Final Project – CMPUT 328
Updated Oct 6th
Classification on MNIST Double Digits dataset

A. OVERVIEW:

In this final project, you are going to do classification on the MNIST Double Digits (MNISTDD) dataset. The MNISTDD dataset contains gray scale images of size 64x64. Each image has two MNIST digits (from 0 to 9) randomly placed inside it like in this visualization:

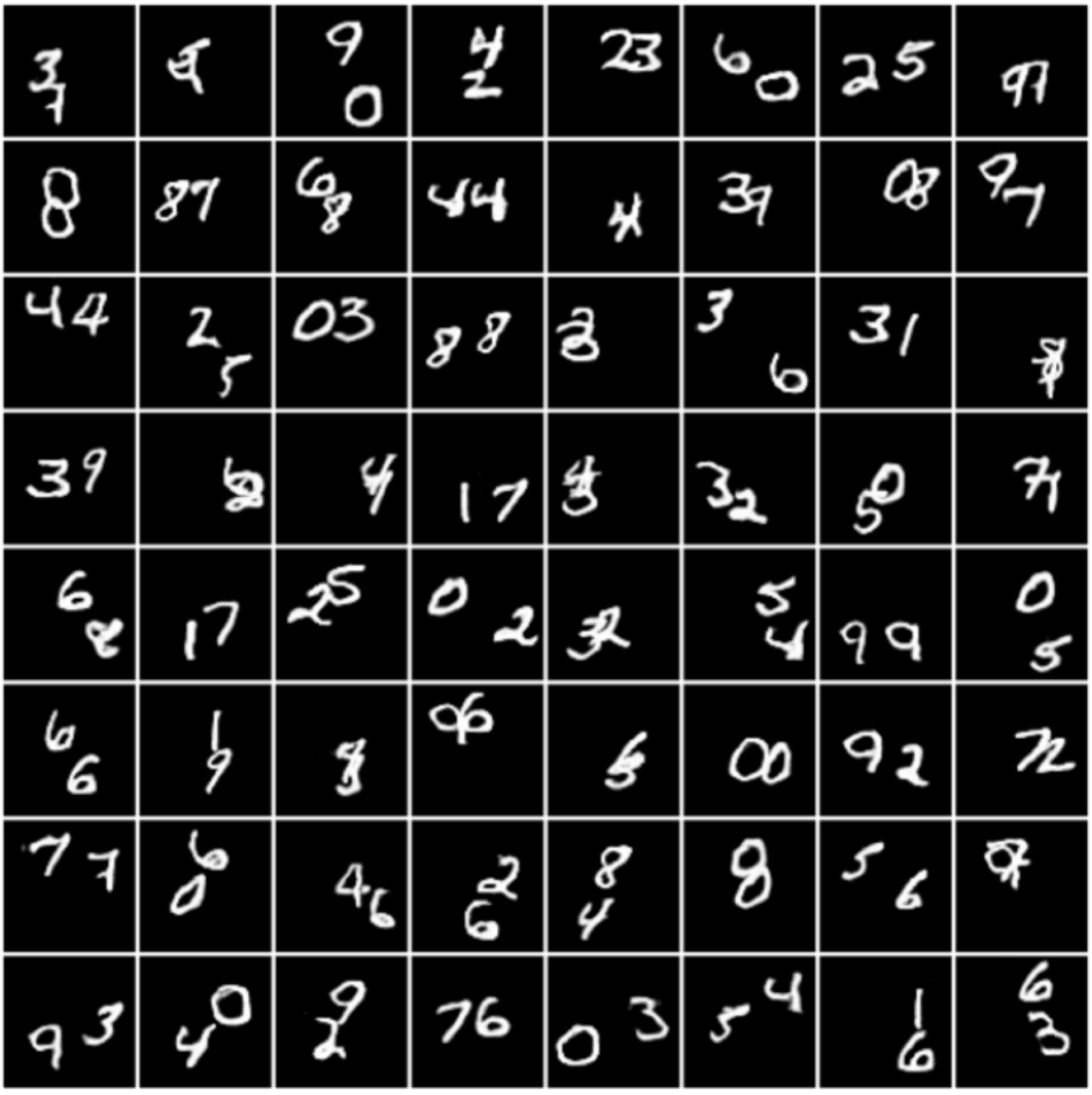

Figure 1: Visualization of the first 64 images in the MNISTDD training set. Each image is 64x64. Please note that the MNIST digits in those images are not taken directly from MNIST dataset, but rather are generated by a Generative Adversarial Networks (GANs) trained on it.

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

Your task in this assignment is to tell which digits are contained in each image.

B. DATASET:

The MNISTDD dataset is divided in to 3 smaller subsets: train, validation and test. Each of the subset contains many samples. A sample consist of:

- **Image**: A 64x64 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 number are always in ascending order.** For example, if digits number 7 and 5 are in the images then this two vector will be [5, 7] and not [7, 5].
- **Bounding boxes**: A **2-by-4 matrix** which contains two bounding boxes that mark locations of two digits in the image. The first row contain location for the first digit in the 2-dimensional vector in **labels** and the second row for the second one. Each row of the matrix has 4 numbers which are **[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 the 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 all bounding boxes will have a size of 28x28 no matter how large or small the digit inside that bounding box is.
  The bounding boxes are provided in case you want to do object detection for the final project (i.e. you detect the location of two digits first then do extra steps to know what kind of digits they are). **You are NOT required to process the bounding boxes in the train or validation set unless you want to and you DO NOT have to compute the bounding boxes for the test set.**

As an example, consider the following 64x64 image that contains two digits 5 and 2:

- Image will be the above image but flattened to a 4096-dimensional vector.
- Labels will be [2, 5]
- Bounding boxes is a matrix that has the first row: [11, 27, 39, 55] and the second row: [6, 2, 34, 30]. which mean the digit 2 located between the 11th and 39th row and between 27th and 55th column of the image. Same goes for digit 5. It is located between the 6th and 34th row and between 2nd and 30th row of the image.

The numbers of sample contained in each set are the same as in the MNIST dataset:

- train: 55000 samples
- valid: 5000 samples
- test: 10000 samples

Each set will have 3 .npy files associate with it. These file can be read using **np.load()**. The contents inside these files are all numpy.ndarray. Let ${SET_NAME} be the name of the subset (train, valid or test). Detailed descriptions of the 3 files are shown below

- **${SET_NAME}_X.npy**: 2d matrix contains the vectorized images. Has dimension of [num_samples, 4096]. Each row is a vectorized image.
- **${SET_NAME}_Y.npy**: 2x matrix contains the labels. Has dimension of [num_samples, 2]. Note that in each row, the labels are always in ascending order.
- **${SET_NAME}_bboxes.npy**: 3d tensor contains the bounding boxes. Has dimension of [num_samples, 2, 4]. For more information, see the description of bounding boxes above.

For example, dimension 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]

You will be given the **train** and **valid** set. You can download both sets in a zip archive directly from e-class (i.e. the file "MNISTDD_train+valid.zip"). Extract this archive, you will find 6 files named: train_X.npy, train_Y.npy, train_bboxes.npy, valid_X.npy, valid_Y.npy, valid_bboxes.npy. The **test** set will **NOT** be released.

C. WHAT TO DO:

1. Two labels classification from an image in MNISTDD dataset:

The problem in this final project is: Given images in the MNSITDD dataset, tell us the two digits that are present in each of these images.

You can use any machine learning method to solve this classification problem. There's no restriction or guideline to what algorithm you can or should use. As long as your code provide a way for us to compute the two digits. You can see the file **mnistdd_classify.py** for a very simple reference. Your code must have a function that:

- **Receive**: an input numpy.ndarray of the test set that has size [NUM_SAMPLES, 4096]. Each row of the array is a 64x64 image that has been vectorized.
- **Return**: a 2D numpy.ndarray that has size [NUM_SAMPLES, 2]. Each row of the returned array contains 2 numbers from 0 to 9 that are the digits in the corresponding image. **Please note that two digits in each row must be in ascending order.**

Note: You don't have to do your training inside this function. You can train your model beforehand, save the model and only use this function to compute the labels using your saved model. This will save you a lot of time during testing and more importantly, may very well help your code to finish before the time penalty described in section E.

2. Assessing the performance:

The returned 2D numpy.ndarray will be compared with our ground truth labels for all image in the test set which is also a 2D numpy.ndarray with size [NUM_SAMPLES, 2]. When a single row your labels are compared with the corresponding row our ground truth labels, there are two kind of matching:

- Complete match: both digits are correct. → You get **3 points** for this kind of match.
- Partial match: only one digit is correct. → You get **1 point** for this kind of match.
- Incorrect match → You get 0 points.

This kind of uneven point system is to reward algorithms that work well in difficult images that have two digits overlap partially or completely.

Your points will be summed across all rows into a final score. Finally, your group performance will be ranked by this score. Depend on your performance, you will get different mark for the **Algorithm** part

of marking. Group with highest standing will be given highest mark.

D. WHAT TO TURN IN:

You must send the code to us by **December 6th, 2017** either by email or USB. This includes your code, your trained model (if applicable) and a note that tell us **how to run your code to get the output labels from the input test array. This note is a MUST**. You would also have to do a final presentation on this day. All members of your team must be there there and present. **If any team member doesn't present on that day, they won't get any mark for the final project.** You can give us your code at the presentation as well.

On **December 8th, 2017** , you must send a report contains details about the machine algorithm you used to solve the problem in this final project: which algorithm, details of the algorithm, how did you apply it to this problem, what kind of accuracy you are getting on train/validation set, etc... Also, a table that show the roles of everyone on your team in this project.

E. MARKING RUBRIC:

1. Mark breakdown:

There are two parts:

- **Algorithm (25%)**: Depend on the performance of your group compared to other groups, you will be given mark for this part accordingly. Please see section C for the point system to rank the groups.

How many mark each group will get for their standing in this competition? TBD

- **Report (5%)**: You have to turn in a report described in section D.

2. Time penalty:

Your code to compute the labels from the test images should run faster than a threshold.

Threshold for this project: **The code must finish before 7200 time units**.

A time unit is determined as the number printed out by the file "**time_test_file.py**" that can be downloaded from e-class. Note that for this number to be correct, **you mustn't run anything else on your PC when you run this file**.

Penalty when your code run past the time threshold: For every **72 time units** past the threshold, you will lose **1%** of your mark in the **Algorithm** part.

3. Late policy:

For this project, you must submit everything on time.