Optimized Road Damage Detection Challenge (ORDDC'2024)

(Team: T24_009_Vung_Pham)

Vung Pham[1], Lan Dong[2], and Linh Bui[3]

[1]Department of Computer Science, Sam Houston State University, Texas, USA
[2]Faculty of Economic Information Systems, Academy of Finance, Hanoi, Vietnam
[3]FPT Polytechnic, FPT University, Hanoi, Vietnam

September 15, 2024

1 Introduction

This report provides a detailed overview of the data processing, model training, and steps required to reproduce the results. The source code and key configurations are available on the project's GitHub repository: `https://github.com/phamvanvung/roaddamagedetection2024`. Readers are encouraged to refer to this GitHub page while reviewing the report, as it includes references to specific Jupyter Notebook files used for the various tasks discussed.

2 Data processing

2.1 Cropping images

he Norway dataset consists of images with varying dimensions, some as large as 2,500 pixels in height and 4,500 pixels in width. Due to hardware limitations, it is not feasible to train models using these large image sizes. Additionally, the majority of road areas, where damages are concentrated, are located in the lower left corner of the images. To address this, we decided to crop the images to a resolution of 1824×1824 pixels, focusing on the lower left corners. This processing is performed by the script `ProcessNorway.ipynb`.

2.2 Annotation conversion and train/validation splitting

The deep learning technique employed for this project is YOLOv7. Consequently, we converted the annotation data from XML files in PASCAL VOC format to $*.txt$ files compatible with the YOLOv7 format. Additionally, the training data for each country was divided into training and validation sets, with a 90% to 10% split, respectively. The data conversion and splitting tasks are handled by the script `PerCountryDataConversion.ipynb`.

2.3 Potholes dataset

The current dataset contains only a limited number of potholes, resulting in low accuracy for pothole detection and classification. To address this, we incorporate the publicly available Pothole dataset from Roboflow, accessible at: `https://universe.roboflow.com/brad-dwyer/pothole-voxrl`. This dataset consists of 665 road images with labeled potholes and is available in various formats compatible with common machine-learning models. For our project, we use the YOLOv7 annotation format. The Pothole dataset has a single class (pothole), with an identification number of 0. In our dataset, however, the pothole class is identified as class 3. To align the class identification, we use the script `PotholesDataProcessing.ipynb` to update the class numbers.

3 Training models and performance tuning

3.1 The edited version of YOLOv7

The primary algorithm used in this project is YOLOv7 (`https://github.com/WongKinYiu/yolov7`). However, we have modified several files to adjust the training process, hyperparameters, and data loaders to enhance both accuracy and inference time. Therefore, for this project, please use the YOLOv7 version available in our GitHub repository (`https://github.com/phamvanvung/roaddamagedetection2024`), rather than the original version.

3.2 YOLOv7 model with additional coordinate attention layers

We have incorporated three Coordinate Attention layers (`https://github.com/Andrew-Qibin/CoordAttention`) into the YOLOv7 neck, training a separate model for each country. These layers are placed just before sending the data to the three corresponding final detection heads. The configuration file for this model can be found at `cfg/training/yolov7-custom-CA.yaml`. Since the annotations across different countries are somewhat inconsistent, we should not be entirely confident in the accuracy of each label. To address this, we apply the label smoothing technique with a rate of 0.1 during training, as specified in the `data/hyp.scratch.p5.custom.yaml` file. The training code for this model is provided in the `YOLOwith3CA.ipynb` Jupyter Notebook.

3.3 A tiny YOLOv7 model

We want to improve accuracy using an ensemble of different YOLOv7 models. The custom-trained YOLOv7 model achieves good accuracy, but its inference speed is relatively slow. To minimize inference time increment and still improve accuracy for small images (those not from Norway), we also trained a tiny YOLOv7 model using its default configuration (`cfg/training/yolov7-tiny.yaml`) and hyperparameters (`data/hyp.scratch.tiny.yaml`). This model is ensembled with the custom YOLOv7 model described previously to improve accuracy. The training code for this model is available in the `YOLOv7tiny.ipynb` Jupyter Notebook.

4 Inference

4.1 Reparameterizing models

Model reparameterization in YOLOv7 is a technique that restructures the network architecture to enhance inference efficiency without altering the model's behavior. It works by merging multiple convolutional layers and batch normalization operations into a single equivalent convolutional layer during inference. This reduces computational overhead and memory usage, leading to faster inference speeds while maintaining detection accuracy. YOLOv7 employs a carefully planned reparameterization strategy that optimizes performance by considering the gradient propagation paths across different network architectures. At the end of the training process (as mentioned earlier), we added code to execute the reparameterization process and saved the reparameterized models in the `weights` folder under the `yolov7` directory, appending the keyword `deploy` to the model name.

4.2 Inferencing and saving results

The default `dataset.py` file in YOLOv7 has been modified to include two new classes: *LoadNorwayImages* and *LoadOtherImages*. The key difference between these classes is that the former loads larger images (those with a width and height greater than or equal to 1,824 pixels), while the latter handles all other images. Additionally, the `detect.py` script has been customized and saved as `batchdetectsinglefornorway.py`. This new detection method performs batch-based inferences, using the trained tiny YOLOv7 model to detect road damages in large images (such as those from Norway) and an ensemble of reparameterized models for other images. The inference results are stored in a list (instead of separate *.txt files) and processed before saving the submission file in *.csv format. We also apply test-time augmentation to enhance performance. Finally, the provided `inference_script_v2.py` script is used to perform the final inference.

5 Reproducibility

To execute the codes from Github, you should follow these steps

- Create a folder called *roaddamagedetector* (project folder)
- Download the datasets from the organization and extract them into 'RDD2022_0' folder.
- Download/clone our GitHub repository for the edited version of YOLOv7 and related Jupyter Notebooks.
- Perform data processing steps as listed in Section 2.
- Train models as described in Section 3.
- Perform inferences using the steps in Section 4.