Generative Adversarial Networks and Cycle-GAN Deep Learning 2020

Gabriel Kasmi
ENSAE, ENS Paris-Saclay
gabriel.kasmi@ensae.fr

Hugo Thimonier
ENSAE, ENS Paris-Saclay
hugo.thimonier@ensae.fr

1 Litterature Review

Since Goodfellow et al. (2014) (2) have introduced Generative Adversial Networks (GAN) many extensions have been proposed to try and improve the vanilla model's performances. Let us first discuss the original GAN, then we will go on discussing Deep Convolutional GAN (DC-GAN) as well as Wasserstein-GAN (WGAN).

1.1 The vanilla GAN

Before entering the maths, let us briefly recall what are Generative Adversial Networks. A GAN is comprised of two adversial networks : a generative network (G) and a discriminative network (D). Both networks are adversial in the sens that the former (G) aims at capturing the data distribution while the latter (D) estimates the probability that a sample came from the training set rather than generated by G. Both G and D are multilayer perceptrons in this framework.

Let us define p_g as the generator's distribution over the data x. Also consider a prior on input noise variables denoted $p_z(z)$ and a mapping to data spaces as $G(z; \theta_g)$, where G is a differentiable function represented as a multilayer perceptron with parameter θ_g.
Also consider a second multilayer perceptron, the discriminative network, $D(x; \theta_d)$ that outputs a scalar. In that framework, $D(x)$ is simply the probability that x came from the data rather than p_g. Thus, we aim at training D so as to maximize the probability of assigning the correct label to data from the training set and data generated by G. Simultaneously, we train G in order to minimize $\log(1-D(G(z)))$: $D(G(z))$ is the probability that $G(z)$, the generated sample, came from the training set, G is trained to maximize this value, thus to minimize $1 - D(G(z))$ which is equivalent to minimizing $\log(1 - D(G(z)))$. This is equivalent to the following two-player minimax game with the value function $V(G, D)$:

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

The training is performed alternating between k steps of optimizing on D and one step on G. This allows D to be maintained near its optimal solution if G changes slowly enough. A small detail but which can have large consequences is the possibility for the model in early learning not to provide a large enough gradient for G to learn correctly. For instance, when G's performances are poor, at the early stage of the game, D is able to reject with high confidence samples generated by G as they are very far away from the true data. Thus, in that case as $D(G(z))$ is very close to 0 and $\log(1 - D(G(z)))$ as well. Thus, the training on G is performed maximizing $\log(D(G(z)))$.
G defines a probability distribution p_g as the distribution of $G(z)$ when $z \sim p_z$. The objective of training is to make p_g converge to p_{data}.

Goodfellow et al. (2014) (2) show that in that framework one can obtain $p_g = p_{data}$. For instance, the optimal discriminator D for any given generator G is defined as

$$D^*_G(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \tag{2}$$

The proof of this can be seen in appendix A.

To go on and proof that indeed the global minimum is achieved when $p_g = p_{data}$ let us note that training for D as described above can be interpreted as a maximization of the log-likelihood for estimating the conditional probability that $P(Y = y|x)$ where Y simply indicates whether x comes from p_{data} (with $y = 1$) or from p_g (with $y = 0$). Hence the minimax game described earlier can be reformulated as minimizing in G the following expression:

$$\begin{aligned} C(G) &= \max_D V(G,D) \\ &= \mathbb{E}_{x\sim p_{data}}[\log(D^*_G(x))] \\ &\quad +\mathbb{E}_{z\sim p_z}[\log(1-D^*_G(G(z))] \\ &= \mathbb{E}_{x\sim p_{data}}[\log(D^*_G(x))] \\ &\quad +\mathbb{E}_{x\sim p_g}[\log(1-D^*_G(x)] \\ &= \mathbb{E}_{x\sim p_{data}}[\log(\frac{p_{data}(x)}{p_{data}(x)+p_g(x)})] \\ &\quad +\mathbb{E}_{x\sim p_g}[\log(\frac{p_g(x)}{p_{data}(x)+p_g(x)})] \end{aligned}$$

Let us now show that the global minimum is achieved iff $p_g = p_{data}$. First consider that in the case where $p_g = p_{data}$ we have that $D^*_G(x) = 1/2$. Thus in that case

$$C(G)|_{p_g=p_{data}} = \log(1/2) + \log(1/2) = -\log(4)$$

knowing that the candidate for the minimum is $-\log(4)$ we proceed to substract and add $\log(2)$ times each density $p_g(x)$ and $p_{data}(x)$. This gives,

$$\begin{aligned} C(G) &= \int_x (\log(2)-\log(2))p_{data}(x) \\ &\quad +p_{data}(x)\log(\frac{p_{data}(x)}{p_{data}(x)+p_g(x)}) \\ &\quad +(\log(2)-\log(2))p_g(x) \\ &\quad +p_g(x)\log(\frac{p_g(x)}{p_{data}(x)+p_g(x)})dx \\ &= -\log(2)\int_x p_g(x)+p_{data}(x)dx \\ &\quad +\int_x p_{data}(x)(\log(2)+\frac{p_{data}(x)}{p_{data}(x)+p_g(x)}) \\ &\quad +p_g(x)(\log(2)+\frac{p_g(x)}{p_{data}(x)+p_g(x)})dx \\ &= -\log(4)+\int_x p_{data}(x)\frac{p_{data}(x)}{(p_{data}(x)+p_g(x))/2} \\ &\quad p_g(x)\frac{p_g(x)}{(p_{data}(x)+p_g(x))/2}dx \end{aligned}$$

which gives in the end

$$\begin{aligned} C(G) &= C(G)|_{p_g=p_{data}} \\ &\quad +KL(p_{data}||\frac{p_{data}+p_g}{2}) \\ &\quad +KL(p_g||\frac{p_{data}+p_g}{2}) \end{aligned}$$

and by construction of the Kullback Leibler divergence (KL), it is positive or null. Thus, $p_g = p_{data}$ is the minimum. Hence, the generative model perfectly replicates the data generating process.
Let us define the *Jensen-Shannon* (JS) divergence as

$$\begin{aligned} JS(p_{data},p_g) &= KL(p_{data}||\frac{p_{data}+p_g}{2}) \\ &\quad +KL(p_g||\frac{p_{data}+p_g}{2}) \end{aligned}$$

One can thus rewrite the original problem described in equation (1) as the following :

$$\begin{aligned} \min_G \max_D V(D,G) &= \min_G V(D^*_G,G) \\ &= \min_G C(G) \end{aligned}$$

which is equivalent to

$$\min_G JS(p_{data},p_g) \quad (3)$$

the equivalence between 1 and 3 led to several extension and especially Wasserstein-GANs that whe shall present in the next section, but that mostly consists in replacing the JS divergence by the Wassertein-1 distance.

1.2 Wasserstein GAN (1)

Let us recall the definition of the *Eart-Mover* (EM) distance, also known as the Wasserstein-1 distance :

$$W(\mathbb{P}_r,\mathbb{P}_g)) = \inf_{\gamma\in\Pi(\mathbb{P}_r,\mathbb{P}_g)} \mathbb{E}_{x,y\sim\gamma}[||x-y||]$$

where $\Pi(\mathbb{P}_{data},\mathbb{P}_g)$ denotes the set of all joint distributions $\gamma(x,y)$ whose marginals are $(\mathbb{P}_r,\mathbb{P}_g)$. $\gamma(x,y)$ indicates how much "mass" must be transported from x to y in order to transform the distribution $\mathbb{P}_r$ to $\mathbb{P}_g$). The EM distance is the "cost" of the optimal transport plan.

Consider $\mathbb{P}_r$ a fixed distribution over $\mathcal{X}$, which in the case of GAN would be the true distribution of the sample (p_{data} earlier). Let z be a random variable over $\mathcal{Z}$, and let $g : \mathcal{Z}\times\mathbb{R}^d \to \mathcal{X}$ be a function, $g_\theta(z)$ with z the first coordinate and θ the second, and now define $\mathbb{P}_\theta$ the distribution of $g_\theta(\mathcal{Z})$ *i.e.* the distribution of the output of the generator (p_g earlier).

The Kantorovich-Rubinstein duality tells us that one can rewrite equivalently the EM distance as

$$W(\mathbb{P}_r,\mathbb{P}_\theta)) = \sup_{||f||_L\le 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)] \quad (4)$$

where the supremum is over all the 1-Lipschitz function $f : \mathcal{X}\to\mathbb{R}$. If we were to replace 1 by K to consider all K-Lipschitz functions, $||f||_L \le K$ as the set over which the supremum is considered, then we would end up with $K.W(\mathbb{P}_r,\mathbb{P}_\theta))$.
Hence, if one disposes of a parameterized family $\{f_w\}_{w\in\mathcal{W}}$ of K-Lipschitz functions for some K, thus one can solve the following problem

$$\max_{w\in\mathcal{W}} \mathbb{E}_{x\sim\mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z\sim p(z)}[f_w(g_\theta(z)]$$

if there exists a $w\in\mathcal{W}$ such that the suprememum is attained, then such process would yield a calculation of $W(\mathbb{P}_r,\mathbb{P}_g))$ up to a constant. Moreover, one could consider differentiating $W(\mathbb{P}_r,\mathbb{P}_\theta))$ by back-proping through the previous equation via estimating $\mathbb{E}_{z\sim p(z)}[\nabla_\theta f_w(g_\theta(z)]$.
To estimate the function that solves the maximization problem described in equation (4) one can consider training a neural network parametrized with weights w lying in a compact $\mathcal{W}$, and then backprop through $\mathbb{E}_{z\sim p(z)}[\nabla_\theta f_w(g_\theta(z)]$ as

would a typical GAN do.
The WGANs are a transformation of the vanilla GANs described in the previous section. The previously called discriminator, is now transformed into a critic. Its previous role was to disentangle which are the false samples generated by the generator and which are the true ones. In the WGAN framework, the critic is now trained to learn a K-Lipschitz continuous function to help compute the Wasserstein distance. It is trained to learn w to find a good f_w and the loss function is configured as measuring the Wasserstein distance between p_{data} and p_g. Wasserstein GAN (WGAN) proposes a new cost function using Wasserstein distance which has a smoother gradient everywhere : WGAN learns no matter how wall the generator is performing. In fact, as discussed in the previous section one of the major issues of vanilla GAN's is that if the generator does perform not well enough in generating sample close to the true data, the gradient for the generator diminishes and the generator is unable to learn.

1.3 Deep Convolutional GAN

The key motivation for this paper (Radford et al., 2015) (4) is to scale up GANs using CNNs (Convolutionnal Neural Networks) to model images which had been unsuccessful until then. For instance, an other major drawback of vanilla GAN introduced by Goodfellow et al. (2014) (2) is the possibility that the generated images are noisy and incomprehensible. This paper present a novel architecture that allows a higher resolution than the usual blurry images generated by vanilla GANs.

This new architecture, DCGAN, is mostly based on three recently demonstrated CNN architecture modifications. One of the major innovation of this paper is the introduction of strided convolution to replace deterministic spatial pooling functions (such as maxpooling). The authors of the paper got their idea from the all convolutional net (Springenberg et al., 2014) (5). This is used in both the generator, allowing it to learn its own spatial upsampling, and in the discriminator. The second main change that was used to improve the model stability (but that ended up slowing down convergence) is eliminating fully connected layers on top of convolutional features (Mordvintsev et al., 2015) (6). The idea is to directly connect the highest convolutional features to the input and output respectively of the generator and discriminator. In other words, the first layer of the GAN, which takes a uniform noise distribution Z as input is reshaped into a 4-dimensional tensor and used as the start of the convolution stack. For the discriminator, the last convolution layer is flattened and then fed into a single sigmoid output. Thirdly, Batch Normalization (Ioffe Szegedy, 2015) was used to stabilizes learning by normalizing the input to each unit to have zero mean and unit variance (not for the generator's output layer nor the discriminator input's layer). Finally, the activation function used in the generator at each layer is replaced by the ReLu function $\sigma(x) = x_+ = \max(0, x)$, except for the last layer which used the tanh function (hyperbolic tangent) defined as $\sigma(x) = \frac{e^{2x}-1}{e^{2x}+1}$. Similarly, concerning the discriminator the activation function considered is now the Leaky ReLu,

$$\sigma(x) = \begin{cases} x & \text{if } x > 0, \\ 0.01x & \text{otherwise} \end{cases}$$

which is a special case of the maxout activation function that was used in the vanilla GAN's.

2 Cycle-GAN

Image-to-image translation model aim at learning the mapping between an input image and an output image. In order for a model to learn such mapping, one requires to have a training set composed of both aligned images. The dataset required to train such model are very rare as they must contain a pair of image composed of the two states between which you want to learn the mapping. For example, in order for a model to estimate the summer to winter mapping on a landscape, one would need to dispose of a large sample of pair of picture of a landscape in both seasons. Cycle-GAN (3) manage to to circumvent this data problem by translating an image from a source domain $\mathcal{X}$ to a target domain $\mathcal{Y}$ in the absence of paired examples.
Let us translate this into math : Cycle-GANs learn a mapping $G : \mathcal{X} \to \mathcal{Y}$ such that the distribution of images from $G(\mathcal{X})$ is indistinguishable from the distribution $\mathcal{Y}$. Conversely, the model shall also learn $F : \mathcal{Y} \to \mathcal{X}$ and introduce a cycle consistency loss to enforce $F(G(\mathcal{X})) \approx \mathcal{X}$.

2.1 Mathematical Formulation

Consider two domains $\mathcal{X}$ and $\mathcal{Y}$ and a training sample $\{x_i\}_{i=1}^{N}$ where $x_i \in \mathcal{X}$ and $\{y_i\}_{i=1}^{M}$ where $y_i \in \mathcal{Y}$. Data distribution are again denoted as $x \sim p_{data}(x)$ and $y \sim p_{data}(y)$. The model includes two mappings, i.e. two generators, $G : \mathcal{X} \to \mathcal{Y}$ and $F : \mathcal{Y} \to \mathcal{X}$. Consider two adversarial discriminator just like the usual GAN setup, $D_{\mathcal{X}}$ and $D_{\mathcal{Y}}$. The former aims at distinguishing between images $\{x\}$ truly emanating from the $\mathcal{X}$ domain and translated images $\{F(y)\}$, the latter aims at discriminating between $\{y\}$ and $\{G(x)\}$. The model is comprised of two types of losses :

(i) a usual *adversarial loss* for matching the distribution of generated images to the data distribution of the target domain.

(ii) a *cycle consistency loss* to prevent the learned mappings F and G to contradict each other.

For the (i) adversarial loss, the objective can be expressed as in the Vanilla GAN framework. Consider the discriminator $D_{\mathcal{Y}}$, then

$$\min_G \max_{D_{\mathcal{Y}}} \mathcal{L}_{\text{adv}}(D_{\mathcal{Y}}, G) = \mathbb{E}_{y \sim p_{data}(y)}[\log D_{\mathcal{Y}}(x)] + \mathbb{E}_{x \sim p_{data}(x)}[\log(1 - D_{\mathcal{Y}}(G(z)))] \tag{5}$$

where G tries to generate images $G(x)$ similar to those from $\mathcal{Y}$, while $D_{\mathcal{Y}}$ tries to distinguish between those translated images $G(x)$ and true sample y. The objective is the same concerning the generator F : $\min_F \max_{D_{\mathcal{X}}} \mathcal{L}_{\text{adv}}(D_{\mathcal{X}}, F)$. Note that this loss cannot suffice for G and F to match a x_i to a particular y_i. For instance, defined as such the mapping could translate an input to any output in the target distribution. One requires a cycle that could bring x to its original images, *i.e.*

$$x \rightarrow G(x) \rightarrow F(G(x)) \approx x$$

as well as $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$. Having this in mind, a second loss has to minimized in order for this objective to be achieved. Such loss, the (ii) *cycle consistency loss*, is defined as

$$\begin{aligned} \mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{data}(x)}[||F(G(x)) - x||_1] \\ + \mathbb{E}_{y \sim p_{data}(y)}[||G(F(y)) - y||_1] \end{aligned} \tag{6}$$

This allows to define the full objective as

$$\begin{aligned} \mathcal{L}(G, F, D_{\mathcal{X}}, D_{\mathcal{Y}}) = \mathcal{L}_{\text{adv}}(D_{\mathcal{Y}}, G) \\ + \mathcal{L}_{\text{adv}}(D_{\mathcal{X}}, F) \\ + \lambda \mathcal{L}_{\text{cyc}}(G, F) \end{aligned} \tag{7}$$

where λ controls the relative importance of the two objectives. The desired solution F^* and G^* solve :

$$G^*, F^* = \arg \min_{G,F} \max_{D_{\mathcal{X}}, D_{\mathcal{Y}}} \mathcal{L}(G, F, D_{\mathcal{X}}, D_{\mathcal{Y}})$$

2.2 Application : CycleGAN from MNIST to USPS

The aim of this application is to use the Cycle GAN model we have just presented in order to translate images from the MNIST dataset to the USPS dataset.

As Figure 4 and 5 show, the main differences between the two dataset are (i) the blurriness of the USPS images and (ii) the cropping of the image.
The overall model considered is shown in figure 1 while the model generator and discriminator and respectively displayed in figure 2 and 3.

2.3 Model structure

We consider for the discriminator three 2D convolutional layers with parameters : kernel size 5×5, stride 2×2 and padding 2×2. The last 2D convolutional layer on the other hands is constructed with kernel size 4×4 and stride 1×1 and no padding. Between each convolutional layer we apply batch normalization (as recommanded in the DCGAN framework) and a ReLu activation fonction ($\sigma(x) = \max(0, x)$).
The generator is constructed as follows : two 2D convolutional layers with the same parameters as those discribed above concerning the three first convolutional layers of the discriminator. Then a Residual block is considered after those two layers which is simply constructed as a convolutional layer with parameters : kernel size 3×3, stride 1×1 and padding 1×1. Then the following upsample 2D convolutional layers is constructed with parameters kernel size 5×5, stride 1×1 and padding 3×3, while the last one is considered with the same parameters but without any padding. Between each layer a batch normalization is applied and the considered activation function is again the ReLu fonction except for the last layer for which we consider the tanh function ($\sigma(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$).

2.4 Model hyperparamters

The model parameters that we consider are the following :

- λ in equation 7 is set to be : 0.045. The higher the λ the more weight on the cyclical component of the objective function : *i.e.* the more importance we give to the resemblance of the original image with the image that went through both generators.

- The learning rate of the gradient descent is set to 0.0002. It controls how much to change the model in response to the estimated error each time the model weights are updated. A value too small may result in a long training process that could get stuck, whereas a value too large may result in learning a sub-optimal set of weights too fast or an unstable training process. The one we chose allowed us to obtain very satisfactory results as discussed in the next section.

3 Results

Figure 6, 7, 8 and 9 display respectively the generated images from MNIST to USPS and USPS to MNIST after 200 iterations and 1000 iterations.
These figures seem to show that we managed to generate satisfactory images in both states even after 200 iterations. The gain of increasing from 200 iterations to 1000 seems quite small as there are no obvious differences.
The MNIST generated from USPS images display the same characteristics as true MNIST images and look very much like their original image in the USPS dataset, similar results can be observed for USPS images generated from MNIST images. The success of our model mostly lies in the simplicity of the images considered : black and white images, background of the same color and very few different signs represented. However, the generated images are very encouraging and our model has managed to satisfactorily estimate the mapping between the USPS and MNIST datasets.

References

[1] Martin ARJOVSKY and Soumith CHINTALA and Léon BOTTOU, Wasserstein GAN, . arXiv preprint,arXiv: 1701.07875, 2017.

[2] I. GOODFELLOW, J. POUGET-ABADIE, M. MIRZA, B. XU, D. WARDE-FARLEY, S. OZAIR, A. COURVILLE, and Y. BENGIO. Generative adversarial nets. In Advances in neural information processing systems, pages 2672–2680, 2014.

[3] Jun-Yan ZHU, Taesung PARK, Phillip ISOLA, and Alexei A. EFROS. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks", in IEEE International Conference on Computer Vision (ICCV), 2017

[4] Alec RADFORD and Luke METZ and Soumith CHINTALA, nsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, . arXiv preprint,arXiv: 1511.064345, 2015.

[5] SPRINGENBERG, Jost Tobias, DOSOVITSKIY, Alexey, Brox, Thomas, and RIEDMILLER, Martin. Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806, 2014.

[6] MORDVINTSEV, Alexander, Olah, Christopher, and TYKA, Mike. Inceptionism : Going deeper into neural networks. `http://googleresearch.blogspot.com/2015/06/inceptionism-going-deeper-into-neural.html`.

Appendices

A Proof of (1)

The proof of (1) relies on the hypothesis that

$$E_{z\sim p_z(z)}[\log(1-D(G(z)))] = E_{x\sim p_g(x)}[\log(1-D(x))]$$

which allows to have the following equality :

$$\begin{aligned}&\int_x p_{data}(x)\log(D(x))dx + \int_z p_z(z)\log(1-D(G(z)))dz\\ &= \int_x p_{data}(x)\log(D(x)) + p_g(x)\log(1-D(x))dx\end{aligned}$$

define $f(y) = a\log(y) + b\log(1-y)$, then critical points are given by

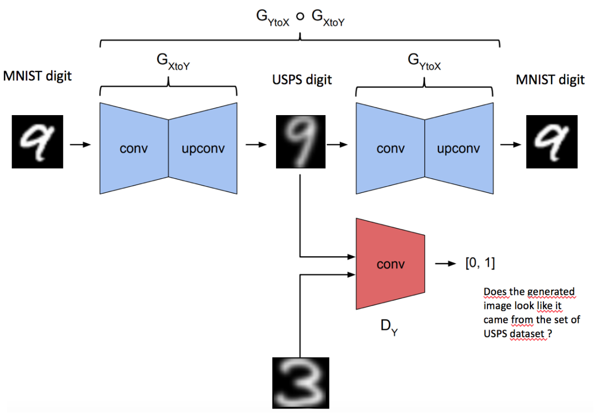
$$f'(y) = 0 \Rightarrow \frac{a}{y} - \frac{b}{1-y} = 0 \Rightarrow y = \frac{a}{a+b}$$

moreover one can show that this is indeed a maximum as the second derivative is negative when $a, b \in (0,1)$.

B Application

B.1 Model Structure

Figure 1: Cycle GAN model : Structure



$G_{YtoX} \circ G_{XtoY}$
MNIST digit
G_{XtoY}
USPS digit
G_{YtoX}
MNIST digit
conv
upconv
conv
upconv
conv
[0, 1]
Does the generated image look like it came from the set of USPS dataset ?
D_Y

Figure 2: Cycle GAN model : The Generator

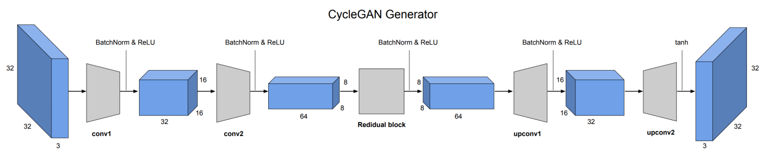
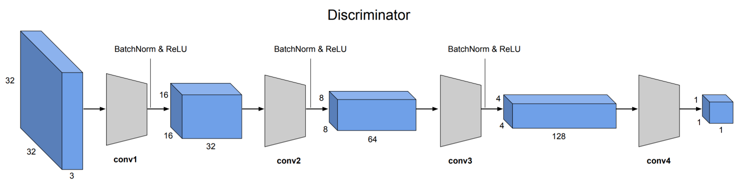


Figure 3: Cycle GAN model : The Discriminator



B.2 Dataset images

Figure 4: MNIST subsample

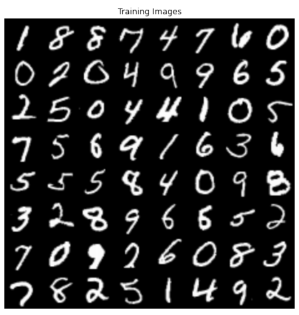
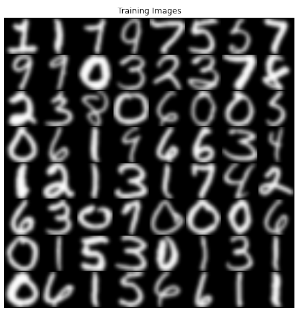


Figure 5: USPS subsample



B.3 Cycle GAN results

Figure 6: USPS images generated from MNIST images (200 iterations)

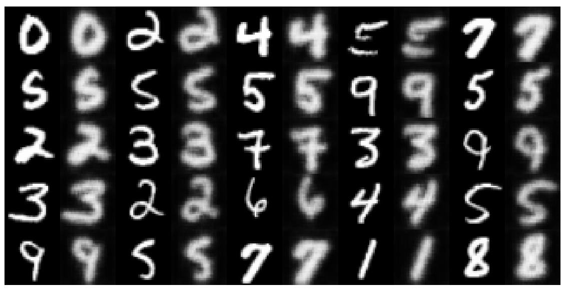


Figure 7: MNIST images generated from USPS images (200 iterations)

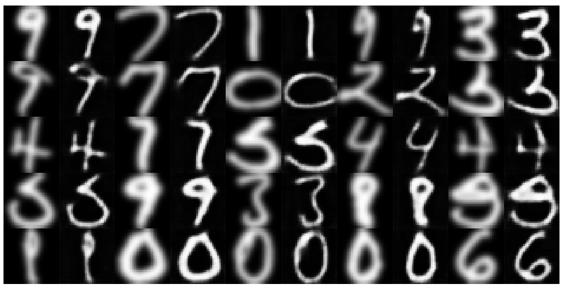


Figure 8: USPS images generated from MNIST images (1000 iterations)

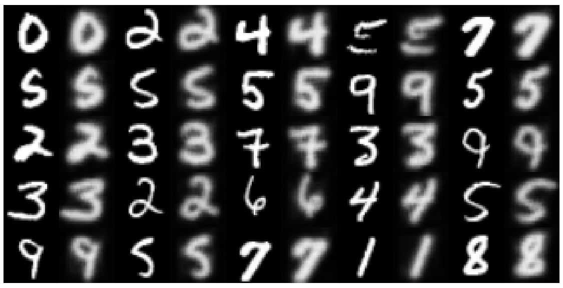


Figure 9: MNIST images generated from USPS images (1000 iterations)