Take-Home Quiz 1

This quiz is due on Monday, Nov. 7 2022 at 11:59pm.

Problem Statement

Image classification is a supervised learning problem: define a set of target classes (objects to identify in images), and train a model to recognize them using labeled example photos. In this problem, you will use the dataset which is a computer-vision-based Printed Circuit Board (PCB) analysis dataset used for classification of defects. It contains two subsets of 12,200 PCB images, captured under representative conditions using a professional camera.

The quality of PCBs will directly impact the performance of many electronic devices. To avoid the shortcoming of manual detection, easily being fatigued, low efficiency, the automated optical inspection (AOI) based on machine vision has been widely used in industry. In this problem, you're challenged to build a multi-class classification model that's capable of detecting different types of defects on these PCBs better than the baseline performance.

Training Dataset

The training dataset consists of 12,000 110x42 color images containing one of 8 defect classes, with unequal class distribution (different amounts of images per class).

Evaluation Dataset

The evaluation dataset is captured under a different machine from the one used for capturing the training dataset. It consists of 200 110x42 color images containing one of 8 defect classes. Your model is expected to be robust to different machines.

Problem Rules

There are general rules valid for take-home quiz can be found here.

- Use of any external data is not allowed.
- Manipulation of provided training dataset is allowed (e.g., by using data augmentation techniques such as flipping or rotation).

- Participants are not allowed to make subjective judgments of the evaluation data, nor to annotate it. The evaluation dataset cannot be used to train the submitted model; the use of statistics about the evaluation data in the decision making is also forbidden.
- Classification decision must be done independently for each test sample.

Baseline Performance

Submissions are evaluated on an averaged F1 score. Your grade on this Quiz will depend on the **F1 score** your model received from the Evaluation Dataset.

Submission File

Once you have finished your model, you need to follow the instructions below to submit your work in order to get the corresponding performance on the Evaluation Dataset (we'll try to release the performance on Portal once a week):

1. For each `ImageName` in the evaluation dataset, you must predict a type of defect (i.e., `label`).
2. Please generate a `.txt` file called `studentID.txt`. The file should contain a header and have the following format:

```
ImageName,label
a762df180,0
24c5cf439,0
7581e896d,0
eb4b03b29,0
etc.
```

3. Generate a zip file of your training and testing codes (`.py` or `.ipynb`) with model weight (`.pth`, `.h5`, or etc.) and `.txt` file called `studentID.zip`.
4. Submit the zip file to Portal.

Software Setup

Working remotely on Google Colaboratory

Google Colaboratory is basically a combination of Jupyter notebook and Google Drive. It runs entirely in the cloud and comes preinstalled with many packages (e.g. PyTorch and Tensorflow) so everyone has access to the same dependencies. Even cooler is the fact that Colab benefits from free access to hardware accelerators like GPUs (K80, P100) and TPUs which will be particularly useful for our take-home quiz.

Requirements. To use Colab, you must have a Google account with an associated Google Drive. Assuming you have both, you can connect Colab to your Drive with the following steps:

1. Click the wheel in the top right corner and select `Settings`.
2. Click on the `Manage Apps` tab.
3. At the top, select `Connect more apps` which should bring up a `GSuite Marketplace` window.
4. Search for **Colab** then click `Add`.

Best Practices. There are a few things you should be aware of when working with Colab. The first thing to note is that resources aren't guaranteed (this is the price for being free). If you are idle for a certain amount of time or your total connection time exceeds the maximum allowed time (~12 hours), the Colab VM will disconnect. This means any unsaved progress will be lost. **Thus, get into the habit of frequently saving your code whilst working on assignments.** To read more about resource limitations in Colab, read their FAQ here.

Using a GPU. Using a GPU is as simple as switching the runtime in Colab. Specifically, click `Runtime -> Change runtime type -> Hardware Accelerator -> GPU` and your Colab instance will automatically be backed by GPU compute.

If you're interested in learning more about Colab, we encourage you to visit the resources below:

- Intro to Google Colab
- Welcome to Colab
- Overview of Colab Features

Working locally on your machine

If you wish to work locally, you should use a virtual environment. You can install one via Anaconda. Ensure you are using Python 3.7 as **we are no longer supporting Python 2**.

Anaconda virtual environment

Note: If you've chosen to go the Anaconda route, you can safely skip this section and move straight to Installing Packages.

We strongly recommend using the free Anaconda Python distribution, which provides an easy way for you to handle package dependencies. Please be sure to download the Python 3 version, which currently installs Python 3.7. The neat thing about Anaconda is that it ships with MKL optimizations by default, which means your `numpy` and `scipy` code benefit from significant speed-ups without having to change a single line of code.

Once you have Anaconda installed, it makes sense to create a virtual environment for the course. If you choose not to use a virtual environment (strongly not recommended!), it is up to you to make sure that all dependencies for the code are installed globally on your machine. To set up a virtual environment called `cs660`, run the following in your terminal:

```
# this will create an anaconda environment
# called cs660 in 'path/to/anaconda3/envs/'
conda create -n cs660 python=3.7
```

To activate and enter the environment, run `conda activate cs660`. To deactivate the environment, either run `conda deactivate cs660` or exit the terminal. Note that every time you want to work on the assignment, you should rerun `conda activate cs660`.

```
# sanity check that the path to the python
# binary matches that of the anaconda env
# after you activate it
which python
# for example, on my machine, this prints
# $ '/Users/gpuuser/anaconda3/envs/cs660/bin/python'
```

You may refer to this page for more detailed instructions on managing virtual environments with Anaconda.

Installing packages

Once you've **setup** and **activated** your virtual environment (via `conda`), you should install the libraries needed to run the assignments using `pip`. To do so, run:

```
# again, ensure your virtual env
# has been activated before running the commands below
# install quiz dependencies.
# since the virtual env is activated,
# this pip is associated with the
# python binary of the environment
pip install SomePackage
```

You may refer to this page for more detailed instructions on how to install Python packages.

Asking Questions on Piazza

As you work through the CS660 take-home quiz, many of you will probably have questions or will encounter issues that you are not sure how to deal with. Piazza provides a convenient mechanism to ask questions, but it can sometimes be challenging to provide assistance if you do not provide the right information when asking a question. Here are a few suggestions that may help you ask questions more effectively on Piazza:

- Before you post a question...
    - **Search before asking**. Before posting a question on Piazza, check whether it has already been answered in a previous post. We realize the volume of posts can be overwhelming, but you should start by using Piazza's search functionality to see if it brings up any relevant posts. For example, suppose you are getting an `IndexError`; you could search just for that word to see if any other students have encountered that same error (and, if you're lucky, an instructor/TA will have already answered it).
    - **Make sure to always check the "pinned" posts**. We will often "pin" a post titled "Must read posts for [assignment name]" with links to posts that address common issues and questions on that assignment (the pinned posts appear at the top of the left sidebar, under "Pinned"). So, make sure you check that post first.
    - **Make sure you're going to ask an actual question**. You need to tell us about a specific issue you're encountering, and why you're stuck on it (e.g., you are not getting the expected result, the tests are failing in a way you do not understand, etc.). Writing a post that says "*I can't get quiz 1 to work, I've pushed my code. Please look at it.*" is not a question! (please note that we're happy to help you work through a task you're having trouble with and may ultimately suggest that you come to office hours for this but, on Piazza, you have to make sure you're asking a specific question).
    - **One question, one post**. Avoid posts that have multiple unrelated questions. Instead, write a separate post for each question. Please note that it is ok to ask multiple questions in one post if they all relate to the same issue.
- What *not* to include in your question:
    - **Never post your code in Piazza**. As noted in Academic Honesty policies, you should never share your code with other students, which means you should never post it on Piazza. If you need us to look at your code, just push it to the Git server and we will look at it there. Please note that, if a test prints out a few lines of code as part of its output, that's ok.

- **No screenshots**. Do not post screenshots of the output. Screenshots are not searchable, and may pose readability issues for some people. Instructors/TAs may also want to copy-paste that output somewhere else, which is not possible if you post a screenshot.

  If you need to share some output with us, copy-paste from the terminal onto Piazza, and use Piazza's "code block" formatting. To copy something on the terminal, just select it (the same way you would do in a word processor: click, and then drag until the end of the output) and press Control-Shift-C.