Deep Learning-Based Garbage Detection Using Aerial Images

EE 405 - GROUP 9

- E / 17 / 243 – Shane Perera
- E / 17 / 370 – Lahiru Wanninayake
- E / 17 / 374 – Aveesha Weerasekara

CONTENT

- GLOBAL PROBLEM
- ROLE OF DEEP LEARNING IN WASTE DETECTION
- CHALLENGES IN GARBAGE DETECTION
- METHODOLOGY
- DATA COLLECTION AND PREPROCESSING
- WHAT WE GAINED FROM THE LITERATURE REVIEW?
- SUMMARY OF OUR MODEL
- DEEP LEARNING MODEL ARCHITECTURE
- DETECTION AND CLASSIFICATION
- TRAINING DETAILS
- RESULTS AND EVALUATION

GLOBAL PROBLEM

- Massive production of disposable goods has led to a significant increase in garbage generation

- World Bank predicts over 3 billion tons of waste per year by 2050

- Low recycling rates and open garbage disposal contribute to environmental pollution

- Plastic waste is a major concern due to long-term environmental harm.

ROLE OF DEEP LEARNING IN WASTE DETECTION

- DL-based systems support waste detecting and classification processes.
- Smart self-sorting waste bins can classify objects on clear backgrounds.
- AI can help identify illegal dumping sites and locate agricultural plastic waste.
- Deep learning is now used with drones and cameras to detect waste in natural environment

CHALLENGES IN GARBAGE DETECTION

- Garbage identification is ambiguous even for humans without context.
- Trash can be atypical or deformed, spotted under uncontrolled natural conditions.
- Garbage objects can vary significantly in size.
- Garbage detection systems should perform well under different lighting conditions
- Well-annotated data is essential.

METHODOLOGY

Use existing datasets to create our own dataset.

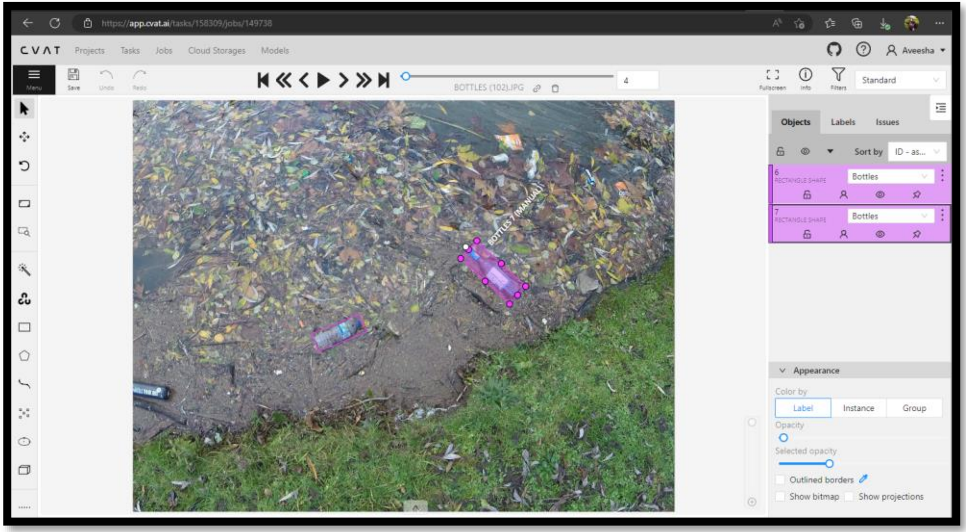
Seven well-defined waste categories: bio, glass, metal and plastic, non-recyclable, other, paper, and unknown.

Utilize FRCNN model for garbage detection and bounding box creation.

Classify the detected objects into the waste categories.

DATA COLLECTION AND PREPROCESSING

- Accurate garbage detection relies on a diverse and representative dataset.
- We have used following datasets to make our own dataset.
  - UAVVaste dataset
  - Aqua trash dataset
- We have used CVAT tool to annotate our dataset.



DATA COLLECTION AND PREPROCESSING

What we gained from the literature review?

Title of the Research Paper	Used Algorithms	Accuracy
Bottle Detection in the Wild Using Low-Altitude Unmanned Aerial Vehicle	Faster RCNN SSD YOLOv2 RRPN	86.4 % 87.6 % 67.3 % 88.6 %
Mask OBB: A Semantic Attention-Based Mask Oriented Bounding Box Representation for Multi-Category Object Detection in Aerial Images	Mask RCNN	96.7 %

SUMMARY OF OUR MODEL

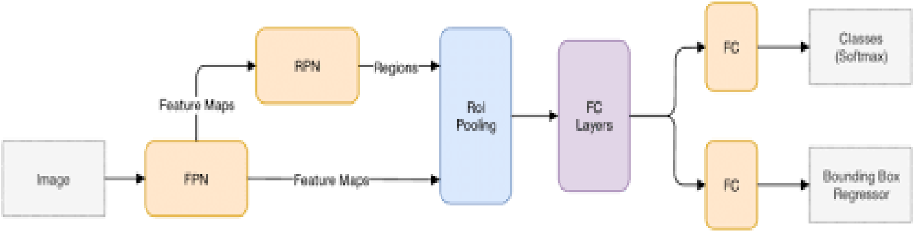
```python
# Display a concise summary of the model
print(summary(model, input_size=(1, 3, 224, 224)))
```

```
===================================================================================================
Layer (type:depth-idx)                                   Output Shape              Param #
===================================================================================================
FasterRCNN                                               [100, 4]                  --
├─GeneralizedRCNNTransform: 1-1                          [1, 3, 800, 800]          --
├─BackboneWithFPN: 1-2                                   [1, 256, 13, 13]          --
│    └─IntermediateLayerGetter: 2-1                      [1, 2048, 25, 25]         --
│    │    └─Conv2d: 3-1                                  [1, 64, 400, 400]         (9,408)
│    │    └─FrozenBatchNorm2d: 3-2                       [1, 64, 400, 400]         --
│    │    └─ReLU: 3-3                                    [1, 64, 400, 400]         --
│    │    └─MaxPool2d: 3-4                               [1, 64, 200, 200]         --
│    │    └─Sequential: 3-5                              [1, 256, 200, 200]        (212,992)
│    │    └─Sequential: 3-6                              [1, 512, 100, 100]        1,212,416
│    │    └─Sequential: 3-7                              [1, 1024, 50, 50]         7,077,888
│    │    └─Sequential: 3-8                              [1, 2048, 25, 25]         14,942,208
│    └─FeaturePyramidNetwork: 2-2                        [1, 256, 13, 13]          --
│    │    └─ModuleList: 3-15                             --                        (recursive
│    │    └─ModuleList: 3-16                             --                        (recursive
│    │    └─ModuleList: 3-15                             --                        (recursive
│    │    └─ModuleList: 3-16                             --                        (recursive
│    │    └─ModuleList: 3-15                             --                        (recursive
│    │    └─ModuleList: 3-16                             --                        (recursive
│    │    └─ModuleList: 3-15                             --                        (recursive
│    │    └─ModuleList: 3-16                             --                        (recursive
│    │    └─LastLevelMaxPool: 3-17                       [1, 256, 200, 200]        --
├─RegionProposalNetwork: 1-3                             [1000, 4]                 --
│    └─RPNHead: 2-3                                      [1, 3, 200, 200]          --
│    │    └─Sequential: 3-18                             [1, 256, 200, 200]        590,080
│    │    └─Conv2d: 3-19                                 [1, 3, 200, 200]          771
│    │    └─Conv2d: 3-20                                 [1, 12, 200, 200]         3,084
│    │    └─Sequential: 3-21                             [1, 256, 100, 100]        (recursive
│    │    └─Conv2d: 3-22                                 [1, 3, 100, 100]          (recursive
│    │    └─Conv2d: 3-23                                 [1, 12, 100, 100]         (recursive
```

SUMMARY OF OUR MODEL

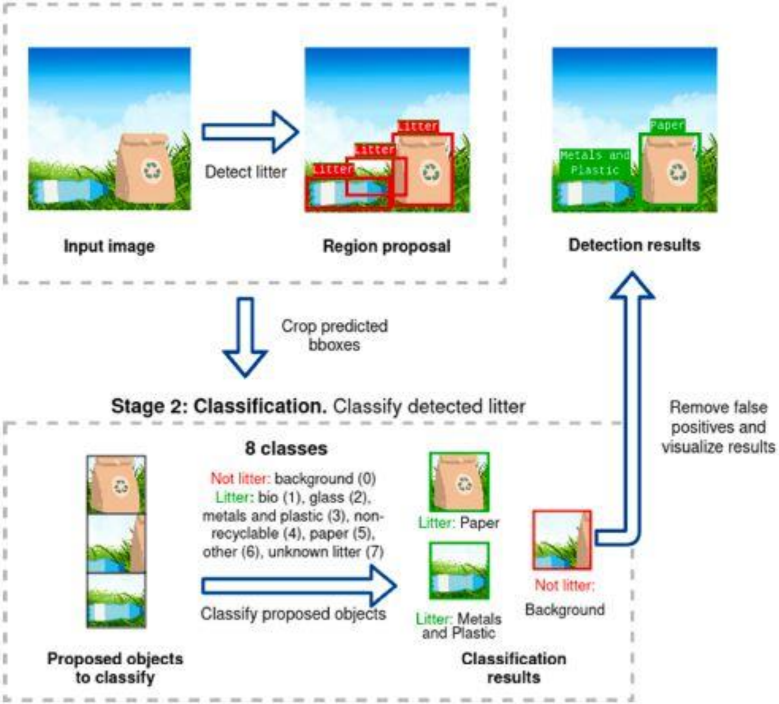
```
├─RegionProposalNetwork: 1-3                           [1000, 4]                 --
│    └─RPNHead: 2-3                                    [1, 3, 200, 200]          --
│    │    └─Sequential: 3-18                           [1, 256, 200, 200]        590,080
│    │    └─Conv2d: 3-19                               [1, 3, 200, 200]          771
│    │    └─Conv2d: 3-20                               [1, 12, 200, 200]         3,084
│    │    └─Sequential: 3-21                           [1, 256, 100, 100]        (recursive)
│    │    └─Conv2d: 3-22                               [1, 3, 100, 100]          (recursive)
│    │    └─Conv2d: 3-23                               [1, 12, 100, 100]         (recursive)
│    │    └─Sequential: 3-24                           [1, 256, 50, 50]          (recursive)
│    │    └─Conv2d: 3-25                               [1, 3, 50, 50]            (recursive)
│    │    └─Conv2d: 3-26                               [1, 12, 50, 50]           (recursive)
│    │    └─Sequential: 3-27                           [1, 256, 25, 25]          (recursive)
│    │    └─Conv2d: 3-28                               [1, 3, 25, 25]            (recursive)
│    │    └─Conv2d: 3-29                               [1, 12, 25, 25]           (recursive)
│    │    └─Sequential: 3-30                           [1, 256, 13, 13]          (recursive)
│    │    └─Conv2d: 3-31                               [1, 3, 13, 13]            (recursive)
│    │    └─Conv2d: 3-32                               [1, 12, 13, 13]           (recursive)
│    └─AnchorGenerator: 2-4                            [159882, 4]               --
├─RoIHeads: 1-4                                        [100, 4]                  --
│    └─MultiScaleRoIAlign: 2-5                         [1000, 256, 7, 7]         --
│    └─TwoMLPHead: 2-6                                 [1000, 1024]              --
│    │    └─Linear: 3-33                               [1000, 1024]              12,846,080
│    │    └─Linear: 3-34                               [1000, 1024]              1,049,600
│    └─FastRCNNPredictor: 2-7                          [1000, 10]                --
│    │    └─Linear: 3-35                               [1000, 10]                10,250
│    │    └─Linear: 3-36                               [1000, 40]                41,000
==========================================================================================
Total params: 41,340,161
Trainable params: 41,117,761
Non-trainable params: 222,400
Total mult-adds (G): 134.01
==========================================================================================
Input size (MB): 0.60
Forward/backward pass size (MB): 1484.01
Params size (MB): 165.36
Estimated Total Size (MB): 1649.97
==========================================================================================
```

DEEP LEARNING MODEL ARCHITECTURE



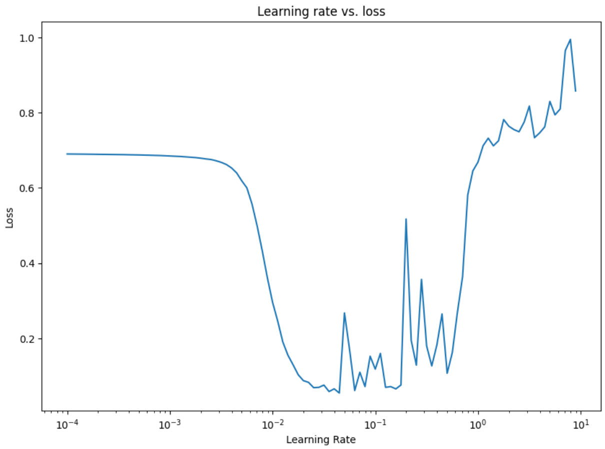
- Faster RCNN model was used.
  - Resnet50 was used as the backbone with consists a Feature Pyramid Network (FPN)
  - The Region Proposal Network (RPN) generating region proposals
  - The proposed regions are then passed through ROI Pooling layer
  - These feature vectors are fed into the Classification Head
  - Simultaneously, the Bounding Box Regression Head predicts adjustments to the bounding box locations based on the region proposals.
  - The final object detections are obtained by combining the region proposals, class probabilities, and bounding box adjustments

DETECTION AND CLASSIFICATION



TRAINING DETAILS

- We used 4500 images from the dataset that we created.
  - Training data - 3600 images
  - Testing data - 900 images
- Hyperparameters used during training.
  - Optimizer – Stochastic Gradient Descent (SGD)
  - Learning rate – Ideal learning rate was found as 0.05
  - Number of epochs – 5 at a time

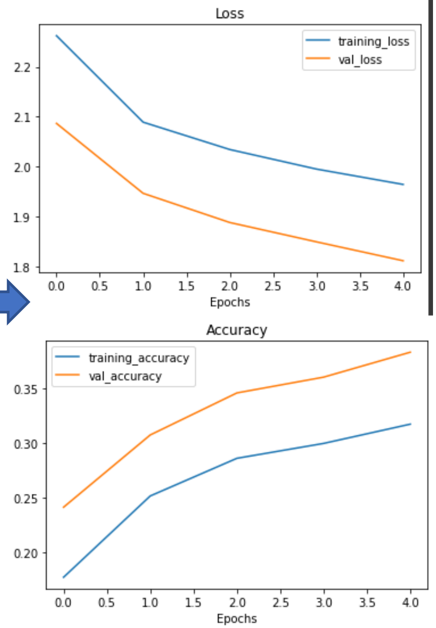


- Performance metrics for evaluation.
  - Mean Average Precision (mAP)
    - Precision - Accuracy of detecting bounding boxes(IOU calculated)
    - Recall – Ability of the image to classify the all instances of a particular object class.

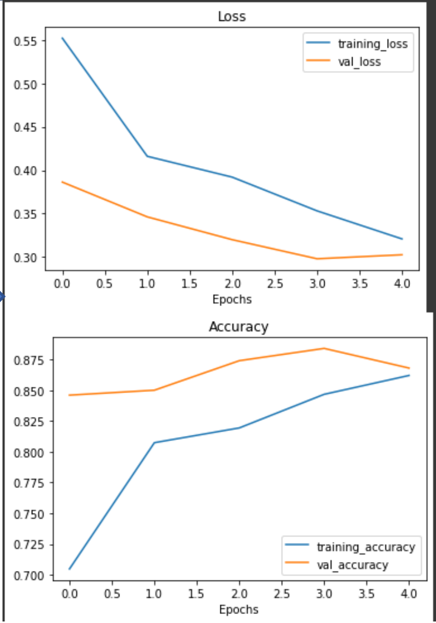
RESULTS AND EVALUATION

RESULTS AND EVALUATION

Validation and Accuracy plots for **FIRST** five epochs

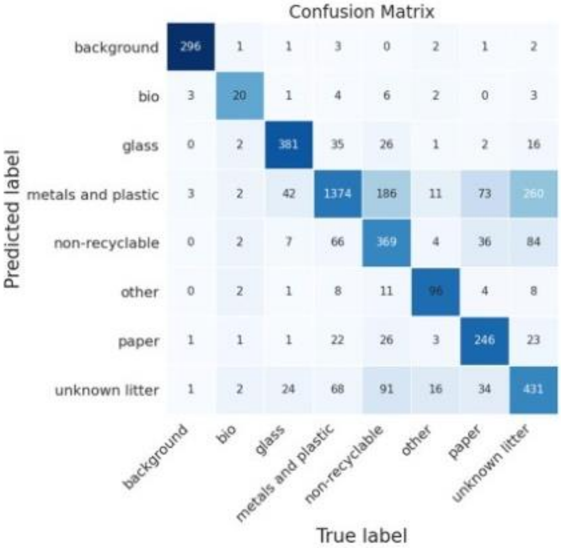


Validation and Accuracy plots for **LAST** five epochs



RESULTS AND EVALUATION

Confusion Matrix For Our Model



Thank You!