EEL5840/EEE4773 Fall 2024
Fundamentals of Machine Learning
Final Project

Title: Handwritten Digit Object Detection

Project Due: Wednesday, December 4, 2024, 11:59 PM
Group Size: up to 3 individuals
Material Due: 4-page final report and code implementation deployed to a GitHub repository

1. Description

In the final project, you will develop a machine learning system to detect multiple objects (handwritten digits) within a set of input images. 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, TensorFlow or PyTorch. *We need to be able to run your implementation on our machines. So, be sure to get approval from us for any special packages! If we 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 system. There's a total of 10 digits (10 classes), $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Each input image can have up to 3 handwritten digits.

Each student is responsible for collecting a total of 100 images: 30 images with one handwritten digit, 30 images with two digits, and 40 images with 3 digits. A group with 3 members will have a collective of 300 images.

Use the Notebook file "Digit Generator.ipynb" to randomly generate a list of digits that you will hand-write and take pictures of. Once you have this list of randomly generated digits, get started on handwriting them on **different** paper pages (unruled paper, ruled paper, grid paper, different color paper, or a combination of these is fine). Use your mobile phone to take pictures of each page. Each group member will collect 100 images in a folder called "images_member_i", where $i = \{1, 2, 3\}$ corresponds to an arbitrary member ID. The figure below shows examples of collected images.

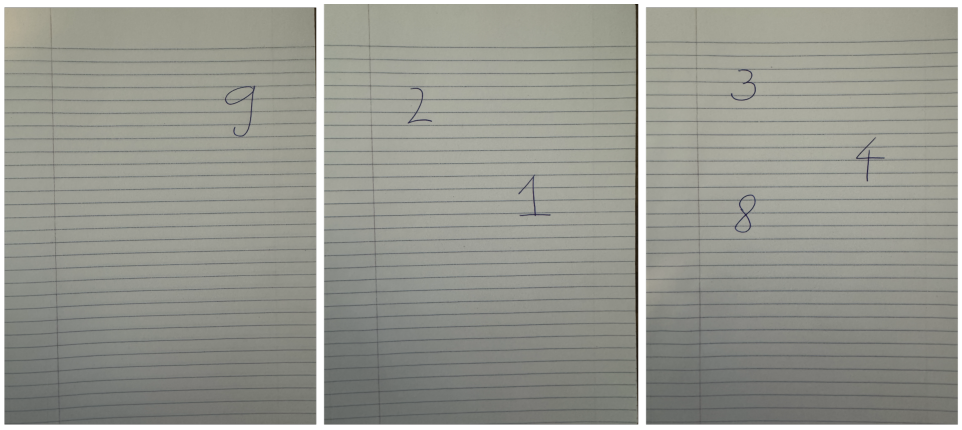

Figure 1: Example of three (3) collected images using a mobile phone.

Now, you will use the Notebook file "Final Project - Data Collection.ipynb" to resize your images as 640×640 RBG images. Follow instructions in the Notebook file to create a folder called "resized_images_member_i", where $i = \{1, 2, 3\}$ corresponds to the ID set from earlier.

2.1. Data Annotation with makesense.ai

Once you have your resized images folder, you will create a new project with `https://www.makesense.ai/`.

1. Click on "Get Started" (bottom right corner).
2. Import all resized images from your "resized_images_member_i" folder.
3. Select "Object Detection".
4. Click on "Load labels from file", and import the file "labels_file.txt".
5. Click "Start Project".

6. For each image in the dataset, use the cursor to draw a bounding box around the digit, and select the appropriate label on the right panel for each digit. Repeat this process for all images in the dataset (total of 100 per member).

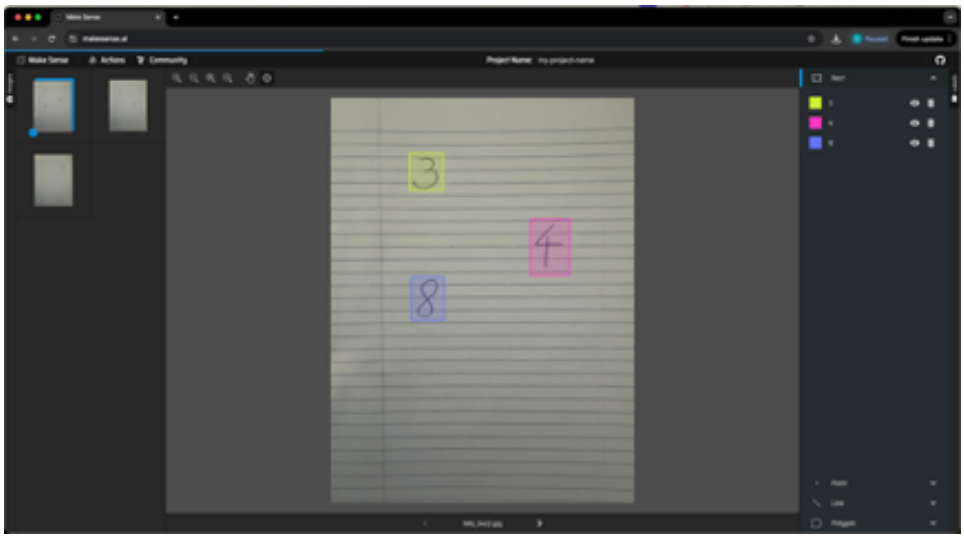

Figure 2: Example of bounding box label annotation.

7. Once all images are annotated, click on "Actions" (top left corner) followed by "Export Annotations".

8. Export annotations as as zip folder in YOLO format. Unzip this folder and rename it to "dataset_member_i", where $i = \{1, 2, 3\}$.

Go back to the Notebook file "Final Project - Data Collection.ipynb" and follow instructions for generating your group's "dataset.zip" and "resized_images.zip" files.

2.2. Training Dataset

After collecting all the data from all teams, we 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 train your system/s. We will hold the ***easy* test set** until after you submit your code implementation and report. This blind test set will be used for grading.

We will also create a separate ***hard* test set** containing images with unknown classes. This ***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, LaTeX or Overleaf) 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, we recommend you to use **Google Docs** using your UFL account. This way, you can all make synchronous and simultaneous edits in your project report.

4. Project Code Implementation

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

Your final code submission should contain 3 files:

- README file - directly edited in your GitHub team repository. A template will be provided.

- 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 the predicted labels and a metric score value.

- 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 the predicted labels and a metric score value.

5. Grading Details

Your grade will be determined using the following rubric:

- 10% Data collection

  - You will be graded on data collection. Each group member should collect 100 pictures with the specifications mentioned in section 2. However, the data collection milestone will be graded as a team submission. A team of 3 members is expected to submit 300 *unique* images and their (*correct*) annotations. A team of 2 members, 200 images and annotations. An individual working individually, 100 images and annotations.

- 25% Implementation
  - Turn in code that runs correctly and easily on our machines. 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 us before the due date to ensure we have those libraries.
  - Turn in code that follows the submission requirements described above.

- 25% Jaccard Index on "easy" blind test data set
  - The "easy" test set is composed of the held-out blind test set with all annotations.
  - Your code should produce the bounding box locations and their corresponding class label.
  - Full points on this component will be obtained if the Jaccard index, also known as the Intersection over Union (IoU), on the blind test data is at least 0.75 or at the average Jaccard index score 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 machine learning system to perform handwritten digit object detection. The teams with the best Jaccard index score on the "hard" data set will earn **extra credit**. The "hard" data set will also have all 10 digits (labels 0, 1, 2, 3, 4, 5, 6, 7, 8, 9) and an *unknown* class (label -1). There will be test images that will contain

handwritten objects that do not appear in the training data. So, you will want to come up with a way to identify when an object is "unknown" or was not in the training data. The label you should return for this object is -1.

Please have your test function output bounding boxes for all objects and the corresponding object labels 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 **Wednesday, December 4 at 11:59 PM**. In Canvas, you should submit your GitHub URL **AND** the project report.

Be sure your repository contains the following files: train.py, test.py, README.txt, and any saved models that will be needed when running test.py.