Due Date : April 20th, 2018

Instructions
- *For all questions, show your work!*
- *This part (practical) is to be done in teams of 2 or 3.*
- *Use a document preparation system such as LaTeX.*
- *Submit your answers electronically via the course studium page.*

(100 points) Image Generation
One of the fundamental tasks in AI is to learn a joint probabilistic distribution that models the real world data. Recent years have shown an explosion of research into using deep learning and computer vision algorithms to generate images. Some of this work has been motivated by understanding algorithms, for instance visualizing what a classification network "sees" in an image . However the strange and beautiful images that came from this research triggered a new push towards image generation for aesthetic ends.

An ongoing challenge in the image generation community is the question of evaluation. In computer science we often want objective metrics by which to compare our algorithms, but for aesthetic tasks it's not obvious that such a metric necessarily even exists.

Combining the above two aspects, the goal for this practical assignment is as follows :

1. **Generating Faces :** The CelebFaces Attributes Dataset CelebA contains over 200,000 celebrity images with annotations. The task for this assignment is to generate new face samples using the dataset with deep generative models that we have seen in the class. Since you're going to be generating faces, you won't need the annotations. You will have to generate faces of resolution 64×64. The dataset can be downloaded here and the helper code to get you started with the dataset pre-processing can be found here. *Some gentle advice : you could start with a small subset of the training data to accelerate debugging.*
2. **Model :** For the assignment, you will have to implement **one** of the following models :
   (a) Variational Auto-Encoders (VAE)
   (b) Generative Adversarial Networks (GANs)
3. **Architecture :** In either case, a latent dimension on the order of 10^2 should suffice. For the generator/decoder, you may take inspiration from the DCGAN architecture. For the encoder of VAE, you may simply use a symmetric structure. To increase/double the feature map size, compare the following alternatives :
   (a) Deconvolution (transposed convolution) with paddings and strides.
   (b) Nearest-Neighbor Upsampling followed by regular convolution.
   (c) Bilinear Upsampling followed by regular convolution

   Visually inspect the difference between the different schemes to increase feature map size, and explain the difference in terms of arithmetics of these operations. You may stick to one operation and proceed with the following analyses.
4. **Variants :** For the analyses below, compare your vanilla GAN/VAE with **one** of the variants provided below :

- For GANs
  (a) WGAN: (let W_D be the weight parameters of the discriminator)
$$L_D^{WGAN} = \mathbf{E}_{p_{data}}\big[D(x)\big] - \mathbf{E}_{p_{gen}}\big[D(G(z))\big]; \quad L_G^{WGAN} = \mathbf{E}_{p_{gen}}[D(G(z))]$$
$$W_D \leftarrow \text{clip}(W_D, -0.05, 0.05)$$
  (b) LSGAN :
$$L_D^{LSGAN} = \mathbf{E}_{p_{data}}[(D(x)-1)^2] + \mathbf{E}_{p_{gen}}[D(G(z))^2]; \quad L_G^{LSGAN} = \mathbf{E}_{p_{gen}}[(D(G(z))-1)^2]$$
- For VAEs
  (a) Normalizing flow as approximate posterior $q(z|x)$
  (b) Importance weighted autoencoder with 5 importance samples

5. **Qualitative Evaluations :**
   (a) Provide visual samples from your models. Is the variant you choose better than the vanilla model in terms of quality of the samples (e.g. blurriness, diversity) ? If yes, why does it help, and if not, explain why and suggest some ways to improve it.
   (b) Try to make changes in the latent space (for z) and observe whether changes in different dimensions in z result in different visual variations. Comment.
   (c) Pick two random points z_0 and z_1 in the latent space. For $\alpha = 0,\ 0.1,\ 0.2 \ldots 1$ compute $z' = \alpha z_0 + (1-\alpha)z_1$ and plot the resulting samples. Do the same thing with two samples x_0 and x_1 that correspond to z_0 and z_1, respectively, and visualize $x' = \alpha x_0 + (1-\alpha)x_1$. Explain the difference with the two schemes to interpolate between images.
6. **Quantitative Evaluations (GAN) :** Some of the best generative models don't have a suitable tractable quantitative evaluation metric (they just provide samples). There are many likely metrics used in the research community to evaluate generated samples. For example Inception score proposed by Tim Salimans et al. (2016) in the mentioned paper to evaluate generative models such as VAEs and GANs use a pre-trained classifier network and sampled images. Other metrics include
   (a) The Mode Score
   (b) Wasserstein distance
   (c) Maximum Mean Discrepancy (MMD)

   You can find the references to the above metrics here and here. Implement ONE of the metrics (a-c) as well as Inception score and report your results. Discuss the pros and cons of the metric that you choose and the Inception score.
7. **Quantitative Evaluations (VAE) :** For VAE based models, evaluate your model based on bits-per-pixel (bpp), which is another form of log likelihood (approximated by importance sampling) that we covered during the lecture :
$$\log p(x_i) \approx \log \sum_{k=1}^{K} \frac{p(x_i, z_k)}{q(z_k|x_i)}; \quad z_k \sim q(z|x_i)$$
   Choose K=2000. Note that the standard way to preprocess the discrete data (0-255) is to add a uniform noise Unif(0,1) to each pixel and then divide by 256. To compute the log likelihood of the pre-scaled noisy image, you need to account for the scaling by substracting some constant. Take a look at Sec 2.4 of this paper. To convert log likelihood to bpp, you need to change the base of log to 2 and divide by number of pixels (64*64=4096). For comparison, a generative model known as Real NVP reports 3.02 bpp on CelebA.

More gentle advice : to train a generative model like GAN or VAE, stability is usually an issue that you need to deal with. For instance you might need to tune the number of times you update the parameters of the discriminator before updating the generator once for GAN. Also, step size of the optimizer and smoothness (via gradient penalization for instance) are probably some things that may help you train a model more easily. For VAE, be careful with the activation you choose to ensure the standard deviation is positive (exponentiation works but might not be optimal for stability). Also, there's a question as to whether you want to model the standard deviation of the decoder for each pixel, each datapoint. Some parameter sharing might help with training at the cost of expressiveness. Most importantly, there are some metrics that you can monitor throughout training, such as the discriminator's cross entropy loss for GAN and the KL divergence between the prior and approximate posterior for VAE. Make good use of these metrics !