EEL 5840/EEE 4773 Summer 2022
Fundamentals of Machine Learning
Final Project

Title: Traffic Sign Classification

Project Due: Monday, August 1, 2022, 11:59 PM
Group Size: up to 4 individuals
Material Due: final report and code implementation

1. Description

In the final project, you will develop a machine learning system to classify traffic signs. The data set is to be collected by all students enrolled in this course. You can implement this system yourselves or using a package/library. You can use any packages that come as a default option with Anaconda or PyTorch. *I need to be able to run your implementation on my machine. So, be sure to get approval from me for any special packages! If I cannot run the code, you will lose a significant number of points.*

2. Data Set

Each group will collect part of the training set that everyone will use to train their machine learning models. There's a total of 10 traffic signs (10 classes). Figure 1 shows examples of each class:

We will use the following integer encoding for each class:

Each student will collect pictures of a total of 10 trials per traffic sign, giving a total of 100 images per student. So, for a group with 4 members, there should be a total of 400 images.

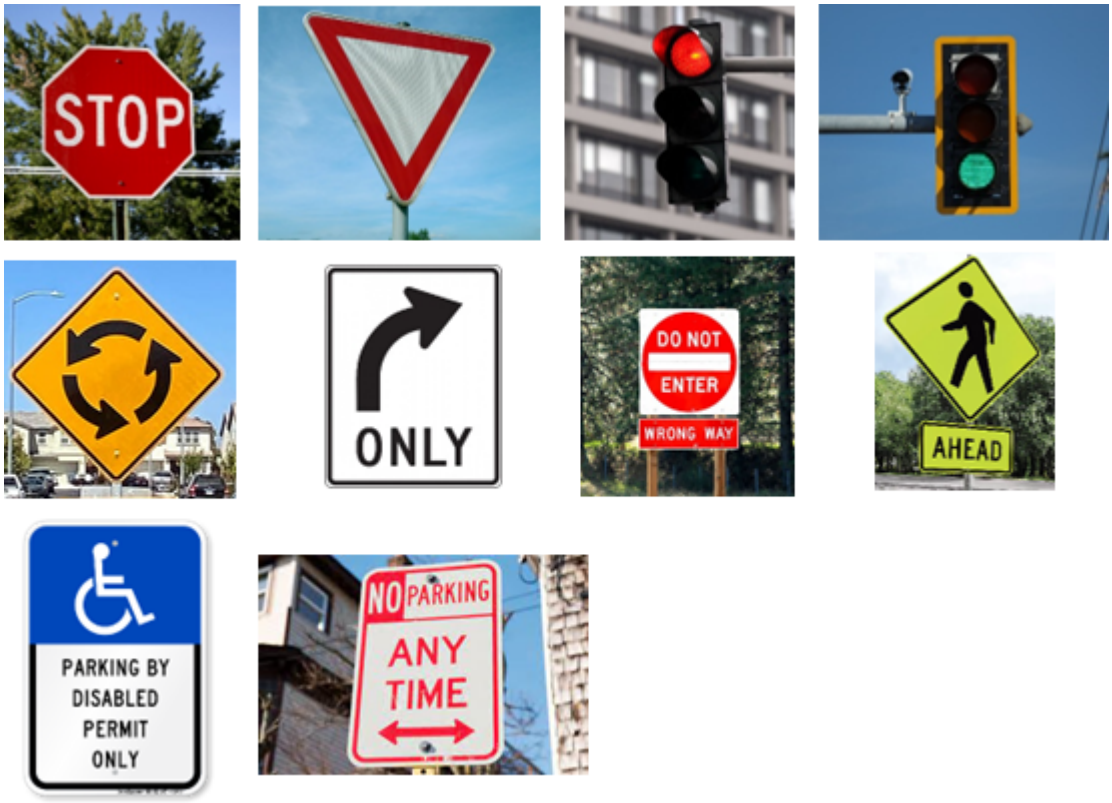

Figure 1: List of all traffic sign classes.

Flower Species	**Label**
Stop	0
Yield	1
Red Light	2
Green Light	3
Roundabout	4
Right Turn Only	5
Do Not Enter	6
Crosswalk	7
Handicap Parking	8
No Parking	9

Table 1: Integer encoding for each traffic sign class.

2.1. Coding System

I recommend you to save your files using a coding system, e.g. ID-trial-label.

- First give a number from 1 to 4 to each team member, this is the ID. Then, for example, when team member with ID 4 is recording hers/his 5th picture of class 2, the file name should read "4-5-2.jpg".

After collecting all the data from all teams, I will merge it and randomly partition it into a **training set** (about 70%) and an ***easy* test set** (about 30%). You will all be given the same **training set** to fit your model. I will hold the ***easy* test set** until after you submit your code implementation and report. This blind test set will be used for grading.

I will also create a separate ***hard* test set** that will contain images from the classroom but also include other images with characters or other objects outside the provided labels. This textbf*hard* test set will be used for extra credit contest - see details below.

3. Project Report

You should write a report that includes the sections listed below. Your report should follow the IEEE transactions format (single spaced, double column).

- You can find a (word doc or LaTeX) template for the IEEE transactions format here: `https://www.ieee.org/conferences/publishing/templates.html`

Focus your report on your training and testing strategies for the contest and any unique implementations. The **maximum number** of pages for the report is 4. If there are any pages beyond page 4, they will be discarded and not read or graded. It should be written with correct English grammar and spelling. Be precise - use pseudo-code or equations to be precise.

For full credit consideration, your report should include the following sections:

- *Abstract.* A summary description of the contents of the report and your findings.
- *Introduction.* Overview of your experiment/s and a literature review. For the literature review, include any references to any relevant papers for your experiment/s. So, whatever you decide to do, search the ACM

and IEEE (or other) literature for relevant papers to read and refer to.

- *Implementation.* Describe and outline any specific implementation details for your project. A reader should be able to recreate your implementation and experiments from your project report. If you participate in the extra credit contest, be sure to describe the methodology on how you will identify unknown classes that were not in the training data.

- *Experiments.* Carefully describe your experiments with the training data set and any data augmentation set you constructed or existing data sets. Include a description for the goal of each experiment and experimental findings. This is the bulk of what you will be graded on - if your experimental design is not sound or your experiments do not make sense, you will lose points.

- *Conclusions.* Describe any conclusions or things you learned from the project. Your conclusions must follow from what you did. Do not copy something out of a paper or say something that has no experimental support in the Experiments section.

- *References.* Listing of all references in IEEE bibliography format.

When writing the report as a group, I recommend you to use **Google Docs** using your UFL account. This way you can all make synchronous and simultaneous edits in your project report. Of course, you should always keep the latest version of your report in your GitHub repo.

4. Project Code Implementation

You can use any packages that come as a default option with Anaconda or PyTorch. *I need to be able to run your implementation on my machine. So, be sure to get approval from me for any special packages! If I cannot run the code, you will lose a significant number of points.*

You can implement your algorithm using Jupyter Notebook or your favorite integrated development environment (IDE). Your final code submission should contain 3 files:

- README file - directly edited in your GitHub team repository

- train.py or a Notebook (.ipynb) with a function "train". This function should contain the code used to train your final model.
- test.py or a Notebook (.ipynb) with a function "test". This function should receive data and labels in the same format as the training data and output an accuracy value and the predicted labels.
- if you compete in the contest, you can create a separate file for testing on the hard test set (or include it in test.py). This function should receive data and labels in the same format as the training data and output an accuracy value and the predicted labels.

5. Grading Details

Your grade will be determined using the following breakdown:

- 10% Data collection
  - You will be graded on data collection. Each person should collect 100 pictures with the specifications mentioned in section 2.
- 25% Implementation
  - Turn in code that runs correctly and easily on my machine. This requires a very clear README and easy to modify parameter settings. This also requires clearly listing what packages/libraries are needed to run your code - and checking with me before the due date to ensure I have those libraries.
  - Turn in code that follows the submission requirements described above
- 25% Accuracy on "easy" blind test data set
  - The "easy" test set is composed of the held-out blind test set with all labels.
  - Your code should produce the integer-coding label encoding defined in Table 1.
  - Full points on this component will be obtained if you correctly classify 90% of the blind test data or have a classification accuracy rate greater than the average classification accuracy rate of the class (whichever is lower).

- 40% Project report
  - This component will be graded based on the requirements outlined in section 3.

6. Extra Credit Contest

The goal of this project is to implement an end-to-end system to perform traffic sign classification. The teams with the best classification accuracy on the "hard" data set will earn extra credit. The "hard" data set will also have all 10 classes (labels 0, 1, 2, 3, 4, 5, 6, 7, 8, 9) and class unknown (label -1). There will be test data points from classes that do not appear in the training data. So, you will want to come up with a way to identify when a test point class is "unknown" or was not in the training data. The label you should return for this case is -1.

Please have your test function output a class label that matches the class value in the provided training data. These should be: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 and -1.

7. Submission Details

Turn in your project report and your code on your group GitHub repository on **Monday, August 1 at 11:59 PM**. In Canvas, you should submit your GitHub **AND** the project report.

Be sure your repository contains the following files: **train.py** (includes a *function* that will run your training code on an input data set X and desired output vector Y. Any parameter settings must be easy to find and modify.), **test.py** (includes a *function* that will run your testing code on an input data set X. Note: Your test.py code should already be trained and have parameters set! Any parameter settings must be easy to find and modify. It should return a vector with the class label associated with each input data point X) and a concise **README.txt** file that clearly illustrates how to run your code. Your classification accuracy on a small test data set will factor into your project grade.