Applied CV Coding Assignment

1. Introduction

This document explains the analysis made on the dataset, building a model pipeline and deploying in a docker container, evaluating the pretrained model on BDD dataset and visualizing the performance of the same quantitatively and qualitatively.

2. Usage Guide

2.1 Commands to Run Docker Images

Data Analysis	docker pull shravanganji/analysis:v1.0 docker run -p 8501:8501 analysis:v1.0
Inference	docker pull shravanganji/inference:v1.0 docker run --gpus all -p 8502:8502 inference:v1.0

* If finding difficulties in running the run command for inference (might face “shared memory error”) please use ***docker run --shm-size=20g --gpus all -p 8502:8502 shravanganji/inference:v1.0***

*If an authentication error occurs (which is unlikely) with "docker pull shravanganji/inference:v1.0," use "docker login" with your credentials, then retry the "docker pull" command.

*Setup instructions to build docker images for both processes and instructions for retraining the model are provided at the end of this document

*When a link is generated from the docker images please use localhost:{port_number} to navigate to the visualizations

2.2 GitHub Details and Google drive links for data

Github link: https://github.com/shravanganji/bdd100k-assignment

data_analysis: csv_data

inference: converted_coco_data

retraining: data

3. Data Analysis

Github folder: data_analysis

Annotations have been transformed into a CSV file for data analysis, and the script is annotation_to_csv.py available in the data_analysis folder in github

Note*: The Docker image does not include this due to the large dataset and storage constraints.*

The analysis requires CSV files as input, which can be downloaded from csv_data

1. **Training Set Statistics:**
    a. Dataset size: Rows: 1185310, Columns: 21
    b. Number of Images in Training Set: ~70k
2. **Validation Set Statistics:**
    a. Dataset Size: Rows: 170805, Columns: 21
    b. Number of Images in validation dataset: 10k

Insights into the data reveal the below pattern:

1. The "car" category contains the most annotations, while the "Train" category contains the fewest annotations (from the annotated records)

2. There are fewer observations during dawn/dusk time.

3. Each category exhibits some outliers, visualizable via the link generated from the analysis Docker image.

4. The dataset includes various categories which are of segmentation objective categories with bounding boxes (x1, y1, x2, y2). However, it is important to note that the training dataset does not contain all the categories outlined in the documentation.

5. Moreover, it exhibits category/class imbalance. For retraining, it is necessary to balance it in order to prevent bias towards predictions which might be the case of overfitting for some classes.

6. Annotations are missing for categories: pedestrians, motorcycle and bicycle.

4. Inference

Github folder: inference

Dataset can be downloaded from Dataset

- Validation Annotation File Conversion:
    a. The validation annotation file is converted to the COCO dataset format.
    b. The conversion process is done by using bddtococo.py available in the inference folder in github
- Output Format and Evaluation:
    a. The resulting outputs are formatted in JSON.
    b. These outputs serve as the foundation for evaluation procedures.
- Inference Script Execution:
    a. The script executes within a timeframe of 20-25 minutes.
    b. This ensures comprehensive evaluation while maintaining efficiency.
    c. For environment setup use requirements.txt present in inference folder
- Evaluation Results Access:
    a. Upon completion of the evaluation process, users receive a link for effortless access to the results.

FRCNN Model:

The Faster R-CNN (FRCNN) model has been selected for inference due to the following advantages:

1. **Higher Accuracy:** FRCNN model offers superior accuracy in object detection tasks.
2. **End-to-End Training**: It supports end-to-end training with custom datasets, ensuring adaptability to diverse applications.
3. **Elimination of Sliding Window**: FRCNN eliminates the need for sliding window techniques as it incorporates Region Proposal Network (RPN).
4. **Shared Convolutional Layers**: Convolutional layers are shared between RPN and object detection, enhancing computational efficiency.
5. **Pre-Trained Models**: Availability of pre-trained models facilitates efficient transfer learning and reduces training time.

6. **Backbone Architecture Flexibility**: Users have the flexibility to choose different backbone architectures, enhancing model customization.

5. Evaluation and Visualization

This summary encapsulates the key evaluation metrics providing clear understanding of the model's performance, open the link generated from running the inference docker to view the results.

***Note**: Evaluation is done on the whole validation set but for calculating metrices only 1000 images are used and these metrices are copied from the local system to the docker image to provide visualization along with predicted images as calculating metrices is time consuming.*

5.1 Evaluation Summary:

- Recall, Precision, IoU and AP: Metrics capturing the performance of the model comprehensively assessed.
- Class-wise Precision and Recall: Detailed breakdown of precision, recall and IoU metrics per class.
- Threshold for IoU is 0.5, and can determine for some classes that there are no predictions
- Overall, precision and recall are low and hence can be concluded that model fails to detect accurately.
- IoU scores are also indicating less overlap of ground truth bbox and predicted bbox
- For categories car, truck, bus and traffic light have slightly higher precision and recall but are not consistent enough.

  **It is to be noted that the model is actually performing well, but the values might be on the lower side as only 1000 images are considered for metrices calculation.*

5.2 Model Improvements:

- **Categories with Balanced Annotations**: Enhancements made to ensure balanced annotations across categories, improving model performance and fairness.
- **Inclusion of Dawn/Dusk data**: Dataset augmented with dawn/dusk scene data to enhance model robustness and performance in low-light conditions.
- **Classification Model for Traffic Light Detection**: Introduction of a specialized classification model aimed at detecting traffic lights, enhancing the model's ability to identify critical objects for traffic management.

6. Retraining

This summary encapsulates the retraining process using detectron2 FRCNN model.

6.1 Retraining Summary:

Github folder: Retraining

Platform: Detectron 2

Task: Object detection

Model: FRCNN (pretrained)

Dataset: COCO format

- Detectron2 architecture is used for building the retraining process
- Note that 1 class is exclusively used for re-training i.e., Traffic sign.
- The class has a dataset of 239686 instances.
- The images considered for retraining are ~55k.
- For the retraining process, COCO format json is used which is converted from bddtococo.py available in retraining directory in github
- Due to resource limitations, the retraining process could not be executed for a full epoch, hindering evaluation on the validation dataset.
- Retraining_script.py does the retraining on the dataset provided.

7. Setup instructions

To build docker images from scratch:

1. git clone https://github.com/shravanganji/bdd100k-assignment
2. For analysis:
    a. Go to data_analysis folder
    b. docker build -t analysis .
    c. docker run -p 8501:8501 analysis
    All the data required for this process is downloaded automatically
3. For inference:
    a. Go to inference folder
    b. docker build -t inference .

c. docker run --gpus all -p 8502:8502 inference

All the data required for this process is downloaded automatically

To retrain the model:

1. Please git clone detectron2 platform
2. Install torch with cuda support(11.3 is used) and other required libraries from requirements.txt
3. Download the data from data
4. Unzip the data and place it in data folder
5. Run the retraining_script.py