GAN-based model to denoise images

Aglieco Francesco, Comparetto Alessandra, Gagliardi Giuseppe

Students of the Master's Degree program in Computer Engineering

Politecnico di Torino

A.A. 2021/2022

{francesco.aglieco, alessandra.comparetto, giuseppe.gagliardi}@studenti.polito.it

Abstract

Digital images suffer from being distorted when performing acquisition, compression, storage and transmission. The noise introduced by those actions is very complex to remove. Nonetheless, this is often an important step in order to work with high quality and clear images. The core of our study is to create a model that is able to face Gaussian, Salt & Pepper and Speckle noise and to remove them from the images. By generalizing the class of noise introduced on the images, the model will be more robust and moves few steps forward into a more ideal objective that is the removal of natural noise. Firstly, a generative-adversarial network structure was designed. Secondly, the network was trained using a large dataset of images. Finally, the performances of the model are evaluated using a smaller dataset.

1. Introduction

Whichever task involves the usage of images (object recognition, action recognition, segmentation, ...) will always have to deal with artefacts. Reconstructing parts of the content that are corrupted is a necessary step to achieve good performances in the tasks quoted before.
Traditional algorithms (like Gaussian filtering, spatial pixel feature denoising algorithm, bilateral filtering etc.) can slightly improve the image quality by removing part of the noise, but they are not able to perform well in generalizing the process when different classes of noises are presented. They are designed to remove noise based on the properties of images and noise.
The artificial neural networks, instead, can achieve a better result by using many convolutional layers to extract features. Even if many researches have made a lot of progress with image de-bluring and super resolution, those networks rely on heavy training and consume a lot of resources. Moreover, even if the training gets longer, the image obtained may lost content details. Therefore, the choice of a GAN-based model which should give a better visual experience and satisfactory results, as proven by other studies like [3].
In order for the network to learn, pairs made of a clean image and a noisy one are needed. The noisy images are created by artificially adding several kinds of noises (Gaussian, Salt and Pepper, Speckle). Adding artificial noise to images has a clear advantage: an unlimited amount of training data can be created.
The network born from this study is able to denoise images by taking a noisy image as input, then the real clear image and the de-noised one are sent (separately) to the discriminator to obtain the for each of them the true/false likelihood corresponding scores.

2. Dataset

Since it's important to have a pretty good amount of images in order to train the model, for this study COCO[1] dataset was chosen. It's a large-scale object detection, segmentation, and captioning dataset, that counts 328 thousands images. They come in different sizes, all of them affected by natural noise. Each image has its own set of labels that are useless in this study, so they were ignored.
The choice of this dataset was driven by the need of a big and diversified dataset to properly train the model. Although at first ImageNet seems like a good choice, using it would have meant not to be able to exploit efficiently the benefits of the VGG content loss when it is employed as generator loss.
Only 1100 images from the COCO dataset were picked out

[1] Link to Coco dataset here

to build the dataset for this study. This decision was driven by the limited amount of RAM and memory available on the development platform used (Google Colab). Out of all the images chosen, 100 were used to test and 1000 are used to train.
Although data augmentation could be useful to improve performances and outcomes of the model (by forming new and different examples to train the model), the resources available for this study wouldn't allow such practice. Nonetheless, results show that the portion of COCO dataset chosen for this study was big enough to train the model.
As mentioned before, the images have different sizes. This required to perform a crop so that each item would be $(256, 256, 3)$ in order to work with a fixed dimension. With the intent of having a more robust network, the cropped area was chosen randomly.
While loading the dataset and creating the test and trains sets, only images with 3 channels bigger than 256x256 were accepted.
In addiction, another dataset made of 300 images was used in order to evaluate the performances achieved by the model after the train. This is the BSD300[2] whose test set is made of 100, while the train set is made of 200. This dataset is used for image segmentation and boundary detection. The items in this dataset have shape (321, 481, 3) or (481, 321, 3).
Being a small dataset, it's perfect to test the performance of the model.

3. Methods

A generative adversarial network (GAN) involves two neural networks that compete to become more accurate: a generative network and a discriminative network. The Generator is used to generate new plausible examples from the problem domain. The Discriminator is used to classify examples as real (from the domain) or fake (generated).
Starting from the idea that the traditional approaches for denoising have a lot of limitation, for this study the idea was to exploit the massive potential that a GAN model has, being a fairly new technology with lots of room to explore.
Other approaches that could be used to perform this task have been considered not good enough:

- Multi-layer perceptron approaches present an incredible amount of parameters, consume a lot of resources in order to finish the training, despite not being always able to achieve excellent results.
- Convolutional Neural Networks (CNN) represent a good option for deep model training, but may be affected by gradient disappearance or gradient explosion during the learning process if the network is too deep.

[2]Link to BSD dataset here

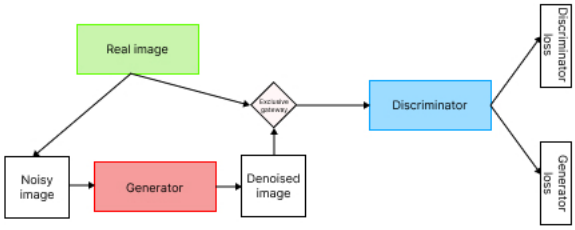

Figure 1. GAN structure used for this study.

All things considered, a residual network based on a GAN, represent a good approach. Even if the resources needed are a lot, the results are sure to be more rewarding.
After choosing such a model, the first design challenge to overcome, would be to find the optimal way to design and connect the two parts that a GAN is composed of: Generator and Discriminator.
The usual structure of a GAN presents a Generator with one input (which is a random vector) and one output (which is the generated image). The random vector is used by introducing noise, we can get the GAN to produce a wide variety of data, sampling from different places in the target distribution. As far as the Discriminator is concerned, it would have one input, the original image or the generated one and classifies them.
This structure just described, is not the one used for this study. The Generator will receive the noisy image since its job is not to create a random image but to remove the noise as much as possible. Furthermore, the Discriminator receives only one input at the time either the original or the generated image. The GAN model proposed for this study is represented in Fig. 1.

3.1. Noise analysis

Being this study focused on denoising, a core point to be discussed regards the noises.
Three reference noises were considered, each of them with a different level of intensity.
Specifically:

- **Gaussian noise**: signal noise that has a probability density function equal to that of the normal distribution. Principal sources of Gaussian noise in digital images arise during acquisition e.g. sensor noise caused by poor illumination and/or high temperature, and/or transmission e.g. electronic circuit noise [7].
  Variance chosen between **0.005 e 0.08**,
- **Salt & Pepper**: consists of only the maximum or minimum intensity values (0 or 255) in the dynamic range. It could arise when transmitting images over noisy digital links [2].

Figure 2. Clear test image from COCO Dataset.

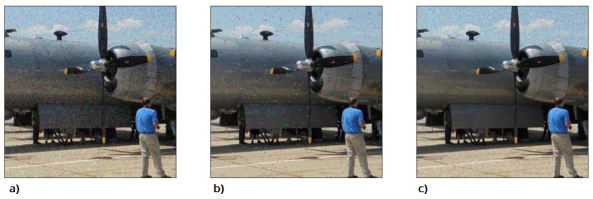

Figure 3. Noisy images: a) Gaussian Noise, b) Salt&Pepper, c) Speckle

Generally, the removal of Salt & Pepper consists of two problems: how to detect the noisy pixels, how to repair them.
Noise intensity is managed by an internal parameter which depends on the noise adding function used. Values chosen for the proportion of image pixels to replace with noise is between **0.02 and 0.07**,

- **Speckle**: the noise that arises due to the effect of environmental conditions on the imaging sensor during image acquisition. Speckle noise is mostly detected in case of medical images, active Radar images and Synthetic Aperture Radar (SAR) images [6].
  Variance chosen between **0.005 e 0.2**

In Fig. 2 and Fig. 3 it is possible to see how the image changes with respect to the noise applied. To plot these images, the average intensity of noise was chosen.

3.2. Generator network

The generator's job is to create images so that the discriminator is unable to recognise which images are "real" (coming from the dataset, what we consider clean images) and which are "generated" (supposedly clean images).
For this study we designed a residual U-Net starting from the code coming from two sources: [1] and [4]. The first code presents a Pix2Pix Generative Adversarial network for image-to-image translation. The structure of the UNet proposed for the Generator of this GAN is the one of a standard UNet but could not be applied for this study. Having 7 encoder and 7 decoder blocks, it was too deep and had too many output parameters. Having limited resources and training with only 1000 images, it would have been very risky to use such a complex network (with over 54 millions of parameters). In order to avoid overfitting, the generator's architecture was simplified, removing two inner layers. This decision was absolutely beneficial to reduce the 47 millions parameters gap between Discriminator and Generator.
The Discriminator presented in the study has two sources, because the underlying structure is a PatchGAN. For this study the structure of the Discriminator is simpler, implemented by a one-source image classification CNN.
The second source, instead, presents a Generative Adversarial Networks that performs super-resolution on optical images. The topic exposed there is not entirely similar to the one faced by this study; therefore the only reusable element would be the Content Loss, which uses a pre-trained (and frozen) VGG network (pre-training performed on ImageNet dataset).
The first part of the network (the contracting part) is made of 5 encoder blocks followed by 5 decoder blocks (for the expansive part).
To prevent the disappearance or explosion of gradients, batch normalization was added do each block mentioned before except for the first one.
Each encoder block is composed of a random weight initialization, one convolutional layer, the batch normalization layer and the leaky ReLu activation.
Each decoder block is composed of a random weight initialization, the upsampling layer, the batch normalization layer, the ReLu activation.
As mentioned before, the reference structure for the model is a UNet, skip connections have been inserted as shown in Fig. 4 in order to avoid the issues related to vanishing gradient.
The number of parameters reached with this structure is almost 20 millions.

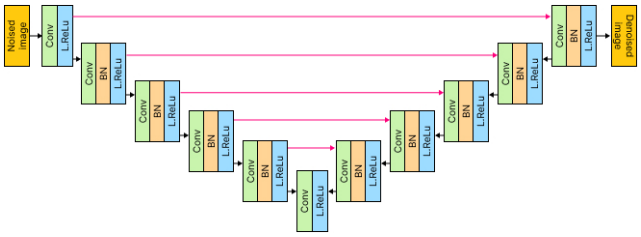

Figure 4. Generator structure: Simplified UNet

3.3. Adversarial network: discriminator

The adversarial network's job is to give a score both to the clear image and to the generated one. Ideally, a well-trained discriminator will give a score close to 1 for the true images and close to 0 for the fake ones. In this case, true images are the clear ones, fake images are the noised ones

after passing though the generator.
If discriminator's behaviour follows the schema just explained, it means that the adversarial network is able to effectively distinguish between clear and supposedly denoised images.
The structure used is fairly simple, having only one source of input and consisting of five groups of three layers: convolutional layer, batch normalization layer and the leaky ReLu activation.
The output is regulated by a sigmoid activation, since we expect the output of this part of the GAN to be between 0 and 1. This output represents the likelihood that the input picture is a clear image.
The number of parameters reached with this structure is almost 30 millions.

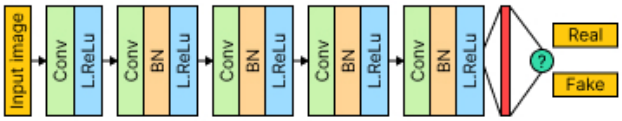

Figure 5. Discriminator structure

The following Tab. 1 and Tab. 2 contain the number of parameters for the two parts of the GAN model (respectively generator and discriminator) and the total number of parameters for the entire model.

Layer (type)	Output Shape	Param #
InputLayer	[(None, 255, 256, 3)]	0
Generator	(None, 256, 256, 3)	29253251
Discriminator	(None, 1)	20588225

Table 1. Number of parameters for Generator and Discriminator.

Total params:	49,841,476
Trainable params:	29,247,491
Non-trainable params:	20,593,985

Table 2. Total number of parameters of the model.

3.4. Hyper-parameters

Hyper-parameters are chosen in order to find the best possible compromise between the training processes of Discriminator and Generator which should be fair and equal as much as possible. A fast learning Discriminator that works with a slow Generator will lead to a poorly performing network. Resulting losses will soon stall, reaching a low plateau that affects the accuracy as well. Specifically it becomes too high, highlight that the discriminator is never wrong with its prediction. Using Adam as optimizer for both the GAN model and the Discriminator, two sets of three hyper parameters are used for this study:

- α or Learning Rate: The proportion that weights are updated. Larger values lead to faster initial learning before the rate is updated. Smaller values slow learning down during training.

- β_1: It controls the moving average. Usually 0.9 is chosen, meaning that it is calculating the average over the last 10 iterations' gradients and the older gradients are discarded.

The following Tab. 3, shows the sets of values chosen for the hyper-parameters. Studies like [3], [4], [5] presented similar values.

Model	α	β_1
Discriminator	0.009	0.9
GAN	0.005	0.5

Table 3. Hyper-parameters used in this study.

3.5. Training process

The training used in this study follows the typical training process of a GAN network.
First, the discriminator is trained passing through it the clear images and then the denoised images. When this first step is done, the generator is trained, making sure that the discriminator is not able to update its parameters.
A crucial point in the training process of the GAN, is to put in disadvantage the discriminator so that it doesn't become too good too quickly. Generator and discriminator must improve their performances evenly.
An operation performed on the images is the addition of the noise. Initially, the class of noise is decided randomly between Gaussian, Speckle and Salt & Pepper. Then, the amount of noise to apply to the image is chosen using a uniform distribution function that varies in a range that we considered not to be too aggressive or shallow as described in 3.1.
It's important to highlight that this process is made throughout the training of the GAN model. When the original image is selected to be used to train the discriminator, we save its reference, so that the noised version can be passed to the generator in the second phase of training, when the discriminator is frozen.
Training was performed by spitting the train set into 125

batches of 8 images. Sadly, due to lack of resources available in the developing platform, it was only possible to train for 30 epochs.

3.6. Loss function

Being composed of two parts (generator and discriminator) the usual loss function of a GAN network is composed as well of two parts: the generative loss and the discriminative loss. We can express it with the following formula:

$$L_{total} = \lambda_g L_{VGG} + \lambda_d L_{BCE}$$

Where L_{VGG} represents the content loss. It uses the 4th layer of a VGG network trained with ImageNet and then uses it to extract the main features from the images (both the clear and the denoised one), then evaluates the distance between the two. As mentioned before, the adoption of this particular loss was driven by another study [4] whose tasks was super resolution. Keeping in mind that this is not exactly the same as performing denoising on images, conceptually the goal is the same: reconstruction of images (whether it uses low-resolution images or noisy ones).
Instead, L_{bce} represents the binary cross-entropy used for the discriminator. The parameters λ_g and λ_d are the corresponding coefficients, which were set manually to 0.1 and 0.01.

3.7. Evaluating performances: metrics

Considering other studies conducted on denoising of images, like [5], the two metrics used to evaluate performances are:

- **Peak signal-to-noise ratio** (PSNR)[3] : represents the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation.

  $$\text{PSNR} = 10 \cdot \log_{10} \frac{\text{MAX}_I^2}{\text{MSE}}$$

  PSNR is commonly used to quantify reconstruction quality for images and video subject to lossy compression and it is an approximation to human perception of reconstruction quality. Acceptable values for wireless transmission quality loss are considered to be about 20 dB to 25 dB.

- **Structural Similarity** (SSIM)[4] : is a perception-based model that considers image degradation as perceived change in structural information. Spatially close pixels have strong inter-dependencies which bear important information about the structure of the objects in the visual scene. The difference with other techniques such as MSE or PSNR is that these approaches estimate absolute errors. The SSIM values ranges between 0 to 1, where 1 means perfect match the reconstruct image with original one. Generally SSIM values 0.97, 0.98, 0.99 for excellent quality reconstruction techniques.

[3]PSNR definition source here
[4]SSIM definition source here

In Tab. 4 and Tab. 5 the values of the two metrics (both average and maximum) are reported. Those have been calculated after training epoch 16, which is the best one among the 30 executed. It is possible to infer from the results, that the model performs fairly well with respect to reference values.

Noise	PSNR (dB)		SSIM	
Gauss	AVG	25.635	AVG	0.707
	MAX	30.519	MAX	0.889
Salt&Pepper	AVG	25.845	AVG	0.721
	MAX	30.794	MAX	0.897
Speckle	AVG	26.115	AVG	0.737
	MAX	31.453	MAX	0.903

Table 4. PSNR and SSIM values (average and max) referring to train set with mean noise intensity.

Noise	PSNR (dB)		SSIM	
Gauss	AVG	25.359	AVG	0.722
	MAX	30.413	MAX	0.884
Salt&Pepper	AVG	25.576	AVG	0.734
	MAX	30.798	MAX	0.889
Speckle	AVG	25.808	AVG	0.751
	MAX	31.358	MAX	0.899

Table 5. PSNR and SSIM values (average and max) referring to test set with mean noise intensity.

In the second phase of the performance evaluation, BSD300 dateset was used. The 300 images were used to calculate the average and maximum values of PSNR and SSIM reached by the trained model. From the values of Tab. 6 it is clear that the model proposed in this study is more than capable to remove noise even when images of a different nature are used.

4. Experiments

Before discussing any of the experiments conducted, it is very important to highlight that throughout the development of this study, the limits imposed by the developing platform used heavily influenced our choices.
Initially, contrary to what presented until now, the GAN loss

Noise	PSNR (dB)	SSIM
Gauss	AVG 26.113 MAX 31.302	AVG 0.724 MAX 0.911
Salt&Pepper	AVG 26.359 MAX 31.746	AVG 0.742 MAX 0.918
Speckle	AVG 26.705 MAX 32.523	AVG 0.767 MAX 0.927

Table 6. PSNR and SSIM values (average and max) referring to test set with mean noise intensity using BSD300.

didn't use the content loss but, in its place, a mean squared error (MSE) loss was used. The results achieved in this first phase, were not nearly as good as expected. During the learning phase, both the Discriminator and GAN losses had incredibly high values that would slightly decrease at each epoch. Since it was not possible to train for a very long time, our team needed to find an efficient solution to this problem.

This is the moment in which the content loss was introduced. The advantages brought to the network by this change have incredibly impacted the performances: PSNR and SSIM values were finally satisfactory even in early epochs. Alongside the pros, came the cons, since calculating the VGG made the training process exceptionally slow and GPU hungry.

Another change that surely helped the robustness of the model was choosing the noise randomly. As mentioned in 3.1, minimum and maximum intensities for the noises were chosen so that it would not be too hard of too easy for the model to remove them. After choosing the intensity ranges for each type of noise, it seemed like a good idea to select the intensity to apply using a normal distribution centered on the mean value of the range. In practice, this choice made the model too specialized on the mean intensity for each range, without giving to it enough robustness to deal with the highest one. This initial idea was therefore discarded and in the final version of the training process, the noise intensity is chosen with a random uniform function.

It goes without saying that many trials were executed in order to find the best possible hyper-parameters for this study. In Tab. 7 one of the initial configuration for the hyper-parameters is shown (version 0). The adversarial loss had less importance on the overall loss, learning rates were smaller and both the β parameters were left to the default setting.

Selecting the best epoch for both of the configuration mentioned, with Fig. 6 and Fig. 7 it is possible to compare how the losses behave with different hyper-parameters. The Discriminator loss in version 0 immediately reaches a low-value plateau, which means that it does not learn anymore

Version	Hyper-params	
0 (old)	Dis. weight	1
	λ_g	0.1
	λ_d	0.001
	α_{disc}	0.007
	α_{GAN}	0.0005
	β_{disc}	0.9
	β_{GAN}	0.9
1 (current)	Dis. weight	1
	λ_g	0.1
	λ_d	0.01
	α_{disc}	0.009
	α_{GAN}	0.005
	β_{disc}	0.9
	β_{GAN}	0.5

Table 7. Hyper-parameters values for two different versions of the model. Highlighted the changes.

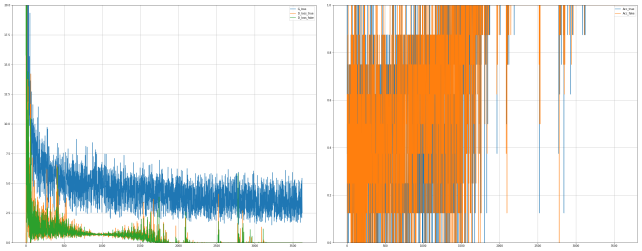

Figure 6. Plot losses and accuracy for version 0

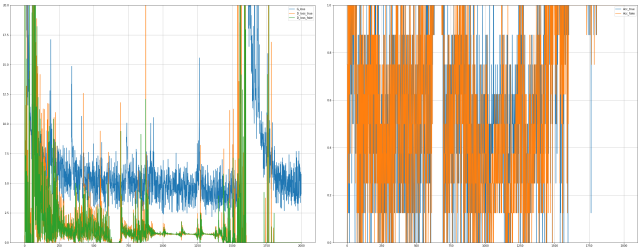

Figure 7. Plot losses and accuracy for version 1

and from that point on it is not useful to guide the Generator. Instead, version 1 losses show how the model is more robust with respect to the trend of discriminator loss to become smaller as the training progresses. The combination of a smaller β and an higher α makes sure that the adversarial loss reaches a close to zero value in a later training epoch.

Another relevant experiment that our team conducted, was reducing the weight assigned to the adversarial loss. This test was done to assess how much the Discriminator's prediction affects the performances of the Generator. Due

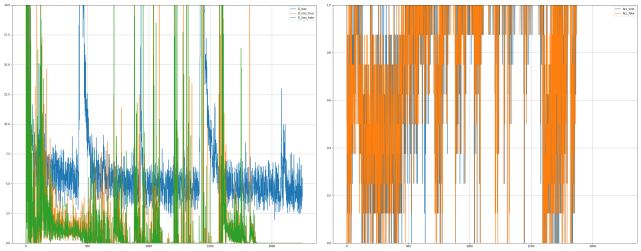

Figure 8. Plot losses and accuracy when λ_d is equal to 0.001

Noise	PSNR (dB)	SSIM
Gauss	AVG 23.351 MAX 27.017	AVG 0.601 MAX 0.831
Salt&Pepper	AVG 23.455 MAX 27.029	AVG 0.609 MAX 0.835
Speckle	AVG 23.639 MAX 27.549	AVG 0.621 MAX 0.839

Table 8. PSNR and SSIM values (average and max) referring to test set when λ_d is equal to 0.001 and mean noise intensity.

to the simplistic nature of the Discriminator with respect to the ones presented in [1] and [3], it was important to be sure that the task played by this part of the model was not irrelevant and negligible. The approach used to verify this, was to diminish the current λ_d value from 0.01 to 0.001 leaving all the other hyper-parameters as they were in version 1 (Tab. 7).
Not only the losses had a much more unstable behaviour (Fig. 8) but at the same time the average and maximum values of PSNR and SSIM decreased (see Tab. 8). This shows how the Discriminator's structure is suitable to decently help the Generator denoising the images. If that was not the case, better performances would be reached when the adversarial loss' weight is lower. Unfortunately, more drastic changes were to be expected, which means that the Discriminator does not fulfil its job up to its ideal potential.

5. Conclusion

In this paper is presented a GAN model for image denoising, which achieves good perceptual quality. As mentioned throughout this paper the challenges faced were several, starting from the design of the model. Once discovered that a simple one was not able to achieve good performances, the attention shifted on finding new ideas that would improve them. However, this meant also finding strategies to overcome the limited resources available on the developing platform. The time spent on this project gave to our team the insight of a real-case scenario GAN model, which implies learning how to deal with encoder and decoder blocks interacting with adversarial part, where the choice of hyper-parameter is a crucial aspect. Overall, the results presented are lower than the state of art, which could be due to the small amount of train epoch performed and to the simplistic stsructure of the Discriminator. A future extension could include developing a better structure for the Discriminator or changing it entirely to exploit a PatchGAN. We believe that, after these modifications, it will be possible to increase the noise intensity and still achieve good values of PSNR and SSIM.

References

[1] Jason Brownlee. How to develop a pix2pix gan for image-to-image translation, 2019. Link. 3, 7

[2] Boncelet Charles. Image noise models, 2005. Link. 2

[3] Ziyuan Wang et al 2020 J. Phys.: Conf. Ser. 1550 032127. An image denoising method based on deep residual gan. 2020. Link. 1, 4, 7

[4] Julien Guillaumin Fatimazahra Imani. Single image super-resolution using gans - keras implementation, 2018. Link. 3, 4, 5

[5] Rina Komatsu and Tad Gonsalve. Comparing u-net based models for denoising color images. 2020. Link. 4, 5

[6] Alenrex Maity; Anshuman Pattanaik; Santwana Sagnika; Santosh Pani. A comparative study on approaches to speckle noise reduction in images, 2014. Link. 3

[7] From Wikipedia. Gaussian noise. Link. 2