Mini project 2 Report

Introduction

Machine learning models to detect a segment object (TensorFlow Zoo):

An image often includes multiple useful information for us to find and learn. Traditional manual detection by a human is not suited for thousands of hundreds of images with huge data stream, so machine learning models should be considered for this situation. But analyze multiple objects from one single image and create machine learning models to label them accurately is still a challenge in computer vision (ObjectDetection, 2019).

TensorFlow Zoo is a collection of powerful machine learning models to detect segment objects. It provides a collection of detection models pre-trained on multiple datasets, such as, COCO dataset, the Kitti dataset and the Open Images dataset. They are useful for initializing the personal models when user training their own datasets or out-of-the-box inference if the user is interested in categories which are already in those datasets (GitHub: TensorFlow detection model zoo).

Analysis of results including pros and cons:

1. Easy to get started:
   The only preparation needs to be done is installing python, Jupyter notebook, TensorFlow, Matplotlib and the related data set API. The user can choose to run it locally or on the cloud (GitHub: Installation of TensorFlow models).

2. Defining own model architecture:
   TensorFlow provides DetectionModel interface in object_detection/core/model.py for users to easily implement their own model. Including five functions:
   Preprocess: shift the images before using as input in the detector
   Predict: Produce “raw” prediction tensors which can be passed to further functions
   Postprocess: Convert predicted output into final detections
   Loss: Based on the groundtruth to compute the loss
   Restore: Load a checkpoint into the TensorFlow graph
   (GitHub: So you want to create a new model)

3. Faster speed and high accuracy

After establishing the pre-trained model, it will report running time in millisecond per 600x600 image (including all pre and post-processing), but still due to the hardware performance (GitHub: TensorFlow detection model zoo).

4. Result and performance difference:
   Result or performance may have a difference between personal training and standard result, due to the hardware and threshold setting (GitHub: TensorFlow detection model zoo).

Recommendations for using such systems:

1. Beware to learn the related algorithms before using the model or establish own models
2. Make sure to understand the pros and cons to each data sets
3. Make sure to learn how to use each function and parameters properly
4. Choose the data set which is closer to the current situation in the training

Conclusions:

TensorFlow Zoo is a good collection of plenty of machine learning models which used for segment objects detection and it is very easy to set up and predict the very good result with lower cost. And users can change the model easily to suit for their own data.

References:

GitHub, So you want to create a new model

Retrieved from:
https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/defining_your_own_model.md

Accessed 14 October, 2019

GitHub, Installation of TensorFlow models

Retrieved from:
https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/installation.md

Accessed 14 October, 2019

Model Zoo, ObjectDetection

Retrieved from: https://modelzoo.co/model/objectdetection

Accessed 14 October, 2019

GitHub: TensorFlow detection model zoo

Retrieved from: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md

Accessed 14 October, 2019

Summary of other group members:

1. Regular Machine Learning Model:

   TensorFlow, PyTorch, Keras

2. Recommendation for TensorFlow:
   1) The static model allows for parallelism and faster, more efficient training
   2) TensorFlow has better computational graph visualizations
   3) TensorFlow has a huge collection of data set backed by Google
   4) TensorFlow can be used on various hardware machines
3. Mask-RCNN:

   Mask R-CNN can be implemented on Python 3 and TensorFlow. The model generates bounding boxes and segmentation masks for each instance of an object in the image.

   Mask-RCNN extends Faster-RCNN via adding a branch for inferring segmentation masks on each Region of Interest (ROI), in parallel with the existing branch for classification and bounding box regression.

4. Unsupervised Learning
   1) Training samples whose categories are unknown
   2) Main methods: Cluster analysis
   3) Clustering divides the data set into different categories based on the data characteristics, so that the data within the category is relatively

similar/correlated, and the data similarity/correlation between the categories is relatively small. We focus on two properties. The first property is consistency and the second property are association

4) Pros:
   (1) No label & clustering: it can start working as long as it knows how to calculate similarity
   (2) Reduce dimension: Unsupervised learning need to extract features, or use layer/item clustering to reduce the dimension of data features
   (3) Non-independent: the cause of the offset between positive and negative samples is smaller than supervised learning
   (4) Interpretable: Unsupervised clustering is well explained on the reason for classification because the elements in one group have similar features and consistency
   (5) Expandability: The unsupervised algorithm is scalable to a n-dimensional model.
5) Cons:
   (1) Unpleasant accuracy and validity
   (2) Data mining
   (3) Abnormal detection
6) Applied Field: detect a segment objects, advertisement