# INFO8010: Image Super-Resolution

**Francois Cubelier**,[1] **Nicolas Radis**,[2] and **Dimitri MARCHAND**[3]

[1] *francois.cubelier@student.uliege.be (s170818)*
[2] *nicolas.radis@student.uliege.be (s152695)*
[3] *dimitri.marchand@student.uliege.be (s196235)*

## I. INTRODUCTION

Image super-resolution (SR) refers to the task of enhancing the resolution of an image from low-resolution (LR) to high-resolution (HR). It is a classical problem in computer vision and it has many applications. It is used for improving medical imaging systems, satellite imaging, in surveillance, astronomical imaging, camera technology, and so on. As a result, in recent years, there has been a lot of research and great progress in this field. There are a lot of ways to solve this problem. Indeed, the solution will be different depending on the technique, the loss functions, the metrics and the datasets used.

The problem can be define by the following equation:

$$I^{LR} = D(I^{HR}; \sigma)$$

where $I^{LR}$ is the low-resolution image, $D$ is the degradation function, $I^{HR}$ is the high-resolution image, and $\sigma$ is the noise. The goal is to find the inverse function of degradation with only the HR and LR image datasets. In fact, the degradation function and the noise are unknown, we are provided only the high and low resolution images.

Among the many techniques that can be found, the most popular that achieved impressive results in 2015, which is also the first method using deep learning, is SRCNN (Super Resolution Convolutional Neural Networks)[1]. It is a pre-upsampling super resolution method which uses traditional techniques like bicubic interpolation and deep learning to refine an upsampled image. One year later, an improvement of this technique, called VDSR (Very Deep Super Resolution) [2] has been designed. Seeing the large amount of computational power required for the pre-upsampling methods, the same authors of SRCNN come with a new method: FSRCNN (Fast Super-Resolution Convolutional Neural Networks)[3]. This new method is faster and achieve better results than SRCNN. Indeed, the feature extraction process occurs in the lower resolution space instead of the high one, as it was the case in pre-upsampling methods. Therefore, the computational power required is significantly reduce. Moreover, instead of using simple bicubic interpolation for upsampling, a learned deconvolutional filter is used, which improves the model. The following years a lot of different other methods appeared with their advantages and drawbacks. Among them, we can cite residual networks [4][5], multi-stage residual networks [6], recursive networks [7][8], progressive reconstruction networks [9], multi-branch networks [10][11], and attention-based networks [12][13]. All these cited works and methods have all the same shortcomings. Indeed, the networks in these methods all try to optimize the pixel difference between predicted and output HR images. Even though this metric is alright, humans do not distinguish images by pixel difference, but rather by perceptual quality. It is not as easy to correctly measure the quality of an image. There exists metrics like PSNR (Peak Signal to Noise Ratio) to do so. However, a good score with those measures does not mean that the image has a good quality for the human eye.

In order to overcome this problem of perceptual quality, a new solution arisen : GAN-based architectures (Generative Adversarial Networks). Indeed, GANs will try to optimize the perceptual quality to produce images which are pleasant to the human eye. The most popular one is SRGAN [14]. This solution came up with lower PSNR scores than previous methods, but the perceptual quality was better. Indeed, the authors used the MOS (Mean Opinion Score) in order to rate the perceptual quality of the generated images. Basically, it consists to survey humans to rate the images between 0 and 5. GAN based methods are among the best one to solve the super resolution problem. Therefore, we have chosen to base our work on these methods, and, in particular, on SRGAN [14].

In this project, we propose experiments on GAN for the super-resolution problem. In particular, starting from the SRGAN [14], we will tune different parts like the loss function, the generator, and the discriminator. Moreover, we will apply the R1 regularization [15] on the discriminator's gradient . During our experiments, we progressively tried to bring some modifications to our base model, among which : removing normalization, using new loss functions, using non-classical labeling for GAN. We make these modifications in order to improve the results or at least to observe the changes they are producing. These observations are made in the Results section IV.

Basically, the main steps we are going through in order to achieve super resolution are the followings. The first step is to prepare the data to deal with. It consists in HR images that we will crop at a fixed size. The LR images are then created by downsizing the HR images by a scaling factor with a bicubic interpolation method. It allow us to use nearly any images of any datasets, and thus ensures a good scaling. The goal of generator will be to upscale the handmade LR images while the goal

of the discriminator will be to say whether an image is a real HR or one generated by the generator. Once the training is over, we can use the generator to upscale any LR image and obtain an HR image.

## II. RELATED WORKS

### A. Image Super Resolution

As briefly mentioned in the introduction, there are a lot of different techniques for super resolution with their benefits and drawbacks.

The first method using deep learning for image super resolution has been brought by Dong et al. with SRCNN[1]. It is a simple CCN architecture which consists in three layers: one for patch extraction, non-linear mapping, and reconstruction. Basically, given a LR image $Y$, the first convolutional layer of the SRCNN extracts a set of feature maps. The second layer maps these feature maps non-linearly to HR patch representations. The last layer combines the predictions within a spatial neighbourhood to produce the final high-resolution image $F(Y)$. In terms of metric, it used MSE loss function to train the network and PSNR to evaluate the results. Kim et al. [2] came up with an improvement: VDSR. Very Deep Super Resolution is using a deep network with small $3 \times 3$ convolutional filters instead of a small network with large convolutional filters. The network tries to learn the residual of the output image and the interpolated input, rather than learning the direct mapping (like SRCNN).

New methods with less computational power required and better results appeared. It consists in doing the feature extraction in the lower resolution space instead of in the high resolution one. The same authors of the SRCNN came up with FSRCNN [3]. It has better performance with lower computational cost than SRCNN. There are three main differences between the two. First, it adopts the original LR image as input without bicubic interpolation, and a deconvolution layer is introduced at the end to perform upsampling. Second, the non-linear mapping step in SRCNN is replaced by three steps in FSRCNN, namely the shrinking, mapping and expanding step. Third, FSRCNN adopts a deeper network structure and smaller filter sizes. ESPCN [16] introduces the concept of sub-pixel convolution to replace the deconvolutional layer for upsampling and solves different problems associated to it.

Methods using residual networks have also been studied. Among them, we can find EDSR[4] and CARN[5]. To deal with the task of feature extraction separately in the LR space and HR space, a multi-stage design is considered in a few architectures to improve their performance. For instance, BTSRN [6] has been proposed by Fan et al.

Other works using recursive networks also exists. Kim et al. [7] with their Deep Recursive Convolutional Network (DRCN) showed a highly performant architecture that allows for long-range pixel dependencies while keeping the number of model parameters small. Later on, an improvement has been proposed: DRRN [8]. It outperformed DRCN aswell as SRCNN, VDSR and ESPCN (measured with PSNR).

Attention-Based networks have been proposed in order to give selective attention to different regions in an image. Indeed, so far most of the networks mentioned earlier was giving equal importance to all spatial locations and channels. For instance, SelNet[12] and RCAN[13] obtained better results than the others on the PSNR and SSIM measures.

The main issue with all the techniques discussed so far is that they optimize the pixel difference between predicted and output HR images. It is not ideal because humans do not distinguish images by pixel difference, but rather by perceptual quality. For this reason, GANs are among the best model to use in order to have a pleasant super resolution. Ledig et al. presented SRGAN [14] which obtained lower PSNR values but was producing images of better quality from the human point of view. It is a GAN-based network optimized for a new perceptual loss. They replace the MSE-based content loss with a loss calculated on feature maps of the VGG network, which are more invariant to changes in pixel space. ESRGAN [17], an improvement to SRGAN, has been presented by Wang et al. They propose a few tricks to improve the results further. First, they introduced the Residual-in-Residual Dense Block (RRDB) without batch normalization as the basic network building unit. Indeed, batch normalization may help a lot for many computer vision tasks, but for image processing related tasks, such as super-resolution, it can create some artifacts. Second, with the SRGAN, the discriminator model was trained to detect whether its input is real or fake. However, in ESRGAN, the authors used a relativistic discriminator that tells whether the input looks more realistic than fake data or less realistic than real data.

### B. Metrics

The mean squared error, $l_2$, is one of the most popular error metric in many fields. Indeed, it has very convenient properties for optimization problems. However, as mentioned in the introduction, it is known that $l_2$, and consequently the PSNR (Peak Signal to Noise Ratio) do not correlate well with human's perception of image quality [18].

The Structural Similarity Index Measure (SSIM)

by Wang et al. [19] is a better tool to measure the quality of an image. It actually measures the perceptual difference between two similar images, in the case of super resolution, the difference between the real HR image and the one generated by the generator. This measure is way more relevant than PSNR, because it is based on visible structures of the image. Indeed, the human visual system is more sensitive to changes in the structure of the image than to other techniques such a MSE and PSNR that estimates more of the absolute errors.

Another measure is the Visual Information Fidelity (VIF) [20], which is based on the amount of shared information between the reference and distorted image. The Gradient Magnitude Similarity Deviation (GMSD) [21] is also a quite good measure that has performance similar to SSIM but that have the disadvantage of having to compute the standard deviation over the whole image.

Finally, another way, not objective at all, not practical to implement, not scalable, but which is the best way to determine the quality of an image for a human, is to directly ask the concerned ones. Mean Opinion Score (MOS) is the measure that the authors used among others to evaluate the SRGAN[14]. It consists in surveying humans to rate the quality of the generated images from 0 to 5.

### C. Loss functions

The discussion on the different metrics also applies for the loss functions used. MSE and SSIM can be used as the loss function but have the same advantages and drawbacks mentioned earlier. More recently, people also started using pre-trained convolutional neural networks as perceptually-relevant loss functions. It works by taking a pre-trained model, typically VGG-19 model trained on ImageNet [22], then take it's first few layers and compute the difference between the feature maps produced by those layers. The difference between the feature maps can be minimized to train another model, just like any other loss function. The layers that generate those feature maps stay frozen during training and acts as a fixed feature extractor. Using this loss function is quite effective [23] but it is more expensive in term of computational power. For this reason, we started with classical loss like MSE, then we used SSIM and finally a model-based loss.

## III. METHODS

This section aims to list the models, *i.e.* the architectures tested in this project as well as describing the training methods.

### A. Base SRGAN

The base method used is a SRGAN [14], the combination of a fully convolutional generator (ResNet) and a mostly convolutional discriminator (figure 1). This is a typical base architecture for super-resolution. The generator loss has 2 parts : 1 adversarial loss and a perceptual loss. The generator up-scales the image by a factor 2 or 4.

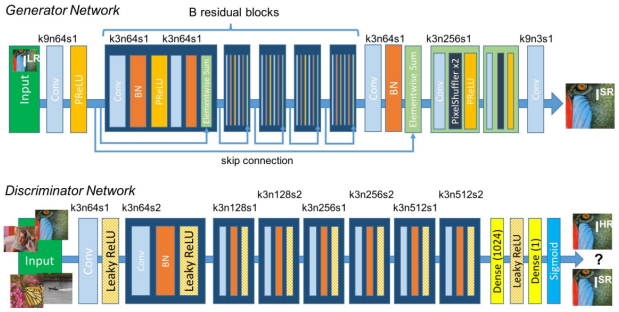



FIG. 1: SRGAN proposed from [14]

### B. Modified SRGAN

#### 1. *Generator*

The base architecture is described in [14]. The generator is mostly the same expect a few differences (figure 2). It was decided to keep the same kernel size (9 for the large one and 3 for main one) and activation function (PReLU). Each upscale block is still a pixel shuffle block[16], Other layer like deconvolution were not investigated. At the very end of the network, we used 2 convolutional layers instead of 1. Note that in this project only 2 to 4 times upscaling was tested (1 to 2 upscale blocks).

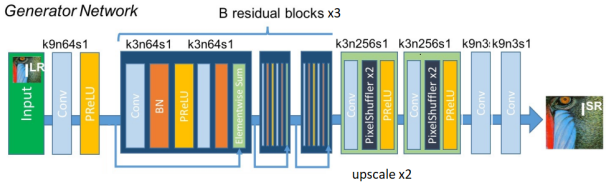



FIG. 2: SRGAN modified for a 4x scaling factor, generator part

#### 2. *Discriminator*

The discriminator also follows the same base architecture as in [14] (figure 3). The only difference is at the end of the convolutional network. After all downsizing convolution, we used a global average pooling layer. Then

there is a 2 layer MLP to the scalar output and a dropout layer was added. The last sigmoid layer was removed and inserted directly in the loss function in order to have more numerically stable computation which allows to use mixed-precision computation.

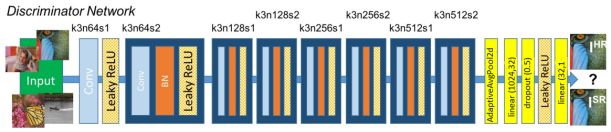



FIG. 3: SRGAN modified, discriminator part

### C. R1 regularization

The $R1$ regularization [15] techniques have proven to be quite effective at stabilizing and helping the convergence of GANs. It is a gradient penalty for GAN's training. We used this regularization on the discriminator's gradient at each batch iteration.

$$R_1(\phi) = \gamma E_{p_D(x)}[||\nabla D_\phi(x)||^2]$$

With $\phi$ the parameters of the Discriminator $D_\phi$ and $\gamma$ a regularization factor (0.01 for most test).

### D. Data

#### 1. Div2K

The Div2k dataset [24] is a well know dataset for benchmarking super-resolution.

#### 2. COCO

COCO [25] is a dataset of thousands of images design for object detection and segmentation.

### E. Model parameters

There are several parameters that can vary in this model. Here we discuss the main ones and their effects.

One classical layer in convolutional block is the normalization layer. These normalization layer typically help sharpen the resulting images. However it has a computational cost. In [17], a proposition to remove the normalization layers is made and use RRDB block instead of simple residual blocks. We tried to remove normalization layer in IV D to see the effects.

Another parameter of the generator is the number of convolutional blocks. Increasing the number of blocks increase the computational cost and the training time, while providing better results. In this work, we decided to keep only 5 residual block for all models.

The type of residual block can also have an impact. For our experiments we used the typical residual blocks.

The number of channels of the main convolutional block is also one interesting parameter. For our experiments, we used 64 channels like [14].

### F. Loss functions

#### 1. Discriminator loss function

For the discriminator we kept the binary cross entropy (with logits) which is used to compute the 2 part of the GAN loss $\log(D(x))$ and $\log(D(G(z)))$, with $x$ the high resolution images and $z$ the low resolution one. The BCE loss (1) correspond to $\log(D(x))$ for $y_i = 1 \quad \forall i \in \{1, N\}$ and $\log(1 - D(G(z)))$ for $y_i = 0 \quad \forall i \in \{1, N\}$.

$$BCE(x, y) = \frac{1}{N}\sum_{i=1}^{N} y_i \log(x_i) + (1 - y_i)\log(1 - x_i) \quad (1)$$

Then we apply a gradient regularization $R_1(\phi)$ (III C) The total discriminator loss function is thus:

$$\begin{aligned} L_{\phi,\theta}(I^{HR}, I^{LR}) =& \log(D_\phi(I^{HR})) + \log(D_\phi(G_\theta(I^{LR}))) \\ &+ R_1(\phi) \end{aligned}$$

#### 2. Generator loss function

The Generator adversarial loss function $\log(D_\phi(G(z)))$ can also computed thanks to the BCE loss with all $y_i = 1$. The total loss (2) balances the adversarial loss and a perceptual loss.

A content loss function is also very useful for super-resolution, this loss will directly compare the output image with the target image. Many loss functions have been proposed over the years. For our experiments, we decided to mainly use loss functions that are less computationally expensive. In used the classical MSE (3), as well as SSIM (4) and finally LPIPS loss (6) [23].

$$\begin{aligned} L(I^{LR}, I^{HR})) =& \alpha \log(D_\phi(G_\theta(I^{LR}))) \\ &+ L_{perceptual}(I^{LR}, I^{HR})) \end{aligned} \quad (2)$$

$$L_{MSE}(I^{LR}, I^{HR}) = \frac{1}{wh}\sum_{x=1}^{w}\sum_{y=1}^{h}(I^{HR}_{x,y} - G_\theta(I^{LR}_{x,y}))^2 \quad (3)$$

$$L_{SSIM}(I^{LR}, I^{HR}) = 1 - \frac{(2\mu_{GLR}\mu_{HR} + c_1)(2\sigma_{GLR}\sigma_{HR} + c_2)(cov_{GLR,HR} + c_3)}{(\mu_{GLR} + \mu_{HR} + c_1)(\sigma_{GLR} + \sigma_{HR} + C_2)(\sigma_{GLR}\sigma_{HR} + c_3)}$$

$$L_{MSE\_SSIM}(I^{LR}, I^{HR}) = L_{MSE}(I^{LR}, I^{HR}) + \beta L_{SSIM}(I^{LR}, I^{HR}) \quad (4)$$

LPIPS measures computes the a perceptual difference of 2 images by comparing multiple features maps of pre-trained models for these 2 images. If one designates $\hat{y}^l$ and $\hat{y}^l_0$ the unit normalized feature stacks obtained at layer $l$ of a model for 2 images $x$ and $x_0$, the LPIPS can be expressed as 5 with $\omega_l$ scaling vectors. In our experiments, we used an AlexNet based LPIPS loss function. We also decide to use $\beta = 0.2$ for balancing the loss (4) and (6).

$$L_{LPIPS}(x, x_0) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} ||\omega_l \odot (\hat{y}^l_{hw} - \hat{y}^l_{0hw})||_2^2 \quad (5)$$

$$L_{MSE\_LPIPS}(I^{LR}, I^{HR}) = L_{MSE}(I^{LR}, I^{HR}) + \beta L_{LPIPS}(G(I^{LR}), I^{HR}) \quad (6)$$

### G. SRGAN training

The models were trained on both DIV2K and COCO. Note that the images are randomly cropped at a fixed size which increase the effective size of the datasets. For the training, we chose a batch size of 32 and to crop the high resolution images (typically $96 \times 96$). The low resolution images are then created by downsizing the high resolution image by the scaling factor with a bicubic interpolation method.

For the learning rate, we chose the start by a 0.0001 for the first half of the epochs, then 0.00001 for the remaining ones.

In order the reduce memory usage, mixed precision computation was used whenever possible. In particular it was not used with the SSIM Loss which does not support it.

When training the models, some jump where detected in the loss function of the discriminator. Gradient norm clipping was used at some point for the discriminator.

For classical GANs, the labels are 0 for fake images and 1 for real images (normal labels). We investigated other empirical labeling methods like "randomized labels" 0 are replaced by $t_0 \sim \mathcal{U}(0.0, 0.2)$ and 1 are replaced by $t_1 \sim \mathcal{U}(0.8, 1.0)$. One sided smoothing [26] replaces the 1's by 0.9's.

At each epochs, the model is evaluated on a test set with the SSIM score.

## IV. RESULTS

In this section, we present some of the experiments that we did with the proposed models.

The first and second experiments try to illustrate the difference between models with different balance between perceptual and adversarial loss.

The third GAN is our based model applied to 2X upscaling while the fourth is a normalization-free version of the generator.

From the fifth model, we investigate only 4X upscaling.

The fifth is the base model for 4X upscaling.

The 6th GAN uses a SSIM based loss function while the 7th uses a LPIPS based loss function for 4X upscaling.

A summary of the models trained for these experiments is shown in table I

### A. First try: weak adversarial loss

For the first experiments that we did with the SRGAN, we started with a weak adversarial loss. This means that with started with a low $\alpha = 0.0002$ (in loss 2 ). This parameter balance the 2 part of the generator loss: the perceptual loss and the adversarial loss. The model was therefore essentially driven by the mean square error loss for this experiment. The generator did not take the discriminator into account and the discriminator managed to properly classify the real and fake images at the end. The final images were still a little blurry. the losses of the generator and the discriminator are shown in figure 4.

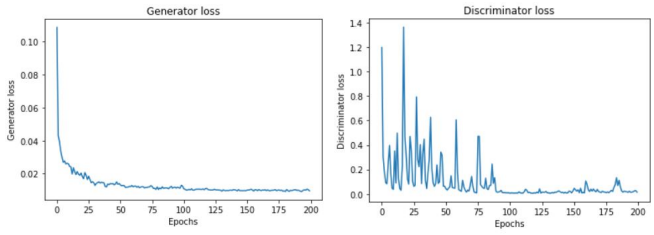



FIG. 4: First GAN training for 2 times upscaling with $\alpha = 0.0002$, no clipping, no regularization and MSE loss.

### B. Second try: increase adversarial loss

For the second experiment, we tried to increase the adversarial loss in order to see if the model could take advantage of it. The $\alpha$ parameter was increased by a factor 5 compared to the first experiment. Indeed we could see (figure 5) that the discriminator loss would not collapse to 0 as in the first experiment. The discriminator seems shaky and varies a lot while the generator stabilizes.

| Model | normalizations | Upscaling factor | Loss (III F 2) | labels (III G) | iterations | training set (III D) |
|---|---|---|---|---|---|---|
| GAN 1 | yes | 2 | MSE | normal | 5000 | DIV2K |
| GAN 2 | yes | 2 | MSE | normal | 7500 | DIV2K |
| GAN 3 | yes | 2 | MSE | normal | 7500 | COCO |
| GAN 4 | no | 4 | MSE | randomized | 12000+ | COCO |
| GAN 5 (base) | yes | 4 | MSE | randomized | 12000+ | COCO |
| GAN 6 (SSIM) | yes | 4 | MSE+SSIM | one sided smoothing | 10000- | COCO |
| GAN 7 (LPIPS) | yes | 4 | MSE+LPIPS | one sided smoothing | 10000- | COCO |

TABLE I: Summary of the models tested during the experiments (IV)

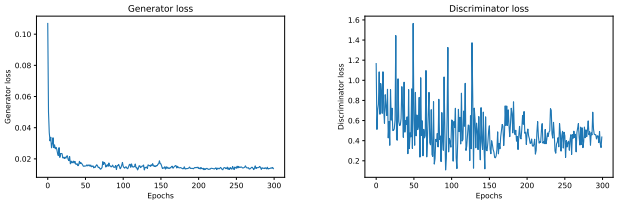



FIG. 5: Second GAN training with $\alpha = 0.01$.

### C. GAN 3: base model on 2X upscaling

Training on a dataset of only 800 high resolution images might not be enough even though random crop are used. In order to train the next models we used a COCO dataset.

This experiments uses our base model for 2 times upscaling. The classical MSE Loss is used with R1 regularization and no label smoothing. The resulting training losses are shown in figure 6.

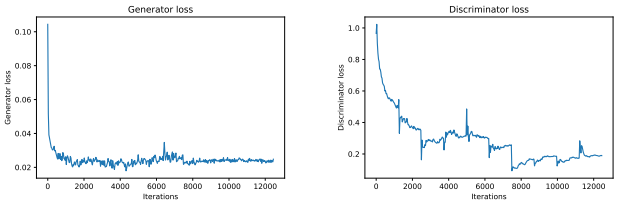



FIG. 6: Third GAN training: base model for 2X upscaling

### D. GAN 4: Removing normalization for 4X upscaling

One improvement that can be added to some convectional architecture is to remove normalization for computational efficiency. For that purpose some convolutional block have been investigated. We decided to keep the residual blocks and remove normalization. This would indeed improve the performance. However it does not seems to work well with simple residual block, Residual in Residual Dense Block (RRDB) [17] might be more suited. The final training losses are again shown in 7.

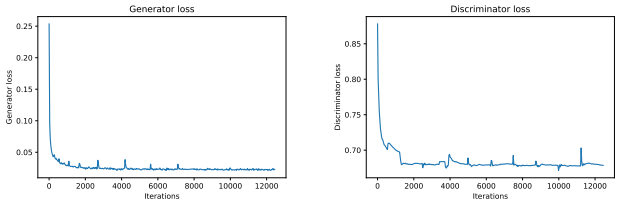



FIG. 7: Fourth GAN training: normalization free version

### E. GAN 5: base model on 4X upscaling

Seeing that the 2 times upscaling was working well. We decided to turn to 4X times upscaling which is more challenging.

For this experiment, the base model was used for 4 times upscaling. We also tried to introduce randomization of label to make the work of the discriminator harder. Every "1" label was replaced with a random number $t_1 \sim \mathcal{U}(0.8, 1.0)$ while each "0" label was replaced by $t_0 \sim \mathcal{U}(0.0, 0.2)$. As one can see in figure 8, the models seems to 'stabilize' relatively fast.

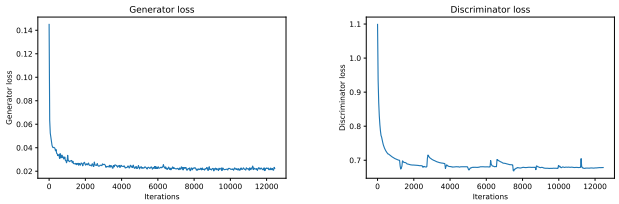



FIG. 8: Fifth GAN training: base model for 4X upscaling

### F. GAN 6: Using ssim in the loss function

SSIM is a measure of image quality. In this experiment, we try to directly optimize it in combination with MSE

loss. In the formula of this loss function 4, $\beta = 0.2$ was chosen experimentally in order to balance both losses. One sided label smoothing was also used, i.e. "1" labels were replaced by "0.9" labels [26].

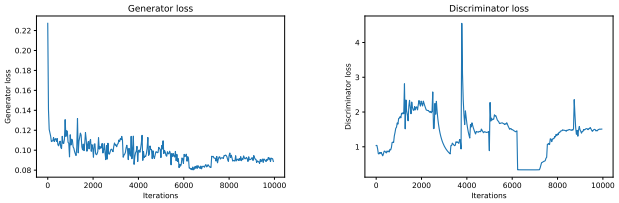



FIG. 9: Sixth GAN training with MSE-SSIM loss function.

### G. GAN 7: Using LPIPS in the loss function

In this final experiment, we used a loss function mixing the MSE and a AlexNet based LPIPS loss. This experiments still uses a one sided label smoothing (with 0.9). A $\beta$ of 0.2 was used in (6). Compared the other experiments, the generator loss seems to decrease more slowly (figure 10). Some higher spikes were also detected in the discriminator loss function even though gradient norm clipping was used (clip norm = 1.0).

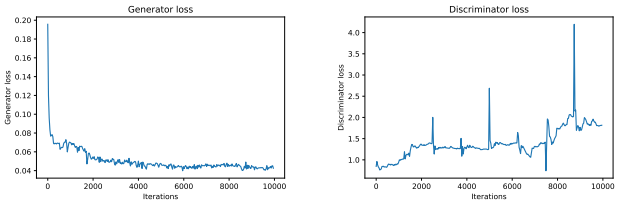



FIG. 10: Seventh GAN training with AlexNet based LPIPS loss function

### H. Comparison of models

This section proposes an evaluation of the model obtained with PSNR (table II) and SSIM scores (table III) on the DIV2K dataset. Even tough these classical metrics seems to decrease with the new features added to the model, the qualitative analysis (IV I) of the images gives other insights.

### I. Final Images qualitative comparison

This section presents a comparison of the GANs obtained with the images in figure 11.

| Model | PSNR | Upscaling factor |
|---|---|---|
| GAN 3 (base, 2X) | 25.29 | 2 |
| GAN 4 (base w/o norm.) | 26.74 | 4 |
| GAN 5 (base) | 26.62 | 4 |
| GAN 6 (SSIM) | 24.47 | 4 |
| GAN 7 (LPIPS) | 24.04 | 4 |

TABLE II: PSNR scores for the different models (IV) obtained on DIV2K dataset

| Model | SSIM | Upscaling factor |
|---|---|---|
| GAN 3 (base, 2X) | 0.72 | 2 |
| GAN 4 (base w/o norm.) | 0.69 | 4 |
| GAN 5 (base) | 0.69 | 4 |
| GAN 6 (SSIM) | 0.62 | 4 |
| GAN 7 (LPIPS) | 0.53 | 4 |

TABLE III: SSIM scores for the different models (IV) obtained on DIV2K dataset

#### *1. 2X upscaling*

The base model GAN 3 achieves reasonable performances, the images obtained are quite persuasive. However, the 2X upscaling task might is not the hardest.

#### *2. 4X upscaling*

The base model (GAN 5) create slightly blurry images. The normalization free version (GAN 4) creates more blurry images.

The GAN 6 (SSIM based loss) attempts to create some sharper details which creates less blurry images but with some artifacts on edges.

The GAN 7 (LPIPS based loss) also attempts to create some sharper details which are more persuasive (but also more creative) than the ones of GAN 6. However these new details do not necessarily match the targets ones very well.

## V. DISCUSSION

Even though the results obtained by the final model GAN 7 are not very impressive, it seems to be on the right track. Increasing the size of the generator, adding better convolutional block or regularization techniques and training the model longer would probably significantly increase the performances.

Such a subject requires to distinguish human and IA perception of shapes and similarities. Hence a comparison of human and IA similarity metrics is a necessary work. In this work, we relied more on human perception of images to draw conclusion. However, a better evalua-

(a) GAN 3 (b) GAN 4

(c) GAN 5 (d) GAN 6

(e) GAN 7 (f) Real

FIG. 11: Images comparison for the models tested in IV
on a butterfly's wing (DIV2K)

tion scheme would have been more suited, which would help to confirm the results obtained. The evaluations could be performed on different datasets and for different models produced by the trainings. Moreover, many parameters are changed during the experiments which makes the comparison harder. Making less parameters vary would allow for a better comparison.

### A. Future improvements

Super-resolution has seen huge improvement in the last years. While the most advanced GANs already provide impressive results, there are still some work to do in image super-resolution. In particular finding better metrics for image quality which match human perception is one important and difficult subject. LPIPS metrics seem to be the more promising methods.


[1] K. He X. Tang C. Dong, C. Loy. Image super-resolution using deep convolutional networks, 2015.

[2] K. Lee J. Kim, J. Lee. Accurate image super-resolution using very deep convolutional networks, 2016.

[3] K. He X. Tang C. Dong, C. Loy. Accelerating the super-resolution convolutional neural network, 2016.

[4] H. Kim S. Nah K. Lee B. Lim, S. Son. Enhanced deep residual networks for single image super-resolution, 2017.

[5] K. Sohn N. Ahn, B. Kang. Fast, accurate, and lightweight super-resolution with cascading residual network, 2018.

[6] J. Yu D. Liu W. Han H. Yu Z. Wang X. Wang T. Huang Y. Fan, H. Shi. Balanced two-stage residual networks for image super-resolution, 2018.

[7] K. Lee J. Kim, J. Lee. Deeply-recursive convolutional network for image super-resolution, 2016.

[8] Y. Tian W. Wang J. Xue Q. Liao W. Yang, X. Zhang. Deep learning for single image super-resolution: A brief review, 2019.

[9] N. Ahuja M. Yang W. Lai, J. Huang. Fast and accurate image super-resolution with deep laplacian pyramid networks, 2017.

[10] J. Li Y. Huang H. Wang Y. Hu, X. Gao. Single image super-resolution via cascaded multi-scale cross network, 2018.

[11] X. Gao Z. Hui, X. Wang. Fast and accurate single image super-resolution via information distillation network, 2018.

[12] M. Kim J. Choi. A deep convolutional neural network with selection units for super-resolution, 2017.

[13] L. Wang B. Zhong Y. Fu Y. Zhang, K. Li. Image super-resolution using very deep residual channel attention networks, 2018.

[14] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network, 2017.

[15] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge?, 2018.

[16] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network, 2016.

[17] S. Wu J. Gu Y. Liu C. Dong C. Loy Y. Qiao X. Tang X. Wang, K. Yu. Esrgan: Enhanced super-resolution generative adversarial networks, 2018.

[18] X. Mou D. Zhang Lin Zhang, Lei Zhang. A comprehensive evaluation of full reference image quality assessment algorithms, 2012.

[19] H.R. Sheikh E.P. Simoncelli Z. Wang, A.C. Bovik. Image quality assessment: from error visibility to structural similarity, 2004.

[20] A.C. Bovik H.R. Sheikh. Image information and visual quality, 2006.

[21] X. Mou A.C. Bovik W. Xue, L. Zhang. Gradient magnitude similarity deviation: An highly efficient perceptual image quality index, 2014.

[22] A. Zisserman K. Simonyan. Very deep convolutional networks for large-scale image recognition, 2015.

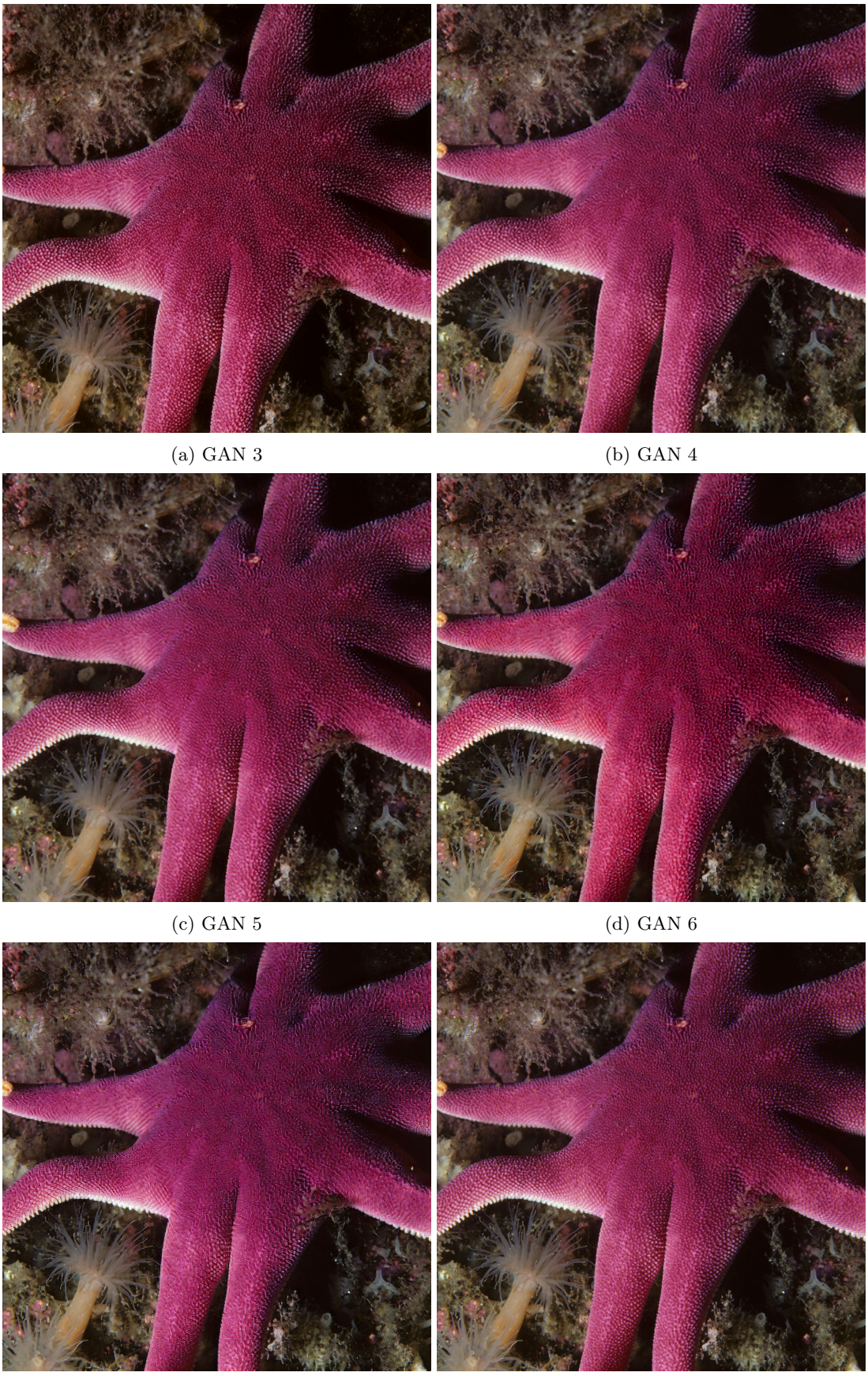
[23] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018.

[24] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[25] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[26] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans, 2016.


## Appendix A: Additional Images

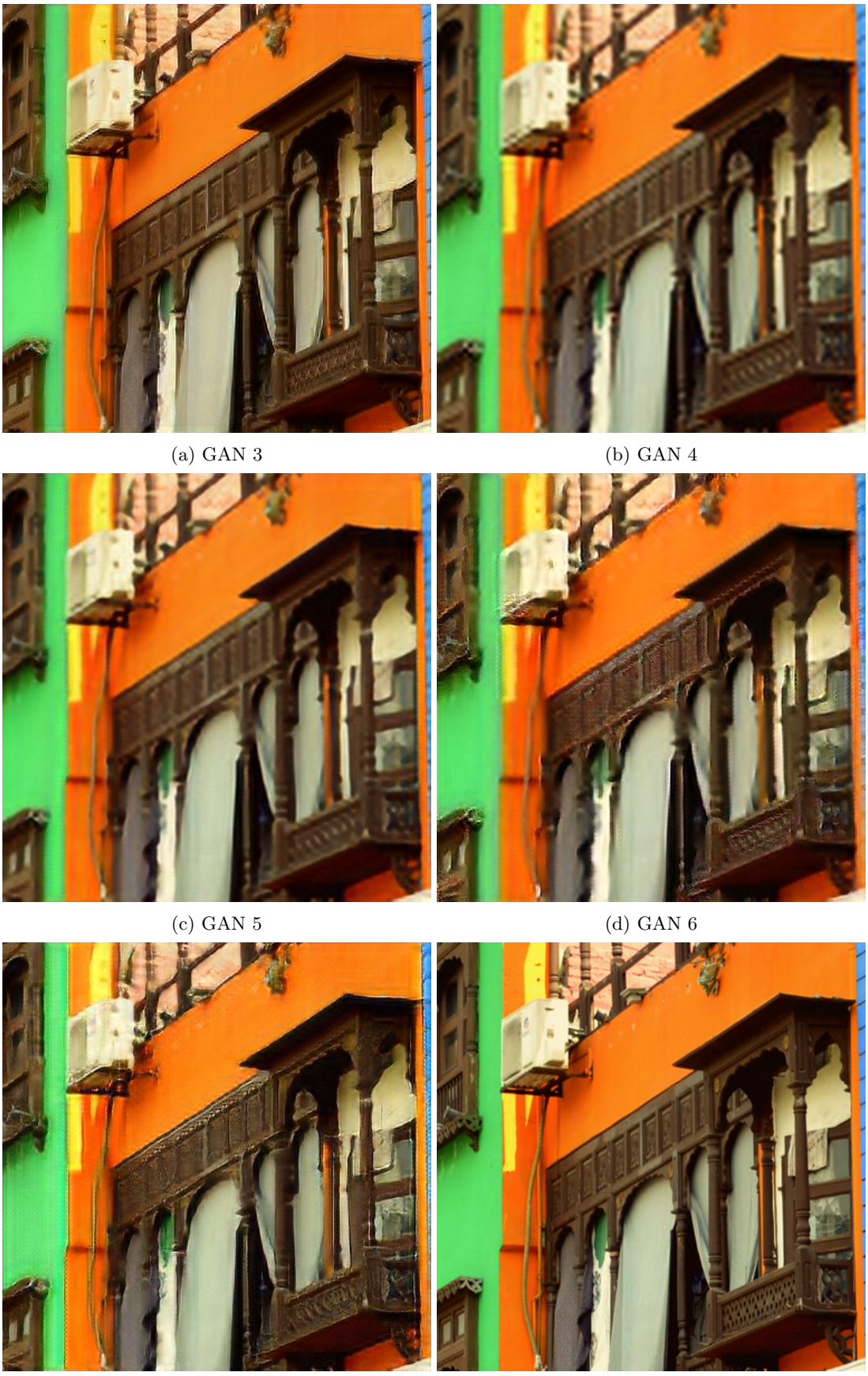


(a) GAN 3 (b) GAN 4

(c) GAN 5 (d) GAN 6

(e) GAN 7 (f) Real

FIG. 12: Images comparison for the models tested in IV on a starfish (DIV2K)



(a) GAN 3 (b) GAN 4

(c) GAN 5 (d) GAN 6

(e) GAN 7 (f) Real

FIG. 13: Images comparison for the models tested in IV on a building (DIV2K)