ECSE 415 - Introduction to Computer Vision
Assignment 4: Object Recognition

DEADLINE: Monday, 17 November 2025, 11:59 PM

Please submit your assignment solutions electronically via the **myCourses** assignment dropbox. The submission should include a **single** Jupyter notebook (**.ipynb** file) and a **report** as a separate **.pdf** file. More details on the submission format can be found below. Submissions that do not follow the format will be penalized by 10%. Attempt all parts of this assignment. The assignment will be graded from a total of **100 points**. You can use **PyTorch**, **TenserFlow**, **OpenCV**, **Matplotlib**, **Scikit-Image**, and **Numpy** library functions for all parts of the assignment. Students are expected to write their own code. You may use any tools for your assignments, but you are responsible for any misrepresentation, inaccuracy, or plagiarism. Citations of non-existent material or obvious factual inaccuracies may result in no credit. (Academic integrity guidelines can be found **here**). Assignments that are handed in late without permission will receive a penalty of 10% per day.

Note: Members within each group will receive the same score for the assignment.

Submission Instructions

On myCourses:

- Submit a single **Jupyter notebook** consisting of the solution of the entire assignment and a **report** as a separate pdf file.
- Comment your code appropriately.
- Give references for all codes which are not written by you. (Ex. the code is taken from an online source or from tutorials)
- Do not forget to run **Markdown** ('Text') cells.
- Do not submit input/output images. The output images should be displayed in the Jupyter Notebook itself.
- Make sure that the submitted code runs without errors. Clearly describe any specific requirements for executing the code in your notebook.
- If you use external libraries in your code, please specify their names and versions within the notebook.

On Kaggle:

- Submit the predicted bounding boxes and class labels as a **CSV** file to the competition.

Preliminaries

1. **Group Enrollment:** Enroll in a group of up to 2 people on myCourses to access the assignment files.
2. **Join the Kaggle Competition:** Join the competition through this link. (If you don't have a Kaggle account, you will need to create one first.)
3. **Team Naming:** Use your myCourses group number as your Kaggle team name (e.g., *Group 3*).

Objective

The goal of this assignment is to develop and evaluate an object recognition model on a road sign detection dataset, with the objective of gaining practical experience in applying deep learning methods to object recognition tasks.

Dataset Description

Image Files: The Road Signs Detection dataset contains multiple categories of traffic signs (e.g., Speed Limits, Stop, Red Light, Green Light, etc.). Images are in `.jpg` format with a resolution of 416x416 and may contain one or more traffic signs per image.

Annotation Files (Labels): Each image has a corresponding YOLO-format text file (`.txt`) containing one or more object annotations per line. Each line represents a detected traffic sign with the following format:

```
<class_id> <x_center> <y_center> <width> <height>
```

All coordinates are normalized (i.e., values between 0 and 1). For example:

```
6 0.45 0.52 0.12 0.20
```

This means:

- **6**: Class index corresponding to a specific sign (e.g., "Stop").
- **0.45, 0.52**: Normalized x- and y-coordinates of the bounding box center.
- **0.12, 0.20**: Normalized width and height of the bounding box.

This dataset contains multiple traffic sign categories. Each class is assigned a unique index, which is used in the YOLO annotation files. The mapping of class indices to class names is as follows:

Class Indices and Names:

0: Speed Limit 80	**1**: Speed Limit 50	**2**: Green Light	**3**: Speed Limit 90
4: Speed Limit 40	**5**: Speed Limit 120	**6**: Stop	**7**: Speed Limit 60
8: Speed Limit 70	**9**: Speed Limit 20	**10**: Speed Limit 110	**11**: Red Light
12: Speed Limit 30	**13**: Speed Limit 100		

Dataset Split: The dataset is divided into **train** and **test** sets:

- **Train:** Contains both images and their corresponding label files.
- **Test:** Contains only images (no labels), used for competition submission.

Dataset Download: Download the dataset from the *Kaggle competition page* and unzip it into your working directory. The expected structure is as follows:

```
traffic_signs/
    train/
        images/    # Training images
        labels/    # YOLO-format annotation files
    test/
        images/    # Test images (no labels)
```

1 Data Preparation (10 points)

1.1 Explore the Dataset:

- Download the dataset from Kaggle and unzip it.
- Display 5–10 sample images from different classes, overlaying the bounding boxes and class labels.
- Plot a bar chart showing the number of instances per class.
- Identify any class imbalance and discuss how it might affect model training in your report.

1.2 Preprocess the Data:

- All images are already standardized at a 416x416 resolution. You may optionally resize them to a smaller resolution (e.g., 256x256 or 128x128) to reduce computational cost and accelerate training, while maintaining aspect ratio if possible.
- Normalize pixel values to the range [0, 1] or [-1, 1] depending on your model requirements.
- Split the dataset into training and validation subsets (e.g., 70% training, 30% validation), while preserving class balance in each subset.

2 Baseline Methods (10 points)

2.1 Implementation of Baseline Object Recognition

Implement a baseline object detection method using **YOLOv8**. Follow these steps:

- **Model Initialization and Fine-tuning:** Start with a pre-trained YOLO model (trained on a large dataset such as COCO) and fine-tune it on the training subset of the provided dataset. Adjust hyperparameters such as learning rate, batch size, number of epochs, and input image resolution as needed.
- **Evaluation Protocol:** Evaluate your fine-tuned model on the validation set using the following metrics:
  - **F1 score:** Compute per class and report the average.
  - **mAP@50:** Mean Average Precision at IoU threshold 0.5.
  - **mAP@50–95:** Mean Average Precision averaged across IoU thresholds from 0.5 to 0.95.
  - **Confusion Matrix:** Show class-level misclassifications.

2.2 Analysis of Baseline Results

After evaluating the baseline model, perform the following:

- **Visualize Predictions:** Display bounding box predictions along with class labels on a subset of validation images to assess localization and class accuracy.
- **Report Metrics:** Present all evaluation metrics (F1 score, mAP@50, mAP@50–95) in tables, including per-class and overall averages. Also, add the confusion matrix to your report.
- **Comment on Performance:** Describe any common errors, such as missed or misclassified objects, and discuss how the baseline performance compares with your custom model from Section 3.

3 Model Implementation (30 points)

3.1 Choose a Model:

Implement an object recognition model suitable for traffic signs other than YOLO, such as Fast R-CNN, RetinaNet or an alternative (**with references**).

3.2 Optional Data Augmentation:

Apply augmentations (e.g., rotations, flips, zoom, color jitter) to improve model generalization.

3.3 Train the Model:

- Train the model on the training set using appropriate loss functions and optimizers.
- Track training and validation loss over epochs and monitor model accuracy.

4 Model Evaluation (10 points)

4.1 Evaluate Performance

Assess the model on the validation set using F1 score, mAP@50, and mAP@50–95. Additionally, generate a confusion matrix to visualize class-level misclassifications.

4.2 Visualize Results

Display a selection of 5–10 validation images showing the original image along with predicted bounding boxes and class labels for visual assessment.

4.3 Generalization Test

To evaluate how well the model generalizes beyond the dataset, test it on an external image of a traffic sign that belongs to one of the 14 classes (e.g., taken personally or from online sources). Include the image, the predicted bounding box and the class label in the report as qualitative examples (note that these do not affect the Kaggle competition score).

5 Prediction & Kaggle Competition (10 points)

5.1 Generate Test Predictions and Prepare Submission File

Use your trained model to predict bounding boxes and class labels for all test images in the Road Signs Detection Dataset. Save the predictions into a single CSV file suitable for Kaggle submission. The CSV should contain:

- Columns: `ID, class_label, x_center, y_center, width, height`.
- Each row's `ID` should correspond to `imageID_idx`, where `image_ID` is the image ID and `idx` identifies multiple objects in the same image.
- `class_label` should contain the predicted class ID.
- `x_center, y_center, width, height` should contain normalized bounding box coordinates.
- Ensure all IDs from `sample_submission.csv` are included, even if no objects are detected; otherwise, Kaggle will raise a "number of rows" error during submission.

5.2 Kaggle Submission

- Submit your CSV file to Kaggle to see your public leaderboard score, which is calculated on 30% of the test images. Final rankings will be based on the remaining 70%, revealed after the competition closes.
- **Submission Limit:** Each team can submit up to 5 times per day, so plan accordingly.

Grading: Your grade for this section will be determined by your ranking, with the top 3 teams receiving 10 points, and the lowest rank receiving 0.

6 Report (30 points)

Prepare a separate PDF report summarizing your findings, including detailed analysis, visualizations, and critical discussion.

6.1 Introduction

- Describe the Road Signs Detection Dataset, including the number of images, classes, and typical object sizes.
- Explain the object recognition task and its relevance to real-world applications and practical use cases.
- Summarize dataset exploration results:
  - Number of images and total objects in the dataset.
  - Displayed sample images with overlaid bounding boxes and class labels to illustrate the dataset.
  - Bar chart showing the number of instances per class, highlighting any class imbalances.
  - Discussion of how observed class imbalances might affect model training and performance.

6.2 Model Architecture

- Describe the architecture of your chosen model and any pre-trained weights used.
- Include references to papers or documentation supporting the model choice.
- Detail input preprocessing, augmentation techniques, and any hyperparameters used for training (learning rate, batch size, number of epochs, image size).
- Explain how predictions are generated (bounding box format, class probabilities) and any post-processing steps.

6.3 Results

- Present quantitative evaluation metrics:
  - Classification metrics: F1 score (per class and overall).
  - Localization metrics: mAP@50, mAP@50–95.
  - Confusion matrix to visualize class-level misclassifications.
- Include visual examples of predicted bounding boxes overlaid on images, highlighting both correct detections and errors.
- Compare your model's performance to the YOLO baseline, commenting on improvements or weaknesses.
- Assess generalization by presenting qualitative results on out-of-dataset or external images.
- Optionally, include plots of training/validation loss, precision-recall curves, or IoU distributions to support your analysis.

6.4 Challenges and Solutions

- Describe difficulties encountered during data processing (e.g., inconsistent annotations, class imbalance, image corruption).
- Discuss challenges during model training (e.g., overfitting, slow convergence, memory limitations).
- Explain problems during evaluation or submission (e.g., mismatched CSV format, missing IDs, incorrect bounding box normalization).
- For each challenge, describe the solution or workaround implemented, and any lessons learned.