Gedeon Muhawenayo
gmuhawenayo@aimsammi.org
African Masters in Machine Intelligence
Computer Vision II

LAB2:Training Detectron2 on a custom dataset

Summary:
In this LAB, I got to know how I can fine-tune Detectron2 on a custom dataset. I have started by preprocessing a dataset for Detectron2. Then next I went through steps of model initialization and training, and finally I did model evaluation. This LAB has helped me to clearly understand transfer learning with Detectron2, I have seen how model fine-tuning gives good results with relatively small training data. Basically, this Lab has widened my thinking and practical hands-on in object detection and segmentation tasks.

Part A: Data

To demonstrate this process of using Detectron2 on a custom dataset, we have used the fruits nuts segmentation dataset which only has 3 classes: data, fig, and hazelnut. To use a custom dataset while also reusing detectron2's data loaders I had to register my dataset to tell Detectron2 how to obtain data, and I have registered metadata of my dataset which is useful for augmentation, evaluation, visualization, logging, etc.

Visualizations of the training annotations from Part A

Fig 1. Example of annotated image

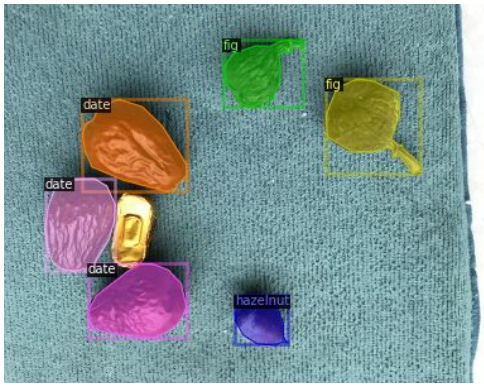


Fig 2. Example of annotated image

Part B: Model initialization and training schedule

I have trained two Mask R-CNN models, with a ResNet50 FPN backbone, the initial weights were coco for the first model and imageNet for the second model. The following figures compare their total training loss.

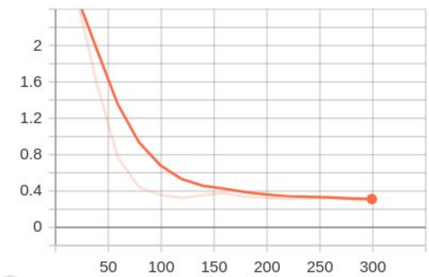


Fig 3. COCOinit model total loss

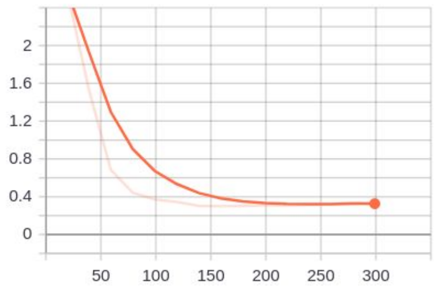
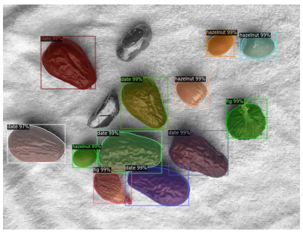
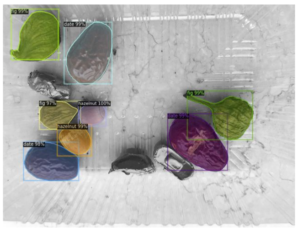
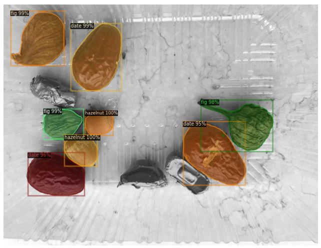


Fig 4. imageNe initt model total loss

From different training curves I have visualized in tensorboard, under the same parameters(learning rate, batch_size, iterations) I have observed that `COCOinit` model is better that `INinit` model. After 300 iterations the loss for `COCOinit` was **0.3021** while for `INinit` it was **0.324**. So, since the main target of the model training is to minimize the total loss, initializing the model with COCO is better than with ImageNet.

Part C: Inference and evaluation of the trained models

Comparing the inference time, both models have an average inference time of **0.08sec**. The following table summarizes the predictions and observations on both models.

<table>
<tr><th>Model</th><th>COCO init model</th><th>Imagenet init model</th></tr>
<tr><td>predictions</td><td> </td><td> </td></tr>
<tr><td>Observation</td><td colspan="2">Both models work well because they are well able to make predictions with an accuracy of over 90%, However when I look at general performance, the coco init model has better performance than the ImageNet init model. In many instances coco init model has more accuracy than ImageNet init model.</td></tr>
</table>

AP evaluation metric

To measure the performance of my models; precision, recall and IoU threshold were needed, I have used the AP metric, the following table summarises the results obtained on each model, either per-category(class) and on overall predictions. I have observed that the COCOinit model is better, This is because while they were training detectron2 on coco dataset they initialized on imageNet. Furthermore coco dataset was designed for detection and segmentation which is more similar to our task than imageNet designed for classification.

<table>
<tr><td rowspan="4">COCO init model</td><td rowspan="2">Bbox</td><td>Overall results</td><td>| AP | AP50 | AP75 | APs | APm | APl |
| 84.095 | 100.000 | 94.389 | nan | 78.194 | 89.656 |</td></tr>
<tr><td>Results per-category</td><td>|category| AP |category| AP |category| AP |
|date | 88.471 |fig | 81.955 |hazelnut| 81.859 |</td></tr>
<tr><td rowspan="2">segm</td><td>Overall results</td><td>| AP | AP50 | AP75 | APs | APm | APl |
| 93.510 | 100.000 | 100.000 | nan | 90.782 | 94.693 |</td></tr>
<tr><td>Results per-category</td><td>|category| AP |category| AP |category| AP |
|date | 99.385 |fig | 89.059 |hazelnut| 92.084 |</td></tr>
<tr><td rowspan="4">ImageNet init model</td><td rowspan="2">Bbox</td><td>Overall results</td><td>| AP | AP50 | AP75 | APs | APm | APl |
| 81.904 | 100.000 | 94.389 | nan | 79.180 | 81.965 |</td></tr>
<tr><td>Results per-category</td><td>|category| AP |category| AP |category| AP |
|date | 85.557 |fig | 74.574 |hazelnut| 85.581 |</td></tr>
<tr><td rowspan="2">Segm</td><td>Overall results</td><td>| AP | AP50 | AP75 | APs | APm | APl |
| 92.716 | 100.000 | 100.000 | nan | 89.703 | 94.495 |</td></tr>
<tr><td>Results per-category</td><td>|category| AP |category| AP |category| AP |
|date | 98.990 |fig | 87.376 |hazelnut| 91.782 |</td></tr>
</table>