Course: Deep Learning **([16/12/2021])**

Submission Assignment #[4]

Coordinator: Jakub Tomczak *Name:* James Bonello, Marton Fejer, Piotr Jastak*, Netid:* [jbo480], [mfr201], pjk600

1 Part 1

Question 1 *Write objective functions for VAEs and GANs. Use standard normal prior p(z) for VAE.*

1.0.1 VAE Objective Function

$$\log p_\theta(\mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\Big[\log p_\theta(\mathbf{x}|\mathbf{z})\Big] - KL\Big(q_\phi(\mathbf{z}|\mathbf{x})\big|\big|p_\lambda(\mathbf{z})\Big)$$

Note that $p_\lambda(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$, and θ, ϕ, and λ are neural network parameters.

The objective function for a VAE involves two components: the reconstruction error, and the regularization term. The reconstruction error is the log-likelihood of the decoder given the latent vector drawn from the variational posterior. It tells us how close we came (on average) with approximating $\mathbf{x}$ given $\mathbf{z}$ drawn from the latent space. The regularization term is actually the Kullback-Leibler divergence of the variational posterior $q_\phi(\mathbf{z}|\mathbf{x})$ and the marginal prior $p_\lambda(\mathbf{z})$. This tells us how much divergence there is between our approximation for the latent space given the inputs, and the assumed marginal distribution for the latent space. We can also refer to this objective function as the ELBO, and we find by definiton of Jensen's inequality that it gives us a lower bound for the log-likelihood function. Furthermore, it is important to note that we do have a true posterior distribution $p(z|x)$ for the latent space given the input, and the KL divergence between this and the variational posterior gives us an upper bound to our log-likelihood. The true posterior distribution is intractable (i.e too complex to compute efficiently), and this is why the variational posterior is used in its place. The closer this variational posterior is to the true distribution, the higher the lower bound for the log-likelihood function.

1.0.2 GAN Objective Function

$$min_G max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}\big[\log D(x)\big] + \mathbb{E}_{z\sim p_z(z)}\big[\log(1 - D(G(z)))\big]$$

The idea follows a mini-max approach where the discriminator and generator play a repeated game to reach an equilibrium state. The discriminator continuously tries to guess real from fake images that the generator gives it until both reach an equilibrium such that they are both in their optimal state.

Question 2 *Derive VAE objective with the Normalizing Flow as a prior. What are the advantages of such model formulation?*

$$\log p_\theta(\mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\Big[\log p_\theta(\mathbf{x}|\mathbf{z})\Big] - KL\Big(q_\phi(\mathbf{z}|\mathbf{x})\big|\big|p_\lambda(\mathbf{z})\Big)$$

Let f be an arbitrary and invertible transformation such that:

$$f : Z \to V$$

where Z and V have the same dimensionality over an arbitrary field (eg. $\mathbb{R}$. Therefore, $V = f(Z)$ and $Z = f^{-1}(V)$. In a normalizing flow model, for $\mathbf{v} \in V$ we have that

$$p_\psi(\mathbf{v}) = p_\lambda\big(f^{-1}(\mathbf{v})\big)\left|\det\left(\frac{\delta f^{-1}(\mathbf{v})}{\delta \mathbf{v}}\right)\right|$$

For simplicity, we denote the Jacobian matrix $\frac{\delta f^{-1}(\mathbf{v})}{\delta \mathbf{v}}$, as $\mathbf{J}$. Note that ψ is a collection of parameters for the probability distribution over V. Now, since we have applied a change of variables from Z to V, we can replace $p_\lambda(\mathbf{z})$ with $p_\lambda(f^{-1}(\mathbf{v}))$, and we get

$$\log p_\theta(\mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\Big[\log p_\theta(\mathbf{x}|\mathbf{z})\Big] - KL\Big(q_\phi(\mathbf{z}|\mathbf{x})\big|\big|p_\lambda(f^{-1}(\mathbf{v}))\Big)$$

Let us look closer at the Kullback-Leibler divergence term. We know that

$$KL\Big(q_\phi(\mathbf{z}|\mathbf{x})\big|\big|p_\lambda(f^{-1}(\mathbf{v}))\Big) = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\Big[\log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\lambda(f^{-1}(\mathbf{v}))}\Big]$$

Expanding further,

$$\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\Big[\log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\lambda(f^{-1}(\mathbf{v}))}\Big] = \sum q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\lambda(f^{-1}(\mathbf{v}))} = \sum q_\phi(\mathbf{z}|\mathbf{x})\big(\log q_\phi(\mathbf{z}|\mathbf{x}) - \log p_\lambda(f^{-1}(\mathbf{v}))\big)$$

Then,

$$\sum q_\phi(\mathbf{z}|\mathbf{x})\big(\log q_\phi(\mathbf{z}|\mathbf{x}) - \log p_\lambda(f^{-1}(\mathbf{v}))\big) = \sum q_\phi(\mathbf{z}|\mathbf{x}) \log q_\phi(\mathbf{z}|\mathbf{x}) - \sum q_\phi(\mathbf{z}|\mathbf{x}) \log p_\lambda(f^{-1}(\mathbf{v}))$$

Finally, by definition of the expected value,

$$KL\Big(q_\phi(\mathbf{z}|\mathbf{x})\big|\big|p_\lambda(f^{-1}(\mathbf{v}))\Big) = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\Big[\log q_\phi(\mathbf{z}|\mathbf{x})\Big] - \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\Big[\log p_\lambda(f^{-1}(\mathbf{v}))\Big]$$

By definition of $p_\lambda(f^{-1}(\mathbf{v}))$, we have that

$$\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\Big[\log p_\lambda(f^{-1}(\mathbf{v}))\Big] = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\psi(\mathbf{v})}{\Big|\det \mathbf{J}\Big|}\right] = KL\Big(p_\psi(\mathbf{v})\Big|\Big|\big|\det \mathbf{J}\big|\Big)$$

Therefore,

$$KL\Big(q_\phi(\mathbf{z}|\mathbf{x})\big|\big|p_\lambda(f^{-1}(\mathbf{v}))\Big) = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\Big[\log q_\phi(\mathbf{z}|\mathbf{x})\Big] - KL\Big(p_\psi(\mathbf{v})\Big|\Big|\big|\det \mathbf{J}\big|\Big)$$

Our objective function for the VAE given a normalizing flow prior therefore becomes

$$\log p_\theta(\mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\Big[\log p_\theta(\mathbf{x}|\mathbf{z})\Big] - \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\Big[\log q_\phi(\mathbf{z}|\mathbf{x})\Big] - KL\Big(p_\psi(\mathbf{v})\Big|\Big|\big|\det \mathbf{J}\big|\Big)$$

The main advantage with normalizing flows is that it allows for a series of transformations that can turn a simple distribution into a more complex one, therefore offering more accurate sampling while still being able to easily measure the likelihood function via the invertible transformations.

Question 3 *Please write down components of VAEs and GANs, and explain their roles.*

1.0.3 VAE

There are three major components of a VAE.

The encoder $q(z|x)$ approximates a latent space as close as possible to the true posterior $p(z|x)$ with a neural network such that

$$q(z|x) = \mathcal{N}(g(\mathbf{x}; \mathbf{W}), h(\mathbf{x}; \mathbf{W}))$$

The neural network $f(g, h)$ is parametrized as such to show that it approximates a mean μ and standard deviation σ so as to encode a latent space from which our samples will be drawn.

With this being said, the second component of the VAE is the marginal dsitribution $p(z)$. This is assumed to be the Gaussian distribution.

The last component of the VAE is the decoder $p(x|z)$. Similarly to the encoder, this distribution is approximated by a neural network such that

$$p(x|z) = \mathcal{N}(l(\mathbf{z}; \mathbf{W}), h(\mathbf{x}; \mathbf{W}))$$

The decoder takes in a random sample z from the variational posterior, and decodes it back into an approximated probability space of the inputs $p(x|z)$. Since z is a random sample, the neural network would not be able to properly backpropagate the gradient given that the functions are computed with a random sample. Therefore, the latent variable z is reparametrized into a deterministic set of variables such that the weights are updated in a more stable manner.

1.1 GAN

For any given GAN, a model consists of a Generator and Discriminator module. The discriminator is a normal convolutional network, which takes an image as an input and generates an output probability that the image is real, rather than coming from the generator ('fake'). The input is convolved in two-dimensions, combined with a *leakyReLu* activation function at each layer. Finally, a *sigmoid* activation is applied to the network.

The generator transforms a latent space vector into an image of the input dimensions for the discriminator network. This is achieved by applying two dimensional convolutional transpose layers, together with leakyReLu activations. At the end, a *Tanh* activation function is applied to the output.

At each training step, the discriminator is first trained with all real images, and then all fake images, thereby trying to minimize its loss function. Subsequently, the generator is updated by performing a forward pass, after which the outputs are then fed to the discriminator. The loss produced by the discriminator can then be used to update the

Question 4 *Please pay attention to the adversarial loss. Please explain which loss you are going to use.*

2 Part 2

We have written two versions of code for a VAE with standard Gaussian prior, one using MLP, another using CNNs. We used an 8-dimensional latent space for the MLP-based VAE and a 32-dimensional one for the CNN-based one. We used Leaky ReLU activations in the encoders and ReLU activations in the decoders, based on claims of superior performance by Radford et al. (2016). We used Binary Cross Entropy as the loss function, with sum instead of mean as reduction.

We have implemented a FID score calculation (taking implementation hints from Jason Brownlee's article at Brownlee (2019)), using the pre-trained Inception v3 network. We have calculated the score for the MLP- and CNN- based VAE based on a mean of samples of 32 images (real and generated). The results are as follows:

1. MLP (MNIST): 246.4875
2. CNN (SVHN): 91.2681

Please find the code for our implementations in attached files.

Major problems were faced when attempting to build the VAE with the flow-based prior. This was attempted using both our own custom VAE and one that was provided by the course coordinator on this page. The biggest problem always seemed to revolve around dimensionality, especially when implementing the link's code to work with MNIST. While the code eventually did work with MNIST, it was found that the reconstruction error was storing NaN values. The cause for the NaN values remains to be explored in future work due to time constraints.

3 Part 3

Question 1 *Analyze learning curves during training. If you use separate validation data (NOT test!), please analyze learning curves on it as well. At the end of training (e.g., after a fixed number of epochs), calculate the test FID scores for the VAE and the GAN.*

We have collected the training losses for MLP VAE (used for MNIST data) and CNN VAE (used for SVHN data). (1)

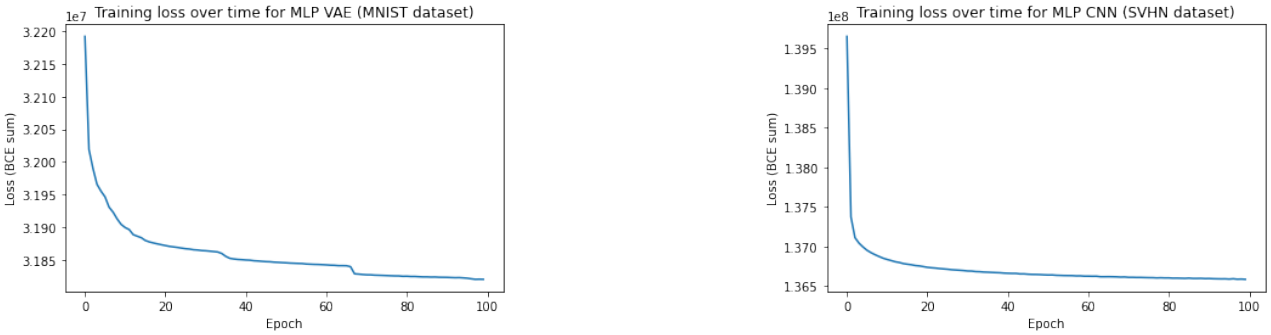
We evaluated the performance of GAN-for both SVHN and MNIST datasets by looking at their generator and discriminator loss. The loss of the two loss functions (of the generator and discriminator) should approach an equilibirium, which is seemingly working quite well for the MNIST dataset. However, for the SVHN dataset, the equilibrium stays further apart suggesting that perhaps more time was needed for the generator to train. (2, 3)

Comparing the training performance of the two approaches, we observe that GANs tended to be "noisy", often severely increasing in training loss before eventually converging. This is less so in the case of the SVHN dataset. In both cases the training and test loss appear closely correlated. VAEs on the other hand only rarely suffered an insignificant uptick in training loss, though we did not use the test dataset for their training.

Question 2 *Sample from the VAE and the GAN. Analyze and compare samples.* We present 100 samples from MNIST-trained MLP VAE, and 100 samples from the SVHN-trained CNN VAE (see 4

We also generated 100 samples from the GAN networks trained with MNIST and SVHN datasets (see 5).

Comparing the two systems, we see that:



(a) Training losses for MLP-based VAE

(b) Training losses for CNN-based VAE

Figure 1: Training losses for MLP and CNN-based VAEs

Figure 2: Generator and Discriminator loss for MNIST-GAN

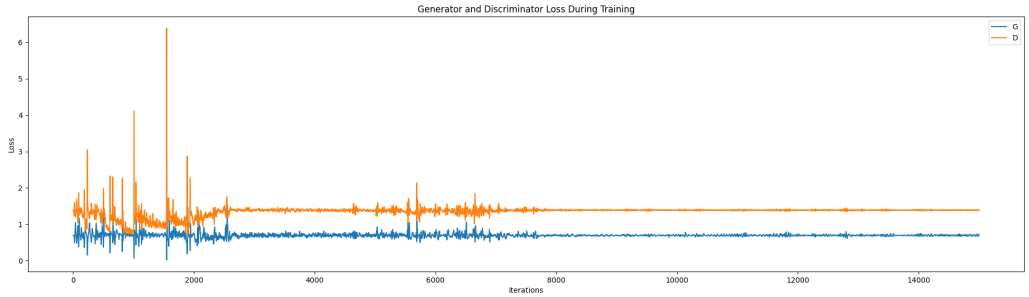
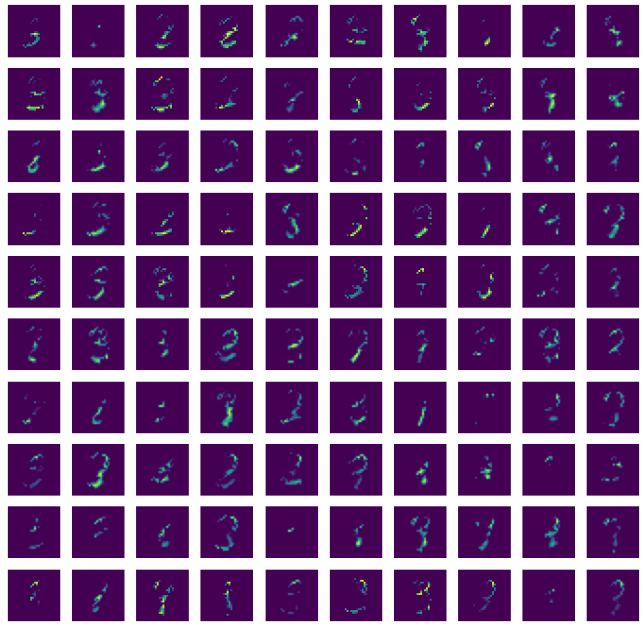
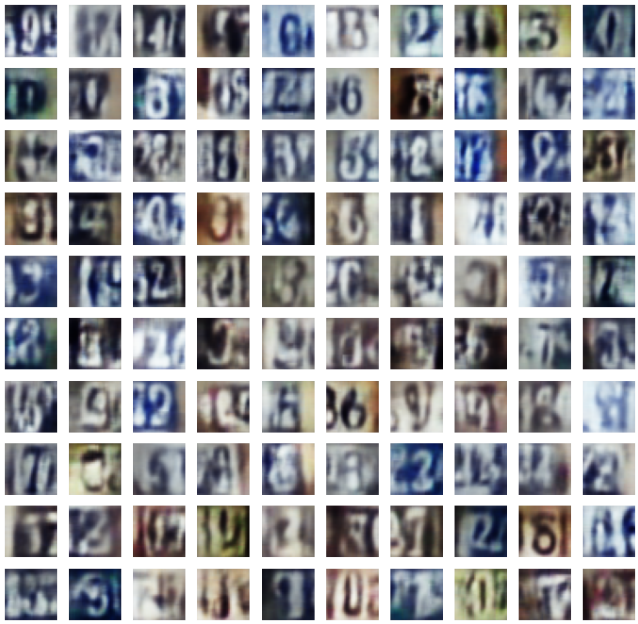


Figure 3: Generator and Discriminator loss for SVHN-GAN



(a) Samples from the MLP VAE



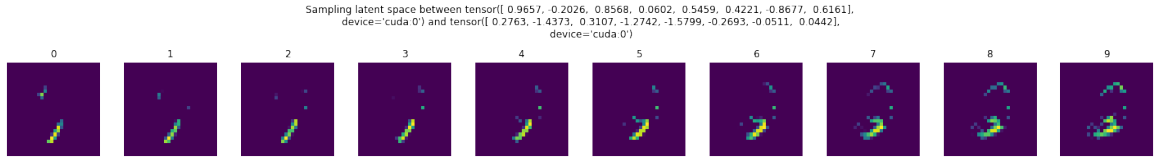
(b) Samples from the CNN VAE

Figure 4: Samples from two VAEs

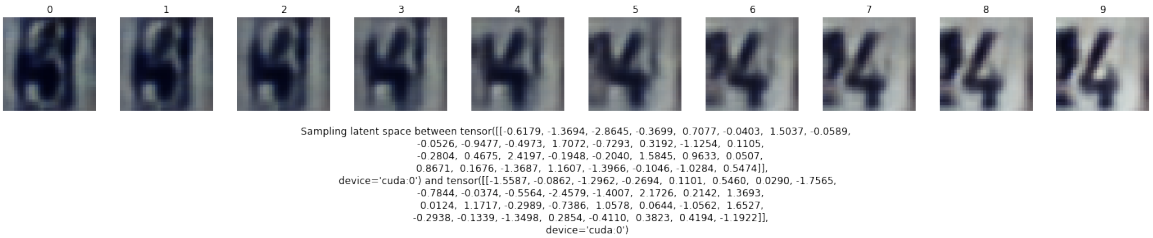
(a) Samples from GAN trained with MNIST

(b) Samples from GAN trained with SVHN

Figure 5: Samples from two GANs



(a) Interpolation in MLP VAE latent space



(b) Interpolation in CNN VAE latent space

Figure 6: Interpolations in two spaces of the MLP- and CNN-based VAEs

(a) Interpolation in MNIST-GAN latent space

(b) Interpolation in SVHN-GAN latent space

Figure 7: Interpolations in two spaces of the MLP- and CNN-based VAEs

1. The output produced by the MLP VAE is often hard to discern. While some glyphs are obviously numbers, others look like noise. In comparison, the MNIST generated sample for GAN is clear, though the numbers are hard to discern against the dark background.

2. The output produced by the CNN VAE, though blurry, appears much more convincing than the output for GAN. With VAE, we see what appears to be different numbers written on number plates, with GAN, we observe mostly noisy images with a 0-like glyph in the middle. The similarity of glyphs in the case of GAN can be due to the mode collapse problem, while the overall lack of image quality is likely due to insufficient training time and size constraints of the network.

Question 3 *Sample two points in the latent space and linearly interpolate between them. Then, for 10 points between the two original points, generate images. Analyze and compare the interpolations.*

We have selected two points in the latent space for MNIST-based VAEs and CNN-based VAEs and interpolated between them. (see 6)

We have also selected two points in the latent space for the MNIST and SVHN-based GANs and interpolated between the points. (see 7)

In the case of MNIST:

1. For GAN, we see what appears to be a 9 morphing into what appears to be a 3, though the darkness of the images makes the numbers very hard to discern.

2. For VAE, we see a vaguely 9-looking glyph morphing into a 0, though the quality of both is quite low, even though they are clearly visible against the background.

In the case of SVHN:

1. For GAN, we see an abstract-looking image that morphs into another, equally difficult to discern image.

2. For VAE, we see an image of what looks like a 3, written in white on a dark background, morphing into a 4, written in the 3's background's color on white.

In case of MNIST, both systems have advantages and disadvantages. GANs produce images which would be clear if they weren't as dark, while VAEs produce images which are clear though are hard to make out.

In case of SVHN, by visual inspection, VAEs clearly trump GANs, producing believable and discernible output, while GANs tend to generate indecipherable images.

Question 4 *Perform 2 and 3 on MNIST and SVHN.*

References

Brownlee, J. (2019). How to implement the frechet inception distance (fid) for evaluating gans.

Radford, A., Metz, L., and Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks.