CISC3024 – Pattern Recognition Project

Classification of Street View House Numbers Using PyTorch

Presentation will be hold on Thursday November 7, 2024
Final report and slides are due to Friday 23:59 pm November 8, 2024

Introduction

The objective of this course project is to apply deep learning techniques to train a classification network on the **Street View House Numbers (SVHN)** dataset (http://ufldl.stanford.edu/housenumbers/). The SVHN dataset is a real-world image dataset obtained from house numbers in Google Street View images. It consists of over 600,000 labeled digit images.

This project will guide you through the process of data preparation, model implementation, training, evaluation, and analysis. By the end of the project, you will have a trained neural network capable of classifying digits from images, along with insights into how different hyperparameters affect model performance.

Task Description (100%)

1. Data Processing and Augmentation (10%)

- **Objective:** Utilize PyTorch APIs to prepare the SVHN dataset for training and testing. Implement image augmentation techniques to enhance the dataset.
- **Requirements:**
    - Load the SVHN dataset using PyTorch's `torchvision.datasets`.
    - Implement custom image augmentation using **Albumentations** library. The augmentations should include (Note: during testing, you should not use image augmentation other than the normalization):
        - **Image Rotation:** Randomly rotate images within a specified angle range.
        - **Random Cropping:** Randomly crop images to simulate zoom-in effects.
        - **Aspect Ratio Change:** Alter the aspect ratio to simulate different viewing angles.
    - Normalize the images and convert them to PyTorch tensors.
    - Ensure that the augmentation parameters (e.g., max rotation, min cropping size, max aspect ratio change) are adjustable through the class's `__init__` method.
    - The augmentation functions that you may need to use are:
        - `A.RandomResizedCrop`
        - `A.Rotate`

        - `A.Normalize(mean=[0.4377, 0.4438, 0.4728],std=[0.1980, 0.2010, 0.1970])`
        - `A.pytorch.ToTensorV2() # if you want to convert to pytorch tensor within the augmentor`

2. Neural Network Setup (10%)

- **Objective:** Build a neural network model based on the architecture provided in Lecture Code 5.
- **Requirements:**
    - Define the neural network architecture using PyTorch's `nn.Module`.
    - Include layers such as convolutional layers, activation functions, pooling layers, and fully connected layers as per the lecture example.
    - Ensure the network is suitable for classifying images into 10 classes (digits 0-9).

3. Coding Training and Evaluation Functions (10%)

- **Objective:** Implement functions to train the neural network and evaluate its performance on the test dataset.
- **Requirements:**
    - Write a training loop that:
        - Processes input batches (putting this part in the data loader is also fine).
        - Performs forward and backward passes.
        - Updates model weights using an optimizer (e.g., SGD, Adam).
        - Tracks and displays training loss.
    - Implement an evaluation function that:
        - Sets the model to evaluation mode.
        - Calculates accuracy on the test dataset.
        - Computes the Receiver Operating Characteristic (ROC) curves and Area Under the Curve (AUC) metrics.
        - Handles both macro and micro ROC AUC calculations.

4. Analysis of Training and Evaluation (20%)

- **Objective:** Analyze how different hyperparameters or network designs affect the model's behavior and performance. Provide quantitative results and visualizations.
- **Requirements:**
    - **Hyperparameter Tuning:**
        - Experiment with different hyperparameters such as learning rate, batch size, number of epochs, optimizer types, and augmentation parameters.
        - Record how changes in these hyperparameters impact the training process and model performance.
    - **Evaluation Metrics:**

        - Report the model's accuracy and ROC AUC scores (both macro and micro).
        - Provide confusion matrices if possible.
    - **Visualizations (50% of this section, i.e., 10% of the whole task):**
        - Include plots such as training loss curves, ROC curves, and bar charts of class-wise performance.
        - Use tables to summarize quantitative results across different experiments.
    - **Analysis Depth:**
        - If only one experimental setting is presented, a maximum of 5% (out of 20%) will be awarded for this section.
        - To receive full credit, analyze multiple settings and discuss the observed behaviors and outcomes.
    - **Report Findings:**
        - Interpret the results and explain why certain hyperparameters may have led to better or worse performance.
        - Discuss any challenges faced during training and how they were addressed.

5. Extra Credit: Interactive Number Recognition with Gradio (8%)

- **Objective:** Create an interactive user interface using Gradio that allows users to draw a digit and have the model classify it. (Note: you may need to save your model's trained weights to the disk so that during the presentation you can directly show the interactive classification demo)
- **Requirements:**
    - Implement a Gradio interface using the `Sketchpad` component.
    - The interface should:
        - Allow users to draw digits using a mouse or touchscreen.
        - Display the model's classification probabilities for each digit (0-9).
    - Ensure the input from the sketchpad is properly preprocessed to match the training data format.
    - Show your demo during your presentation to get the credit (The model needs to make the correct prediction. Too many wrong predictions will decrease your extra credit.)

Note: Completing this task will add 2% to your final course score after adjusting for the project's weight in the final grade.

Report and Presentation (50%)

- **Report Writing (20%):**
    - **Structure:**

        - Methodology: Explain the data processing steps, model architecture, training procedures, and evaluation methods.
        - Results: Present the quantitative results along with visualizations.
        - Discussion: Analyze the results, discuss the effects of different hyperparameters/network design, and reflect on the model's performance.
        - Conclusion: Summarize the key findings and suggest possible improvements or future work.
    - **Formatting:**
        - Use clear headings and subheadings.
        - Include figures, tables, and charts where appropriate.
        - Have proper citation of any external resources or references.
    - **Clarity and Depth:**
        - Write clearly and concisely. Please use ChatGPT or other LLM to polish your language to ensure your report is readable!
- **Presentation (30%):**
    - **Duration:** Prepare a 5-minute presentation.
    - **Content:**
        - Summarize your approach and key aspects of your implementation
        - Present your main findings and analyses.
        - Include visual aids like slides, diagrams, and charts.

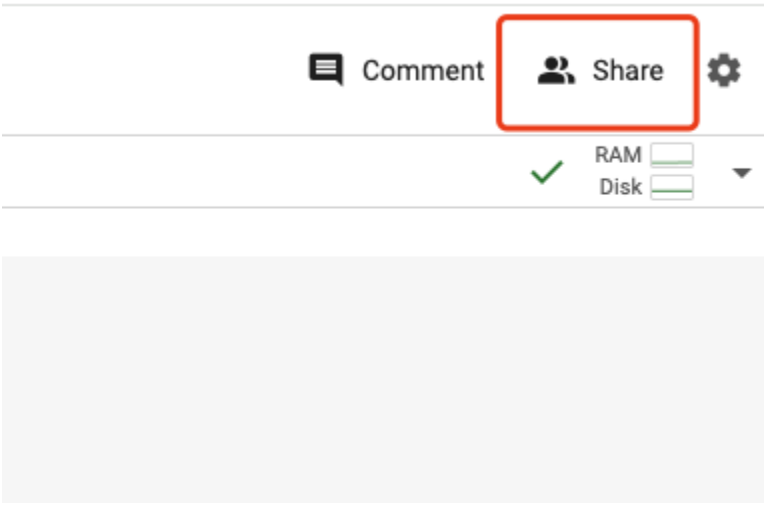
Submission Guidelines

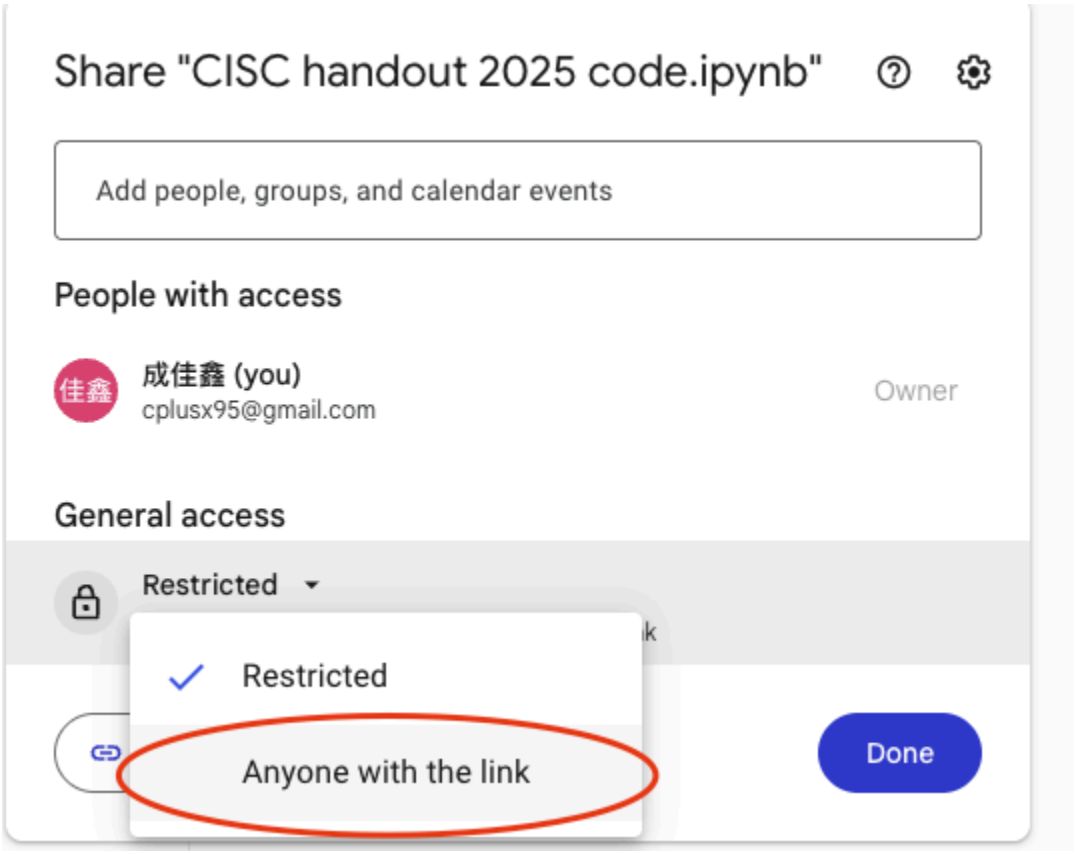
- **Deadline:** Submit all materials before the specified deadline on UMMoodle.
- **Materials to Submit:**
    - **Online Model Link:**
        - Provide a link to your Jupyter Notebook hosted on an accessible platform (e.g., Google Colab, GitHub). See below content for sharing from Google Colab. For the GitHub link, ensure that your repository is visible from the public.
        - Ensure that all code cells are executed and outputs are visible.
    - **Report:**
        - Submit a PDF version of your report.
        - Ensure all figures and tables are properly formatted and legible.
    - **Presentation Slides:**
        - Submit your presentation slides in PDF or PowerPoint format.
- **Submission:**
    - You can have at most TWO students in a team. (You can also finish the task alone.)
    - Each student should submit his/her final project to the UMMoodle even if you are working with another student. The grading will be the same for both students in the group.
    - Clearly list all team members' names and student IDs on the report and presentation.

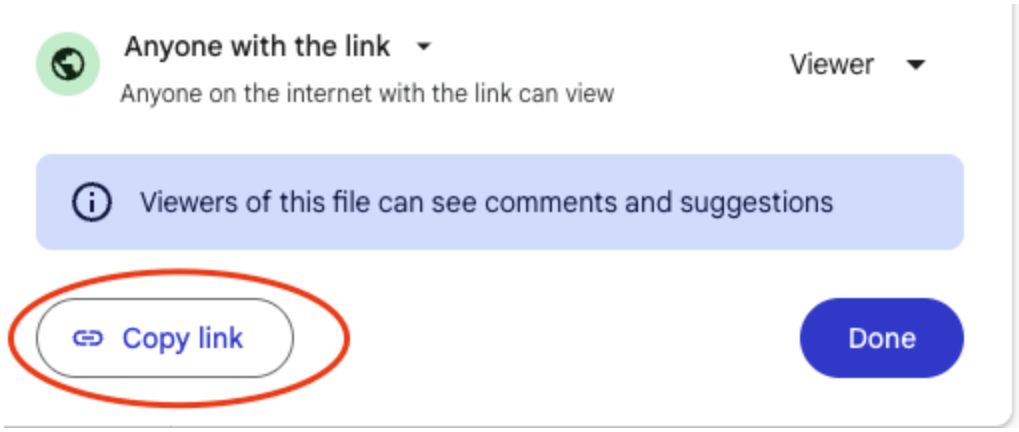
- **Presentation:**
    - Each team will present their project in class.

Sharing from Google Colab

Turn on the link share of your Colab Notebook and paste the link in your report.







Grading Reminder

The following are the grading guidelines:

1. Data Processing and Augmentation (10%)

2. Neural Network Setup (10%)

3. Coding Training and Evaluation Functions (10%)

4. Analysis of Training and Evaluation (20%)

- **Hyperparameter Tuning and Analysis (10%)**
- **Visualizations (10%)**

  Note: Insufficient analysis (e.g., only one experiment setting) will result in a maximum of 5% for this section.

5. Report Writing (20%)

6. Presentation Preparation (30%)

Extra Credit: Interactive Number Recognition with Gradio (8%)