**Document Revision History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Author(s)** | **Reviewer(s)** | **Description** |
| 1.0 | 13/05/2025 | 1.Jagadeesh Mouli Jyothula S V | ETA\_UI | Initial Draft |

**EPIC**

**Wildlife Detection and Classification Using Adaptive Filtering and YOLO Models** is an AI-powered computer vision project focused on developing a real-time wildlife monitoring system named **VANYAlytics**. This system leverages advanced image processing techniques like **Adaptive FAPG filtering** for denoising, and deep learning models such as **YOLOv8 and YOLOv11** for accurate detection and classification of wild animals in forest environments. It is designed to support forest officials, researchers, and conservationists by enabling intelligent biodiversity monitoring and mitigating human-wildlife conflict.

The core objective of this project is to automate the identification of wildlife in challenging environments characterized by noisy backgrounds, low lighting, and diverse species appearances. Wildlife image data is collected from trusted sources and processed using adaptive filters to improve clarity and quality. This filtered data is then annotated and augmented using **Roboflow**, enabling the creation of a high-quality training dataset. YOLO models are trained and evaluated on this data to ensure high accuracy in real-world conditions.

A key feature of the system is its real-time inference capability. End-users can upload an image and receive immediate detection results, complete with bounding boxes and class labels. The system’s modular AI pipeline ensures scalability, making it adaptable for additional species and broader deployment scenarios. The model training workflow is optimized for both accuracy and speed, ensuring practical deployment in surveillance operations.

In summary, this project builds a robust, scalable, and intelligent wildlife detection system for forest surveillance and conservation. The expected outcomes include structured and annotated wildlife datasets, high-performing YOLO models, and a real-time prediction interface. Together, these components form a complete pipeline that enhances wildlife protection efforts through automation, precision, and accessibility.

|  |
| --- |
| **OVERALL REQUIREMENTS** |

**PRODUCT BACKLOG**

**User Story - 1**

|  |  |
| --- | --- |
| **Description** | **Story Points** |
| As a data engineer, I want to collect wildlife images from reliable datasets and preprocess them using adaptive FAPG filtering, so that the images are clean, denoised, and ready for model training. |  |
| **Priority / MoSCoW parameters:**  **Must Have:**  1. Collect images from Snapshot Serengeti and Kaggle datasets.  2. Resize all images to 640×640 for YOLO compatibility.  3. Apply Adaptive FAPG filtering for noise removal and clarity enhancement.  **Should Have:**  1. Maintain logs of local noise estimation and filtration strength (β).  2. Calculate PSNR, MSE, and SSIM for filtered images to assess quality.  **Could Have:**  1. Visual comparison (before vs after filtering).  2. Automated β calculation based on image characteristics. **Won't Have:**  1. Image augmentation or model training at this stage. | |
| **Acceptance Criteria:**  **Dos / Positive points:**  1. Do resize all images consistently to 640×640.  2. Do ensure noise reduction using Adaptive FAPG with local variance analysis.  3. Do verify and log the filtering parameters and their impact on image quality.  **Don'ts / Negative points:**  1. Don’t perform data augmentation or model training now.  2. Don’t bypass filtering or leave out any images in resizing. | |

**User Story - 2**

|  |  |
| --- | --- |
| **Description** | **Story Points** |
| As a machine learning practitioner, I want to annotate and augment wildlife images using Roboflow, so that a clean and diverse dataset can be prepared for training object detection models. |  |
| **Priority / MoSCoW parameters:**  **Must Have:**  1. Upload filtered images to Roboflow.  2. Annotate each image with bounding boxes and class labels.  **Should Have:**  1. Use Roboflow’s data augmentation features (flipping, rotation, noise).  2. Export datasets in YOLO format.  **Could Have:**  1. Apply auto-labeling for large image sets and validate manually.  **Won't Have:**  1. Export to formats other than YOLO or use of unfiltered images. | |
| **Acceptance Criteria:**  **Dos / Positive points:**  1. Do annotate each image with correct animal class.  2. Do perform validation after auto-labeling.  3. Do apply and log augmentation techniques.  **Don'ts / Negative points:**  1. Don’t use raw unfiltered images.  2. Don’t skip annotation steps or export incorrectly formatted data. | |

**User Story - 3**

|  |  |
| --- | --- |
| **Description** | **Story Points** |
| As a data scientist, I want to train both YOLOv8 and YOLOv11 models using the annotated dataset, so that I can evaluate their detection performance on filtered wildlife images. |  |
| **Priority / MoSCoW parameters:**  **Must Have:**  1. Train YOLOv8 and YOLOv11 on the filtered dataset.  2. Evaluate performance using metrics like precision, recall, mAP@0.5 and mAP@0.5:0.95.  **Should Have:**  1. Compare detection performance on filtered vs. unfiltered inputs.  2. Save the best-performing model checkpoint.  **Could Have:**  1. Use hyperparameter tuning (batch size, epochs, learning rate) for optimization.  **Won’t Have:**  1. Deployment or real-time interface integration in this sprint. | |
| **Acceptance Criteria:**  **Dos / Positive points:**   1. Do validate model performance on test data and log results. 2. Do compare YOLOv8 vs YOLOv11 output with and without filtering.   **Don'ts / Negative points:**   1. Don’t overwrite model weights without versioning. 2. Don’t train on augmented data only—use original + augmented mix. | |

**User Story – 4**

|  |  |
| --- | --- |
| **Description** | **Story Points** |
| As a wildlife officer or end-user, I want to upload a new image and get real-time detection results with bounding boxes and class labels, so that animal presence can be identified quickly in field conditions. |  |
| **Priority / MoSCoW parameters:**  **Must Have:**  1. Accept user input (image) through a script or UI.  2. Run the trained YOLOv8/YOLOv11 model on input and return bounding boxes and labels.  **Should Have:**  1. Display detected classes with confidence scores.  2. Support switching between YOLOv8 and YOLOv11. teams.  **Could Have:**  1. Provide side-by-side output comparison of model variants. **Won't Have:**  1. Live video or drone feed integration in current scope. | |
| **Acceptance Criteria:**  **Dos / Positive points:**  1. Do display detection results with visual overlays.  2. Do ensure model loads correctly and gives results in <5 seconds.  **Don'ts / Negative points:**  1. Don’t allow images that are not preprocessed or incorrectly sized.  2. Don’t display incomplete or invalidated results. | |