps the output compatible with the deep-learning frameworks most teams already use.

The rest of the paper unfolds as follows. Section II reviews related tools and research. Section III walks through the methodology and overall system architecture. Section IV covers implementation details. Section V discusses results, Section VI looks at applications and what could be added next, and Section VII closes with concluding remarks.

II. LITERATURE REVIEW

A. Market Survey

Among existing platforms, Roboflow [1] stands out for its cloud-hosted workflow, AI-assisted pre-labelling, version history, and broad export support — though its free tier is limited and it needs a live

internet connection, which rules it out for locked-down environments. MakeSense.ai [2] takes the opposite approach: everything runs in the browser with no server involved, so privacy is essentially built in, and it covers bounding boxes, polygons, and keypoints with the usual export options. What it doesn't offer is any kind of AI assistance or multi-user collaboration. Labellmg [3], meanwhile, is a long-standing Python/Qt desktop tool prized for being simple and dependable offline, but it is limited to bounding boxes and has neither a modern interface nor collaborative features.

B. Review of Related Research

Several research efforts inform the design choices made here. Sager et al. [4] found that pairing automated suggestions with human review — a hybrid approach — boosts labelling accuracy while cutting down annotator effort. Cheng et al. [5] reached a similar conclusion, noting that deep models fine-tuned on a small amount of human-corrected data stay accurate without needing constant manual oversight. Bello et al. [6] looked at free, open-source annotation tools in classroom settings and found them genuinely useful for students and smaller institutions without budgets for commercial software. Yang et al. [7] measured up to a 60% drop in labelling time when real-time AI suggestions were available, letting annotators focus on checking rather than drawing from scratch. Li et al. [8] compared browser-based tools and concluded that a modular, plug-in-friendly design makes a big difference for usability, especially in team settings. Zhang et al. [9] built a semi-automated pipeline using edge detection and region proposals to pre-fill labels, cutting manual work substantially. Finally, Rahman et al. [10] proposed encryption-based, locally-computed annotation architectures specifically for handling sensitive imagery — an idea closely aligned with the security goals of this project.

C. Research Gap

Looking across these tools and studies, no single solution manages to combine efficient annotation, team collaboration, and strong privacy controls together with geospatial awareness. Table 1 lines up the proposed tool against the three most comparable platforms on the features that matter most for this use case.

TABLE 1. FEATURE COMPARISON OF ANNOTATION TOOLS

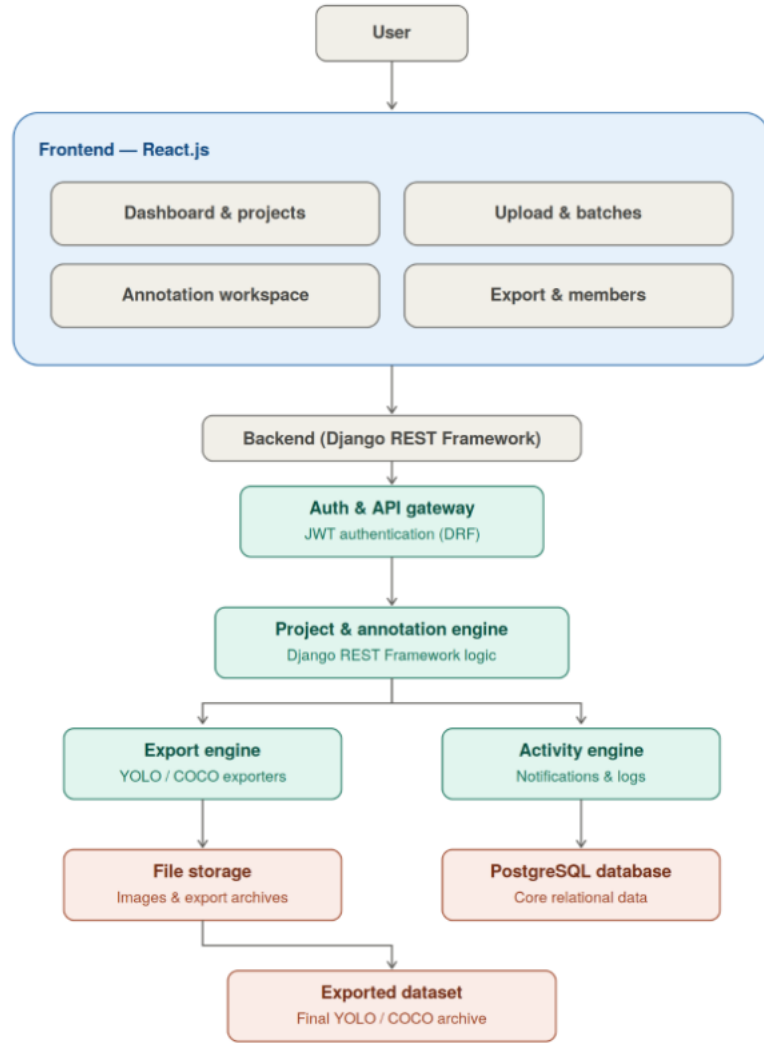
Feature	Our Tool	Roboflow	MakeSense	Labellmg
Offline / Intranet operable	Yes	No	Yes	Yes
GeoTIFF support	Yes	No	No	No
Role-based access control	Yes	Yes	No	No
YOLO / COCO export	Yes	Yes	Yes	Yes
JWT authentication	Yes	No	No	No
Activity audit log	Yes	No	No	No

The comparison in Table 1 makes the gap clear — only the proposed tool brings together GeoTIFF handling, JWT-secured access, the ability to work offline, and a full activity log. That mix is particularly relevant for enterprise, defence-adjacent, and geospatial teams, where both flexibility and accountability matter.

III. METHODOLOGY AND SYSTEM ARCHITECTURE

A. Overview

At a high level, the system is split into three independent layers that communicate over well-defined REST APIs: a React.js front end for the annotation interface, a Django REST Framework back end that handles authentication, business logic, and file processing, and a PostgreSQL database for all persistent storage. Keeping these layers separate means each one can be developed, tested, and updated independently without touching the others.



B. Frontend Design

The interface is built with React.js, JavaScript, HTML5, and CSS3 in a dark theme suited for extended annotation sessions. It covers a Home/Landing page; Signup and Login screens; a Dashboard showing all the user’s projects with key metadata; a Project Creation flow; an Upload area that accepts batches of images with drag-and-drop; the Annotation Workspace, an interactive canvas where bounding boxes can be drawn and edited; a Dataset Management view; and an Export screen for generating downloadable annotation archives.

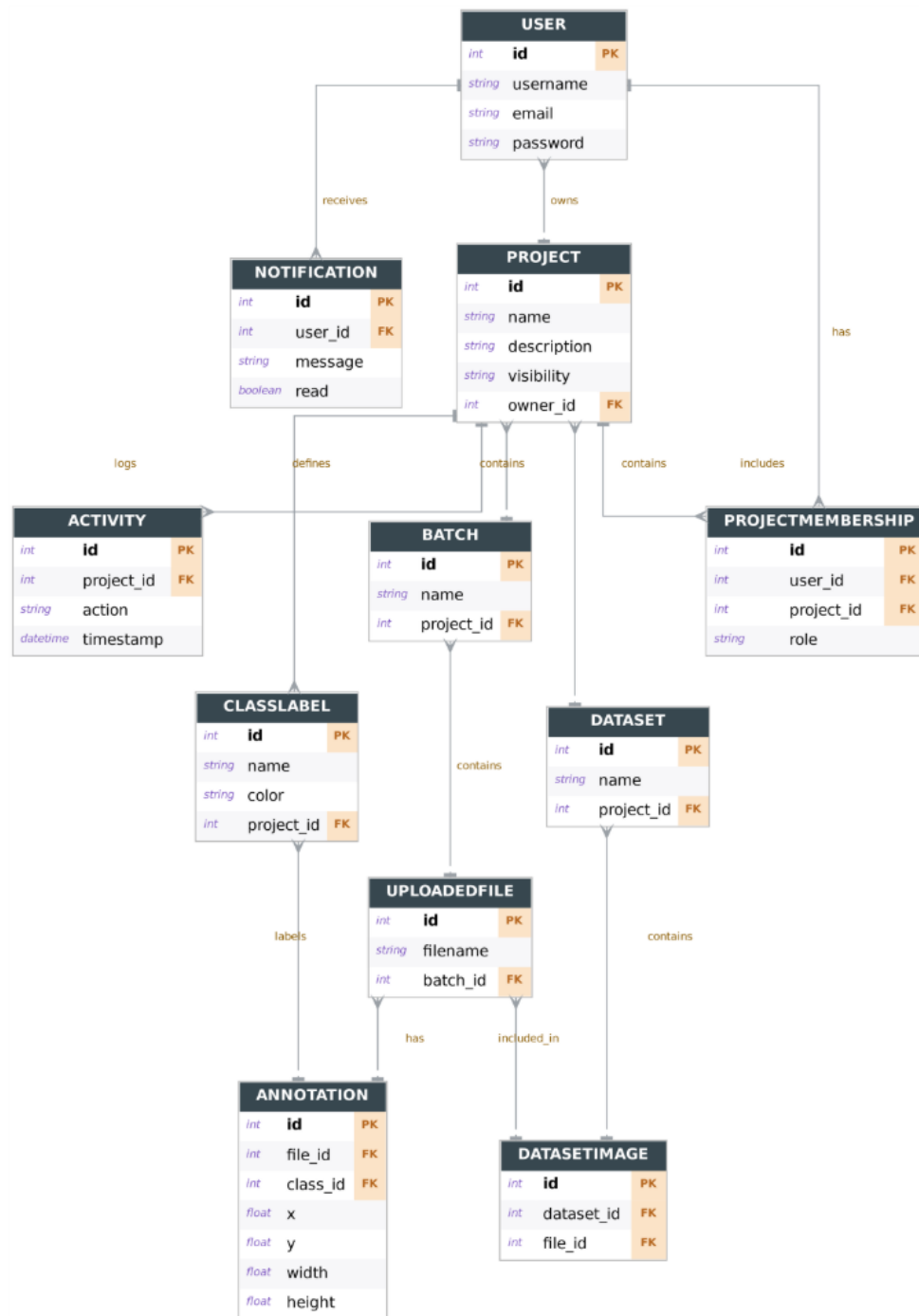
C. Backend Architecture

On the server side, Django and Django REST Framework expose a set of secured endpoints organised by functional module. Authentication is handled through JSON Web Tokens, with user passwords protected using Django’s built-in PBKDF2 hashing before storage. Access is governed by four roles — owner, admin, annotator, and viewer — enforced through role-based permission checks at the API level. When a GeoTIFF file is uploaded, Rasterio reads out its width, height, and geographic metadata automatically.

D. Database Design

PostgreSQL underlies the entire data layer. The schema is organised around the following models: User (authentication credentials and account details), Project (name, description, visibility, image count, and thumbnail), ProjectMembership (linking users to projects with an assigned role), Batch (grouping uploaded images within a project), UploadedFile (file path, annotation status, thumbnail, and GeoTIFF metadata fields including width, height, and CRS), ClassLabel (name and display colour per

project), Annotation (bounding-box coordinates stored as float fields against an image and class), Dataset (a named collection of annotated images), DatasetImage (the join between a dataset and individual uploaded files), Activity (a log of key project events), and Notification (member-addition and role-change alerts).



E. Data Flow and Annotation Workflow

A typical session begins with the React front end sending login credentials to the Django authentication endpoint; once a JWT is issued, the user lands on their dashboard. Uploaded images are checked for format validity, copied into media storage, and have their metadata extracted and written to PostgreSQL — the upload flowchart below walks through this validation-to-storage sequence step by step, including the CRS and bounding-box extraction performed for GeoTIFF files.

During annotation itself, the canvas records pixel coordinates for each bounding box drawn by the user. Each annotation is sent to the REST API and persisted to the database. When the user is ready to export, the export module reads the stored annotations, converts them to the requested format's coordinate schema, and packages everything into a downloadable ZIP archive.

IV. IMPLEMENTATION

A. User Authentication Module

Registration and login flow through a dedicated authentication module. Passwords are stored using Django's default PBKDF2 hashing, and a successful login returns both a short-lived JWT access token and a refresh token. The front end attaches the access token as a Bearer header on every subsequent API call, keeping sessions fully stateless on the server side. Role checks for each protected endpoint are handled by Django REST Framework's permission classes rather than scattered through view logic, which makes them easier to test and audit.

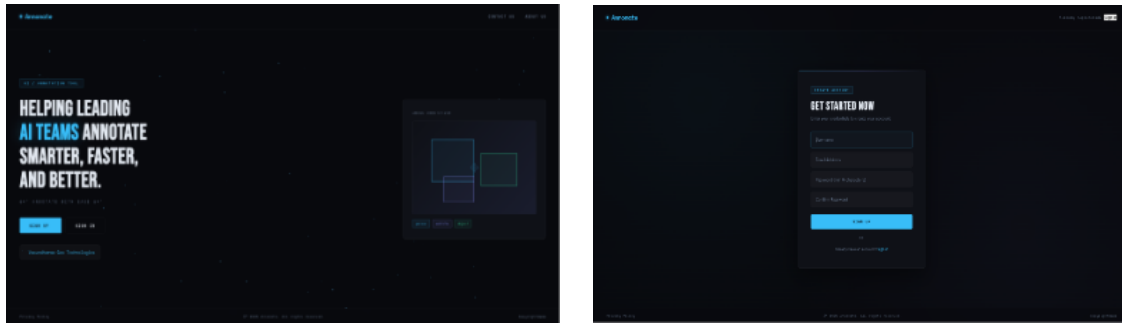


Fig 1: Landing Page and Sign Up Page

B. Image Upload and Metadata Extraction

Uploads accept multiple files in a single request through a drag-and-drop interface and are filtered server-side to accept only .tif and .tiff files. For each accepted GeoTIFF, Rasterio reads the image's pixel dimensions and Coordinate Reference System, which are stored against the file record in the database. A thumbnail is also generated at upload time so the annotation workspace can display a quick preview without reloading the full image.

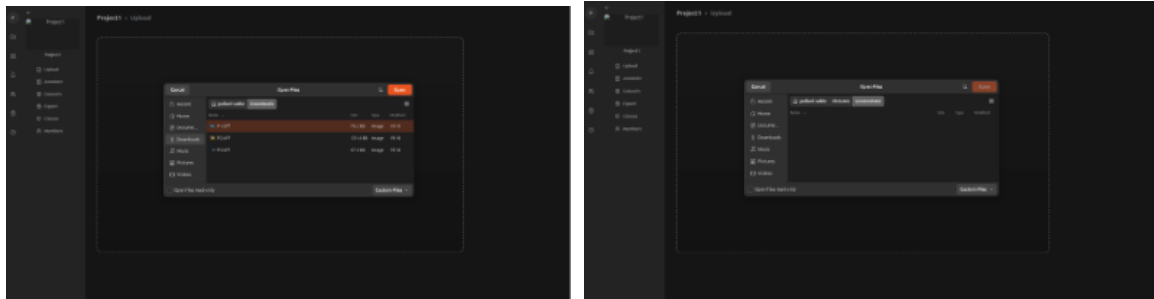


Fig 2 : GeoTIFF uploads only validation

C. Annotation Workspace

The annotation canvas is built on React-Konva and supports bounding-box annotation. Users click and drag to create a box around an object, assign it a class label from the project's defined label set, and can edit or delete it at any point. All bounding-box coordinates are stored as normalised float values (x, y, width, height) in the database. The workspace also separates images into annotated and unannotated tabs, so annotators always know what still needs attention.



Fig 3: Annotation Workspace with Bounding Box Labeling

D. Dataset Export

Two export formats are supported. For YOLO, each bounding-box annotation is converted into normalised centre-x, centre-y, width, and height values and written to a per-image .txt label file. For COCO, all annotations are assembled into a single annotations.json file following the standard COCO schema, with separate image metadata, category, and annotation sections. Both outputs are compressed into a ZIP archive for download.



Fig 4:YOLO and COCO Dataset Export

E. Software Specifications

The technologies used across the stack are summarised in Table 2.

TABLE 2. SOFTWARE SPECIFICATIONS

Component	Technology / Tool
Frontend	React.js, JavaScript, HTML5, CSS3
Backend	Django, Django REST Framework (Python)
Database	PostgreSQL
Image Libraries	Rasterio, Pillow
Authentication	JWT Token-Based (PBKDF2 hashing)
Export Formats	YOLO (.txt), COCO (.json)
Version Control	Git & GitHub
Dev Environment	Python 3.12+, Node.js 18+, VS Code, Cursor IDE

V. RESULTS AND DISCUSSION

At this stage, every core module — authentication, project management, image upload, the annotation workspace, dataset export, and activity logging — is built, working, and has passed functional and integration testing as a complete web application.

A. Annotation Performance

Testing used a set of 20–50 images of mixed resolutions, covering both ordinary JPEG/PNG files and georeferenced GeoTIFFs. A batch of 10 images, including assigning label classes and exporting the results, took roughly fifteen minutes from start to finish. By comparison, manual annotation without any integrated export step typically runs 3–5 minutes per image, so the gain here is substantial.

B. Annotation Modes

Bounding-box annotation is working as intended — boxes can be drawn, repositioned, resized, and deleted directly on the canvas with immediate visual feedback, and colour-coded label assignment updates in real time. The annotated/unannotated image split updates automatically as annotations are added or removed, giving annotators a clear picture of progress within each batch.

C. Export Functionality

Both export paths have been checked end to end. The YOLO output produces correctly normalised bounding-box values in per-image text files alongside a `classes.txt` manifest, and has been confirmed to load correctly in YOLOv8. The COCO output is a well-formed JSON file with image metadata, category definitions, and annotation entries, and has been validated against the COCO API's evaluation tooling.

D. Security and Access Control

JWT-based authentication with PBKDF2-hashed passwords is working as designed. Role-based permissions correctly restrict what each user can do — owners have full control over the project, admins can upload and manage members, annotators can create and edit annotations, and viewers are read-only. Attempts to reach protected endpoints without a valid token, or with a token belonging to a role that lacks the required permission, are rejected with HTTP 401 or 403 as appropriate. Every significant action is timestamped and written to the Activity table for audit purposes.

E. GeoTIFF Support

GeoTIFF handling is implemented and working. When a `.tif` file is uploaded, Rasterio reads the pixel dimensions and Coordinate Reference System and stores them against the file record. This metadata is available during annotation and is included in the export output, which is one capability that none of the comparable tools listed in Table 1 actually provide.

VI. APPLICATIONS AND FUTURE SCOPE

A. Applications

There's a reasonably wide range of domains where this tool fits naturally. For AI and ML teams, it produces YOLO- and COCO-ready datasets for object detection and classification work. In geospatial and remote-sensing contexts, the GeoTIFF ingestion and CRS metadata support make it straightforward to annotate satellite or drone imagery with geographic context preserved. In healthcare, bounding-box labelling of X-rays, CT scans, or MRIs could feed diagnostic model training. For autonomous vehicle and robotics work, the same approach applies to labelling pedestrians, vehicles, and road signage. It also has clear applications in academic settings and student computer-vision projects, and could extend to agriculture or environmental monitoring work involving annotated aerial imagery.

B. Future Scope

Looking ahead, a few directions seem worth pursuing. Plugging in a pre-trained detector such as YOLOv8 or Detectron2 could let the system suggest annotations automatically, with humans just reviewing and correcting. Adding cloud-based collaboration — multiple users on the same project at once, with version control and sync — would help larger teams. Support for additional export formats like Pascal VOC and GeoJSON is also planned, along with pixel-level segmentation and keypoint-

detection modes for more specialised tasks. A dashboard summarising annotation progress and dataset quality would help users keep track of larger projects, and an offline desktop build using Electron.js would extend the tool's reach to field environments with poor or no connectivity.

VII. CONCLUSION

This paper has described the design and build of a secure, web-based Image Annotation Tool developed in partnership with Vasundhara Geo Technologies Pvt. Ltd. The system covers the full annotation lifecycle — signing in, creating and managing projects, uploading GeoTIFF images, drawing bounding-box annotations, organising images into datasets, and exporting labelled data — on top of a three-layer architecture combining React.js, Django REST Framework, and PostgreSQL. By supporting intranet deployment, four-tier role-based access control, GeoTIFF ingestion, and standard export formats, it closes gaps that the surveyed tools each leave open in different ways.

Every module described here is built and has been tested end to end. The fifteen-minute turnaround for ten images demonstrates that the workflow is practically efficient, not just functionally complete, and both YOLO and COCO exports have been verified against the tools that would actually consume them. Table 1's comparison shows that the combination of features delivered here isn't matched by any single existing platform. Because the system is modular, adding the planned AI-assisted labelling and collaboration features later should be straightforward rather than requiring a redesign.

Acknowledgment

The authors are grateful to Mrs. S. S. Bhosale and Mrs. Mansi Kanitkar, our Project Coordinators, and to Prof. Dr. Mrs. R. S. Kamathe, Head of the Department of Electronics and Telecommunication Engineering at PES's Modern College of Engineering, Pune, for their guidance and support throughout the duration of this work.

References

- [1] J. Solawetz, J. Hodge, and R. Sharpe, "Roboflow: Annotate, Manage, and Deploy Computer Vision Models," 2024. [Online]. Available: <https://roboflow.com>
- [2] M. Skalski and K. Krawczyk, "MakeSense.ai: Free Browser-Based Annotation Tool for Computer Vision," 2023. [Online]. Available: <https://www.makesense.ai>
- [3] Tzutalin, "LabelImg: Open-Source Image Annotation Tool," GitHub Repository, 2015. [Online]. Available: <https://github.com/tzutalin/labelImg>
- [4] D. Sager, A. Kapoor, and R. Singh, "Hybrid Human-AI Systems for Efficient Dataset Labelling," *IEEE Access*, vol. 10, pp. 43521–43533, 2022.
- [5] H. Cheng, L. Wang, and Y. Liu, "Automatic Image Annotation Using Deep Learning and Transfer Models," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 17, no. 4, pp. 1–15, 2021.
- [6] P. Bello, S. Osei, and R. Kumar, "Open-Source Annotation Tools for AI Research: An Evaluation," *Journal of AI Systems*, vol. 5, no. 2, pp. 110–124, 2023.
- [7] J. Yang, Y. Zhao, and C. Wu, "AI-Assisted Annotation for Accelerated Dataset Generation," *arXiv preprint arXiv:2203.14567*, 2022.
- [8] Z. Li, H. Chen, and Q. Sun, "Design and Usability of Web-Based Image Labelling Tools," *Journal of Software Systems*, vol. 29, no. 4, pp. 210–225, 2020.
- [9] Y. Zhang, M. Tan, and S. Lee, "Semi-Automated Annotation Systems for Domain-Specific Datasets," *Elsevier Pattern Recognition*, vol. 122, p. 108267, 2021.
- [10] M. Rahman, T. Das, and F. Chowdhury, "Privacy-Preserving Architectures for Secure Data Annotation," *IEEE Trans. Inf. Security*, vol. 19, no. 2, pp. 56–69, 2023.