Wildlife Detection using YOLOv8 for Analytics with Amazon Location Services

DATA/MSML 650 Interim Progress Report

Vivek Ediga, Aditya Sastry, Prakhar Tiwari, Yajat Uppal, Ateeq Ur Rehman

Friday, November 7, 2025

Problem Statement

Conserving all of Earth's biodiversity is essential to our survival. Wildlife supports healthy and resilient ecosystems, which in turn sustain human health. One of the most effective ways to support biodiversity conservation is by understanding where species live and how they are distributed. This knowledge allows us to identify key habitats for protection and support resource management.

Species distribution data often comes from occurrence records, which can be captured using edge devices such as trail cameras. However, these devices often have limitations with storage capacity and vulnerability to data loss through damage or tampering. Latency is another major concern, as noteworthy events could go unnoticed for days. These events might include rare wildlife behaviour or the presence of poachers and other anthropogenic threats to wildlife.

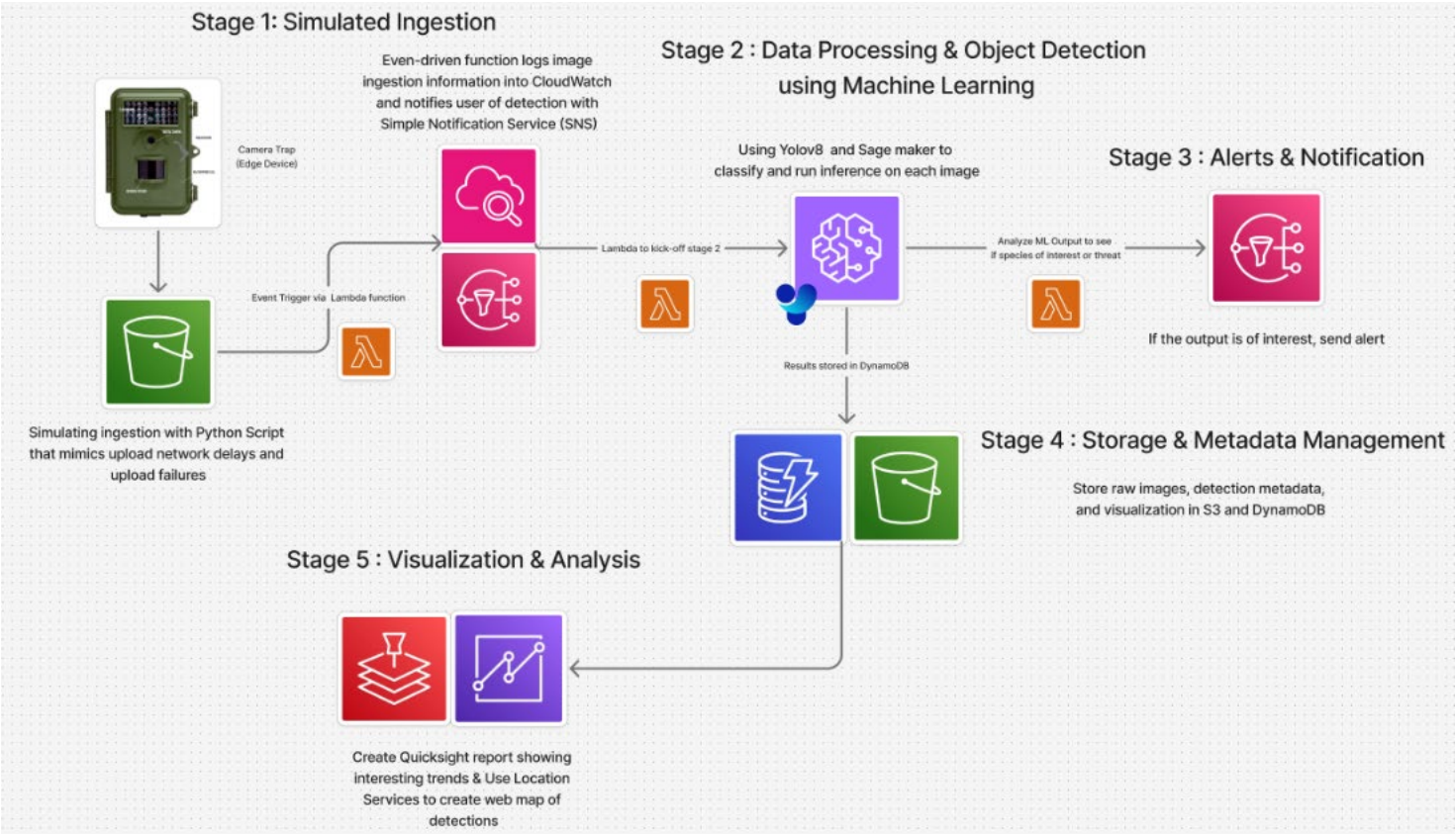
To ensure all data is reliably captured and available to support species distribution, conservation planning, and protection, a more robust and resilient data pipeline must be established. Therefore, we are proposing a cloud-based pipeline that facilitates the ingestion and offloading of data from edge devices, such as trail cameras, for wildlife or threat detection, analytics, and visualization.

Plan and Architecture

Our proposed architecture consists of three main stages:

- **Stage 1:** Simulated Ingestion - Mimicking camera trap data streaming to cloud storage
- **Stage 2:** Data Processing & Object Detection using YOLOv8 Machine Learning Model
- **Stage 3:** Alerts & Notifications System
- **Stage 4:** Storage & Metadata Management
- **Stage 5:** Visualization and Analysis

Architectural Diagram



Progress

GitHub Repository: GitHub Repository: https://github.com/adi-sastry/650_Project/tree/main

Completed Work

Infrastructure Setup and Utility Scripts

Before working on specific stages of our architecture, we developed a comprehensive set of utility scripts that are crucial for setting up the infrastructure for our pipeline. These scripts provide essential functionality for dataset management, resource provisioning, and cost control:

- **Dataset Download Script:** Downloads the Spatiotemporal Wildlife Dataset in the correct file structure via the Kaggle API.
- **Resource Provisioning Script:** Creates functions to provision AWS resources, including S3 buckets and IAM policies for camera trap data simulation. This script is designed to be extensible as we progress to later stages.
- **Resource Cleanup Script:** Deletes all provisioned resources to ensure we avoid unneeded charges while using the AWS free tier.

Stage 1: Simulated Ingestion (95% Complete)

The simulated ingestion component is nearly complete. We have successfully created a Python script that mimics an edge device, such as a camera trap, and uploads images of

Cheetahs (Acinonyx jubatus) from the validation set of the Spatiotemporal Wildlife Dataset to an S3 bucket.

The simulation script incorporates several realistic features to accurately represent real-world deployment scenarios:

- Simulates network delays to mimic unreliable connectivity in remote wildlife areas.
- Implements upload failure scenarios and automatic retry logic.
- Includes comprehensive metadata with each image upload: latitude, longitude, positional accuracy, temperature, elevation, and timestamp.

Stage 2: YOLOv8 Model Integration and Deployment (50% Complete)

Objective

The goal of Stage 2 is to integrate a YOLOv8-based object detection module into the existing cloud workflow. This stage focuses on building, training, and deploying an object detection model capable of recognizing key wildlife species in the dataset, while maintaining seamless integration with cloud resources (S3 for data storage and IAM for access control).

Implementation Steps Completed

- **Environment Setup:** Installed necessary dependencies, including ultralytics, boto3, and PyYAML. Configured environment using the requirements.txt file created in Stage 1, ensuring compatibility between local and cloud environments.
- **Data Pipeline Integration:** Defined dataset structure and YAML configuration file to specify train, validation, and test image directories. Integrated dataset upload and retrieval from AWS S3 using IAM-authenticated API calls. Ensured all datasets are stored in cloud buckets for reproducibility and team access.
- **Model Creation (In Progress):** Implemented a custom YOLOv8 model using the ultralytics library. Configured the model architecture for specific object categories relevant to the project, focusing on felid classification. Initial training runs were conducted locally using sample data to verify the correctness of the configuration and pipeline.

Outstanding Work

Stage 1 Completion (5% Remaining)

- Development of an event-driven Lambda function to log image ingestion information into CloudWatch and notify users of detections using Simple Notification Service (SNS)

Stage 2 Completion (50% Remaining)

- Complete model training on the full dataset with appropriate hyperparameter tuning
- Comprehensive evaluation and testing with bounding box visualization.
- Validation of model checkpoints and output storage on S3
- Integration with Stage 1 data ingestion pipeline

Stage 3: Alerts & Notifications System

- Design and implementation of the complete alerts and notifications architecture

Problems Encountered

Lambda Function Deployment Issues

We encountered significant challenges when implementing the event-driven Lambda functions to log image ingestion information. The primary issue was related to IAM role permissions and resource access control.

Problem Description:

When attempting to create Lambda functions using boto3, we discovered that the IAM role we created to deploy the Lambda functions was not assumable by the AWS Lambda service. This prevented us from successfully deploying our event-driven architecture.

Root Cause:

After thorough troubleshooting, we identified that our IAM user lacked the "iam:PassRole" permission. This permission is essential as it allows users and resources to assume and use IAM roles. Without this permission, we cannot grant Lambda functions the necessary role to access S3 buckets, CloudWatch, and other AWS services.

Resolution Plan:

We need to attach the "iam:PassRole" policy to our user account before creating the Lambda role. Due to time constraints, we were unable to fully complete the implementation of this policy attachment. Resolving this issue is our number one priority moving forward, as it represents the critical connection between Stage 1 and Stage 2 of our architecture.

Impact:

Once this permission issue is resolved, it will enable us to:

- Complete the event-driven architecture for Stage 1
- Deploy Lambda functions for automated image processing in Stage 2
- Implement the notification system for Stage 3
- Understand the proper patterns for deploying Lambda functions across our entire pipeline.

References

Hannah Ritchie, Fiona Spooner, and Max Roser (2022) - “Biodiversity” Published online at OurWorldinData.org. Retrieved from: 'https://ourworldindata.org/biodiversity' [Online Resource]

“Documenting Species and Ecosystems.” *NatureServe*, NatureServe, www.natureserve.org/documenting-species-ecosystems#:~:text=Knowledge%20of%20where%20species%20and,comparability%20of%20data%20among%20programs. Accessed 3 Oct. 2025.

Shaw, Julie. “Why Is Biodiversity Important?” *Conservation.Org*, Conservation International, 15 Oct. 2024, https://www.conservation.org/blog/why-is-biodiversity-important#:~:text=Species%20are%20often%20integral%20to,and%20nature%20is%20generally%20overlooked. Accessed 3 Oct. 2025.

Yu, Hui, et al. "Edge computing in wildlife behavior and ecology." *Trends in Ecology & Evolution* 39.2 (2024): 128-130.