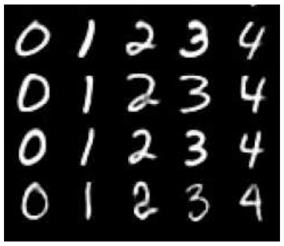
Introduction to Deep Learning - Exercise #2

submission date: 27/6/2024

Programming Task: Classifying and encoding 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. 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 autoencoding and classification networks over this dataset.

Specific tasks:

1. **Autoencoder**(*). Define a convolutional autoencoder to encode (and decode) the images into a small dimensional latent space (around d~12). Explore the use of an FC layer at the coarsest level **(i.e. in latent space)** in order to gain a better reconstruction accuracy and reach the embedded dimension d. Report these tests (input vs. reconstructed images) and the scores obtained. Describe and explain the network architecture you choose for this particular data (#filters / stride factors / #layers / non-linearity etc.). The best practice of implementing this code is by defining separate encoder, decoder and MLP models. Use a mean L1 error to define the reconstruction loss.

2. **Classifier**(*)**.** Use the same architecture as the encoder in #1 in order to define a pre-classification feature extractor and combine it with a small two-layered MLP network to map the latent vector into a 10 classes prediction. Train this network to predict the digit classes using cross-entropy loss. Plot the training and test errors as well as accuracies.

   (*) Note regarding #1 and #2: the unsupervised training in #1 typically requires more epochs to converge than the supervised training in #2. MNIST is a fairly lightweight dataset which will allow you to do both.

3. **Classifier Decoding**. Use the pre-trained classifier encoder network from #2 as a fixed (non-trainable) encoder, connect it with a decoder network (same arch. as the one used in #1) and train the latter to minimize the reconstruction loss. Compare the losses obtained here and in #1 and explain the difference. Moreover, display an array of (>50) reconstructed digits and try to identify differences between the results produced by the AE and classifier-based encoding. In this explanation take into account the following aspects:

a. The different training incentives the two encoders have (the decoders are trained to minimize the same loss) - what is the difference in the latent embedding they produce
b. Where do you see higher in-class (per-digit) variability
c. Where do you see higher intra-class (between digits) distance/separation? Explain both.

4. **Shortage in Training Examples**. Assume we have very few labeled training examples, *only 100* (taken from the training set). Use this small set to train the classifier network (both encoder and classifier MLP). Report the losses and accuracies obtained as function of training time, and whether over-fitting is observed.

   To save you some time, here's how you define a subset of the training set:

```
train_loader = datasets.MNIST("data",
                              train= True,
                             download=True,
                             transform = transform)
indices = torch.arange(100)
train_loader_CLS = data_utils.Subset(train_loader, indices)

train_loader_CLS = torch.utils.data.DataLoader(train_loader_CLS,
batch_size=batch_size,shuffle=True, num_workers=0)
```

5. **Transfer Learning via Fine-tuning.** Use the pre-trained encoder from #1 (which was trained over the entire training data) as an initial guess for a classifier encoding stage along with a randomly-initialized classifier over the 100 examples you used in #4. Plot the losses and accuracies as function of training time and discuss the results obtained, mainly the test loss/accuracy obtained by this and #4 approaches.

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) **LTI**. Show that a convolution with respect to any filter h is time/space invariant.

2) **TI**. Explain whether each of the following layers are time/space invariant or not:
   a) Additive constant
   b) Pointwise nonlinearity (such as ReLU)
   c) Strided pooling by a factor > 1

    d) As a result, is a CNN composed of all these operators (+convolution) time invariant?

3) **Layers' Jacobians**. Calculate the Jacobian matrix of the following layers:
    a) Additive bias vector
    b) General Matrix multiplication (FC)
    c) Convolution layer

Submission Guidelines:

The submission is in **pairs.** Please submit a single zip file named "ex2_ID1_ID2.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 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.
Late submission - 10 points reduction for each day. Submissions will not be accepted after 4 days.