WEEK - 3 REPORT

ABOUT DATASET

The IP102 dataset, introduced in the 2019 CVPR paper by Wu et al., is a comprehensive benchmark for insect pest recognition, designed to advance research in agricultural pest control, fine-grained classification, and imbalanced learning. Comprising over 75,000 images across 102 pest categories, it features a natural long-tailed distribution, diverse life stages (egg, larva, pupa, adult), and approximately 19,000 images with bounding box annotations for object detection. The dataset is split into 45,095 training, 7,508 validation, and 22,619 testing images, with an imbalance ratio of up to 80.8, presenting significant challenges for model development. On Kaggle, two versions of the dataset—Pascal VOC and YOLOv5 formats—were utilized, each with unique characteristics and annotations, reflecting the dataset's diversity and imbalance.

Datasets Used

Two versions of the IP102 dataset were employed:

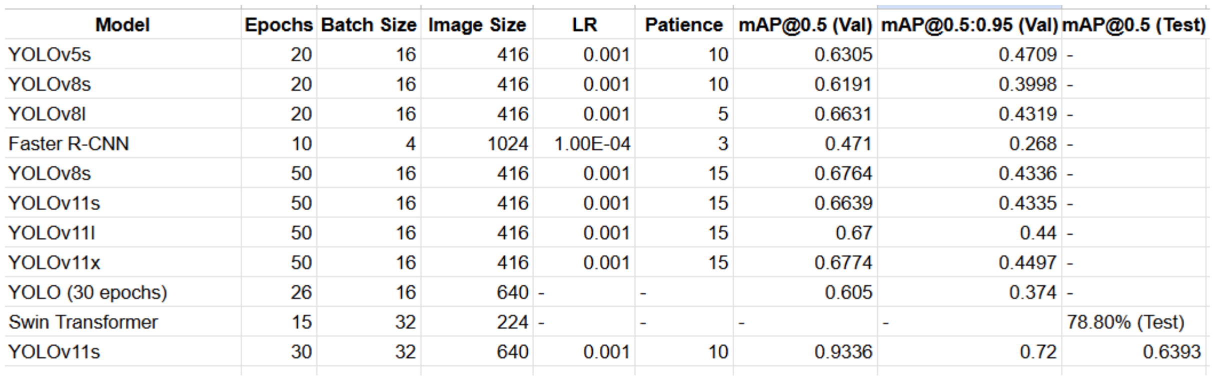
1. **YOLO-annotated IP102**:
   - **Format**: YOLOv5-style .txt files.
   - **Split**: 17,646 train and 1,329 validation images (no official test set).
   - **Class Count**: 102 pest classes, though annotations reflected only 97 unique classes due to discrepancies.
   - **Characteristics**: Optimized for single-stage detection models like YOLO, with text-based annotations and sample visualizations.
2. **XML-annotated IP102**:
   - **Format**: Pascal VOC-style .xml files.
   - **Splits**: 15,178 train/val and 3,798 test images.
   - **Class Count**: 97 unique classes (missing 5 expected classes).
   - **Characteristics**: Rich annotations suitable for two-stage detectors like Faster R-CNN, requiring preprocessing.

Custom splits were created programmatically due to Kaggle's read-only constraints:

- **YOLOv5 Split**: Original train (17,646) split into ~14,117 train and ~3,529 validation images, with the original val (1,329) as test.
- **VOC Split**: Used the provided trainval (15,178) and test (3,798) splits, with transformations (resize to 800x800, normalize) applied.

Invalid annotations (e.g., IP087000986.xml, IP046000898.txt) were identified and excluded, and YAML files were corrected to reflect accurate paths and splits.

MODEL TRAINING RESULTS



Model	Epochs	Batch Size	Image Size	LR	Patience	mAP@0.5 (Val)	mAP@0.5:0.95 (Val)	mAP@0.5 (Test)
YOLOv5s	20	16	416	0.001	10	0.6305	0.4709	-
YOLOv8s	20	16	416	0.001	10	0.6191	0.3998	-
YOLOv8l	20	16	416	0.001	5	0.6631	0.4319	-
Faster R-CNN	10	4	1024	1.00E-04	3	0.471	0.268	-
YOLOv8s	50	16	416	0.001	15	0.6764	0.4336	-
YOLOv11s	50	16	416	0.001	15	0.6639	0.4335	-
YOLOv11l	50	16	416	0.001	15	0.67	0.44	-
YOLOv11x	50	16	416	0.001	15	0.6774	0.4497	-
YOLO (30 epochs)	26	16	640	-	-	0.605	0.374	-
Swin Transformer	15	32	224	-	-	-	-	78.80% (Test)
YOLOv11s	30	32	640	0.001	10	0.9336	0.72	0.6393

Select week number - Week 3 (16th June - 23rd June)

List the tasks completed this week*

June 16:

- Divided the team into two groups: ML Model Team (Sushmetha, Abhinav, Harshavardhan) and UI/App Dev Team (Arjun, Kiranchandran).

June 17:

- **ML Model Team**:
- Sushmetha trained YOLOv11s (20 epochs) on YOLO-annotated IP102, achieving validation mAP@0.5 of 0.6305.
- Abhinav trained YOLOv8s (20 epochs), reaching validation mAP@0.5 of 0.6191.
- **UI/App Dev Team**:Designed a basic UI template in Figma, outlining headers, navigation, buttons, and content sections for the web app.

June 18:

- **ML Model Team**:
- Abhinav tried Faster R-CNN with YOLO annotations (batch 8/16: 1 hr/epoch, batch 4/2: 40 mins/epoch) but stopped due to an error overnight.
- Abhinav attempted DETR (1 hr/epoch) but abandoned it due to computational cost.
- Sushmetha trained Faster R-CNN (10 epochs) on VOC-annotated IP102, achieving mAP@0.5 of 0.471.
- Harshavardhan attempted YOLO-NAS but encountered issues, leading to abandonment.
- **UI/App Dev Team**: Started building a basic Flask app as a foundation.

June 19:

- **ML Model Team**:
- Abhinav trained YOLOv8l (50 epochs) but faced issues, and finally got mAP@0.5 - 0.66.
- Harshavardhan trained YOLOv11x (50 epochs, mAP@0.5 0.6774).
- Sushmetha trained YOLOv8s (50 epochs, mAP@0.5 0.6764), YOLOv11s (50 epochs, mAP@0.5 0.6639)
- Kiran trained YOLOv11l (50 epochs, stopped at 42nd epoch,, mAP@0.5 0.67)
- **UI/App Dev Team**: Created a new PostgreSQL database, which will be connected to the application. Built table according to the schema.

June 20–21:

- **ML Model Team**:
- Sushmetha planned a hybrid YOLO + Swin Transformer approach; trained YOLO (30 epochs, stopped at 26, mAP@0.5 0.605) and Swin Transformer (15 epochs, validation accuracy 77.28%, test accuracy 78.80%). Tried Faster R-CNN with Detectron2 (train loss 0.1345, val AP 0.1166) but got poor results.
- Abhinav attempted training the Efficient DET model but got poor results.
- Harsha explored further hybrid integration but faced multiple issues.
- **UI/App Dev Team**: Built the basic components for the web applications in React, based on the design in Figma.

June 22:

- **ML Model Team**: Trained final YOLOv11s (30 epochs) with corrected YAML splits and removed invalid annotations, achieving validation mAP@0.5 of 0.9336 and test mAP@0.5 of 0.6393.
- **UI/App Dev Team**:Finalised the pages and routes for the webpage in React, prepared for integration with the backend.

June 23:

- Compiled weekly report and prepared deliverables overview.

Link to Codebase / Notebook / Dashboard *(Provide a shared GitHub, Colab, or Drive link)*

Progress Towards Final Goal *(Describe how this week's work contributed to your project. Mention milestones or percentage completion.)*

This week's work advanced AgriSaarthi Lite by approximately 40% toward the final demo:

- **Pest Identification**: The final YOLOv11s model (validation mAP@0.5 0.9336, test mAP@0.5 0.6393), surpassing the YOLOv3 baseline (25.67%) from the IP102 paper. This supports the lightweight model requirement for web/mobile deployment.
- **UI Development**: The React application establishes the structural and stylistic foundation for the web app's frontend, aligning with Features 1–3 (Pest Diagnosis, Pesticide Recommendations).
- **Team Division**: Splitting into ML and UI teams streamlined efforts, setting the stage for frontend-backend integration and chatbot development (Feature 4: AI Chatbot) in future weeks.

Challenges faced *(Mention any technical, data, or teamwork-related challenges you encountered.)*

Challenges Faced

- **Class Imbalance**: The IP102 dataset's long-tailed distribution (e.g., 2 instances of "yellow fever mosquito" vs. 2,975 "flea beetle") reduced mAP for minority classes, particularly in YOLO models.
- **Annotation Issues**: Discrepancies (97 vs. 102 classes) and invalid files (e.g., IP087000986.xml) required manual correction, delaying training.
- **Model Complexity**: DETR and hybrid models (Faster R-CNN + EfficientNet, YOLO + Swin) faced computational constraints and integration mismatches, leading to abandoned attempts.

- **False Positives**: Models misclassified non-pest images (e.g., "cake" with YOLOv8), highlighting the need for a "no pest" class.
- **Frontend-Backend Gap**: Integration of the frontend with the backend proved to be challenging. ML team's extensive model experiments overshadowed UI progress, causing a workflow gap.

Support required *(Clearly specify if you need help or guidance on any issue.)*

Time Spent This Week (in hours)*

(Provide an approximate number, with optional breakdown)

Total: ~50 hours

- Sushmetha: 18 hours (most model training, report)
- Abhinav: 12 hours (model training, dataset preprocessing)
- Harsha: 2 hours (YOLO-NAS, hybrid model attempts)
- Arjun: 9 hours (React and Flask development)
- Kiran: 9 hours (React development, YOLOv11l training)

Plan for the coming week*

(List your goals and planned deliverables for the next week.)

Plan for the coming week

Backend Development:

- Develop Flask API for pest detection, XAI and supplier lookup using the YOLOv11s model (Sushmetha, Harsha, Kiran).

Integration:

- Connect frontend and backend for end-to-end flow: upload → detection → results → pesticide recommendations (Arjun, Abhinav).

Database:

- Populate db/suppliers.db with dummy data (PIN, StoreName, Contact, Address, Lat/Long) (Harsha).

Documentation:

- Update project card and architecture diagram with current progress