IMAGE ENHANCEMENT WITH GENERATIVE ADVERSARIAL NETWORKS

Pierre Le Jeune - s182169

ABSTRACT

Exposure is a fundamental component of a good picture. However, it can be quite challenging to set the camera parameters to get it right. Overexposed shots can be corrected, but this also demands some expertise. In this work, we try to show that this correction can be automated using deep learning. We use conditional adversarial networks in order to correct overexposed images. We mainly build our method on previous work from [1] that introduced a successful GAN architecture for image-to-image translation problems. On top of that, we make use of more recent techniques to improve the quality of the reconstructions, such as spectral normalization, noise injection and a custom loss design.

1. INTRODUCTION

Taking pictures with a camera is not always straightforward, many parameters must be adjusted in order to get a good looking image. One of the most important feature of a picture may be its exposure. The exposure represents the quantity of light received by the sensor of the camera and can be tuned with three parameters: the lens aperture, the shutter time and the film speed. The two first change directly the amount of light that gets into the camera. The latter one changes the sensitivity of the sensor to light. Such a sensor have a limited range of light intensity that can be perceived. This means that any light intensity above (resp. under) this range will be seen represented as white (resp. black) in the picture. These white (resp. black) areas on a are called overexposed (resp. underexposed). Changing film speed allows to shift this range: a small film speed will capture low intensity lights while a big one will capture high intensity lights. Nowadays, it corresponds to the sensitivity of the sensor but it was with analog photography the sensitivity of the photographic film inside the camera. Adjusting these parameters may be tricky, but most of the recent cameras propose automatic suggestions in order to make photographer's lives easier. Nonetheless, in some cases when a scene has both high and low light intensities (e.g. backlighting photography), it is not possible to avoid overexposure or underexposure. Now, this problem can be avoided using high dynamic range images (HDR). These images are generated using multiple 'standard' range images taken with different exposure. This allows to get details from both dim and bright parts of a scene. The combination of these images is called tone-mapping.

However, HDR images require both cameras that are able to generate such images and devices that are able to display it. Most of the screens we use in our day-to-day life cannot show all the tones present in a HDR image. These equipment are still expensive and not widespread. Another solution is to modify images after the shot was taken in order to correct overexposure. The main idea of this work is to find a method able to adjust the exposure of an image in order to get close to an optimum exposure. Here the optimum exposure is an exposure that looks 'natural' for the human eye, i.e. without overexposed (or underexposed) parts. The problem can then be split in two. First, the algorithm must find how much to shift the overall exposure of the picture in order to make it more natural. Then, it needs to reconstruct missing parts that were outside the range. The latter part is clearly the most difficult, the algorithm only gets the overexposed picture and has no information about the missing parts.

In order to address such problems, we decided to make use of generative adversarial networks (GANs), a class of artificial neural networks introduced five years ago in [2]. These kinds of networks have shown their ability to produce high quality samples just from random noise. They have been used in a conditional setting as well [3], conditioned on labels and more recently conditioned on images [1]. These networks will be mainly based on convolutional layers, which have been extensively used in deep learning and computer vision for almost a decade.

2. RELATED WORK

2.1. Generative adversarial networks

Generative adversarial networks (GANs) are a class of deep generative models introduced in [2] that have been very successful in the last few years, achieving state-of-the-art results of many different generative tasks such as style transfer [4, 5], image-to-image translation [1], text-to-image [6], super resolution [7] and many others. The key idea behind GANs is to train two neural networks at the same time. One, called the generator, will generate data, given some random noise, and try to match the dataset distribution. The second one, called the discriminator, will learn to distinguish between real data, from the dataset, and generated samples created by the generator. More precisely, the discriminator will compute the probability that a sample came from the true distribution. In a way, it can be seen that the two networks are competing against each other: the generator tries to fool the discriminator while the discriminator aims to spot the fake samples created by the generator. In order to enforce these behaviours, both networks must be trained based on their mistakes, meaning that the generator must be penalized when the discriminator spots a fake. And when it fails, the discriminator itself is penalized. The networks are trained in order to find a solution to the following minimax game:

$$\min_G \max_D \mathbb{E}_{x\sim p_{data}(x)} \log(D(x)) + \mathbb{E}_{z\sim p_z(z)} \log(1 - D(G(z))) \tag{1}$$

Here p_{data} represents the distribution of the training dataset, which is the distribution that the generator G must learn. p_z is a prior distribution on the random noise fed to the generator as an input. It is most of the time a normal distribution. G and D represent the functions estimated by the generator and the discriminator respectively. In a general setting (i.e. if G and D could be any function) there would be an unique solution to this game, that is, $G(z) = x \sim p_{data}(x)$ and $D(x) = 1/2$.

In practice however, things are not so well behaved. First the functions estimated by neural networks are not arbitrary functions. Therefore, it is not certain that the unique solution to the game is reachable. Then, it has be shown that training directly to the objective described in (1) often leads to convergence issues [8]. In order to prevent that, one can train the discriminator and the generator on slightly different objectives:

$$\mathcal{L}_D = \mathbb{E}_{x\sim p_{data}(x)} \log(D(x)) + \mathbb{E}_{z\sim p_z(z)} \log(1 - D(G(z))) \quad (2)$$

$$\mathcal{L}_G = \mathbb{E}_{z\sim p_z(z)} \log(D(G(z))) \quad (3)$$

$\mathcal{L}_D$ and $\mathcal{L}_G$ are the losses that will be optimized by the two networks. The original GAN framework proposed $\mathcal{L}_G = -\mathcal{L}_D$ but the loss describe in (3) gives better results in practice.

2.2. Conditional generative adversarial networks

Training using labels seems to improve the generative performances of the GANs [3]. The main idea is to give to the generator some information in addition of the random noise. This can be class labels for instance and it will force the generator to create samples from a specific class. Then, the discriminator is also given the class and must distinguish between fake and real samples within each class. This technique improves the quality of the samples and stabilizes the training. This idea can be used as well for image manipulation. Instead of just giving random noise and class labels, one can give images to the generator. The generator must then modify the image in order to match with the training distribution. In this setting, we will talk about reconstructed images rather than generated ones. This is the key idea of image-to-image translation [1] and this is what we have used for this project. In that case, we give two images to the discriminator: the same that was fed to the generator and either the reconstructed one or the original. The discriminator must then output a probability for the latter one to be reconstructed (i.e. fake) or the original (i.e. real).

This method can be applied with various datasets and have already been used for multiple tasks: satellite image to map conversion [1], underwater image color correction [9], JPEG compression artifacts removal [10] and plenty others. Most of the time, this method is used without noise in the generator and is therefore quite different from the usual GAN framework. Moreover, in most applications, an additional supervised loss is used during training. Of course, this improves a lot the quality of the reconstructed images, but it moves the model away from the original unsupervised GAN framework.

2.3. Supervised losses for image-to-image translation

Plenty of losses have been used for such a purpose. First, we used a content loss that simply penalizes the generator when the reconstructed image is different from the target. One way to do so is to add a L1 loss between the reconstruction $G(x)$ and the target reconstruction y.

$$\mathcal{L}_{content} = \mathbb{E}_{x,y}\Big(\|y - G(x)\|_1\Big) \quad (4)$$

Another widespread technique is to make use of a perceptual loss in order to improve the quality of the reconstruction. This have be seen in many previous works [7, 11, 10]. This relies mostly on transfer learning. A pre-trained classification network, typically VGG-19, is used. The reconstruction and target images are fed to this network and the feature maps at different levels in the network are compared and used to assess how alike the two images are. Intuitively, the reconstruction and the target should be perceived the same way by VGG-19. To do so, a new term is added to the generator loss:

$$\mathcal{L}_{perceptual} = \mathbb{E}_{x,y}\Big(\sum_{i\in\mathcal{C}} \frac{1}{W_i H_i} \|\phi_i(y) - \phi_i(G(x))\|_1\Big) \quad (5)$$

The output function of the i-th feature map of the classification network is represented by ϕ_i and $\mathcal{C}$ is the set of feature maps chosen for the loss. In most applications only a few feature maps are used in the perceptual loss. This is not optimal because two images could be similar for some feature maps and rather different for others. In our implementation we will use 5 features maps, regularly spaced in the network, out of the 16 available in VGG-19 (only convolution layers matters here). This gives a good estimate of how similar the two images are perceived by the network and reduces the processing time. Finally, the contribution of each feature map to the loss is normalized by the size of the feature map, in order to give the same weight to each feature map.

3. METHOD

3.1. Network architecture

In order to tackle the overexposure correction we chose a similar architecture as the one proposed in the Pix2pix paper [1].In this work they proposed a fully convolutional U-net generator. It is composed of 4 down-sampling blocks. Each of them is composed of two convolutional layers (with stride 1) and one pooling layer (see Figure 1). The up-sampling part of the network is basically the same with transposed convolutions instead of the pooling layers. This works basically as an autoencoder, the down-sampling part encodes the image characteristics into a latent space, which is then decoded by the up-sampling part. The main difference is that there are residual connections between the encoding and the decoding parts. This allows to keep the original information present in the image along the reconstruction. Batch normalization is used in every layer except for the first and the last, as recommended in [1]. This encoder-decoder structure allows the network to learn a latent representation of the image structure. In the case of the overexposure correction, the input image and the desired reconstruction share a lot of structure. Once the structure is encoded, the decoder can reconstruct the image with a correct exposure.

The discriminator is as well composed only of convolutional layers. Unlike in the generator though, the convolutional layers have a stride of 2 and there are no pooling layers between convolutions (see Figure 1). This is a slight difference from the original GAN framework where the discriminator just outputs a probability of the input being real. Here the output of the discriminator is a 2D-map where each point gives the probability of a $N \times N$ region of the input image to be real. The size of each region is called the field of view of the network. Such a network sees the image as a Markov random field where two pixels separated by greater distance than the width of the field of view are considered independent. Therefore these pixels can be classified independently. It means that the discriminator only focuses on structures of a size relatively similar to the size of its field of view (see Figure 2). This suggest that the field of view of the discriminator will have a key role in how good it performs. Hence, it will probably have a great influence on the overall training (see Section 4 for more detail).

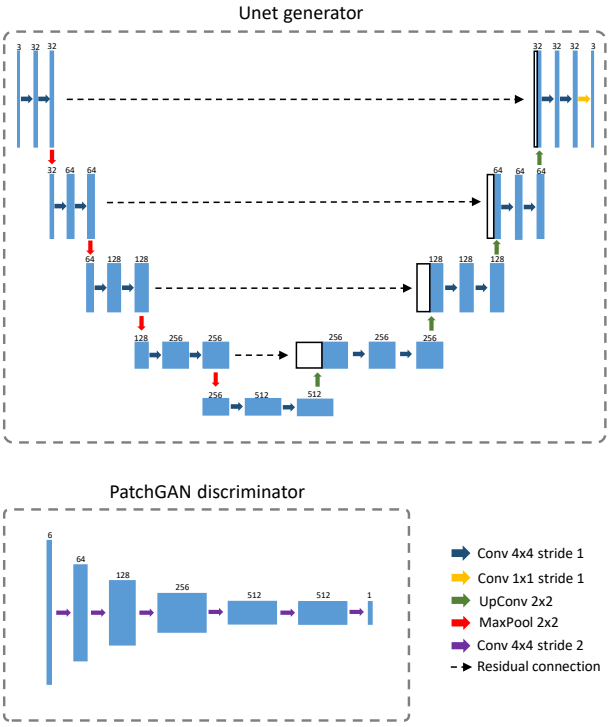


Fig. 1: Architectures of the generator and the discriminator used in our work. Note that these architectures were not used for all experiment. In particular the experiments about the field of view needed slight modifications in the convolutions.

In order to stabilize the training, we decided to use spectral normalization [12] both in generator and discriminator, on all convolutional layers. Spectral normalization is a way to enforce the lipschitzian constraint on a network. This constraint has been proven to be efficient in previous work such as [13]. The main idea behind spectral normalization is to normalize the weight matrices in the network by their spectral norm, i.e. by their largest eigenvalue. This restricts the outputs of the network the be within a certain range:

$$\|G(x) - G(y)\| < \|x - y\| \tag{6}$$

This constraint implies an interesting property on the gradient of G, that is it must be smaller than one. This prevents the gradient to explode and stabilizes a lot the training.

3.2. Loss design

Instead of using the original GAN objective we chose to use the least squares GAN objective [14]. It has been proven to be more stable and, to some extent, produces higher-quality images. This objective can be expressed as follows:

$$\mathcal{L}_D = \frac{1}{2}\mathbb{E}_{x\sim p_{data}(x)}\left(D(x)^2\right) + \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}\left((1-D(G(z)))^2\right) \tag{7}$$

$$\mathcal{L}_{LS} = \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}\left(D(G(z)^2\right) \tag{8}$$

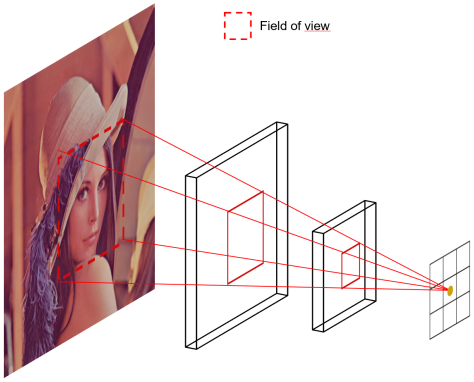


Fig. 2: The field of view of a convolutional network correspond to the area of the input image that is used to compute the value of one point of the output map.

As mentioned above, the loss design is crucial for image manipulation, and in a supervised setting as this one, perceptual and content losses can be used to significantly improve the quality of the reconstructions. Training without these losses is less stable, produces high frequency artifacts in the reconstructions and add blur to the output. This will be discussed in more details in the next section. Hence in order to have some flexibility on the loss design, we defined an overall objective for the generator:

$$\mathcal{L}_G = \mathcal{L}_{LS} + \lambda_c\mathcal{L}_{content} + \lambda_p\mathcal{L}_{perceptual} \tag{9}$$

Here λ_c and λ_p are parameters that allows to tweak the objective to put more weight on perceptual loss or on content loss. The perceptual and content loss that we used are the ones defined in section 2.3.

3.3. Noise injection

As mentioned before, in this setting, there is no noise introduced in the model at any time. This is a great change from the GAN framework that is on the contrary based on noise to generate samples. Adding noise to our model would perhaps give the generator some 'inspiration' in order to fill the overexposed areas where all information is missing. In order to do so, we decided to introduced noise after each convolutional layer. This noise is directly added to the feature map as proposed in [5]. The noise is sampled from a normal distribution and then scaled by a set of learnable, per-channel parameters (see Figure 3). A new sample is drawn for each convolution. The consequences of this are further described in section 4.

3.4. Quality assessment

It is easy for a human-eye to tell if a picture is correctly exposed or not. An image is rightly exposed when it looks natural for us, i.e. as if we were looking to the scene with our eyes. However this is a clearly subjective way of assessing the performance of our model. Plus, it is quite complicated to transpose this criterion for a machine. In order to assess the quality of the reconstruction, we need to find an objective way to assess the quality of the reconstructions. To do so, we propose two different metrics: one will assess the similarity between the reconstructed image and the original one and a second will assess how 'natural' are the generated samples.

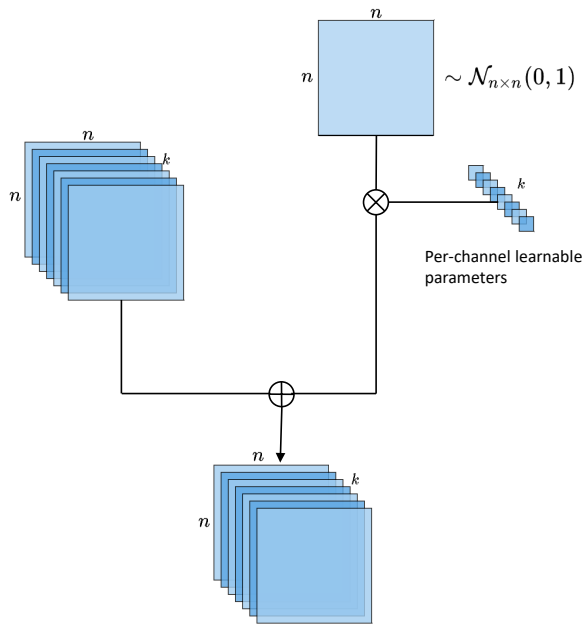


Fig. 3: After each convolution, noise is added to the feature map. The noise is sampled from a normal distribution and is scaled by a learnable set of parameters before being added to each channel.

3.4.1. Histogram similarity score

The first thing that we want to measure is how close the reconstruction is from the target. This could be rather easily be measured by taking the L1 Loss between these two images. However this is certainly not a good way to proceed as there might exist plenty of acceptable reconstructions for one overexposed image. The L1 Loss however will only have one unique solution, that is the reconstruction is identical to the target. Therefore, another metric must be introduced. The one chosen was proposed in [15]. Their work is partly focused on exposure as well, therefore it seems reasonable to use their metric. This score measures the similarity between two datasets by computing distribution histograms of three features of the images: the luminance, the saturation and the contrast, which are closely related to the exposure. These three quantities are computed for each image and histograms are built for each quantity and each dataset. We can then compare how similar the datasets are in order to assess the quality of the reconstructions. When the histograms of the original dataset and the reconstruction dataset are highly similar, it means that the images are looking very similar in terms of luminance, contrast and saturation. It seems that this is a good way of assessing the exposure correction as the overexposed dataset have a poor similarity on these three quantities compared to the original one and the more overexposed the images are, the lower the similarity (see Table 1).

3.4.2. Inception Score

The second metric that we chose is the Inception Score (IS) which was introduced in [16]. This metric has been extensively used in the last few years to assess the quality of GAN's samples as it correlates well with how 'natural' the samples look for a human observer. This is exactly what we want for our reconstruction. The IS is computed

Threshold	0.2	0.5	0.8	1
Luminance Sim.	33.86%	54.13%	66.38%	100%
Contrast Sim.	61.12%	66.92%	77.62%	100%
Saturation Sim.	38.41%	52.03%	69.98%	100%

Table 1: Similarity scores between overexposed dataset with different thresholds and the original ones. The more overexposed, the less similar they are.

using the inception network V3, a pre-trained classification network on ImageNet. It measures how diverse are the generated samples and how well each sample represents a defined object. To do so, we compute the probability of each class, conditioned on an image (i.e. this is just the output of the classification network) and the marginal probability (approximated by averaging all the conditional probabilities) of all images. The conditional probabilities should be focused, i.e. a high value for one class and small ones for all others as the object in the image should be clearly recognizable. The marginal probability therefore should be close to the uniform distribution as all the classes in the dataset should be equally represented in order to have diverse samples. The Kullback-Leiber divergence between the condition and marginal probabilities gives a measure of how dissimilar these distributions are. It will give the highest value for a completely focused conditional distribution and a uniform marginal one. Therefore a high IS will indicate high quality samples.

3.5. Dataset generation

In order to train such networks, we need to have a lot a images and more importantly we need to have pairs of overexposed and correctly exposed (that we will call originals) images. The best way to do so is to simulate overexposure on an existing dataset. Here we chose to use a subset of the Places365 dataset. The first thing to do is to remove all the images that are overexposed or underexposed. It is important to remove overexposure from the dataset as it will be used as example of correct exposure to train the network. We also want to control the amount of overexposure in our experiments so it is crucial to have no overexposure before simulating it, otherwise this amount may be variable. Actually, the amount of overexposure simulated will never be exactly the same between two images as it depends and the luminance of the original image. An image with a high luminance will be more overexposed than an image with a lower luminance, even if the same overexposure is simulated.

The simplest way to simulate overexposure would be to directly clip the pixel intensities above a certain threshold. This of course will generate some overexposure but this is not the way it happens when a real picture is overexposed. As mentioned before, overexposure happens when light intensity is outside the range of detectable light intensities of the camera sensor. Therefore to simulate overexposure, we need to clip the light intensities of the image. However, our eyes are not sensitive linearly to the light intensity. The human-eye is capable of distinguishing tones in very dim and very bright areas at the same time, which suggest that the visual cells' response follows a logarithmic law. This is known as the Weber-Fechner law [17] but it can be approximated by a power law (Stevens's power law). In order to make the picture close to what our eye sees, a power law is applied to convert the light intensities captured by the camera sensor into pixel intensities. This is called the gamma correction, for the exponent parameter in the power law:

$$I_{pixel} = I_{light}^{\gamma} \tag{10}$$

A typical value for this parameter is $\gamma = \frac{1}{2.2} \approx 0.45$ and this is the value we used for all our experiments. Then, if we want to simulate overexposure, we need to clip the light intensities and not the pixels intensities. Therefore, we must convert the pixel intensities into light intensities before clipping and then convert them back. This is illustrated in Figure 4. The last step to simulate the overexposure is to scale the light intensities once they are clipped in order to use all the range of light intensities. Note that in practice cameras use slightly modified gamma curve as a correction. This is called the camera curve and it can vary from one camera to another.

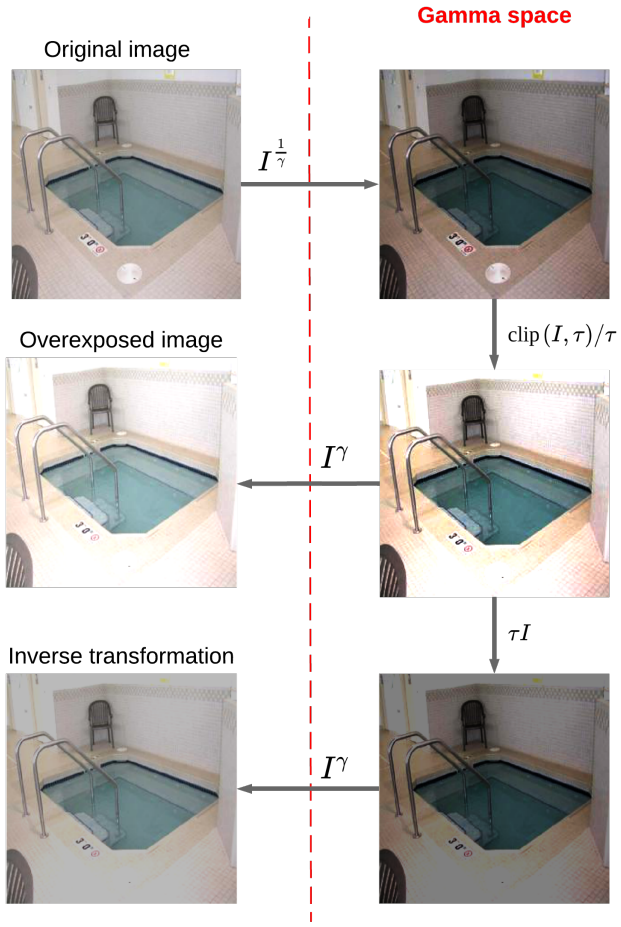


Fig. 4: The overexposure process: each image is first mapped to the so-called gamma space where pixel intensities are proportional to the light intensities of the actual scene. The intensities are then cropped above a certain threshold τ and finally the image is mapped back to the original space.

3.6. Inverse transformation baseline

At this point, one might think that the overexposure can then be corrected quite easily just by reversing the simulation operation presented above. This is demonstrated in Figure 4 and this method does correct the overexposure to some extent. However the overexposed areas are completely lost, and this cannot be addressed this way. Nevertheless this gives a reasonable baseline to compare with.

4. EXPERIMENTS AND RESULTS

4.1. Adding noise to the generator

As mentioned above, we decided to add noise into the generator in order to improve the quality of the reconstructions. This was mainly motivated by the hope that the noise would give some inspiration to the network in order to reconstruct the overexposed areas. This noise was added after each convolution layer in the generator (see Figure 3). However it was not clear if noise was necessary in both the encoder and the decoder part of the generator. At first, we tried with noise only in the decoder part. The encoder part should only wrap the information about the image into the latent space. The reconstruction is done in the decoder part and here only we would expect that noise is needed in order to inspire the network. Or at least we would expect that if we were using a true auto-encoder network. There are some important differences between our generator and an auto-encoder. First, the latent space is quite large and it might not be small enough to force the network to encode all the information of the image and then reconstruct it. Meaning that some part of the reconstruction may happen in the encoder part as well. Second, there are some residual connections between the same-level blocks of the encoder and decoder part. This also relax the constraint on the encoding-decoding scheme as the information about the original image can be retrieved through the residual connections and does not need to be encoded that well. This emphasizes the fact that the reconstruction may not only happen in the decoding part of the generator. Therefore, it may be interesting to add some noise as well in the encoding part. In fact, it performs better with noise everywhere in the generator (see Table 2). Even though it performs better on the selected metrics, it is not clear that the reconstructions are better when looking at it (see Figure 5).

(a) No noise (b) Decoder only (c) Encoder+decoder

Fig. 5: Comparison of the reconstructions with noise in different part of the generator.

	No noise	Decoder only	Encoder + decoder
IS	18.00	18.23	**19.26**
Luminance Sim.	88.44%	89.73%	**89.97%**
Contrast Sim.	88.29%	87.38%	**88.74%**
Saturation Sim.	83.23%	80.62%	**83.55%**

Table 2: Performance comparisons with different noise injection. It seems clear that noise improve the performances and that it should be added both in the encoder and the decoder.

4.2. Loss design

After trying different ways to add noise in the generator, we tried to tweak the loss by changing the weights λ_c and λ_p of its components (see eq. (9)). By default we chose $\lambda_c = 50$ and $\lambda_p = 100$. Of

course, the higher these two weights, the farther we are from a true GAN framework as the influence of the adversarial loss decreases. However, when both λ_c and λ_p are small, the reconstructions are blurry and some generation artifacts are quite visible (see Figure6). Therefore, a compromise has to be found in order to get both nice reconstructions and still make use of the adversarial setting. In the end, it seems that the weights selected by default were satisfactory. Therefore they will be used in all the following experiments.

(a) $\lambda_c = 100$ (b) $\lambda_c = 10$ (c) $\lambda_c = 1$

Fig. 6: Effect of λ_c on the reconstructions. One can see that with small values, the output gets blurry and that generation artifacts are more and more abundant.

4.3. Different fields of view for the networks

We finally discuss the importance of the size of the field of view of the generator and the discriminator. The field of view of a convolutional network corresponds to the size of a patch of the input image that is mapped into one element of the last feature map. This means that each element of the last feature map is only influenced by the pixel that are within this patch (see Figure 2). It seems quite reasonable that this parameter will influence the quality of the reconstructions. When the field of view is narrow, the generator has only little context at its disposal to correct the exposure. This may be problematic when reconstructing areas that are largely overexposed and where no context can be find at all. On the contrary, when the field of view is large, it might be hard for the network to find what is relevant for the reconstruction as too much context is provided. This is true as well for the discriminator. Therefore, it is crucial to find a suitable size for the fields of view of the two networks.

Discriminator FoV	70	94	140
IS	**19.26**	17.99	17.65
Luminance Sim.	**89.97%**	89.23%	89.94%
Contrast Sim.	**88.74%**	87.06%	87.47%
Saturation Sim.	**83.55%**	79.75%	82.84%

Table 3: Score comparison with different field of view sizes for the discriminator. The field of view of the generator was kept constant at 47 pixels.

During the early stage of this work, and because of the discussion about the discriminator field of view in [1], we thought that only the field of view of the discriminator had some influence on the reconstruction. Therefore we tested different sizes and concluded that a field of view of 70 pixels was the best for the discriminator (see Table 3). However, this experiment has some flaws. Firstly, we did not test values smaller than 70 (which was the suggested value from [1]). We found that 70 gave the best performances but only compared to larger sizes, smaller field of view may perform equally or

Generator field of view	70	140	140
Discriminator field of view	70	70	140

IS	21.18	**21.46**	21.2
Luminance Sim.	**89.46**	88.11	89.53
Contrast Sim.	89.67	**90.63**	90.4
Saturation Sim.	83.88	**84.73**	84.48

Table 4: Score comparison with different field of view sizes for the discriminator and the discriminator. Scores are higher than in Table 3 because between these two experiments we added the perceptual loss which provide a consequent boost of the performances.

even better. Secondly, we realized after some time that the generator field of view was also important to the quality of the reconstructions. Therefore, we started a new batch of experimentation to investigate this. In the meantime, we had introduced the perceptual loss on top of the content loss which increased a lot the performances. We did not had the time to conduct again all the experiments from above to have a proper comparison. Nevertheless, we tried to vary the field of view of the generator and it seems that the best combination is, after all, 140 pixels for the generator and 70 for the discriminator (see Table 4). Again, we did not try smaller sizes, but as mentioned before, small fields of view provide less context and are likely to perform worse. In Figure 7 are compared the reconstructions with the different sizes of the fields of view. It is not obvious from these images which architecture performs the best. Even if it is quite subjective, we found that the last one (140/140) was the one that looks the most natural and which is the closest from the original image.

(a) 70/70 (b) 140/70 (c) 140/140

Fig. 7: Effect of the sizes of the fields of view of the networks. From these images it seems that reconstruction (c) with a the generator and the discriminator at 140 pixels produces the best results. However this is not in agreement with the performance scores (see Table 4).

4.4. Results

These experiments helped us improve our architecture quite well. In all the previous experiments, we trained the networks for approximately 10 hours. However it seemed that it was not fully converged. Therefore, we trained our optimal model for one and a half day. It achieved even better results. We also run a full training with random overexposure. Up to now, in all our experiments the images were overexposed by the same amount. The intensities above $\tau = 0.5$ were cropped (in the linear space). Now, a difficulty is added as τ is chosen randomly between 0.5 and 1 for each image. It turns out that it performs a little bit better than with constant overexposure (see Table 5). This may be explained because the random threshold is picked randomly between 0.5 and 1. In average the threshold is then 0.75, so there is less overexposure in that case. Of course, the

network still needs to figure out by itself the amount of overexposure to correct. The reconstructions can be found in Figure 8 The last row presents results from the experience with random overexposure while the two first rows shows images with constant overexposure. One can see that the overall exposure of the images is well corrected and is very similar to original ones in both cases. However the reconstructions of the highly overexposed areas, those where all information was missing, are not so nice. Actually, those are almost not reconstructed at all, only the edges of these areas are a bit reconstructed. But it is hard to tell just from the comparison of these images. A better way to do so is to compare by taking the difference between, the reconstruction and the trivial solution (see Section 3.6), and the original image. It will make it easier to assess both how close is the overall exposure and how good are reconstructed the missing areas. A perfect reconstruction should appear completely black. These comparisons can be found in Figure 9. One can see that the non-black areas are quite smaller with our method, compared to the trivial solution. Strangely, the network tries to correct some areas that should not require any modification. The wall behind the pool in the previous Figure could just be corrected by the overall exposure correction but the network modified it specifically. This may be due to the fact that the network never uses high intensities in its reconstructions. See section 5.2 for more details on this point.

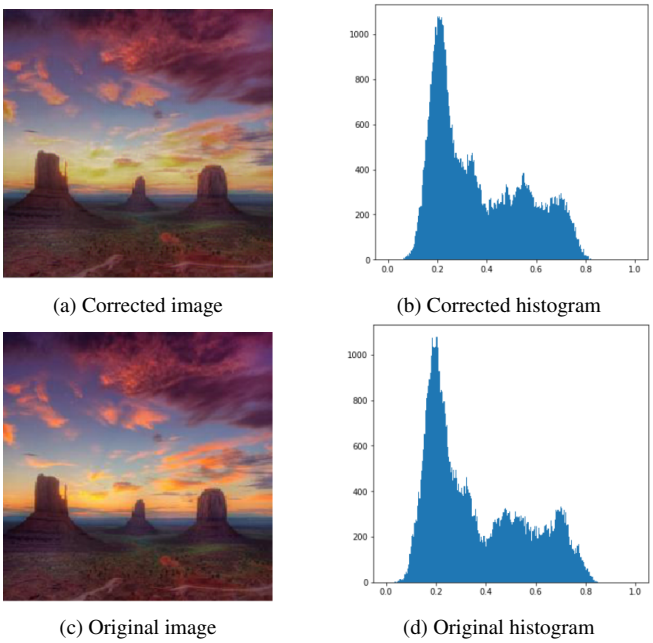
5. DISCUSSION AND FUTURE WORK

5.1. Non-stability of the experiments and confirmation bias

GANs are well known to be unstable during training. In this work we used a couple of tricks to make it more stable and reproducible (LSGAN loss and spectral normalization). However, this does not fully address the problem. There are still some instabilities in the training and running the same experiments twice can give quite different results. We did not have the time to run each of the experiments multiple times in order to make an average of the performances. This would have been more reliable for sure. Instead we only ran the experiments once but when the scores of two experiments are close, it is hard to conclude which method or model does actually perform better than the other. This mostly concerns the field of view experiments where all the results are very close (see Table 4). In a similar way, it is sometimes hard to tell which of the reconstructions looks better than the other. This could be solved by doing some user studies where humans are asked to choose between different reconstruction or to rate them. It would give an unbiased metric to assess the quality of the network. During the experiments, we had a hard time avoiding the confirmation bias suggested by the scores. When two scores were close, it was somehow easier to find flaws in the reconstructions of the model that performed the worse. In such a situation the user studies would have been really interesting, as it would have been more objective and unbiased. But this takes time to conduct and therefore we did not make use of it.

5.2. Generators are afraid of overexposure

An important flaw common to all our models is that the generator does not produce images with high pixel intensities. It always learns somehow a limit of pixel intensities that should not be crossed (see Figures 10 and 11). It is like the generator is afraid of using high intensities in the reconstructions because it looks like overexposure. However, some parts of a picture could have high intensities without being overexposed. This limit prevents the networks to produce image with overexposed area but it is too restrictive. It is most of the time around 0.8. Thus, any image that has light intensities above this value would never be perfectly reconstructed. A way to address this issue may be to add a new term to the loss to enforce the network to produce image with a larger contrast. This may be done in a supervised manner with a term that tries to match the contrast of the reconstruction and the original image.



(a) Corrected image (b) Corrected histogram

(c) Original image (d) Original histogram

Fig. 10: Comparison of histograms of the original image and the correction made by our network. For image with low pixel intensities, the limit cannot be seen and the reconstruction is of a very good quality.

5.3. Working with different resolution

Even if our architecture is capable of dealing with any image size, it is unlikely to perform equally good for any size. The sizes of the fields of view of the networks are crucial for the quality of the reconstructions. But this optimal size is likely to be dependent on the input image size. In order to understand it, let's take two images of the same scene with two different resolutions. On the larger one, the field of view contains less information about the context than on the smaller one. Therefore it may be harder to reconstruct a bigger image. Furthermore, it has been proven that the pix2pix architecture does not work very well with large images [18]. In this work they also proposed a way to address this issue. For the generator, they first trained a regular pix2pix network on low resolution images. Then they added a down-sampling and an up-sampling block at each end of the network in order to increase the resolution of the images. For the discriminator, they used three different networks with different field of view sizes, to be able to deal with different scales. The same problem is also addressed by [19] in a slightly different way.

5.4. Filling the missing part

Finally, the biggest flaw of our method is that it cannot reconstruct properly the missing areas. However, this looks like an inpainting

(a) Overexposed (b) Inverse transformation (c) Our correction (d) Ground truth

Fig. 8: Here are some examples of reconstructions generated with our method (c). This is compared to the inverse transformation defined in section 3.6 (b) and the ground truth (c), i.e. the original image before overexposure. The first column shows the overexposed images. The last raw has been generated with the network trained with random overexposure, for this particular image the random threshold was $\tau = 0.69$.

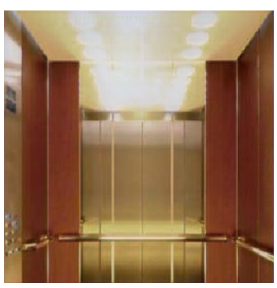
(a) Original (b) Inverse mapping (c) Corrected (d) Original-inverse (e) Original-corrected

Fig. 9: Comparison of the reconstructions using the inverse transformation and our network. In order to make the comparison easier, we took the difference in pixel intensities between the original image and each reconstructions. Zooming in, one can see that even if our method produces some artifacts in the overexposed areas, the edges of this area are better reconstructed than with the inverse transformation. While the rest of the image is similar.

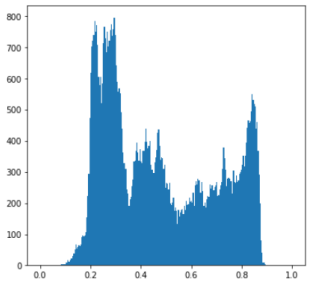
problem: an image has some missing parts and we want to reconstruct these areas based on the overall context. This has been addressed in multiple works such as [20, 21]. The only difficulty here is to provide a mask of the missing area that we want to reconstruct. One way to do so may be to use the output of the discriminator. It gives a probability for each patch of the input image to be real or fake. Therefore we could build a mask with all the patches where the probability of being fake is above a certain threshold. This could then be fed into the inpainting network to reconstruct the missing parts.

	Baseline	Baseline +Noise	Baseline+ noise+ perceptual	Random overposure
IS	18.00	19.26	**21.59**	**21.81**
Luminance Sim.	88.44%	**89.97%**	89.46%	84.56%
Contrast Sim.	88.29%	88.74%	**90.5%**	88.84%
Saturation Sim.	83.23%	83.55%	**85.01%**	83.73%

Table 5: Overall comparison of the different experiments. Note that for the to last columns, the method is the same, only the overexposure is changed. The scores from the last one have been obtained using a random overexposure, i.e. each image was overexposed with a random threshold (between and 1). The last two columns have also been optimized for a longer time (30 hours).

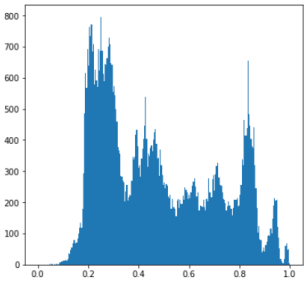


(a) Corrected image



(b) Corrected histogram

(c) Original image



(d) Original histogram

Fig. 11: For image with pixel intensities above the limit, it is clear that the network do not reconstruct it. The quality is obviously influenced.

6. CONCLUSION

In this work, we successfully built an architecture that is able to correct overexposure on picture. We conducted some experiments in order to test and improve our method. In the end, it works quite well when overexposure is relatively small. Once large areas of the picture are missing though, the network has a hard time reconstructing it and is generally able only to correct the edges of such areas. This is not surprising as this falls under another kind of problem: image inpainting. A combination of methods from this field and our work could lead to an end-to-end pipeline that would be able to deal entirely with overexposure.

7. REFERENCES

[1] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[3] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[4] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[5] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[6] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*, 2016.

[7] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[8] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[9] Cameron Fabbri, Md Jahidul Islam, and Junaed Sattar. Enhancing underwater imagery using generative adversarial networks. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7159–7165. IEEE, 2018.

[10] Leonardo Galteri, Lorenzo Seidenari, Marco Bertini, and Alberto Del Bimbo. Deep generative adversarial compression artifact removal. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4826–4835, 2017.

[11] Wenqi Xian, Patsorn Sangkloy, Varun Agrawal, Amit Raj, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Texturegan: Controlling deep image synthesis with texture patches. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8456–8465, 2018.

[12] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[13] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[14] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2017.

[15] Yuanming Hu, Hao He, Chenxi Xu, Baoyuan Wang, and Stephen Lin. Exposure: A white-box photo post-processing framework. *ACM Transactions on Graphics (TOG)*, 37(2):26, 2018.

[16] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.

[17] Vladimir Sacek. Eye intensity response, contrast sensitivity. chapter 13.8. `https://www.telescope-optics.net/eye_intensity_response.htm#level`, 2006.

[18] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018.

[19] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1511–1520, 2017.

[20] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5505–5514, 2018.

[21] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 85–100, 2018.