Neural Networks for Images - Exercise #2

submission date: 23/5/2021

Programming Task: Deep Generative Models

In this exercise you will train generative models of various types that reproduce images of a particular distribution. By running several tests you will learn about the strengths and weaknesses of the different approaches as well as explore several applications of such models. As in the previous exercise, in each of these experiments you will need to implement the network or test, rationalize the results and document your conclusions, as to what happened and why, in a pdf report that you will submit (see Submission Guidelines below).

The different tasks described below will require you to set up an encoder and a decoder (generator) network. The encoder is similar in spirit to a classifying network, i.e., it starts with several conv. layers with x2 pooling (or stride), and ends with one or two FC layers that map every image into an abstract representation in a low-dimensional latent space. The decoder or generator network does the opposite, starts with the FC layers, reshapes the vectors into a coarse grid, and applies transpose convolution operations to produce an image with the input dimensions. We suggest using either 2 or 3 conv layers starting with 16 filters, and increasing this number by a factor of two, while applying an x2 pooling. The FC layers should map to a low dim. space (around 10 in case of MNIST, and 48 to 64 in case of CelebA64 dataset). The generator network should have the analog (transposed) dimensions and number of filters. The final step of the generator should apply a sigmoid (or tanh) activation to ensure the images produced are limited to the input range of values (e.g., for example [0,1] or [0,255]).

It should be easier to work with low res. MNIST images, but should be very interesting to see how these networks operate on more meaningful images, such as the CelebA64 images which are 64x64 pixels RGB images. The latter will require adding one more conv. layer in the encoder and decoder networks to handle the higher resolution. The CelebA64 dataset can be obtained from /cs/dataset/CelebA/CelebA64.npy as a single numpy file containing a 4 dimensional tensor with all the CelebA64 images (image_number,width,height,rgb). Use this dataset for at least one of the tasks below.

This exercise has two parts: GANs and non-adversarial training, formally:

Generative Adversarial Networks

1. **Loss Saturation.** Train the generator using a discriminator in the GAN paradigm taught in class. The discriminator should have a similar architecture to the one described above

for the encoder (but mapping into a yes/no decision space). For this purpose explore the case where multiple training optimization steps are applied to the discriminator per a single iteration for the generator. Try three different GAN losses: (i) the original GAN cross entropy, (ii) the non-saturating version of this loss (as mentioned in class), and (iii) a least-squares loss (L2). See which losses lead to a saturation and explain why this is the case.

2. **Model Inversion.** One elementary operation which can be performed as a basis for various applications (such as image editing) is GAN inversion; namely, given an image I, find its source latent vector z that reproduces that image, i.e., maps G(z) = I where G is the generator. This should be implemented on a trained G, and optimized over the z vector. Explain which features in the image are accurately reconstructed, and which aren't. Please provide an explanation of why you think this is the case.
3. **Image Restoration.** Once we have a generator that parametrizes a class of images from a low-dim. latent space, we can use it as a tool for restoring images. That is, given a degraded image, we can search for the best approximating z vector in latent space, i.e., min_z ||G(z)-I|| and since every z in that set in latent space is supposed to produce a "healthy" image, the z found will restore the image. We suggest you try to fill in missing pixels by masking (setting to zero) some portion of the image pixels, and search for the z that best approximates the ones left. Try three different masks, and explain (with images) what cases were easy to the network to restore and what challenged it.

Non-Adversarial Generative Networks

1. **Auto-Encoders.** AEs can also be considered as generative models as their decoder maps vectors in latent space into images. Train an AE to encode the target class of images into the same latent space dimension as you used above when training a GAN. Given an image I, the encoder explicitly provides you its corresponding z code (no need to solve an optimization for this inversion!). Use this to compute two z codes of two different images, and produce the intermediate images (interpolate) by G(z1*a+z2*(1-a)) and use different values of a between 0 and 1 (where G is the decoder). Repeat this experiment with the GAN (compute the z codes by the inversion above). Which interpolation resulted in better looking images? Explain.
2. **Samping novel images.** The latent space produced by the encoder is arbitrary and we cannot sample z vectors from it (hence we cannot produce novel images from the decoder). As a remedy we can "force" the latent space into a known distribution, for example a Gaussian, which will enable us to sample new z codes. In order to achieve this goal train an auto-encoder by optimizing both its reconstruction loss as well as additional losses over the z codes in latent space. These losses should promote a standard Normal distribution in patent space. Try using, (mean - 0)^2, (variance - 1)^2, and (kurtosis - 3)^2 on each axis on latent space for this purpose. Generate novel images by sampling z vectors from a normal distribution and describe the results you get from this approach compared to a GAN training.

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

Theoretical Questions:

1. Each image in ImageNet is labeled to belong to one class out of 1000 classes. How many mathematical constraints each image in this dataset provides when training a network.
2. Assume C(I) is a convolutional layer consisting of the following operation C(I) = I * f , where f is a single filter and I is a monochrome image. Provide a mathematical expression for the gradient of this layer with respect to: (i) the argument I, and (ii) the inner parameters f. What does it require to implement the matrix-vector multiplication of an error term with the Jacobian of the layer as a procedure (and not a costly matrix-vector multiplication)?
3. Read the paper "Large scale GAN training for high fidelity natural image synthesis" (https://arxiv.org/pdf/1809.11096.pdf) and: (i) explain the main architectural changes and training strategies that enable the success of BigGAN. In particular, (ii) explain the meaning and the implications of the "truncation trick"; (iii) explain what are the Inception Score (IS) and the Frechet Inception Distance (FID) metrics that this paper uses to evaluate the generation quality.

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 and analysis for the practical part. In addition, please include in this compressed archive a README with your names and cse usernames.
Please write readable code, with documentation where needed, as the code will also be checked manually.