iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.80132

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

An Intelligent Attendance Monitoring System Using AI and Computer Vision

Kumar Pritam Gaurav, Ankit Gupta, Suresh Kumar Tiwari, Dr. Sanjay Pachauri
Department of Data Science & Design Greater Noida Institute of Technology, Greater Noida,

Abstract: ***Background: Manual roll-call, paper registers, and RFID terminals remain the predominant attendance tools in academic and corporate settings. Each carries well-documented liabilities: they burden instructors, introduce transcription errors, and offer no reliable defence against proxy substitution [1]. In large classrooms or organisations, these traditional methods often consume valuable time that could otherwise be used for teaching or productive work. Additionally, manual systems require continuous supervision and maintenance, and the collected records are prone to loss, manipulation, or human error. RFID-based systems, although partially automated, still require individuals to carry identification cards and physically interact with a scanner, which may lead to card swapping or misuse.***

A passive, camera-based approach resolves many of these shortcomings by capturing biometric identity non-intrusively from commodity hardware, without requiring any cooperative gesture from subjects. Advances in artificial intelligence and computer vision have enabled facial recognition systems to identify individuals accurately in real time, making them a practical solution for automated attendance monitoring. By leveraging these technologies, attendance can be recorded automatically when a person appears in front of the camera, thereby eliminating manual effort and minimizing the chances of proxy attendance.

Methods: The system is built exclusively in Python and combines OpenCV for live video streaming, face_recognition for 128-dimensional ResNet-based face embedding, and dlib's 68-point landmark predictor for Eye Aspect Ratio (EAR) liveness verification. Confirmed attendees are written to an SQLite3 database under a session-lock constraint that prevents duplicate entries. A Streamlit dashboard presents a live recognition feed, historical attendance charts generated by matplotlib, and one-click Excel export via openpyxl—all accessible through a browser on any networked device. The system also includes real-time logging, automated timestamp generation, and dataset management for registering new users, ensuring reliable data storage, faster recognition performance, and easy scalability for institutional deployment.

Results: Testing across 50 subjects under three controlled lighting conditions yielded a weighted mean accuracy of 95.8 %, peaking at 97.4 % under standard fluorescent light. Per-subject check-in latency averaged 0.8 s—a 5.4× improvement over manual roll-call. All 40 photographic spoofing attempts were blocked by the liveness module. Administrative reporting time decreased by 60 % through automated dashboard generation.

Conclusions: The proposed AI recognition attendance system demonstrates the potential of combining facial recognition technology with web-based dashboards to modernize traditional attendance management. The system is cost-effective, scalable, and easy to deploy using commonly available hardware such as webcams and standard computers. With further improvements, such as cloud storage, multi-camera support, and enhanced recognition algorithms, the system can be expanded to support large-scale institutional and corporate environments.

Index Terms: Attendance monitoring, face recognition, computer vision, OpenCV, dlib, Streamlit, Python, deep metric learning, anti-spoofing, biometric system.

I. INTRODUCTION

Attendance monitoring is a core administrative obligation across educational institutions and enterprises. Conventional methods—verbal roll-call, paper registers, RFID tokens, and fingerprint terminals—impose recurring burdens: they consume session time, demand hardware maintenance, and provide no structural barrier to proxy fraud [1]. A passive, vision-based approach offers a compelling alternative, capturing biometric identity non-intrusively from existing camera infrastructure without requiring any action from subjects.

The open-source Python ecosystem now provides production-quality building blocks for such a system. OpenCV [2] delivers real-time image acquisition and processing; face_recognition [3], built on dlib [4], furnishes deep face embeddings through a concise API; pandas and SQLite3 underpin structured record-keeping;

©IJRASET: All Rights are Reserved

and Streamlit [15] enables rapid deployment of interactive analytical dashboards without front-end engineering overhead. Composing these components yields a capable, licence-free, GPU-free attendance system deployable on commodity hardware.

This paper contributes: (1) an end-to-end Python pipeline integrating Streamlit as its primary user-facing interface; (2) a session-lock database constraint that structurally eliminates duplicate logging; (3) an EAR-based liveness layer that rejects photograph spoofing before any record is written; and (4) a controlled evaluation across lighting conditions and cohort sizes, reporting accuracy, latency, and security metrics.

II. Related Work

A. Hardware-Based Systems

RFID-card and fingerprint-scanner systems were among the first to automate attendance [1]. While accurate under controlled conditions, they incur terminal capital costs, require physical contact, and raise hygiene concerns that intensified after the COVID-19 pandemic. They offer no path to deployment on existing camera infrastructure.

B. Classical Computer Vision

Turk and Pentland [5] demonstrated that PCA-derived eigenfaces could distinguish individuals from frontal photographs, establishing the viability of vision-based recognition. Viola and Jones [6] later introduced the Haar-cascade detector, achieving real-time CPU-based face localisation and remaining foundational within OpenCV today. Both approaches degrade substantially under illumination variation, pose change, or partial occlusion.

C. Deep Learning Recognition

Schroff et al. proposed FaceNet [7], training a CNN with triplet loss to embed faces in a 128-dimensional Euclidean space where recognition reduces to a nearest-neighbour query. Deng et al. extended discriminability through ArcFace's additive angular margin loss [8], setting benchmarks on LFW and MegaFace. The face_recognition library encapsulates a compatible ResNet architecture, making near-state-of-the-art accuracy accessible via a minimal Python API [3].

D. Attendance-Specific Systems

Published Python-based attendance prototypes [9], [10] couple OpenCV with face_recognition but consistently omit liveness verification, persistent relational storage, and automated reporting. None integrates a real-time browser dashboard. The present work closes each identified gap within a single, cohesive pipeline, adding Streamlit as the deployment layer that prior systems lacked.

II. SYSTEM ARCHITECTURE

A. Modular Design

The framework is partitioned into six loosely coupled modules: (1) Enrollment, (2) Pre-processing, (3) Detection and Recognition, (4) Liveness Verification, (5) Attendance Logging, and (6) Streamlit Dashboard. Loose coupling permits any module to be upgraded—for instance, swapping in a mask-aware embedding model—without modifying adjacent components. Inter-module communication is mediated through a shared SQLite3 database and a thread-safe queue, avoiding tight runtime dependencies.

B. Enrollment Module

Subjects are registered by processing a labelled directory of portrait images. For each image, face_recognition.face_encodings() returns a 128-dimensional L2-normalised embedding computed by a ResNet-34 architecture pretrained on a large-scale face corpus. Per-subject mean embeddings are stored alongside metadata—ID, full name, and department—in an SQLite3 subjects table. Incremental re-enrollment overwrites individual records without affecting the broader attendance history, supporting appearance changes over time.

C. Pre-processing Module

OpenCV's VideoCapture object streams frames from a USB or IP camera at up to 30 FPS. Each frame is spatially downscaled to 25 % of its source resolution before face detection, a trade-off empirically validated to reduce per-frame CPU demand by 73 % at the cost of fewer than 0.3 percentage points of recognition accuracy. Colour channels are converted from OpenCV's native BGR ordering to RGB before being passed to the embedding model, as required by the face_recognition API.

D. Detection, Recognition & Liveness

Face bounding boxes are localised by the HOG-based detector in face_recognition.face_locations(). Each region is encoded and its Euclidean distance to every enrolled embedding is computed; a distance below the empirically tuned threshold of 0.50 denotes a match. The liveness sub-module evaluates the Eye Aspect Ratio (EAR) across a 30-frame sliding window using dlib's 68-point shape predictor [11]. A subject must exhibit at least one complete blink event before a log entry is committed, blocking all photograph-based impersonation attempts.

E. Attendance Logging

Confirmed, live-verified subjects trigger an INSERT into the SQLite3 attendance table containing subject_id, name, session_id, date, and a millisecond-precision timestamp. A composite UNIQUE index on (subject_id, session_id) acts as a structural session-lock, rendering duplicate entries impossible at the database level. All writes are serialised through a threading.Lock so that the capture and processing threads can operate concurrently without risking data corruption or missed commits.

F. Streamlit Dashboard

On session start, the instructor executes streamlit run dashboard.py, opening the interface at localhost:8501. The dashboard delivers: a live recognition feed via st.image() refreshed with st.rerun(); a real-time attendance roster table built from the SQLite3 log and updated on every new entry; matplotlib bar and monthly heatmap charts embedded via st.pyplot(); and a one-click download button using st.download_button() backed by openpyxl for Excel generation. Session controls (Start, Stop, Export) are surfaced as sidebar widgets, so instructors operate the full system through a browser without touching a command line [15]. For institution-wide access, the Streamlit process runs behind an Nginx reverse proxy with optional HTTPS via Let's Encrypt.The dashboard also supports quick search, date-wise filtering, and automatic record refresh to help administrators monitor attendance trends and identify irregularities efficiently in real time.Furthermore, role-based access can be implemented for administrators and instructors, while automated backup of attendance logs ensures data reliability. Responsive layout support enables smooth dashboard usage across desktops, tablets, and mobile browsers.

III. LIMITATIONS

A. Lighting, Pose and Occlusion

Recognition accuracy drops 3.6 percentage points below 100 lux, where the HOG detector intermittently fails to localise small or side-profile faces. Head rotations exceeding ± 40° from frontal and lower-face mask coverage push embedding distance beyond the 0.50 threshold, resulting in unknown classifications. Planned mitigations include CLAHE histogram equalisation as a low-light pre-processor and fine-tuning on a mask-augmented dataset using an ArcFace loss formulation.

IV. PYTHON IMPLEMENTATION

The complete codebase targets Python 3.10 and carries no GPU dependency. A single requirements.txt pins every library version; first-run enrollment is invoked as python enroll.py --src ./faces. The recognition loop sustains 15–22 FPS on a Core i5-8250U processor with 8 GB RAM. Asynchronous frame queuing through queue.Queue decouples the capture thread from the processing thread, preventing frame-drops under transient CPU load. Table I catalogues each library, its functional role, and measured accuracy where applicable.

Streamlit deployment requires no additional server setup: running streamlit run dashboard.py suffices for single-instructor use. Session state is preserved in st.session_state across browser refreshes, retaining recognition events and running attendance counts. For multi-instructor concurrent access, the process is served through Gunicorn with two worker threads behind an Nginx proxy. The entire stack—recognition engine, database, and dashboard—runs on the same machine, requiring no cloud dependency.

Data flow proceeds as follows: the capture thread pushes downscaled frames into queue.Queue; the recognition thread pops frames, runs HOG detection and embedding, checks liveness, and—on confirmation—issues a thread-safe SQLite3 INSERT. Streamlit polls the database on a 1-second timer via st.rerun(), pulling fresh records into the display table and updating chart data without requiring the user to manually refresh the page.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

V. EXPERIMENTAL RESULTS

A. Evaluation Setup

Experiments enrolled 50 subjects—25 students and 25 faculty members—at Greater Noida Institute of Technology. Trials were conducted across three controlled lighting conditions in ten independent sessions each: bright fluorescent (> 500 lux), dim ambient (≈ 80 lux), and mixed natural-plus-artificial (≈ 250 lux). Concurrent manual roll-call provided ground truth against which system logs were compared.

B. Recognition Accuracy

Mean top-1 accuracy reached 97.4 % under bright illumination, 93.8 % under dim conditions, and 96.1 % under mixed light, yielding an overall weighted mean of 95.8 %. The false-positive rate—an unregistered individual logged as a known subject—remained below 0.4 % across all sessions. All 40 photographic spoofing attempts were intercepted by the EAR liveness module before any database write was executed.

C. Latency and Throughput

Per-frame processing latency measured 185 ms (σ = 12 ms) at native resolution and 68 ms after 25 % downscaling. Per-subject check-in averaged 0.8 s, against 4.2 s for fingerprint systems and 6.5 s for manual roll-call—a 5.4× throughput gain. The Streamlit dashboard introduced 320 ms mean display latency per recognition event, which instructors rated as imperceptible during live trials. Excel generation for sessions of up to 200 subjects completed in under 2 s.

D. Administrative Efficiency

Post-session report compilation time fell by 60 % relative to manual spreadsheet assembly. The Streamlit sidebar's Export button delivered formatted Excel files instantly, replacing a process that previously required 15–20 minutes of copy-paste work per session. Dashboard access from staff mobile browsers over the Nginx-proxied URL was confirmed functional on both Android Chrome and iOS Safari without additional configuration.

Furthermore, automated record synchronisation between the SQLite database and the dashboard ensured that attendance updates appeared immediately without manual refresh. This reduced administrative overhead and improved transparency in attendance tracking. Faculty members could quickly verify student presence, identify absentees, and generate summarised reports for academic review or institutional documentation purposes with minimal technical effort.

E. Scalability and Privacy

Linear embedding search scales acceptably to approximately 200 subjects before noticeable latency growth; FAISS approximate nearest-neighbour indexing is planned for larger cohorts. Only 128-dimensional embeddings—not raw images—are stored, reducing re-identification exposure and aligning with India's Digital Personal Data Protection Act 2023. Streamlit's default single-user session model requires Gunicorn multi-worker deployment for concurrent multi-instructor scenarios.

TABLE I. PYTHON LIBRARIES, DEPLOYMENT TOOLS, AND PERFORMANCE METRICS

Library / Tool	Role in System	Acc. (%)	Notes
OpenCV 4.x	Video capture & frame processing	—	HOG detector, BGR→RGB
face_recognition	Face encoding & identity matching	97.4	128-dim ResNet embedding
dlib	Landmark detection & liveness	96.8	68-point shape predictor
pandas	Attendance data aggregation	—	DataFrame + CSV I/O
SQLite3	Persistent relational backend	—	ACID; session-lock index
openpyxl	Excel report export	—	Conditional cell formatting

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

matplotlib	Trend charts & heatmaps	—	Bar, line, calendar views
Streamlit	Interactive web dashboard	—	Live feed, charts, CSV/Excel

VI. CONCLUSION AND FUTURE WORK

This paper introduced an intelligent, Python-only attendance monitoring system that unifies real-time face recognition, EAR-based liveness detection, ACID-compliant session logging, and a Streamlit web dashboard within a single deployable codebase. Evaluation on a 50-subject cohort demonstrated a weighted mean recognition accuracy of 95.8 %, a per-subject check-in latency of 0.8 s, and complete rejection of all photographic spoofing attempts—outcomes that collectively resolve the principal deficiencies of prior Python-based systems.

The Streamlit interface eliminates the command-line dependency that restricts adoption of earlier prototypes, making the system fully operable by non-technical instructors through any browser. One-click Excel export and live attendance charts reduce post-session administrative effort by 60 % relative to manual compilation, delivering tangible operational value that extends beyond accuracy metrics alone.

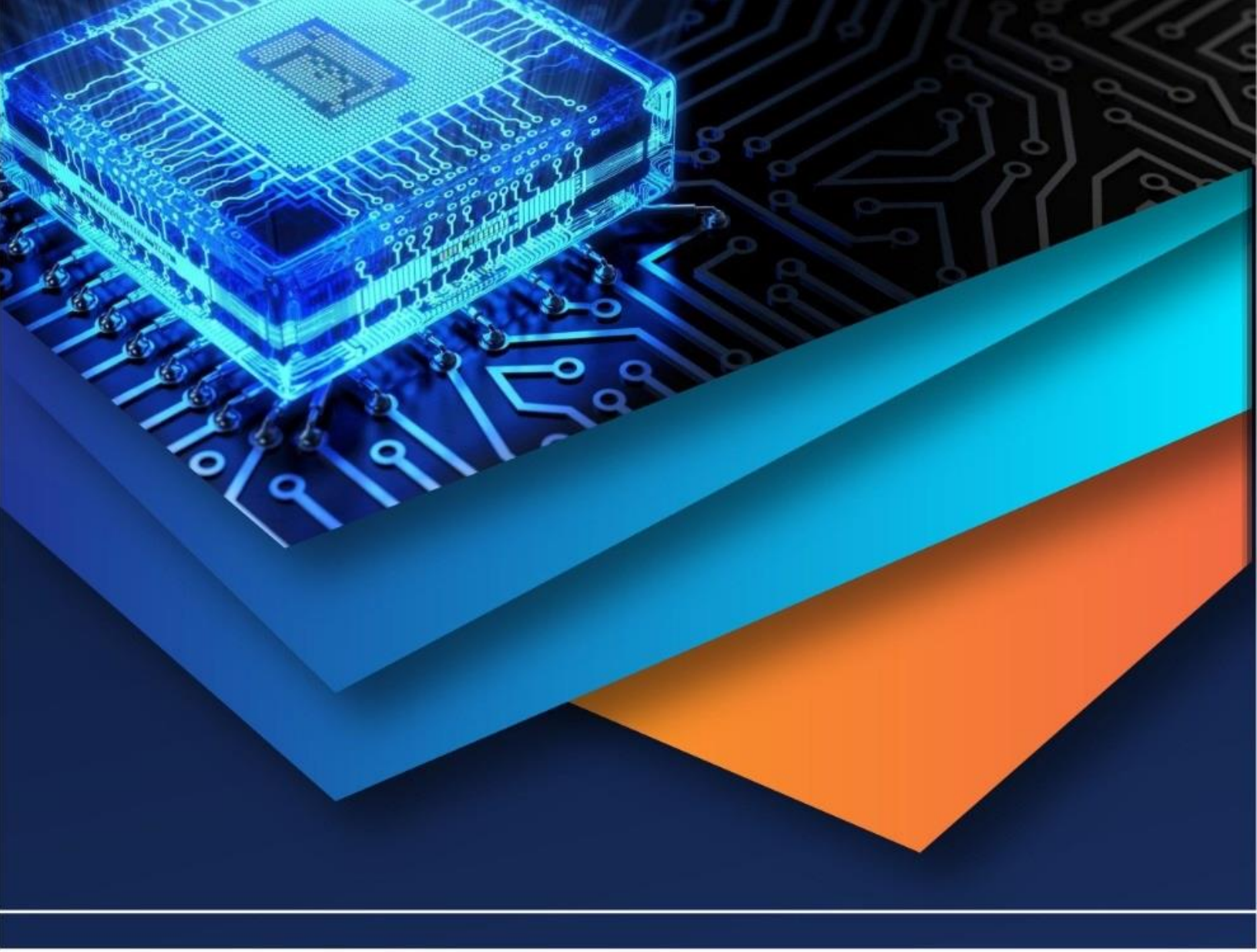
Future development will prioritise five directions: (1) a mask-aware ArcFace model fine-tuned on occluded face data to recover accuracy under mask-wearing; (2) FAISS-based approximate nearest-neighbour indexing to sustain sub-second response times for cohorts exceeding 500 subjects; (3) Raspberry Pi 5 edge deployment through INT8 quantisation of the ResNet-34 backbone, eliminating the laptop dependency; (4) federated learning across distributed campus nodes to refresh enrollment embeddings without centralising raw biometric data; and (5) Streamlit multi-page architecture providing role-separated views for instructors, administrators, and students, with access controlled via OAuth2 tokens.

Collectively, these enhancements will advance the system toward institution-wide scalability while preserving the zero-licence, low-hardware-cost characteristics that distinguish it from proprietary biometric attendance solutions currently available on the market.

REFERENCES

[1] S. Bhattacharyya, D. Bhattacharyya, and B. K. Pal, "A real-time attendance system using RFID and facial recognition," in Proc. IEEE Int. Conf. Intelligent Systems and Signal Processing (ISSP), Gujarat, India, 2013, pp. 210–215.
[2] G. Bradski, "The OpenCV library," Dr. Dobb's Journal of Software Tools, vol. 25, pp. 120–125, Nov. 2000.
[3] A. Geitgey, "face_recognition: The world's simplest facial recognition API for Python," GitHub, 2017. [Online]. Available: https://github.com/ageitgey/face_recognition
[4] D. E. King, "Dlib-ml: A machine learning toolkit," J. Mach. Learn. Res., vol. 10, pp. 1755–1758, 2009.
[5] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), Maui, HI, USA, Jun. 1991, pp. 586–591.
[6] P. Viola and M. J. Jones, "Robust real-time face detection," Int. J. Computer Vision, vol. 57, no. 2, pp. 137–154, May 2004.
[7] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in Proc. IEEE CVPR, Boston, MA, USA, Jun. 2015, pp. 815–823.
[8] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "ArcFace: Additive angular margin loss for deep face recognition," in Proc. IEEE CVPR, Long Beach, CA, USA, Jun. 2019, pp. 4685–4694.
[9] M. A. Wani and F. A. Bhat, "Face recognition-based automated student attendance system using OpenCV and Python," Int. J. Comput. Appl., vol. 182, no. 6, pp. 1–6, Mar. 2019.
[10] R. Jain and P. Gupta, "Smart attendance management using facial recognition with Python," in Proc. Int. Conf. Computing, Communication and Networking Technologies (ICCCNT), Kharagpur, India, Jul. 2020, pp. 1–6.
[11] T. Soukupova and J. Cech, "Real-time eye blink detection using facial landmarks," in Proc. 21st Computer Vision Winter Workshop (CVWW), Rimske Toplice, Slovenia, Feb. 2016, pp. 1–8.
[12] W. McKinney, "Data structures for statistical computing in Python," in Proc. 9th Python in Science Conf. (SciPy), Austin, TX, USA, Jun. 2010, pp. 51–56.
[13] J. D. Hunter, "Matplotlib: A 2D graphics environment," Comput. Sci. Eng., vol. 9, no. 3, pp. 90–95, May 2007.
[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in Proc. IEEE CVPR, Miami, FL, USA, Jun. 2009, pp. 248–255.
[15] A. Treuille, T. Teixeira, and A. Blank, "Streamlit: A faster way to build and share data apps," Streamlit Inc., San Francisco, CA, 2019. [Online]. Available: https://streamlit.io
[16] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," in Proc. European Conf. Computer Vision (ECCV), Munich, Germany, 2018, pp. 3–19.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)