

CMPUT 328 Fall 2020

Assignment 6

Classification and Detection on MNIST Double Digits Dataset

Overview

In this assignment, you are going to do classification on the MNIST Double Digits (MNISTDD) dataset. This dataset contains gray scale images of size 64×64 . Each image has two MNIST digits (from 0 to 9) randomly placed inside it as in this visualization:



Figure 1: Visualization of the first 64 images in the MNISTDD training set. Each image is 64×64 . Please note that the digits in these images are not taken directly from MNIST, but rather generated by a Generative Adversarial Network trained on it.

Two digits in an image may partially or completely overlap each other as shown in Figure 1 (for example, see the last image in 3rd row) though complete overlap only happens in a small fraction of images. Most images do not have digits overlapping or only partially overlapping. Two digits in an image may also be of the same class.

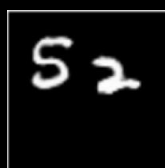
Your task in this assignment is to tell which digits are contained in each image and where are they located.

Dataset

The MNISTDD dataset is divided into 3 subsets: train, validation and test containing 55K, 5K and 10K samples respectively. A sample consists of:

- **Image:** A 64×64 image that has been vectorized to a 4096-dimensional vector.
- **Labels:** A 2-dimensional vector that has two numbers in the range $[0, 9]$ which are the two digits in the image. Note that these two numbers are always in ascending order. For example, if digits 7 and 5 are in an image, then this two-vector will be $[5, 7]$ and not $[7, 5]$.
- **Bounding boxes:** A 2×4 matrix which contains two bounding boxes that mark the locations of the two digits in the image. The first row contains location for the first digit in labels and the second row for the second one. Each row of the matrix has 4 numbers which represent [row of the top left corner, column of the top left corner, row of the bottom right corner, column of the bottom right corner] in this exact order. Note: it is always the case that row of the bottom right corner - row of the top left corner = column of the bottom right corner - column of the top left corner = 28. This means that each bounding box has a size of 28×28 no matter how large or small the digit inside that box is.

As an example, consider the following 64×64 image that contains the digits 5 and 2:



- Image will be the above image but flattened to a 4096-dimensional vector.
- Labels will be $[2, 5]$
- Bounding boxes will be a matrix with the first row: $[11, 27, 39, 55]$ and the second row: $[6, 2, 34, 30]$. This means that the digit 2 is located between the 11th and 39th rows and between 27th and 55th columns of the image. Same goes for digit 5 - it is located between the 6th and 34th rows and between 2nd and 30th columns of the image.

Each set comprises 3 `numpy` files which can each be read using `numpy.load` to obtain the corresponding matrix stored as a `numpy.ndarray`. Following are detailed descriptions of the 3 files where {SET_NAME} denotes the name of the subset (train, valid or test) and N is the number of samples:

- `{SET_NAME}_X.npy`: 2D matrix with dimension $[N, 4096]$ and containing the vectorized images. Each row is a single vectorized image.
- `{SET_NAME}_Y.npy`: 2D matrix with dimension $[N, 2]$ and containing the labels. Note that the labels are always in ascending order in each row.
- `{SET_NAME}_bboxes.npy`: 3D matrix with dimension $[N, 2, 4]$ and containing the bounding boxes. For more information, see the description of bounding boxes above.

For example, following are the dimensions of the `numpy.ndarray` in 3 files of the train set:

- `train_X.npy`: $[55000, 4096]$
- `train_Y.npy`: $[55000, 2]$
- `train_bboxes.npy`: $[55000, 2, 4]$

Task

You are provided with the train and valid sets that can be downloaded from e-class in a zip file named `MNISTDD_train_valid.zip`. This contains 6 files: `train_X.npy`, `train_Y.npy`, `train_bboxes.npy`, `valid_X.npy`, `valid_Y.npy`, `valid_bboxes.npy`.

The test set is NOT released.

You are also provided two python files: `A6_main.py` and `A6_submission.py`. The latter has a single function `classify_and_detect` that you need to complete. It takes a single matrix containing all test (or validation) images as input and returns two numpy arrays `pred_class` and `pred_bboxes` containing the classification labels and detection bounding boxes respectively in the same format as described above.

To reiterate, the two digits in each row of `pred_class` must be in ascending order and the two rows in `pred_bboxes` corresponding to that image must match this ordering too.

`A6_main.py` can be run to evaluate the performance of your method as in terms of the classification accuracy (% of digits classified correctly) and Intersection over Union (IOU) of the detection boxes with the ground truth boxes.

You are free to add any other functions or classes you need including those in other files imported from `A6_submission.py`. Just make sure to submit all files needed to run your code.

You can use any machine learning method to solve this problem. There is no restriction or guideline as to what algorithm you can or should use. The only constraint is that image information must be used in some **meaningful way** for both classification and detection. Simply returning random or fixed values in either of the arrays will disqualify the submission.

You do not need to include training code in your submission.

Marking

This assignment uses **relative marking** where all qualifying submissions will be ranked by each of the following metrics separately:

1. Classification Accuracy (higher is better)
2. Detection IOU (higher is better)
3. Test Runtime (lower is better)

A linear scaling from 50 – 100 will then be used for lowest to highest ranked submissions to generate a score for each metric. The overall score will be the average of the three.

There are no documentation marks for this assignment.

What to Submit

You need to submit a **single zip file** containing the modified `A6_submission.py` along with any other files or checkpoints needed to run it on the test set. Testing will be done by running the main function in `A6_main.py` and it is your responsibility to ensure that your code can be imported and run without errors. Apart from correct functioning, there are two more qualifying performance criteria for inclusion in the ranking process:

1. Classification accuracy and detection box IOU should both exceed 50%
2. Test time should not exceed 2000 seconds on Colab CPU.

Any submission not satisfying either of these will get **no marks** for this assignment.

Check before submitting:

All submissions will be used in an automated evaluation system and any failure to run, for whatever reason, will be heavily penalized and might lead to the submission being disqualified from ranking and getting a zero.

- Make sure that your code **runs without errors on Colab whether GPU is enabled or disabled**
- Remove any Colab specific lines (e.g. mounting Google drive) or any other code that can produce an error when it is imported.
- **Do not submit** `A6_main.py` or any of the `npz` files. Your code needs to run without requiring any changes in `A6_main.py`.
- **Do not rename** `A6_submission.py` to anything else (including changing any part of it to upper or lowercase).
- Training and validation files will not be available during testing so make sure to remove any code that loads these.
- Set paths for loading weights (if any) **relative to** `A6_submission.py` in your submitted zip file.
- Make sure to keep the testing **batch size small enough** to not run out of memory on Colab.
- All included files and folders must be in the root of the zip file (instead of being contained inside another folder).

- Submit a zip file instead of any other archive format (e.g. rar, tar or tar.gz)
- Eclass has a maximum upload size limit of 400 MB so your submission must be small enough to fit in this size. Google Drive (or any cloud) links for any part of it will not be accepted.

Submission deadline is **November 27, 11:59 PM**