Advanced Course on Deep Generative Models **Spring 2024**

Exercise 5: Variational Auto-Encoders

Dan Rosenbaum

Due: 30 June 2024

In this exercise we will use the same notebook we used in exercise 4.

`https://uvadlc-notebooks.readthedocs.io/en/latest/tutorial_notebooks/tutorial12/Autoregressive_Image_Modeling.html`

The goal of this exercise is to implement a variational autoencoder (VAE) model.

In order to do that, use the same training process and model structure as the implemented autoregressive model. This means you will need to implement two main functions:

1. **calc_elbo:** As computing the likelihood is intractable, you should implement a function that calculates the ELBO. This should replace 'calc_likelihood' and be used for training, evaluation and testing. As we defined in class, the ELBO consists of two terms: the reconstruction log likelihood, $\log p(x|z)$, and the KL divergence, $D_{KL}\left(q(z|x)||p(z)\right)$. Note that the original implementation returns the negative likelihood so you can also return the negative ELBO.
2. **sample:** This function is used to generate samples. As the process of generating samples is different than in autoregressive models, the parameters of this function should only be the number or samples requested.

In order to implement a VAE you will need to implement both an encoder network and a decoder network.

- The **encoder** should consist of several layers of convolutions (you can use stride=2 in two of the layers to gradually decrease the dimension of the representation), followed by several layers of fully connected linear layers.
- The **decoder** should consist of several layers of fully connected linear layers followed by several layers of transposed convolutions such that the output has the same shape as the input images.

In addition you will need to define a prior distribution. You should use a standard isotropic Gaussian over the latent vector, without learned parameters. Note that the prior, encoder and decoder are used in different ways for computing the ELBO and for sampling. The encoder is used to define a posterior probability over the latents $q(z|x)$. The posterior should be implemented as a Gaussian with different mean and variance for each dimension of the latent z. The way this can be implemented is by mapping the output of the decoder (using learned linear mappings) to two vectors with the same dimension as the latents. The first vector is used as the mean for each dimension, and the second vector is used as the log of the std (exponentiate it before you pass it to the distribution).

1. Implement a VAE as described above and train it on the MNIST data. The pixel values in the image should be represented as continuous variables, and the reconstruction should be modeled as a Gaussian distribution. In order to consider the pixel values as continuous, you should cast them to a float type, add uniform noise in the range of $[0, 1]$, and then scale them from $[0, 256]$ to $[0, 1]$. In order to implement a Gaussian reconstruction term, the output of the decoder should be the mean for each pixel, and you should add one hyper-parameter that will be used as the constant standard deviation of all pixels in the image. A good value for this parameter is 0.1. When you sample from the reconstruction distribution, you can simply

show the mean of each pixel - this should be the output of the function 'sample' (even though it is not a real sample).

Report the ELBO for the test set, curves showing the ELBO during training (for both training and validation), and curves showing the reconstruction likelihood and KL divergence during training. After training, generate a few samples and show the results. You should be able to train models where most of the samples look like valid digits.

2. In this question you will generate latent traversals, showing the representation power of VAEs. Use the model that you trained, and chose two images of different digits from the dataset. For each image, compute a sample from the posterior $q(z|x)$. This will be the 'representation' of the image. Now given then two latent representations z_1, z_2, compute a linear path between them with a set of 10 points along the way. For each of those 10 points, z^i, compute the reconstruction sample $p(x|z^i)$ (you can use the mean of this distribution like before), and show the resulting images in order. You should see how the first digit transforms into the second digit. Repeat this for 3 different pairs of images.

You should submit a colab notebook that contains the text of the answers, figures and code for all the above questions.