Ahmedabad University

14/03/2025

CSE623: Machine Learning: Theory and Practice

Group: 5   Project no.:11

Weekly Report 3

Name	**Enrollment number**
Daksh Shah	AU2240207
Shalvi Modi	AU2240215
Malav Modi	AU2240214
Mahi Patel	AU2240210
Vinisha Kankariya	AU2220100

Summary:

Using Unmanned Aerial Vehicle (UAV) drone images the project works to detect wild mugger crocodiles (Crocodylus palustris). Research on mugger crocodiles requires individual identification because this species faces vulnerability which means population dynamics need monitoring along with behavioral pattern analysis. The current identification practices depend on invasive tagging methods that create stress in addition to disturbing natural environments of wild animals. The system provides solutions to identification challenges through the deployment of distinctive scute patterns for non-invasive identification processes. Various high-resolution imaging analysis methods now let researchers detect both specific animal subjects and separate different wildlife species effectively. Our system utilizes the YOLOv8 model which creates bounding boxes to establish exact location detection in addition to giving wildlife population monitoring both speed and scalability capabilities. Our system makes use of the model to identify wildlife effectively without dependency on human interaction and generates precise results for classification. This project design features flexibility which allows its use for multiple species dealing with similar conservation threats. The system brings substantial progress to ecological research by connecting automated identification capabilities with advanced image analysis systems.

Task completed:

- Implemented YOLOv11 model on the annotated dataset with bounding boxes applied to mark the crocodile dorsal scute patterns.

- Trained YOLOv11 on the dataset to detect and localize crocodiles efficiently.

Pseudo code:

```
Step 1: Preprocessing
Input: Image Dataset with Bounding Box Annotations
Output: Preprocessed Images with Bounding Boxes

Function Preprocess_Images(image_dataset):
    - Resize images to (640x640)
    - Normalize pixel values [0, 1]
    - Apply Data Augmentation (Flip, Rotation, Brightness Adjustment)
    - Split Dataset into Training, Validation, and Test sets
Return Preprocessed Images

Step 2: Model Initialization
Input: Preprocessed Dataset, Model Configuration
Output: Initialized YOLOv11 Model

Function Initialize_Model():
    - Load Pretrained Backbone Network (ConvNeXt or Swin Transformer)
    - Add Transformer-based Feature Extractor
    - Add Detection Head with Dynamic Convolution Layers
    - Apply Adaptive Anchor Box Mechanism
Return Model

Step 3: Training the Model
Input: Preprocessed Dataset, Model
Output: Trained Model with Optimized Weights

Function Train_Model(model, dataset, epochs):
    For epoch in range(1, epochs):
        For image, label in dataset:
            - Forward Pass
                Feature_Map = model.Backbone(image)
```

Predictions = model.Detection_Head(Feature_Map)
- Loss Calculation
  Localization_Loss = Smooth_L1_Loss(Predicted_BBox, Ground_Truth_BBox)
  Classification_Loss = CrossEntropyLoss(Predicted_Class, Ground_Truth_Class)
  Total_Loss = Localization_Loss + Classification_Loss
- Backward Propagation
  Update Weights using Adam Optimizer

Save Best Weights

Return Trained Model

Step 4: Inference

Input: Test Image, Trained Model

Output: Detected Objects with Bounding Boxes

Function Inference(model, test_image):

- Preprocess Test Image
- Forward Pass through the model
- Apply Non-Maximum Suppression (NMS)
- Draw Bounding Boxes with Confidence Score > Threshold

Return Detections

Step 5: Evaluation

Input: Model Predictions, Ground Truth

Output: Accuracy, Precision, Recall, F1-Score

Function Evaluate_Model(predictions, ground_truth):

- Compute Intersection over Union (IoU)
- Calculate Precision, Recall, F1-Score

Return Evaluation Metrics

Goals for Next week:

1. **Model Training** –Train the model and store image paths, extracted feature maps, and relevant metadata in a structured CSV file for efficient analysis.
2. **Identification of classes** - Use the trained model for identifying different classes.
3. **Similarity-** Finding the cosine similarity between the training dataset and testing dataset