DLF Homework 2

John Klein: 10044346
Kristian Rehn: 10043938

Task 1

A DCGAN is a composite model consisting of a generator and a discriminator. Such a model follows certain guidelines such as replacing the pooling layers with strided convolutions or the use of batchnorm layers in both parts and other guidelines (Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks). The generator tries to create objects that are as authentic as possible, while the discriminator tries to recognize the objects created from the generator. This results in a game against each other, whereby both should improve. In most cases, the generator is then used and the discriminator is only used to train the generator and then discarded.

Task 3

1. Reparametrization trick

Since the sampling process to retrieve the distribution's parameters is non-differentiable, we can not perform backpropagation as we can not apply a gradient descent. The *Reparametrization Trick* therefore suggests to fix the

distribution's parameters and scale them by randomly drawing constants from a prior distribution (normal distribution):

$$z = \mu + \sigma \cdot \epsilon \qquad\qquad \epsilon \sim \mathcal{N}(0, 1)$$

Since we now not sample from a Gaussian $\mathcal{N}(\mu, \sigma)$ but from a normal distribution, we can compute a gradient on z and therefore enables us to use backpropagation.

2. GANs: Optimization and Training Stability

Optimizing GANs is hard because there we have to optimize the generator and discriminator simultaneously, while they are antagonists playing a minimax game. This means it's hard to reach an equilibrium between them and possibly not ever reaching optimization but an oscillation without approaching a stable state.
Another problem regarding optimization occurs when the discriminator is perfect. We get a vanishing gradient problem, where the loss approaches 0 and therefore we can not update the loss since its gradient is non-present.
We also have to account for mode collapse: if the generator finds some outputs which always successfully fool the discriminator, we only get very similar outputs to this particular subset of outputs, resulting in a too homogenous result not representing the whole dataset.

We can improve training stability by employing several tweaks to the original idea. The original paper (Generative Adversarial Nets) laid out a way to address the vanishing

gradient problem reformulating the generator loss from

$$J^{(G)} = \frac{1}{2}\mathbb{E}_{\mathbb{z}} \log(1 - D(G(z)))$$

to

$$J^{(G)} = -\frac{1}{2}\mathbb{E}_{\mathbb{z}} \log(D(G(z)))$$

and hereby avoiding a gradient close to zero when the generator is unable to fool the discriminator. The reformulation causes the gradient to be higher close to zero and flat out as it gets better.

In Improved Techniques for Training GANs the authors describe a technique called One-Sided Label Smoothing to prevent the generator from becoming too confident with its predictions. We smooth positive labels to α, a value usually set to something like 0.9 instead of 1, and keep negative labels at 0. Since a correct classification will not produce a zero loss, the generator will be more inclined to produce different results, thus improving training stability and variety.

3. GANs vs VAEs

GANs and VAEs are both generative models, but work in different ways. GANs employ the generator and discriminator in an adversarial fashion to let the generator produce samples which fool the discriminator. VAEs on the other hand translate the input into a distribution (latent space) and use a decoder to reconstruct it.

GANs minimax the classification error loss while VAEs maximize the variational lower bound (ELBO). Both do not explicitly model the density $P(x)$ but use an approximation: $x \sim P_\theta(x)$.

VAEs are a better solution when you need a full representation of the dataset and want to reconstruct a inputs. This would be the case for denoising tasks or inpainting. Anomaly detection tasks also profit from the benefits of a learned distribution.

GANs are a great solution for high-quality generation tasks. Due to the adversarial setup, they deliver better results in sharpness, quality and realism. Settings which profit from these features are image generation and enhancement.