Assignment 2
COL 870: Deep Learning. Semester II, 2020-21.
Due Date: Thursday May 6, 2021. 11:50 pm.

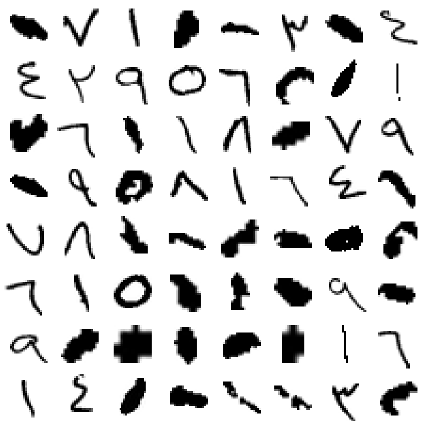
April 15, 2021

In this assignment you will implement a conditional Generative Adversarial Network [Mirza and Osindero, 2014] given a set of character images, with single labeled example from each class. You will also implement a visual Sudoku solver where the characters (digits) in the Sudoku come from the character set used in the conditional GAN problem. The details of the dataset are described in Sec. 1. If you are not sure what Sudoku problems are, you can read the details on wikipedia. Note that in this problem you will be solving (irregular sized) 8×8 sized Sudoku problems, with each mini-block of size 2×4, i.e., spanning a contiguous block of 2 rows and 4 columns.

1 Dataset

You will be given N training samples. Each sample consists of a pair of input and output images. The input is an image of a Sudoku puzzle with numbers 0 to 9 replaced with some hand written character (not necessarily the actual characters $'0', '1', '2', \cdots, '9'$). The grid cells having the character corresponding to digit 0 in the input image denote an unfilled position in the corresponding Sudoku puzzle. Each character occupies 28×28 space in the grid. The output image is of the same size as the input (and with the same number of grid cells), and represents the solved Sudoku puzzle corresponding to the input. Specifically, it is an image where for all the grid cells except for those where input image had the character corresponding to 0 (unfilled positions), are copied with the character present in the input image at the corresponding cell. Each occurrence of the character corresponding to 0 in the input is replaced by an appropriate character such that the output image represents a valid Sudoku board. As stated earlier, do not forget that we are solving irregular sized Sudoku puzzles with 8×8 sized board, and 2×4 sized mini-blocks. Refer to Figure 1 for an example. The dataset for this problem can be downloaded from this link. The folder contains the files as listed below:

- visual_sudoku: This folder has a 'train' folder and inside it there are two


(a) An example of an input image to the visual Sudoku solver, i.e. an image of a Sudoku board. Cells are either filled with the objects from class-1 to class-9, and class-0 to represent unknown.

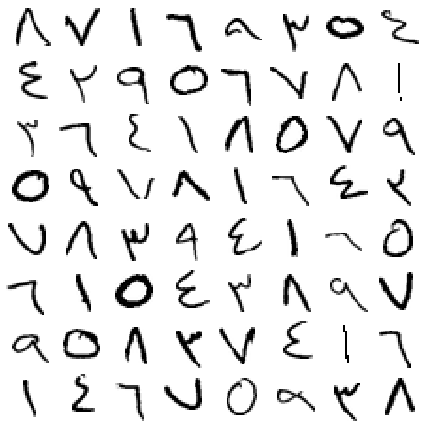

(b) One possible solution or target data of the visual Sudoku solver when fed with input presented in Figure 1a. Class-0 is replaced with an object from appropriate class.

Figure 1: Sample input to the visual Sudoku solver and one possible target corresponding to the input. In both the images, each character occupies a space of 28×28 pixels. Note these are 8×8 Sudoku boards (irregular sized), where each mini-block is of size 2×4 (2 rows, 4 columns). Here is a fun website where you can play with Sudoku puzzles of different sizes

.

folders named ‘query’ and ‘target’. The query folder contains 10000 input images named as ‘0.png’ to ‘9999.png’. Similarly, the target folder contains 10000 target images. ‘i.png’ in the target folder is the solution to the ‘i.png’ image in the query folder $\forall i \in [0, 9999]$

- sample_images.npy: This numpy file contains 10 images, one from each of 10 classes and is of shape (10, 28, 28). The 28×28 matrix in the i^{th} index represents a sample from i^{th}-class.

You are not allowed to use any external data for this problem.

2 Conditional GAN

The space of neural generative frameworks is dominated by Generative Adversarial Networks (GAN) [Goodfellow et al., 2014] and its variants [Arjovsky et al., 2017, Gulrajani et al., 2017, Karras et al., 2018]. Although GAN models can generate realistic novel examples, in a vanilla GAN it is not possible to control the types or class of images that are generated. The conditional generative adversarial network (cGAN) [Mirza and Osindero, 2014], is a modified framework of GAN suitable for conditional generation of images by the generator network. Image generation can be conditioned using the class label, in a supervised manner, allowing the targeted generation of images of a given type. For example, given the class label '6', the model would allow you to generate images corresponding to digit 6 in our set up.

In this part you will implement a cGAN [Mirza and Osindero, 2014]. Input to the generator will be 100-dimensional noise vector concatenated with a 10-dimensional one-hot vector representing target class. For real data you need to extract images from the input sudoku grid. For supervision, you will be provided label corresponding to one image from each of the ten classes. Figure 2 presents the samples of 10 classes from class-0 to class-9 in ascending order (left-to-right). Use the exact architecture, optimizer and hyper-parameter settings as reported in the original paper [Mirza and Osindero, 2014] for MNIST [Lecun, 2010] dataset.

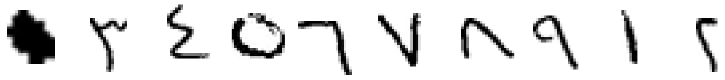


Figure 2: Each image represents one image from each of the ten classes. The left most image represents class-0 and right most image represents class-9.

You need to evaluate your model using the following two metrics.

1. **Generation quality:** Generate 1000 examples from each of the 10 classes and compute FID[Heusel et al., 2017] between the 10k generated samples and 10k real examples extracted from input Sudoku image grid. You may use standard implementation such as pytorch-fid package[1] for FID computation. You can read about the FID score here.

2. **Coherence:** At the end of your submission, we will use a trained classifier to compute the quality of your class conditioned generated images. The conditioning label will be used as the true label for accuracy computation (more on this later).

Hint: Since you are given only a single labeled example from each class, you need to think about how to first generate supervised data for training your conditional GAN model. One idea would be to use a standard clustering technique (such as K-means) to first cluster the character images, and use the single labeled example to label each cluster. This will result in potentially noisy labels, but can be a starting point for training your conditional GAN. Can you think for improved ways of clustering, or other methods to solve this problem where we have very limited training data?

[1]`https://pypi.org/project/pytorch-fid/`

3 Visual Sudoku

In this problem you will design a neural network for solving a combinatorial problem in structured output spaces. However, the training data will not be provided as symbolic data but visual images. Specifically, the training data consists of pairs of input-output images as described in Section 1. At the test time, the input will be an image representing a puzzle, but the output of your model will be the symbolic solution of the puzzle represented by the input image. You will design a visual Sudoku solver i.e. given an image representation of a Sudoku board (as opposed to a one-hot encoding or other logical representation) constructed with 10 different objects, your network should output the solution matrix.

3.0.1 Training Data

Input: Query Sudoku grid as described in Sec. 1. Refer to Fig. 1a.
Target: Target Sudoku grid as described in sec. 1. Refer to Fig. 1b.

3.0.2 Test Data

Input: Query Sudoku grid as described in Sec. 1. Refer to Fig. 1a.
Desired output: $Y \in \mathcal{V}^{8\times 8}, \mathcal{V} = \{1 \dots 9\}$ such that each char is replaced by a digit from $\mathcal{V}$. If a char appears more than once in the input image, it should be replaced by the same digit in Y at all the places. Finally, Y should be a valid Sudoku board.

Note:

1. You may refer to Palm et al. [2018] to get an idea about how to solve Sudoku using Recurrent Relational Networks (RRN).

   Hint: RRNs solve the symbolic Sudoku problem. You will need to extend this to solve the visual Sudoku problem. One naïve way to do this would be to use the labeled data as generated in Sec 2 (conditional GAN), and convert your grid cells to symbolic representation. This will be good as first-cut and is bound to be constrained by noise in your input labels. You should implement this first-cut solution as a starting point. Can you think of improved ways of doing this (see Sec 4 of the problem)?

2. You are not allowed to implement a rule-based Sudoku solver. You may use the rules of the Sudoku to validate if the output of your solver is correct or not (and possibly use it as additional loss). You should treat this as a black box - which takes as input a fully filled Sudoku board, and returns 1 if it is a valid board, and 0 otherwise (no more functionality should be assumed). If doing so, you should clearly mention this in the architecture details in your report.

4 A Combined Solution

In this part, the idea is to devise a method so that the problem statement defined in Sec. 2 and 3 is solved in a joint fashion (like utilizing the constraints of Sudoku to improve conditioning) and see if that improves performance as compared to the solution for Sec. 2 and 3. You can be creative here. We will be happy to discuss more ideas on this on Piazza or in one-on-one interactions once you reach to this part. We strongly suggest that you first implement the first two parts using the Hints provided so that you do not have a cold start.

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Proc. of ICML, 2017.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Proc. of NeurIPS, 2014.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. In Proc. of NeuRIPS, 2017.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In Proc. of NeurIPS, 2017.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In Proc. of ICLR, 2018.

Y. Lecun. The mnist database of handwritten digits. `http://yann.lecun.com/exdb/mnist/`, 2010.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. CoRR, abs/1411.1784, 2014.

Rasmus Berg Palm, Ulrich Paquet, and Ole Winther. Recurrent relational networks. In Proc. of NeurIPS, 2018.