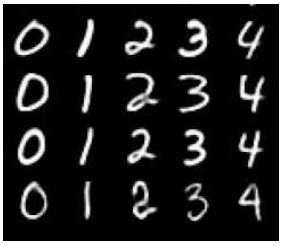
Introduction to Deep Learning - Exercise #3

submission date: 10/1/2021

Programing Task: Autoencoding and Generative models for the MNISTdigit dataset

The dataset consists of 60,000 (+10,000 test) images of scanned hand-written digits (0-9). The dataset contains the digits values as labels. The original images are of size 28-by-28 pixels (but better be resized to 32x32 using zero padding for easier processing). The images are monochromatic, i.e., have a single brightness channel with values between zero (black) and one (white).



In this exercise we design both classification and generative networks over this dataset.

Specific tasks:

1. **Autoencoder (to AE)**. Define a convolutional autoencoder to encode (and decode) the images into a small dimensional latent (around d=10). Explore the use of FC layers at the coarsest level **(i.e. in latent space)** in order to gain a better reconstruction accuracy and the embedded dimension d. Report these tests and the scores obtained. The best practice of implementing this code is by defining Encoder and Decoder modules (torch.nn.Module).

2. **Decorrelation.** In the following experiment we will investigate the connection between dimensional reduction and dependencies (redundancies) in the representation. Do that by computing the Pearson cross correlation between different coordinates of the latent codes (based on a few thousands encoded images), and **use them to** come up with a single value measuring the overall correlation. Plot this value with respect to the latent space dimension d (over at least 4 values of d). Explain the trend you observe and justify it in your report.
3. **Transfer Learning**. Use the pre-trained encoder from the previous section by holding its weights constant, and attaching it to a small MLP (over its output latent space) and train the latter to classify the digits (recall that MNIST is labeled) using a small fraction of the labels in the dataset **(tens of images)**. Compare the performance of this solution next to the option of training the entire network (i.e., allow the encoder weights to train as well) over this small number of **training examples (which is likely to suffer from overfitting…)**. Report all the choices made (e.g., latent space dimension, MLP arch., and number of labels used, etc.), and the results obtained by each of the two training approaches.

4. **Generative Adversarial Networks** (GAN). Implement a GAN *in latent space*, i.e., define an MLP generator network that maps some easy-to-sample distribution into the latet codes you produced above. Once trained, this generator can be combined with the

(pre-trained) decoder to generate novel images. The GAN's latent space can be of an even lower space dimension than the latent codes. Show in your report several sampled digits as well as the following interpolation task. Interpolating between digits can be done by $aL1+(1-a)L2$ where $L1$ and $L2$ are the latent codes (ideally two different digits), and increasing a gradually from 0 to 1. **Show also the results you obtain when interpolating in Q.#1's autoencoder latent space (in which case L1 and L2 should be first generated by two images using the encoder)**. Which interpolation performs better? Explain.

5. **Digit-Specific Generator**. Assume the user would like to specify the digit that will be produced by the generator. In this case, the randomness should apply only to the shape of the chosen digit but not its value. Think of a way to incorporate (as well as train) a user input to control the generator. Training multiple independent generator networks for each digit is not a possibility. Describe your choice and show the variability of the images you get for a particular digit.

We are expecting you to report and elaborate on every practical task in the pdf, with your own words and analysis of what you've done. Include everything that you think is crucial for us to understand your way of thinking.

Theoretical Questions:

1) **Transformers**. CNNs relate nearby values (e.g., pixels) when extracting features at each layer. A transformer layer can relate every pair of tokens. Explain how you would feed an image to a BERT-like arch. such that it will **have the potential** to mimic the action of a convolution operator. Explain how this architecture can be used to mimic an FC layer (or an MLP).

2) **Image Translation**. Generative networks can also be trained to operate over images as an input rather than noise vectors. This way the network can translate images from one class of images, denoted by "A", to another class "B". Consider the following three training strategies:
    a) GAN-like approach: **train** the generator G with images from class A and train the discriminator D to separate G's output from images from class B.
    b) Cycle GAN: train two pairs of generator and discriminator networks such that one pair will map A to B, and the other map B to A. This training uses an additional training loss that requires the composition of the two generators to form an identity mapping.
    c) Conditional GAN: in this strategy the discriminator network needs to distinguish between pairs of images (I,G(I)) where I belong to class A, and pairs of (I,J) where I, again, comes from class A, and J from class B and matches A in some sense. The generator G is therefore required to generate images that appear as if they came from class B and maintain some relation to its input image.

Consider **each** of these cases and explain whether you expect it to applicable over the following datasets:

i) A paired dataset where every image in class A has a counterpart in class B containing the same content but in a different visual style (e.g., photographs and hand-drawn images of the same scene). The correspondence could be very accurate (i.e., all the details in one image are found in its counterpart).

ii) The same as (i) but the correspondence is not as accurate (the drawing can be a loose description of the scene).

iii) Two unpaired datasets containing images of the same topic (e.g., cars) but different visual rendering (e.g., photos and drawings).

In case you find the training strategy applicable to a certain dataset, describe briefly the application.

Submission Guidelines:

The submission is in **pairs.** Please submit a single zip file named “ex3_ID1_ID2.zip”. This file should contain your code, along with an ”ex3.pdf” file which should contain your answers to the theoretical part as well as the figures/text for the practical part. Furthermore, include in this compress file a README with your names and cse usernames.
Please write readable code, with documentation where needed, as the code will also be checked manually.