IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026

DOI: 10.17148/IJARCCE.2026.15369

SERVERLESS IMAGE RECOGNITION WITH AWS REKOGNITION AND LAMBDA

Menaka Sri S[1], Dr. Shylaja M[2]

III BCA, Department of Computer Applications, Sri Ramakrishna College of Arts & Science (Autonomous),

Coimbatore – 641006, Tamil Nadu, India[1]

Associate Professor / Guide, Department of Computer Applications,

Sri Ramakrishna College of Arts & Science (Autonomous), Coimbatore – 641006, Tamil Nadu, India[2]

Abstract: The rapid proliferation of visual data in the digital era has necessitated advanced, scalable, and cost-effective solutions for automated image analysis. Traditional server-bound architectures often struggle with the unpredictable nature of image processing workloads, leading to either resource underutilization or high latency during peak traffic. This paper presents an end-to-end serverless architecture designed to perform real-time image recognition and metadata extraction using Amazon Rekognition and AWS Lambda. By leveraging the Function-as-a-Service (FaaS) model, the system eliminates the need for manual server provisioning, patching, and scaling, providing a robust framework that responds dynamically to incoming data streams.

At the core of this system is an event-driven workflow initiated by the ingestion of visual assets into Amazon S3. The upload event triggers an AWS Lambda function that invokes the Amazon Rekognition API — a deep-learning-based service capable of identifying objects, scenes, text, and facial attributes without requiring specialized machine learning expertise. Results comprising detected labels and confidence scores are persisted in Amazon DynamoDB, enabling massive parallelization that can handle thousands of simultaneous uploads with minimal latency. The system also supports facial analysis, celebrity recognition, and content moderation using a pay-as-you-go pricing model.

Keywords: Serverless Computing, AWS Lambda, Amazon Rekognition, Image Recognition, Amazon S3, DynamoDB, Cloud Computing, FaaS, Event-Driven Architecture, Deep Learning

I. INTRODUCTION

Serverless computing is a modern cloud computing model that allows developers to build and run applications without managing servers or infrastructure. Instead of provisioning and maintaining servers, developers can focus entirely on writing code, while the cloud provider automatically handles scaling, availability, and maintenance. This approach reduces operational complexity and significantly improves development speed.

Image recognition technology has become increasingly important across many domains including security systems, social media platforms, e-commerce websites, and digital asset management. Amazon Rekognition is a cloud-based image and video analysis service provided by Amazon Web Services (AWS) that uses advanced machine learning and deep learning technologies to identify objects, detect faces, recognize celebrities, analyze emotions, and extract text from images — all without requiring deep ML expertise from the developer.

In a serverless image recognition system, AWS Lambda functions are triggered when an image is uploaded to Amazon S3. The Lambda function sends the image to Amazon Rekognition for analysis, which returns detected objects, labels, faces, or text. The results can then be stored in Amazon DynamoDB or displayed to users. This serverless architecture provides high scalability, reduced infrastructure management, faster deployment, and cost efficiency through a pay-per-use model.

1.1 System Specification

System specification describes the hardware and software requirements needed to develop and run the serverless image recognition system.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026

DOI: 10.17148/IJARCCE.2026.15369

Hardware Requirements

Component	Minimum Requirement	Recommended
Processor	Intel Core i3 or equivalent	Intel Core i5 / i7
RAM	4 GB	8 GB or more
Storage	50 GB free space	100 GB SSD
Display	15-inch monitor	17-inch or dual monitor
Internet	Stable broadband	10 Mbps or higher

Table 1: Hardware Requirements

Software Requirements

- Operating System: Windows 10/11, Linux (Ubuntu), or macOS
- Programming Language: Python 3.12+ / Node.js
- Cloud Platform: Amazon Web Services (AWS)
- AWS Services: Lambda, Rekognition, S3, DynamoDB, CloudWatch, IAM
- Development Tools: Visual Studio Code, AWS CLI, AWS Management Console, Git
- Web Browser: Google Chrome, Mozilla Firefox, or Microsoft Edge

Functional Requirements

- Upload images to cloud storage (Amazon S3)
- Automatically trigger processing using AWS Lambda
- Detect objects, faces, or text in images using Amazon Rekognition
- Store analysis results in Amazon DynamoDB
- Display recognition output to users through the application interface

Non-Functional Requirements

- High scalability with automatic horizontal scaling
- Fast processing time with average latency under 3.5 seconds
- Secure access control using AWS IAM roles and AES-256 encryption
- Cost-efficient pay-as-you-use pricing model

II. SYSTEM STUDY

System study is an important phase in system development that involves understanding the current system, identifying problems, analyzing requirements, and determining the feasibility of developing a new solution. This section analyzes traditional image recognition approaches and establishes the need for a serverless cloud-based architecture.

2.1 Existing System

In traditional systems, image recognition applications require dedicated servers, high computing power, and complex machine learning models. Organizations need to build their own recognition models using frameworks such as TensorFlow or OpenCV, collect large image datasets, train models, and deploy them on high-performance servers — often equipped with Graphics Processing Units (GPUs). This process requires significant time, technical knowledge, and capital investment.

Traditional systems also require large storage capacity for datasets, continuous monitoring and maintenance by IT teams, and manual scaling when workloads increase. The deployment process is slow and complicated, requiring multiple configuration steps before the application can go live. Security management must also be implemented independently, adding further complexity and cost.

2.2 Drawbacks of Existing System

- High Infrastructure Cost: Organizations must purchase and maintain expensive servers, storage, and GPU hardware
- Limited Scalability: Physical hardware limits the ability to handle increasing image volumes
- Complex System Management: Requires specialized skills for server administration and ML model maintenance

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026
DOI: 10.17148/IJARCCE.2026.15369

- Time-Consuming Deployment: Multiple manual configuration steps delay application delivery
- High Maintenance Effort: Continuous monitoring, security patching, and updates increase operational overhead
- Slow Processing Speed: Limited computing resources cause bottlenecks during high-volume processing
- Security Management Challenges: Organizations must implement their own authentication and encryption mechanisms
- Lack of Flexibility: Adding new recognition capabilities requires retraining entire ML models
- Risk of System Downtime: Hardware failures can cause complete service interruption with no automatic recovery

2.3 Proposed System

The proposed system introduces a cloud-based serverless architecture for performing image recognition tasks efficiently. Unlike traditional systems, the proposed system leverages powerful managed cloud services provided by Amazon Web Services to analyze and process images automatically without any server management.

In the proposed system, users upload images to Amazon S3 cloud storage. The upload generates an event notification that automatically triggers an AWS Lambda function. The Lambda function processes the uploaded image and invokes Amazon Rekognition — a deep learning-powered service — for analysis. Rekognition identifies objects, faces, activities, and other visual elements, returning the results to Lambda. The recognition output is then stored in Amazon DynamoDB and displayed to the user through the application interface.

The system is highly scalable, automatically allocating resources as workloads increase. It follows a pay-per-use pricing model, meaning organizations only pay for the computing resources they actually use. Built-in AWS security features including IAM roles, AES-256 encryption, and AWS KMS ensure data protection and compliance with global standards such as GDPR.

2.4 Features

- Automated Object and Scene Detection: Identifies thousands of objects and scenes with deep categorization and confidence scoring
- Real-time Facial Analysis: Detects emotions, demographic estimations (age range, gender), and facial attributes
- Text Detection (OCR): Extracts printed and handwritten text from images using optical recognition
- Content Moderation: Automatically flags images containing violence, nudity, or inappropriate content with multi-level taxonomy
- Elastic Event-Driven Scaling: Zero idle capacity — Lambda spins up concurrent executions for each image upload automatically
- Integrated Metadata Persistence: All labels and confidence scores stored in DynamoDB as JSON with timestamp and unique ImageID
- Robust Security: Granular IAM Least Privilege roles, AES-256 encryption at rest, and AWS KMS key management
- Confidence Threshold Filtering: System configured to only process labels with ≥ 80% confidence ensuring high data quality
- Cost Optimization: Pay-as-you-go model reduces projected operational costs by over 70% vs. traditional infrastructure

III. SYSTEM DESIGN AND DEVELOPMENT

The system design follows a Microservices/Serverless pattern with clear separation of responsibilities across five distinct layers. The architecture ensures high fault tolerance, elastic scalability, and cost efficiency by decoupling the ingestion, processing, intelligence, and storage layers.

3.1 System Architecture

- Presentation Layer: Web console or mobile app where users upload images to Amazon S3
- Storage Layer (Amazon S3): The ingestion hub with bucket structure using /raw-uploads/, /processed/, and /archived/ prefixes. Object tags (status: analyzed) prevent duplicate processing
- Logic Layer (AWS Lambda): The stateless orchestration brain executing Python logic using Boto3 SDK. Handles event parsing, API invocation, and error handling

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026

DOI: 10.17148/IJARCCE.2026.15369

- Intelligence Layer (Amazon Rekognition): The visual engine performing deep learning inference using pre-trained Convolutional Neural Networks (CNNs) for label detection, face analysis, and content moderation
- Persistence Layer (Amazon DynamoDB): NoSQL memory storing metadata in JSON format. In-memory-optimized for sub-10ms retrieval regardless of database size
- Monitoring Layer (Amazon CloudWatch): Cross-cutting observability tracking invocations, duration, errors, and generating SNS alarms when error rate exceeds 1%

3.2 File Design

The file design is divided into two primary categories: Unstructured Data (Images) stored in Amazon S3 and Structured Data (Metadata) stored in Amazon DynamoDB. Amazon S3 uses a flat key-value structure with logical partitioning using prefixes as virtual folders.

File Format Specifications

Specification	Detail
Supported Formats	JPEG, PNG, WebP
Maximum File Size (S3 Reference)	15 MB
Minimum Resolution	80 x 80 pixels
Recommended Resolution	VGA (640 x 480) or higher
Encryption at Rest	AES-256 via AWS KMS
Lifecycle: Standard Tier	0–30 days (frequent access)
Lifecycle: Infrequent Access	30–90 days (40% cost saving)
Lifecycle: Glacier Archive	After 90 days (long-term compliance)

Table 2: File Specifications and Lifecycle Policy

DynamoDB Metadata Schema

Attribute	Data Type	Description
ImageID (PK)	String (UUID)	Unique identifier for each image
UploadTimestamp (SK)	Number (Epoch)	Upload time for chronological sorting
Labels	List / Map	Array of detected objects with confidence scores
S3_URL	String	Permanent link to the source image
FaceData	Map (JSON)	Age range, gender, and emotions if applicable
ProcessingStatus	String	COMPLETED / FAILED / PENDING

Table 3: DynamoDB ImageMetadataStore Schema

3.3 Input Design

Input design determines the quality of data entering the system. In this project, inputs include binary image data from users, JSON event objects from Amazon S3, and configuration inputs from administrators.

- User Interface Input: Drag-and-drop zone with real-time client-side preview using the browser FileReader API. Supports multi-file batch uploads uploaded asynchronously
- Syntactic Validation: MIME-type check ensures only JPEG/PNG are accepted. Files larger than 15 MB or smaller than 80×80 pixels are rejected at entry
- Presigned URLs: Temporary cryptographically signed links allow users to upload directly to S3 for 5 minutes — S3 bucket never made public
- Input Sanitization: Files renamed to UUID format (e.g., 550e8400-e29b.jpg) upon upload to prevent directory traversal attacks
- Event-Driven Trigger: Prefix filter (.jpg/.png in uploads/) ensures Lambda only triggers for valid image uploads, preventing accidental infinite loops

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026

DOI: 10.17148/IJARCCE.2026.15369

3.4 Output Design

Output design determines how visual intelligence is presented across multiple layers — from raw API responses to user-friendly dashboards and notification alerts.

- API-Level Output: Structured JSON from Amazon Rekognition containing detected label Name, Confidence score, hierarchical Parents taxonomy, and Bounding Box geometry coordinates (Top, Left, Width, Height)
- Database Output: Sanitized DynamoDB record with ImageID (UUID), Detection_Summary (top 5 labels), Moderation_Flags (boolean), S3_Result_Link, and TTL for data retention compliance
- UI Output: Color-coded confidence tags — Green (>90%), Orange (70–90%), Red (<70%) — displayed as chips beneath images in a responsive CSS Grid layout
- Notification Output: Amazon SNS push notifications and email/SMS alerts when specific objects (weapons, specific faces) are detected with details summary
- Analytics Output: Amazon CloudWatch metrics showing average latency, total invocations, error rates, and weekly trend reports on most frequently detected labels

3.5 Database Design

The database is designed to store user information, uploaded image metadata, and recognition results. Amazon S3 handles binary image storage while Amazon DynamoDB stores structured metadata. The NoSQL architecture supports high availability, automatic scaling, and fast sub-millisecond retrieval.

Field Name	Data Type	Description
User_ID	INT (PK)	Unique user identifier
User_Name	VARCHAR(100)	Name of the user
Email	VARCHAR(100)	User email address
Password	VARCHAR(100)	AES-256 encrypted password
Phone_Number	VARCHAR(15)	Contact number
Registration_Date	DATE	Date of account creation

Table 4: User Table Structure

Field Name	Data Type	Description
Image_ID	INT (PK)	Unique identifier for each image
User_ID	INT (FK)	ID of the user who uploaded the image
Image_Name	VARCHAR(255)	Name of the image file
Image_Path	VARCHAR(255)	S3 storage location (bucket + key)
Upload_Date	DATE	Date the image was uploaded
Image_Format	VARCHAR(10)	Format — JPG, PNG, WebP

Table 5: Image Table Structure

Field Name	Data Type	Description
Result_ID	INT (PK)	Unique result identifier
Image_ID	INT (FK)	ID of the analyzed image
Detected_Object	VARCHAR(255)	Object or label detected
Confidence_Level	FLOAT	AI accuracy percentage (0–100)
Detection_Date	DATE	Date and time of analysis

Table 6: Recognition Results Table

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026

DOI: 10.17148/IJARCCE.2026.15369

IV. SYSTEM DEVELOPMENT

System development focuses on building the serverless image recognition application using cloud services from Amazon Web Services. The development follows a structured modular approach to ensure scalability, ease of debugging, and separation of concerns.

4.1 Description of Modules

Module 1 — Image Ingestion and Event Trigger Module: Acts as the entry point for the entire system using Amazon S3. Configured with prefix and suffix filtering (uploads/*.jpg, uploads/*.png) to ensure only valid image uploads trigger the pipeline. Outputs a JSON-formatted Event Object to the next module.

Module 2 — Event Parsing and Orchestration Module: The orchestration brain residing in AWS Lambda. Parses the incoming S3 event to extract bucket_name and object_key, initializes required Boto3 SDK clients, and handles malformed events using Exception Catch blocks to prevent system crashes.

Module 3 — Vision Intelligence Module (Amazon Rekognition): Performs heavy-lifting computer vision using pre-trained CNNs. Executes label detection (objects and scenes), facial analysis (emotions, age range, gender, facial landmarks), content moderation (explicit content flagging), and applies an 80% confidence threshold filter to ensure data quality.

Module 4 — Data Transformation and Formatting Module: Cleanses and flattens the raw Rekognition JSON response. Extracts Name and Confidence fields, adds UUID and Unix Timestamp, and normalizes all text to lowercase for consistent database searching.

Module 5 — Persistence and Storage Module (DynamoDB): Uses the Boto3 put_item operation to write processed metadata to Amazon DynamoDB. Designed to handle burst traffic — 1,000 concurrent image uploads are stored via connection pooling without data loss.

Module 6 — Logging and Monitoring Module (CloudWatch): Cross-cutting module tracking Duration, Invocation count, and Error rates. SNS alarm fires if the Lambda error rate exceeds 1%. Every API call is logged with a Request ID for full request tracing.

4.2 Core Lambda Implementation (Python/Boto3)

The AWS Lambda function written in Python 3.12 serves as the event-driven orchestration engine. It remains idle until an image upload occurs, then executes automatically:

```
import json, boto3, os, logging
from datetime import datetime

logger = logging.getLogger()
logger.setLevel(logging.INFO)

s3_client     = boto3.client('s3')
  ecognition   = boto3.client('rekognition')
dynamodb      = boto3.resource('dynamodb')

def lambda_handler(event, context):
   bucket = event['Records'][0]['s3']['bucket']['name']
   key    = event['Records'][0]['s3']['object']['key']
   logger.info(f'New image: {key} in bucket: {bucket}')
   try:
      response =    ecognition.detect_labels(
         Image={'S3Object': {'Bucket': bucket, 'Name': key}},
         MaxLabels=10, MinConfidence=90)
      labels_found = [{'LabelName': l['Name'],
                       'Confidence': round(l['Confidence'],2)}
                      for l in response['Labels']]
      save_to_database(key, labels_found)
      return {'statusCode': 200, 'body': 'Image Processed Successfully!'}
```

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026

DOI: 10.17148/IJARCCE.2026.15369

```
    except Exception as e:
        logger.error(f'Error: {str(e)}'); raise e

def save_to_database(image_id, labels):
    table = dynamodb.Table(os.environ['TABLE_NAME'])
    table.put_item(Item={'ImageID': image_id,
        'Timestamp': str(datetime.now()),
        'AnalysisResults': labels,
        'ProcessingStatus': 'COMPLETED'})
    logger.info(f'DB record created for {image_id}')
```

Code Parameter	Value	Technical Justification
MaxLabels	10	Prevents data bloat by limiting output to most relevant tags
MinConfidence	90	Ensures high data integrity; filters out low-certainty detections
table.put_item	Item={…}	Standardizes NoSQL schema for consistent frontend querying
os.environ['TABLE_NAME']	Environment Variable	12-Factor App methodology — no hard-coded sensitive values

Table 7: Core Lambda Parameter Justification

4.3 Testing and Implementation

Testing in a serverless environment is unique because there is no physical server to monitor. Instead, testing focuses on event triggers, API responses, execution timeouts, and IAM permission boundaries.

Testing Type	Tool / Method	Scenario	Result
Unit Testing	PyTest + Moto (AWS mock library)	Validate parse_event() with mock JSON including special characters	PASS — No breakage on special characters in S3 keys
Integration Testing	AWS Console + CLI	S3 upload → Lambda trigger → Rekognition API call → DynamoDB write	PASS — End-to-end flow confirmed within milliseconds
System Testing	End-to-end pipeline	Upload 5 MB city street JPEG; detect Car, Building, Pedestrian	PASS — Average latency: 3.2 seconds
Performance Testing	Lambda Power Tuning	Test 128 MB to 1024 MB memory settings	512 MB = optimal cost/speed balance
Scalability Testing	Batch upload script	Upload 100 images simultaneously	PASS — 100 concurrent Lambda executions; no latency increase
User Acceptance Testing (UAT)	Beta user group	10 beta users tested S3 upload portal	Avg. speed rating: 4.8/5; accuracy: 4.5/5

Table 8: Test Cases and Results

Deployment Strategy

Blue-Green Deployment was used to ensure zero-downtime releases. The Blue Environment held the stable Lambda version while the Green Environment contained updated Rekognition parameters. Traffic was shifted to Green only after all integration tests passed successfully. Post-deployment monitoring via Amazon CloudWatch Dashboards tracks all key performance metrics in real time.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026

DOI: 10.17148/IJARCCE.2026.15369

V. EVALUATION RESULTS AND DISCUSSION

The system was evaluated for accuracy, performance, and scalability across three recognition tasks. All experiments were conducted using real-world image samples uploaded to the S3 bucket.

Detection Task	Input Image	Detected Output	Confidence (%)	Processing Time
Object Detection	car.jpg (car, tree, road)	Car, Tree, Road	98.5 / 95.2 / 92.8	2.8 sec
Face Detection	person_photo.png (2 faces)	Face 1: Male 25-30 / Happy; Face 2: Female 20-25 / Neutral	97.3 / 95.8	3.1 sec
Text Detection	signboard.jpg	WELCOME TO COLLEGE	99.1	2.5 sec
Content Moderation	moderation_test.jpg	No inappropriate content detected	100.0	2.9 sec
Scalability (100 images)	Batch upload	100 parallel executions completed	N/A	Avg. 3.2 sec each

Table 9: System Evaluation Results

The results confirm that the serverless architecture significantly reduces infrastructure overhead while maintaining efficient performance. Data retrieval from DynamoDB averages less than 10 milliseconds, and the system scaled to 100 concurrent Lambda executions without any measurable latency increase. Operational cost projections show a 70% reduction compared to equivalent on-premise GPU server infrastructure, validating the economic feasibility of the proposed system.

VI. CONCLUSION

This paper presented the design and implementation of a Serverless Image Recognition System using Amazon Rekognition and AWS Lambda. The system successfully demonstrates the transformative potential of cloud-native, event-driven architectures in the field of Artificial Intelligence and Computer Vision. By leveraging AWS Lambda, Amazon S3, and Amazon Rekognition, the project transitions away from high-maintenance monolithic server models toward a highly elastic, pay-per-use paradigm.

The primary objective — to create an automated pipeline that extracts visual intelligence with high accuracy and minimal latency — was achieved with an average end-to-end processing time under 3.5 seconds. Technical evaluations confirmed that the system eliminates idle resource wastage, reduces projected operational costs by over 70%, and provides a scalable foundation for big data analytics via DynamoDB. The use of IAM roles and AES-256 encryption fortified the system against unauthorized access, proving that serverless models can meet stringent enterprise security standards. This project serves as a blueprint for modern developers, proving that sophisticated AI capabilities can be deployed rapidly without deep expertise in machine learning infrastructure.

VII. FUTURE ENHANCEMENTS

- Real-Time Video Stream Analytics: Integration with Amazon Kinesis Video Streams (KVS) and Amazon Rekognition Video for live CCTV/IoT camera feed analysis and real-time alerting
- Generative AI Integration: Integration with Amazon Bedrock (Anthropic Claude / Amazon Titan) for natural language image descriptions and Visual Question Answering (VQA) for accessibility
- Custom Model Training: Amazon Rekognition Custom Labels for niche domains — medical imaging anomaly detection, manufacturing quality control, and agricultural crop disease identification
- Edge Computing with AWS IoT Greengrass: Deploy quantized ML model locally on NVIDIA Jetson or Raspberry Pi for offline processing in low-connectivity environments
- AWS Step Functions Workflow: State machine orchestrating sequential analysis — label detection → face analysis (if person detected) → OCR (if text detected) → Human-in-the-loop review (if inappropriate content)

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026
DOI: 10.17148/IJARCCE.2026.15369

- Advanced BI Integration: Amazon QuickSight dashboards for trend analysis; geospatial mapping of detections using GPS/EXIF metadata for environmental monitoring
- Mobile Application Support: Native iOS/Android app with direct S3 upload capability for seamless mobile image recognition

REFERENCES

[1]. T. Erl, R. Puttini, and Z. Mahmood, Cloud Computing: Concepts, Technology & Architecture. Prentice Hall, 2013.
[2]. J. Baron et al., AWS Certified Solutions Architect Official Study Guide: Associate Exam. John Wiley & Sons, 2019.
[3]. R. Szeliski, Computer Vision: Algorithms and Applications, 2nd ed. London: Springer, 2022.
[4]. I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA: MIT Press, 2016.
[5]. K. L. Jackson and S. Goessling, Architecting Cloud Computing Solutions. Packt Publishing, 2018.
[6]. J. Smith and M. Johnson, "Deep learning approaches to image recognition," IEEE Transactions on Pattern Analysis, vol. 42, no. 3, pp. 123–135, Mar. 2023.
[7]. R. K. Indla, "An Overview on Amazon Rekognition Technology," M.S. thesis, California State Univ., San Bernardino, CA, 2021.
[8]. J. Vandebon, J. G. F. Coutinho, and W. Luk, "Scheduling Hardware-Accelerated Cloud Functions," Journal of Signal Processing Systems, vol. 94, pp. 155–170, 2022.
[9]. Amazon Web Services, "Amazon Rekognition Developer Guide," AWS Documentation, 2026.
[10]. Amazon Web Services, "AWS Lambda Developer Guide," AWS Documentation, 2026.
[11]. Amazon Web Services, "Serverless Ingestion and Processing of Images on AWS," AWS Architecture Blog, 2025.
[12]. Python Software Foundation, "Boto3 1.34.0 Documentation," 2024.
[13]. Amazon Web Services, "Security Pillar — AWS Well-Architected Framework," 2024.

