DYOM AY 20/21 S2 Final Assignment

Option 1 - Building and Deploying a CNN Classifier Almost from Scratch

The task for this option is to build, train, evaluate and deploy an Image Classifier.
You will have to do the following:

1. Choose a framework (Tensorflow/Keras or Pytorch)
2. Choose a dataset, from either of the following 2 options
    a. Your own dataset with the following requirements
        i. More than 10 classes
        ii. At least 500 images per class
        iii. Dataset must contain images with at least 300px at the shortest dimension. Width or height whichever is shorter must be at least 300px long.
    b. Flickr Image Style Dataset
        i. Follow download instructions here under "Download flickr style dataset"
        ii. You can reduce the dataset (use less than the given number of classes) as long as it meets the same criteria as the choose your own dataset option.

3. Train a CNN for classification via Transfer Learning.
    a. You can either use the CNN as a feature extractor [example] or via fine tuning [example].
    b. More information on Transfer Learning can be found here
    c. The final trained classifier should be able to achieve at least 70% accuracy on the test set.
4. Evaluate the trained classifier
    a. Use relevant metrics (e.g.: accuracy, confusion matrix, top-k accuracy)
5. Create a simple command line tool or python script that takes in a file path to an image as input and produces the top-3 predicted classes and their probability as output.
6. Ensure your results are reproducible.
7.

Submission:

1. Dataset source
2. Code clearly showing the following
    a. Preprocessing steps

    b. Training steps
    c. Evaluation steps with relevant metrics
    d. Inference function that reads an image file and returns top-3 classes and their probability
3. Simple usage instruction document

Option 2 - Deploying an Object Detection Tool/Service

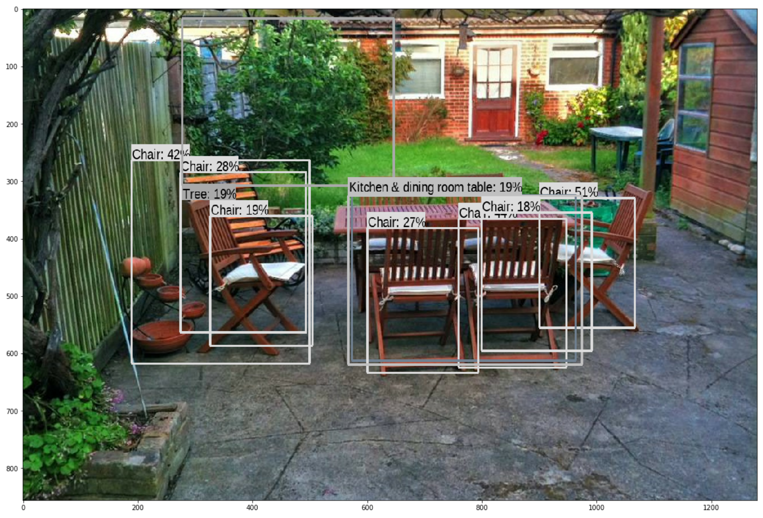
The task for this option is to deploy an object detection tool or service using existing pre-trained models.

The pretrained Object Detector and detection function can be found in the following colab notebook. Alternatively, you can use your own object detection model or code.

You will have to do the following:

1. Use the object detection model and detection code to create your own deployable
    a. Tool, or
        i. Command line tool that takes in an image and produces an image with drawn bounding boxes (Bbox drawing function provided in the colab notebook).
            1. **In**:

2. **Out**:



b. Service
    i. Client-Server Architecture
    ii. Server:
        1. REST API or Web Application
        2. Takes as input an image and displays or returns an image with bounding boxes drawn. (See above for example)
        3. Communication between client and server (Rest API) must be in JSON format.
        4. Hint: To transmit an image between client and server which is encapsulated in JSON, you can use base64 encoder to encode/convert image to become a string.
    iii. Client:
        1. Simple web client or command line client
        2. that takes as input an image path and uploads/sends that image to the server (REST API) for object detection.
        3. Receives JSON response from server with coordinates to plot on the image or the image with the plots itself.

2. For both Service and Tool you are to do the following
    a. Consider User Experience
        i. Keep inference time as minimal as possible
        ii. Make usage as hassle free as possible
    b. Consider Usefulness
        i. If you want to make a car detector, you don't have to plot other detected objects
        ii. If you want to make a device (handphone, laptop, etc) detector you don't have to plot irrelevant objects.
        iii. Info: List of objects classes can be found here
        iv. Info: Information about the dataset the detector is trained on can be found here.

Submission:

1. Client and server code for Service
2. Tool code for Tool
3. Inference time:
    a. Using a few sample images, record how long each inference takes. Include this information in the documentation.
4. Simple usage instruction documentation

Additional Resources

[1] Building a simple Keras + deep learning REST API