PowerAI Vision / Visual Insights Workshop

Lab exercise 2 – Object detection

In this exercise you will practice Object Detection. With Object Detection, a ML algorithm learns to locate and identify defined objects in a scene. It is a sophisticated and very useful technique for several AI applications. The goal of the exercise is to familiarize with the user interface, practice the pipeline and OD related tasks.
The datasets used in the lab are available on github (the instructor shares the link) – download the files to your computer: poles1.zip, poles2.zip, poles3.zip, poles_diff.zip., poles_test.zip.

1. **Sign on** to PowerAI Vision.
2. Creata a **dataset** called ’Electric’.
    a. On the welcome screen, press ’Get Started’ (alternatively, select Data Sets from the menu bar). The Data Sets screen appears.
    b. Press the plus sign to ’Create a new dataset’ on the leftmost tile. Name the data set ’Electric’.
    c. Click on the tile ’Electric’. A ’Data set / ’Electric’ page will open.
    d. Click on ’Import files’ under the leftmost tile ’Drop files here’. Select the ’poles1.zip’ file for upload.
    e. The ’Electric’ data set is populated. At the bottom of the pane, select ’Items per page: 100’ to view as many images as possible. This will improve productivity.
        i. Note: In real cases, do not forget to set aside some pictures for testing your model later. Those pictures should not be used for training. For the lab, the instructors already created a separate test image collection.
3. Labeling **preparation**. Because object labeling is tedious, we would use an iterative process that improves the effectiveness of our work: with minimal labeling, train a simple, rough model, that will be used to help label more images; we will do only the correction manually.
    a. Open the ’Electric’ dataset. Five objects will be identified, most of them are insulator types: **’Pin’, ’Shackle’, ’Strain’, ’Bell’ and ’Lamp’**.
    b. Revise the images. At least 5 images are required by object type. It is best if contrast, brightness, sharpness and other visual attributes make the pictures clear for the first model. Try to manually improve the images that needs correction (work outside PAIV). If a picture is hopeless, you’d rather delete it.
4. Define and **label** objects.

        i. Note: we will use bounding box in the first round, just because it is quicker. Should you have time, try Polygon selection and use the Detectron model for training.
        ii. Note: before you start, click on an image then click on 'Label objects' and familiarize yourself with the interface. Look at options (represented by a small gear) and Keyboard shortcuts. The shortcuts availability depend on the platform / keyboard you use.
        iii. Instead of selecting the objects one by one, select all and open the labeling interface. This is a more efficient workflow.
    b. On each image, perform the following steps.
        i. Select all images and click 'label objects'
        ii. If there is no object category, select 'Add new' and create an object. You will use the above listed objects (Pin, shackle, strain, bell and lamp).
        iii. Select the object tag and draw a bounding box around the object.
        iv. All objects of which the majority is visible should be labeled. Do not label objects that are not clearly representing what you want to recognize. Do not leave much empty space around the object. Do not let the bounding box get out of the image boundary.
    c. Once you finished labeling, check if there are enough labels per category. At least 5 image ocurrences are required per object (objects in the same image are not counted separately). You will find that 'Pin' is repreented in more than 80 cases while the count of 'Bell' is hardly enough among the 32 images.
    d. Augment the data with horizontal flipping, rotation and tweaking color (Rotation: 20°, Hue:50, Brightness:10). Create a new dataset e.g. 'Electric_aug'.

5. Select the new dataset and start **training** for Object detection (Faster R-CNN)
    a. *Note: For object detection, the training can be stopped without losing the model. Once training loss flattens, there is no need to wait any longer, because the model will not improve.*
6. Once the model is ready, deploy it.
    a. Note: Though this model is not good, precise manual preprocessing and labeling gives it a good starting point to get close to a useful one. This model will be used as an assistant in automatic labeling. The errors it makes should be manually corrected. Therefore, there is a tradeoff between the time and resources (images, labeling, training time and afterwards, correction time) invested in the primary model and the time it saves by quickly allowing us to move to the second model.
7. Go back to Data sets and duplicate 'Electric'. Assign the name 'Electric2' to the copy.

   a. Populate the data set with images from poles2.
      i. Note: If you have no time, you can upload pole3 immediately. Otherwise, you can use pole 2 to create a further refined model, and with that, experiment on pole3, using auto-labeling again.
   b. Select Auto label, and from the deployed models list specify the recently trained and deployed model.
   c. Let Auto label run for some time.
   d. On each image, perform the following steps:
      i. Click on the image and select 'label objects'
      ii. If there is no object category, select 'Add new' and create an object. You will use the above listed objects (Pin, shackle, strain, bell and lamp).
      iii. Check and correct the object bounding boxes and labels assigned by auto labeling.
      iv. Complete labeling for missing boxes and objects, as well as corrections.
      v. Select the object tag and draw a bounding box around the object.
      vi. All objects of which the majority is visible should be labeled. Do not label objects that are not clearly representing what you want to recognize. Do not leave much empty space around the object. Do not let the bounding box get out of the image boundary.
   e. Once you finished labeling, augment the data with horizontal flipping, rotation and color (use previous values). Create a new dataset, e.g. 'Electric2_aug'.

8. Select this dataset and start **training** for Object detection (R-CNN). Before that, if you need to free up resources (training will require a GPU), you can delete the previously deployed model (it was bad anyway).
   a. Note: For object detection, the training can be stopped without losing the model. Once training loss flattens, there is no need to wait any longer, because the model will not improve.
9. If you opted for separately working with the poles3.zip file, repeat the deployment/autolabel/correction/training phase. When you get your final model, it should be a fair one, so you can deploy it.
10. Test the model for object detection using images that were not involved in the training process.

    - - - IF TIME ALLOWS : - - - RECOMMENDED EXPERIMENTATION AND LEARNING

11. Restart the process with Polygon selection and Segmentation (Detectron) training. The results and the testing could be slightly different:
12. Evaluate the model and compare the results. Testing shows tangible difference.
13. Export / delete / reimport / retest the model.

14. Export the (even partially) labeled dataset. Look at the exported .zip file, and the respective XML descriptions. Optionally, delete and reimport the dataset.
15. Check how ’difficult images’ can affect models.
    a. Record the performance of your best model against all test images.
    b. Create a new data set (’Electric_diff’), that uses poles1, poles2, poles_diff. Poles_diff contains selected images that are hard to recognize, e.g. there is a dominant pattern in the background with quite limited amount of data.
    c. Use auto-label with your best model and perform the necessary corrections.
    d. Train your model.
    e. Test the model and compare the performance against those that were trained with ’easy’ images. Where are the differences? What are the main causes?

This concludes the exercise.