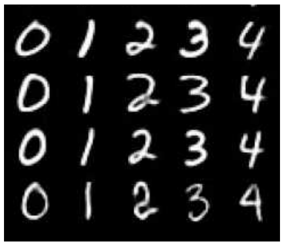
Neural Networks for Images 2023 - Exercise #2

Submission Date - 23/4/2023

Programming Task: Auto-Encoding (AE) and transfer-learning over the MNIST digits dataset



The MNIST 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 you may want to pad them as you see necessary for easier encoding-decoding). The images are monochromatic, i.e., have a single (grayscale) channel.

In this exercise we will explore aspects of auto-encoding and its uses for dimensionality reduction and for transfer-learning.

Practical Tasks:

1. **Auto-Encoding**. Define a convolutional AE to encode the images into (and decode from) an abstract low-dimensional latent space, of dimension d (around d=10) containing no spatial structure. The best practice of implementing this code is by defining an encoder and a decoder as separate Pytorch modules. The encoder may employ an architecture similar to the one you implemented in the previous exercise, but using strided convolutions rather than maxpool operations, and without the final softmax, of course. The decoder should typically mirror the encoder's architecture, but instead of strided convolutions that reduce the spatial resolution you should use strided transpose convolutions (ConvTranspose2d) that increase the resolution, and a final sigmoid activation (to obtain images in [0,1]).
   Your task is to explore the reconstruction error over the test set when (i): using lower and higher latent space dimension d, and (ii) using a fixed latent dimension d but with encoder/decoder architecture with more or fewer layers/weights. Report these tests and the best score obtained (i.e. plot your results as a function of the different experiments performed).

2. **Interpolation.** Once you have trained the AE, you can use it to interpolate between two digits in latent space. That is, let I_1 and I_2 be images of two *different* digits, perform the interpolation $D((E(I_1) * \alpha) + (E(I_2) * (1 - \alpha)))$ for $a \in [0, 1]$ where D denotes the

decoder and E the encoder. (i) Include the resulting images obtained by such latent space interpolation. (ii) Try different pairs of digits. (iii) Try repeating this operation with an AE trained using the higher embedding dimension you tried above. (iv) Which is better? Provide an explanation why the quality increased or decreased.

3. **Decorrelation.** In the following experiment we will investigate the connection between dimensional reduction and dependencies (redundancies) in the representation. Carry this out by computing the Pearson correlaions between different coordinates in the latent codes (based on a few thousands encoded images), and use them to come up with a single value (that you choose) for measuring the overall correlation. Plot this value with respect to the latent space dimension d (over at least 4 values of d). Explain your choices and the trend in correlation versus d that you observe. Provide an explanation in your report.

4. **Transfer Learning**. Use a pre-trained encoder as a fixed network (i.e., exclude its weights from the following training optimization), and attach to it a small MLP (in latent space) and train only the latter to classify the digits (recall that MNIST is labeled). Do this with a small fraction of the annotated training data in the MNIST 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 as the classification MLP) over the same (very small) number of training examples. While both training schemes use the same number of labels, which option performs best? 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.

We are expecting you to report and elaborate on every practical task in the PDF, using 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. Show that the composition of linear functions is a linear function. Show that the composition of affine transformations remains an affine function.

2. The calculus behind the Gradient Descent method:
    a. What is the stopping condition of this iterative scheme,

$$\theta^{n+1} = \theta^n - \alpha \nabla f_{\theta^n}(x)$$

b. Use the second-order multivariate Taylor theorem,

$$f(x+dx) = f(x) + \nabla f(x)\cdot dx + dx^T \cdot H(x)\cdot dx + O(\|dx\|^3),$$
$$H_{ij}(x) = \frac{\partial^2 f}{\partial x_i \partial x_j}(x)$$

to derive the conditions for classifying a stationary point as local maximum or minimum.

3. Assume the network is required to predict an angle (0-360 degrees). How will you define a prediction loss that accounts for the circularity of this quantity, i.e., the loss between 2 and 360 is not 358, but 2 (since 0 is equivalent to 360). Write your answer in a programmable mostly differentiable pseudo-code.

4. Chain Rule. Differentiate the following terms (specify the points of evaluation of each function):

   a.
   $$\frac{\partial}{\partial x} f(x+y, 2x, z)$$

   b.
   $$f_1\Big(f_2\big(\ldots f_n(x)\big)\Big)$$

   c.
   $$f_1\Big(x, f_2\big(x, f_3(\ldots f_{n-1}(x, f_n(x)))\big)\Big)$$

   d.
   $$f\Big(x + g\big(x + h(x)\big)\Big)$$

Submission Guidelines:

The submission is in **<u>singles</u>.** Please submit a single zip file named “ex2_ID.zip”. This file should contain your code, along with an ”ex2.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 compressed file a README with your name, ID, and CSE username.
Please write readable code, with documentation where needed, as the code will also be checked manually.