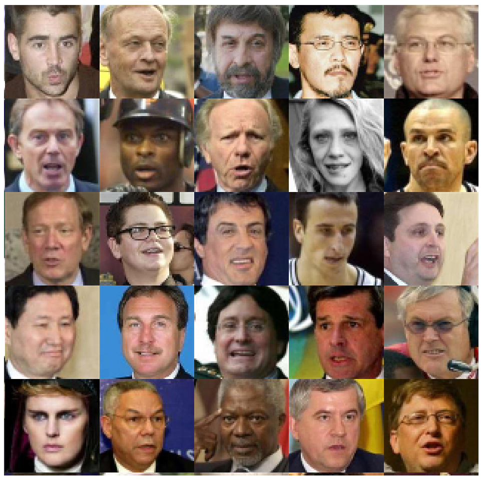
Neural Networks for Images - Exercise #3

Submission Date - 2/6/2022



Programming Task: Deep Generative Models

In this exercise you will train generative models of various types to produce 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 previous exercises, 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. Your report should be submitted as a pdf file, as explained in the Submission Guidelines below.

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

Generative Adversarial Networks (GANs)

The tasks described below will require you to set up a generator and a discriminator. The generator network can follow the DCGAN architecture, namely starts with an FC layer that reshapes the random input vector into a coarse grid with several channels, and applies transpose convolution operations to produce tensors with double spatial resolution and half the number of channels, in each layer, until eventually a 3-channel image of the desired resolution is generated. The final step above should apply a sigmoid activation to ensure the images produced are limited to the input range of values (e.g., [0,1]). The discriminator is similar in spirit to a classification 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 a scalar or 2D decision space.

It should be easiest to work with low res. MNIST images, but should be more interesting to see how these networks operate on more complex and varied images, such as the CelebA64 images which are 64x64 pixels RGB images. The latter will require adding one more conv. layer in the generator and discriminator 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 whatever dataset you feel your GPU resources are fit for.

1. **Loss Saturation.** Train the generator together with the discriminator following the classic GAN paradigm taught in class. 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 that 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.

Non-Adversarial Generative Networks

1. **Wasserstein Auto-Encoders (AEs).** AEs can also be considered as generative models as their decoder maps latent space vectors into images. However, the latent space created by the encoder has an arbitrary shape, or distribution, which we cannot sample from. In order to shape the latent space's distribution to be a "simple" and sample-able distribution, let us explore the use of additional losses over the encoder. In particular, let us require that the empirical mean, standard-deviation and kurtosis of latent vectors are identical to those of a standard Gaussian distribution, which are vectors of zeros, ones and threes respectively. You can read more about the kurtosis in relation to Gaussian distribution here.

   a. Train this AE to encode the target class of images into the same latent space dimension as you used above when training a GAN. Compare the results you obtain with the ones produced by the GAN above. Describe the qualitative differences between the images produced and explain them in relation to the different mechanisms of the underlying methods producing them? **Note**, that since the new losses are statistical estimators, large batches must be used (64 or greater). However, since these losses are defined only over the encoder (and not the decoder), the back-prop procedure can be made shorter and require less memory if this optimization is separated from the reconstruction optimization. Running them interleaved can be a solution for those of you who face GPU memory issues.

   b. Given an image I, the encoder explicitly provides you with its corresponding z code (no need to solve an optimization for this inversion!). Use this to compute two codes, z1 and z2, that correspond to two different images, and produce the intermediate images (interpolate) by G(z1*a+z2*(1-a)) and use different values between 0 and 1 (G denotes the decoder). You ran this experiment with a naive

AE in the previous exercise. Which interpolation results in better looking images? Explain.

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. Explain how the value of P(x) can be computed in the GLOW settings. Write down the explicit formula given the components computed/available in this method. Explain why P(x) cannot be computed in GAN and GLO.

2. At the beginning of a GAN training, G and D are faced with problems with very different complexity. G must create authentic images (hard), D must differentiate between true images and the poorly-produced images that G creates at initialization. Hence, D converges much quicker to a decisive classifier with logit values close to 0 and 1. These values are obtained by small and large values right before its final output sigmoid activation (that maps its values to [0,1]). Show analytically, what happens to the gradients of V with respect to G in this scenario.

3. GAN is trained by solving the following problem: min_G max_D V(D,G) ~ min_G [max_D V(D,G)] . This implies that the optimization of G occurs over a converged / maximized D. In practice, it means that for every gradient descent (GD) step of G, multiple steps must be made over D. We know however, that this may lead to a saturation of the gradients of G as shown above. In order to avoid that, in practice a single GD iteration of D is applied every GD iteration of G. In a sense, one could qually argue that a different problem is solved, namely, **max_D min_G** V(D,G) ~ **max_D [min_G** V(D,G)**]**. Explain what this problem is solving by:
    a. Explaining the what the inner problem **min_G** V(D,G) solves
    b. Explain what the outer problem needs to be solved given the converged inner problem. Explain whether you think it is easy or hard for a limited capacity neural network.
    c. What pitfat do you see in this inverted formulation assuming D fails to achieve the exact solution it needs to.

Submission Guidelines:

The submission is in **<u>singles</u>.** Please submit a single zip file named "ex3_ID.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 and analysis 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.