FINAL REPORT

Summary of problem statement, data and findings

Coffee and Tea estates in places like the Nilgiris are often bordering forests where one can find few wild animals, but one can surely find lots of elephants. These elephants are not interested in Coffee and Tea but most of the estates are sometimes on their path to reach a banana plantation, source of water or the estates themselves have banana trees etc. Now once these elephants enter the estates then they are very tough to get rid of without seeing them cause destruction and eventually leave once they are satisfied.

Currently barbed wire fences or electrically activated fences are what are the solutions employed to prevent them from entering but they don’t seem to keep these huge mammals away. What such estate owners often look for is the ability to get to know about the arrival of elephants early enough so that they can keep them away or do whatever is needed at their end to be better prepared to handle these situations.

So, elephant intrusion detection is of huge value to such estate owners who incur massive losses due to such intrusions. An intelligent way of detecting the arrival of elephants, ideally early enough, capturing details such as the number of elephants, proximity, if accompanied by human, presence of baby elephants is needed to build a repulsive system.

Overview of the final process

The most important feature is building the right set of training dataset, keeping in mind localized factors. The dataset must be resized for compatibility and augmentation can be used if needed. Our approach to solve this problem includes both classification and regression.

The bounding boxes with the regression tend to return lower metrics values and that can often mean vague indication of performance. To work around this situation, the methodology we intend to use for this pilot is manual investigation instead of system generated performance metrics. Once the model is built and trained, we will manually validate the output for validation manually instead of using system generated metrics.

The salient features are:

- Localization
- Ability to detect multiple objects of interest
- Ability to detect object with partial visibility
- Estimate proximity
- Estimate object size
- Detect presence of baby elephants
- Detect presence of human
- Estimate number of elephants

Following the deep learning framework, we will use a logical algorithm to determine the level of alerts and display them as different colours for the purpose of this pilot. In future, this can easily be converted into action workflows.

Step-by-step walk through the solution

Our first step towards the solution was to find out the methods (algorithms) that exist today which could be our starting point. Knowing very well that Elephant classification is already available today in various models that have been trained on the ImageNet dataset , for eg: VGG 16, RestNet, MobileNet etc.

Our solution needed a model that along with classification could accurately detect elephants, i.e localize them with good accuracy (80% and above). We were sure we did not need to write our model from scratch but instead use any of the existing models and if needed employ the method of transfer learning.

So here is how we set about solving our problem:

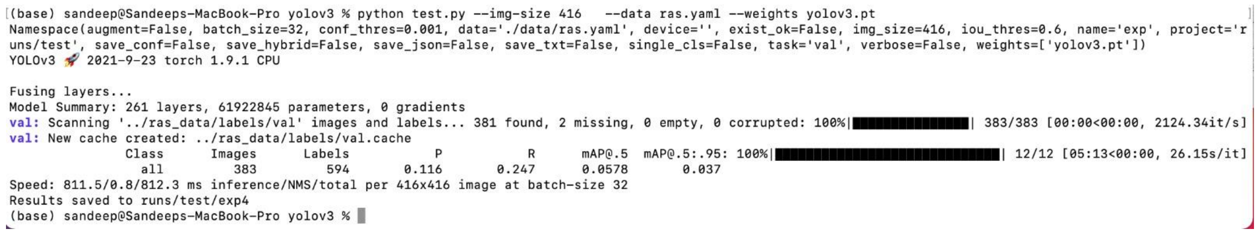
1) We scrapped the web to get images of elephants(single, in a herd, baby elephants etc). These sets of images were going to be used as our test images and if needed as our training set also. But one challenge for us was that these images did not have the annotations (labels with bounding box co-ordinates) if we had to train , so we would need to create the annotations if needed.
2) We used the Resnet50 model to test the predictions on a few images and as a classifier it seemed to be doing a good job. We knew Resnet50 in itself does not do object localization but this exercise was more to gain confidence that Elephant classification is achievable.
3) In our discussion with the mentor, we briefly discussed the possibility of using Yolov3 itself as our base model as it does object detection which is what we needed.

4) We implemented the base model of Yolov3 (referenced from web) and used the default Yolov3 weights and used this model to do run inference on our images. We observed (visual) that the accuracy of detection for elephants was good but did not perform that well on partially visible elephants and elephants very close to each other. This is when we decided to try training the Yolov3 using transfer learning. We annotated our images using “labelimg”.
5) First we tried to use a combination of Resnet50 as backbone for classification and Yolov3 for regression but we could not get to combine the two and move ahead. We had to drop this idea.
6) We then thought of instantiating the complete Yolov3 model , load its weights, freeze top layers and train the bottom few layers but ended up getting some errors which we were not able to fix(we sent this code to our mentor for cross checking)
7) Then we started looking for other options to train Yolov3 and that is when we came across https://github.com/ultralytics/yolov3 which provided a framework to train and test using Yolov3 for a custom data set. We then went through the material and code to figure out how to train a custom data set and more importantly how to train only a few layers keeping the weights the same as other layers.
8) Unfortunately when we did transfer learning on our custom data set (which is not very big) we observed that the results were very bad with low MAP values (less than 10 %). The reason mostly for that was that our dataset was not big enough. We tried tuning the learning rates, reducing the number of layers we freeze but results were not encouraging. Sad part about this whole exercise is that we tried lots of stuff but nothing much to show. Due to lack of documentation and any help we had to dig into the code in order to do transfer learning, try to get metrics for our dataset (again we could not figure out how to get metrics for just 2 classes in our case (yolov3 has 80 classes by default)
9) At this point the only option we had was to use the default yolov3 weights for our final application.
10) Parallely we had started working on the GUI using Tkinter. Once we had the GUI , we started working on the application that is needed to handle the requirements in our problem statement.

Model evaluation

The final model that we had decided to use was Yolov3. Unfortunately we could not get any accuracy metrics for our data set using the Ultralytics framework. So we had to rely on visual accuracy of detection of elephants and humans.

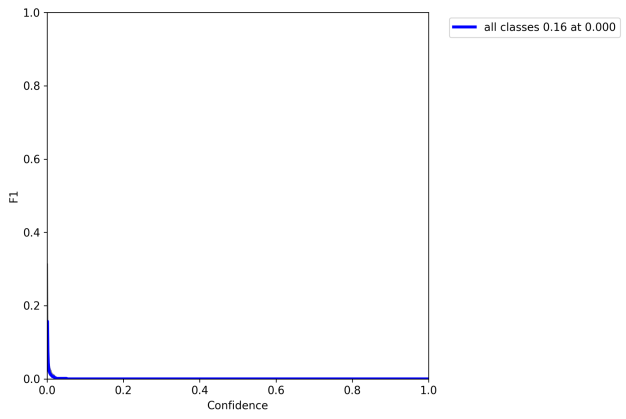
Here is the metrics we got when we did transfer learning(mAP values are very low):

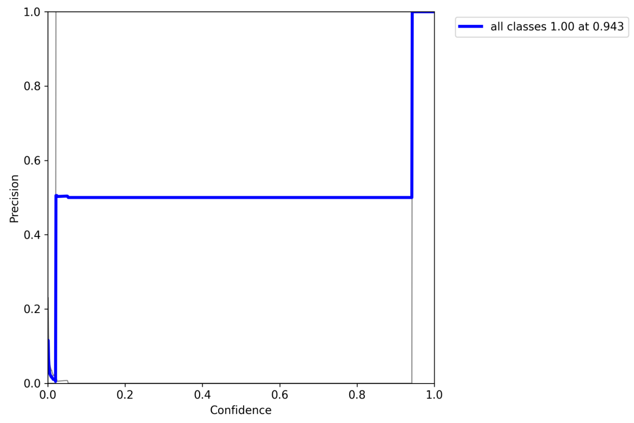


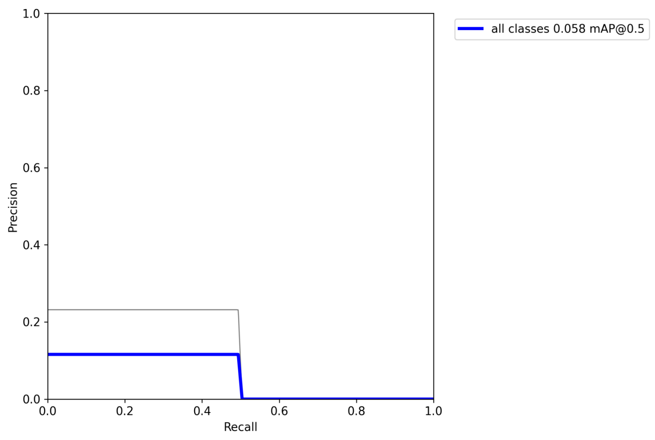
```
[(base) sandeep@Sandeeps-MacBook-Pro yolov3 % python test.py --img-size 416   --data ras.yaml --weights yolov3.pt
Namespace(augment=False, batch_size=32, conf_thres=0.001, data='./data/ras.yaml', device='', exist_ok=False, img_size=416, iou_thres=0.6, name='exp', project='r
uns/test', save_conf=False, save_hybrid=False, save_json=False, save_txt=False, single_cls=False, task='val', verbose=False, weights=['yolov3.pt'])
YOLOv3 🚀 2021-9-23 torch 1.9.1 CPU

Fusing layers...
Model Summary: 261 layers, 61922845 parameters, 0 gradients
val: Scanning '../ras_data/labels/val' images and labels... 381 found, 2 missing, 0 empty, 0 corrupted: 100%|███████████████| 383/383 [00:00<00:00, 2124.34it/s]
val: New cache created: ../ras_data/labels/val.cache
               Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100%|█████████████████████████████| 12/12 [05:13<00:00, 26.15s/it]
                 all         383         594       0.116       0.247      0.0578       0.037
Speed: 811.5/0.8/812.3 ms inference/NMS/total per 416x416 image at batch-size 32
Results saved to runs/test/exp4
(base) sandeep@Sandeeps-MacBook-Pro yolov3 % 
```

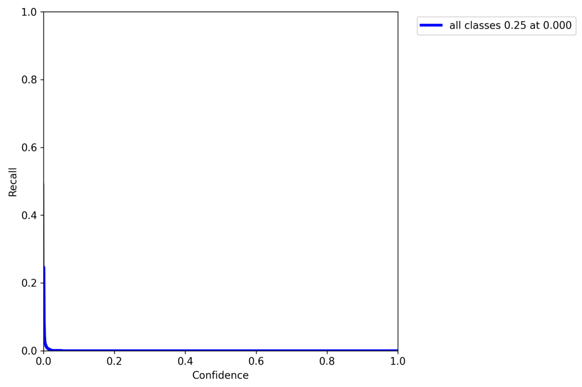
Below are some other metrics we got when we tested the mode for accuracy on our custom dataset:

Below are the F1 curve, P curve, PR curve & R curve.









Comparison to benchmark

Initially when we started on this problem statement, we did not have any statistical benchmark because our problem statement type was such that it needed more visual accuracy testing than statistical (over and above what we know certain models statistically are capable of).

Though we intended to try to improve on the performance (accuracy of detection) using transfer learning but we were not successful in doing that (as mentioned above)

Visualizations

We carried out Exploratory Data Analysis to identify the following which are important for our problem statement:

1) How does the model detect humans + elephant
2) Does model detect a baby elephant as an elephant or some other object (cow, dog)
3) Is Model able to detect multiple elephants standing close by or visible as a small spec in the image (elephants present far off from the camera)

Based on the above , we were successfully able to create a GUI based application to detect and localize elephants(single and in herds), baby elephants and humans.

We also developed a logic to determine the proximity of the elephant(s) and presence of baby elephant(s) in the images.

Please watch the attached video which provides a Color code alert system to warn about an elephant intrusion.

Color codes are as below:

1. **GREEN** - No presence of elephants in the image
2. **YELLOW** - If the elephant(s) is/are far or presence of humans in the image
3. **ORANGE** - If more than 3 elephants are detected or if the Proximity is medium
4. **RED** - If a herd of elephants is detected or if the Proximity is near

Implications

The solution that we show here as a POC can very well be deployed in the real world where there can be an edge device which can run inferences using YOLOv3 or a similar model ,on input images received from a camera.

The challenges would be to do the inferences in real time , which means there should be a powerful hardware which can run such AI algorithms OR if this is being done over the cloud , then the turnaround time needs to be quick(high speed internet connectivity is key)

Limitations

The backbone of the solution, the yolov3, is trained on the dataset that does not cover the localized factors, such as the landscape of tea and coffee estates, types of elephants (African vs Indian elephants).

We believe the model performance can be improved materially by adding custom training based on the localized images. The challenge is to prepare an adequate number of training images, which is quite time consuming.

We tried to take the benefit of transfer learning by using a custom training dataset to train the last few layers of the Yolo-v3 model and freezing all earlier layers. There are not too many resources available to understand custom training for yolo-v3, we had to deep dive into the code to figure out ways to custom train yolo-v3 model. Again, due to the lack of documentation, it's quite time consuming effort to figure out different hyper parameters combinations and layers to freeze to get optimized results. We felt the time was too short for us to completely understand the yolo-v3 model and tune. As a result, our model falls short on accuracy with localized images. However, in theory, we feel quite confident on the approach and with adequate time we believe the model overall performance can be further optimized.

Closing Reflections

For such problems where an elephant is an object which is part of the imagenet dataset ,using transfer learning is the ideal process to start from and find out methods to fine tune the model and improve accuracy. But the biggest challenge is to prepare the dataset which is difficult in a project of short duration.