Assignment 2: Object Detection for Autonomous Driving

In this assignment, you will develop an object detection system for autonomous driving. You are required to **implement a simple detection network** called *AD-Detector (Naïve Version)* with ResNet as Backbone and YOLO as detection head. Through this assignment, you are expected to learn about designing and optimizing an object detection network.

This codebase is only for this course. Please **don't upload your answers or this codebase** to **any** public platforms (e.g., Github) before permitted. This assignment is adapted with permission from HKU DASC7606 (Zhixuan Liang & Haibao Yu), All rights reserved.

1. Problem Description and Assignment Objectives

1.1 What is Object Detection for Autonomous Driving?

Object detection is one of the most essential tasks in autonomous driving.
This task aims to recognize and localize traffic participants, like cars in the road scene, from images or point clouds. This assignment focuses on solving the detection problem from a single image.
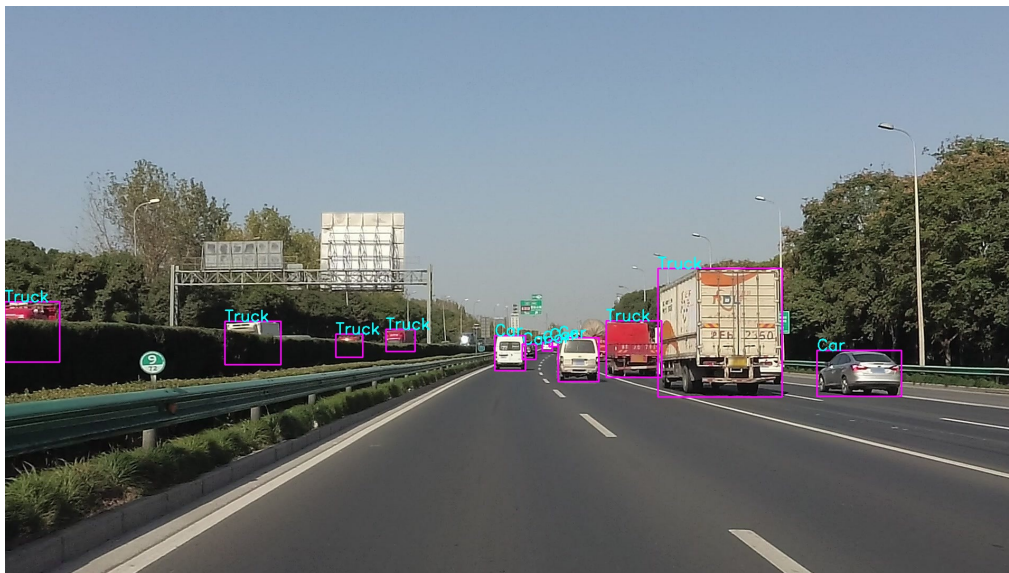

Figure 1. A Visualization Example of Object Detection for Autonomous Driving.

1.2 What will you learn from this assignment?

- Understand the basic theories of Deep Learning exspecially for object detection task, such as the networks of **ResNet** and **YOLO v1**, the detection loss function, and important concepts of object detection (e.g. **ROC Curve and Mean AP**).

- Apply object detection algorithm to solving **autonomous driving problem**, such as detecting cars and pedestrians in the traffic environment.

- Gain experience of implementing neural networks with a popular deep learning framework **PyTorch**.

- Develop a deep learning system **from scratch**, including **network design, model training, hyperparameter tuning, training visualization, model inference and performance evaluation.**

- *Spontaneously and deeply explore advanced solutions for object detection task. (Italics indicate bonus requirements, the same below.)*

1.3 What should you prepare for this assignment?

- Master the basic use of Python and Pytorch.

- Be familiar with the **ResNet**.

- Be familiar with the **YOLO v1**.

  (You can learn about them step by step in the process of completing this assignment.)

2. Assignment Tasks

- You must fill in the blanks and submit the completed codes for this assignment with the provided codebase. This assignment offers an incomplete codebase for object detection with modified **YOLO v1**.

- You must submit the test data's model outputs with your completed codes, the test data annotations will be released before deadline. You should utilize hyperparameter tuning and other widely-used techniques to improve detection performance.

- You must submit a maximum of 6-page technique report to explain how you improve the object detection performance. Besides, some visualization results (e.g., loss curve) are encouraged in this report.

2.0 Dataset and Codebase Preparation

The dataset is available here. The dataset is composed of training and validation images with their corresponding annotations. Please refer to src/data/README.md for the dataset strcuture explanation.

Unzip and place the downloaded data in your Assignment 2 folder, the codebase is organized as followings.

```
├── dataset
│   ├── annotations        # annotations in json file
│   ├── train              # training images
│   ├── val                # validation images
│   └── test               # testing images
└── src
    ├── data
    │   ├── dataset.py      # dataloader
    ├── model
    │   ├── block.py        # backbone
    │   ├── head.py         # detection head
    │   └── ADdetector.py   # object detection network
    ├── utils
    │   ├── loss.py
    │   └── util.py
    ├── train.py            # model training
    ├── predict.py          # model inference
    └── eval.py             # performance evaluation
```

2.1 Build and Train Your Detection Model

Task 1: Filling in the Backbone with ResNet

You should fill in two blanks in file src/model/block.py to complete the basic blocks of backbone network, including the block design and forward function.
You should try to apply the ResNet into the Backbone.
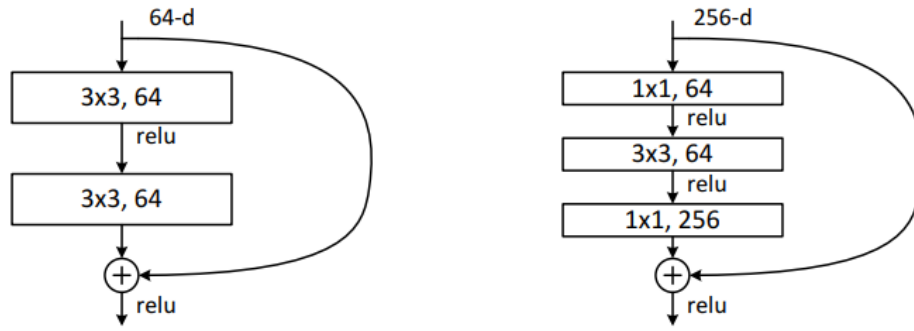The ResNet architecture is as the following.

Figure 2. A Visualization Illustration of Basic Block and Bottleneck Block in ResNet.

Task 2: Filling in the Detection Head

You should fill in one blank in file src/model/head.py.

Task 3: Filling in the Network

You should fill in three blanks in file src/model/ADdetector.py.

Task 4: Filling in the Object Detection Loss Function

You should fill in three blanks to complete object detection loss in file src/utils/loss.py. The object detection loss includes five terms as following (which is Equation (3) in YOLO v1). You are required to complete two terms of them.

$$
\begin{aligned}
loss = &\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2
\end{aligned}
$$

To help you better understand and implement the loss function, we can split this equation into four parts. You will find it helpful when you do the task 4.

- The first two terms forms the location loss,

- The third term is the IOU loss for boxes containing the objects,

- The fourth term represents the loss for boxes not containing the objects (Not IOU or No Objects),

- And the last term is the classification loss.

Hint: In implementation, the fourth term is split into two part: (Part I) Not response loss: for the box containing the object but not the one with the maximum iou; (Part II) No onject loss: for the box not containing objects.

Task 5: Filling in the Training Pipeline

We have implemented dataset preprocessing codes for you. We strongly suggest you to read the src/data/dataset.py carefully before completing the following tasks.
You should fill in two blanks to complete the training pipline in file src/train.py.

After completing the training pipeline, you can run with the following command.

```
python src/train.py --output_dir 'checkpoints'
```

Task 6: Filling in the Prediction Pipeline and Non-Maximum Suppression

You should fill in two blanks to complete the prediction pipeline and the NMS in the file src/utils/util.py.
Non Maximum Suppression is a computer vision method to select a single entity from many overlapping entities. For more details, refer to nms algorithm.

After completing the inference pipeline, you can run in the following command. The visualization result will be saved in vis_results folder. And you should show some of your results in the final report.

```
python src/predict.py --image_path "./dataset/val/xxxx.jpg" --vis_dir "./vis_results"
```

Task 7: Filling in the Evaluation Pipeline (To Understand mAP)

You should fill in one blank to complete the mAP calculation in file src/eval.py.
The mAP is one of the most essential evaluation metrics in object detection. For more details, refer to mAP calculation.

After completing the evaluation pipeline, you can run in the following command. **IMPORTANT: You should submit the result.pkl generated in this part. (Submission details are illustrated in Sec 3.1)**

```
python eval.py --split 'test' --output_file "./result.pkl"
```

Hint: please pay attention to the threshold set in the evaluation pipeline. We suggest setting $pos\_threshold=0.1$, **and we will perform lower truncation on** $pos\_threshold$ **(that is, all** $pos\_threshold<0.1$ **will be truncated to** 0.1)

2.2 Submitting the Model Outputs of the Test Data

You should train your model on the **training** set, and valid your model on the **val** set, and you should generate and submit the outputs of the test set with your trained detection model.
test data will be released one day before deadline.

You can generate the final result using the following command. **Please specify the image size here if you change it to a larger scale.**

```
python src/for_submit.py --split 'test' --output_file "./result.pkl" --model_path <your best ckpt
file> --image_size <your image size>
```

Task 8: Improving the Detection Performance

In this task, you should try different techniques to improve the detection performance, such as using grid search for the proper hyperparameters, modifying data preprocessing methods, optimizing the loss for each class, changing the batch size, replacing the optimizer and so on. You should generate the prediction outputs with the trained detection model on the test data. The output organization should follow the form in the next Section.

Bonus Task (10% more beyond full mark): Improving the Detection Performance with More Advanced Architecture

You are encouraged to explore and implement more advanced network architectures on this task. If you have any tries on this part, you should submit all your codes, model output, and the report to explain your understanding about how to improve the detection performance with more advanced architecture. (You should explain why these improvements can enhance the performance in your own words instead of just cloning some codes and performing inference.) Bonus score will be given based on your overall contributions on these three parts.

2.3 Submitting the Reports

Task 9: Explaining the Efforts to Improve the Detection Performance in Experiment Report

You should submit a experiment report to describe how you take efforts to improve the object detection performance. You should also provide the loss curve of your training procedure and figure out when the model converges in the curve. You can further explain whether training epoch and batch size affect the result in this report.

3. Submission Forms and Score Scheme

3.1 Submission File Forms

The submission files should follow the specific organized forms as following. Since TAs have to rerun your code on another test dataset, marks will be **deducted** (multiplied by 80%) if your code cannot be run because your submission of code does not follow our specified instructions. TAs may contact you to solve related problems.

```
└── uid_name_assignment_2 # Your submission file name.
    ├── src                # The completed source codes corresponding to Sec. 2.1. You should keep
the same organization with the original file structure.
    ├── bonus1_src         # The source codes for more advanced architecture if there are any.
    ├── outputs            # Your model outputs on test data.
    │   ├── result.pkl      # It will be generated automatically in src/eval.py. Please follow the
format in source code.
    │   ├── bonus1.pkl      # The results of your more advanced implementation if there are any.
    │   ├── bonus2.pkl
    │   └── ...
    └── report.pdf         # Combine the experiment analysis report and the survey analysis report in
one pdf file.
```

3.2 Score Scheme

The final score of the assignment is composed of four parts: (1) the codebase you complete (40%); (2) the detection performance (40%); (3) the experiment analysis report (20%).

1. For the completed codebase (40%): The mark will be given mainly based on completeness under the requirements.

2. For the performance part (40%): TA will rerun your code on another test dataset (same distribution as training set and verification set) and give your marks based on the mAP of your model.

- mAP larger than 42% will get the full mark of this part.

- mAP between 38% and 42% will get 90% mark of this part.

- mAP between 34% and 38% will get 80% mark of this part.

- mAP between 30% and 34% will get 70% mark of this part.

- mAP between 26% and 30% will get 60% mark of this part.

- mAP larger than 0.2% will get 50% mark of this part.

- Others will get 0% mark.

3. For the experiment analysis report (20%).
   The marks will be given mainly based on the richness of the experiments & analysis.

- Rich experiments + detailed analysis: 90%-100% mark of this part.

- Reasonable number of experiments + analysis: 70%-80% mark of this part.

- Basic analysis: 50%-60% mark of this part.

- Not sufficient analysis: lower than 50%.

4. $\text{Final mark} = min\{100, (\text{sum of the THREE parts} + \text{bonus mark})\}$

4. Working on the Environment

Standard codes have been tested with the following settings. It's just a reference for you, and there is no need for you to run in the completely same environment.

- python 3.9
- pytorch 1.9.1
- torchvision 0.10.1
- numpy
- scipy
- tqdm
- opencv-python

Computing Resource Reference: Training on a single NVIDIA RTX 3090 GPU with a batch size of 16 (use less than 12GB VRAM) takes less than ~2 minutes per epoch.

5. Important Date

- Assignment 2 Release: Nov 17 (Mon)
- Test Data Release: Nov 28 (Fri)
- Submission Deadline: Nov 30 (Sun)