Final Project Report
pix2pix
Team AVBZ

Antonin Vidon
MS Data Science
Columbia University
New York, NY
av3023@columbia.edu

Hugo Artigas
MS Business Analytics
Columbia University
New York, NY
ha2605@columbia.edu

Waël Boukhobza
MS Data Science
Columbia University
New York, NY
wab2138@columbia.edu

***Abstract*—Colorizing black and white images has been one of the most exciting applications of deep learning. The goal of this project was to build a model able to perform this task as well as other image-to-image translation operations, with a flexible architecture. To do so, we used a state-of-the-art architecture of the research area : the conditional GAN `pix2pix` [1].**
Our approach was to first try to replicate the results of `pix2pix` original paper. The main dataset used was Paris facade images and the goal was to reconstruct pictures from colored architectural sketchs. As it is quite tedious to quantitatively assess a cGAN's performance, we compared and mixed the use of different losses (*L1*, *L2* and the Binary Cross Entropy loss) and conducted a range of ablation experiments.
To perform colorization, we used the *Country211* dataset, created black and white images and then trained our `pix2pix` implementation on it. We succeeded in achieving visually convincing and promising results. The model retrieves the colors of main objects or places such as the yellow color of the sun yellow or the blue of the sea.

***Index Terms*—conditional GAN, Image colorization and reconstruction, U-Net architecture, PatchGAN discriminator.**

I. Introduction

Generative Adversarial Network (GAN) has turned out to be extremely efficient in performing image-to-image translation tasks. The main idea of GANs is to simultaneously train two models; a generator G that captures a certain data distribution, and a discriminator that determines whether a sample came from the original distribution or from a generated image. Conditional GAN is a type of GAN that involves the conditional generation and discrimination of images. In cGAN, the generator and the discriminator also take as input a prior knowledge about the desired output. For example, in the particular case of colorization, this information corresponds to the black and white image.

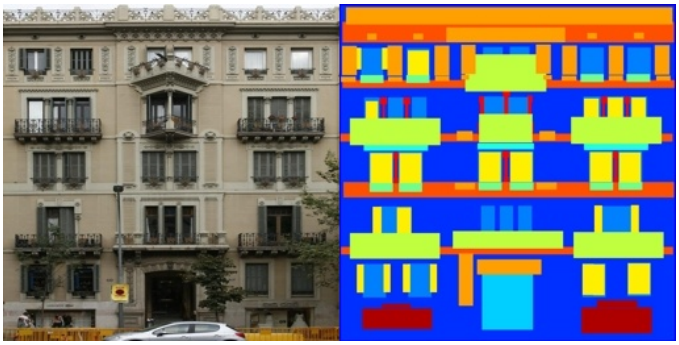
`pix2pix` proposes a general solution to many image-to-image tasks based on the cGAN. This model has turned out to be extremely efficient and general. It succeeds to perform well on a lot of different tasks such as image colorization, sketches to images, architecture labels to photo, semantic segmentation... The architecture of this generator is based on the *U-Net* encoder-decoder network to keep track of high frequency information and its discriminator uses an architecture called *PatchGAN* which divides the images in patches and classifies each patch as real image or fake one.
In this context, we focused our work in two steps. The first one was to implement the `pix2pix` architecture, understand the influence of the different kind of losses present in the model and try to replicate a part of the initial results [1]. Then, we used this model to perform black and white picture colorization.

II. Data

With our `pix2pix` implementation, we aimed to resolve the two following tasks: creating realistic facade images from semantic segmentation images and coloring black and white images. Thus, we needed to collect all the relevant datasets for these tasks.

This process was not a challenge for Facades since `pix2pix` authors shared their datasets [2]. There is 400 training, 100 validation and 110 testing images in Facades. Each image is 256×512 and their structure is always the same: in the left-hand part of the image we have the ground-truth 256×256 picture and in the right-hand part we have the label Maps. Obviously, `pix2pix`'s GAN has only access to the label map during inference time. See Fig. **??** for examples taken from the datasets.

Nonetheless, the dataset for the black and white task – which is our main focus task for this project – was not provided; we thus had to build it ourselves. To do so, we looked for an already existing dataset that would provide a good variety of colorful images (i.e. not just cat and dogs); the authors from `pix2pix` used ImageNet dataset, but its size (150GB) might be overbearing given our limited computing resources. Hence, the dataset we looked for also had to have a reasonable size. For all of these reasons, we chose the *Country211* [3] dataset from OpenAI. It is a subset from the *YFCC100m* (Yahoo Flickr Creative Commons



(a) Facades

100 Million) dataset, keeping only images that have GPS coordinate corresponding to a ISO-3166 country code. It matches all of our requirements and weighs 11GB, which is a reasonable size given our computing resources. It consists of 31,650 training, 10,550 validation and 21,100 testing images of various sizes.

We then converted all of these images to the `pix2pix` format as described for Facades and Maps datasets. To do so, we resized each image to a 256×256, then created their label Maps for our particular coloring task, which is here is just a black and white version of the image; we used Python library `PIL` which is adapted for this kind of task. We then horizontally concatenated each image to its corresponding black & white label map, which finally gives us our adapted images for `pix2pix`. See Fig. 2 for an example of such transformation.

We did not always use all of the images for training and parameterizing our models, only a portion of it; we then gradually increased the number of images used as we found best parameters until we included all of training/validation images from Country211 for our final training run.

This closes the Data part, and we will now describe the `pix2pix` model as well as our implementation of it.

III. Model

In order to complete the image-to-image translation task, we use the same approach as `pix2pix` [1], exploiting an adversarial paradigm and training both a generator and a discriminator.

A. Generator

The generator component is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks. The type of architecture is referred as "U-Net" in the original paper [1].

The input image is first downsampled through a sequence of convolution blocks and then upsampled through a sequence of deconvolution blocks. Each blocks, contains, in addition to its main layer, a module processing the filter maps with one

(a) Before transformation

(b) After transformation

Fig. 2: Examples of an image from *Country211* dataset before and after our black & white transformation in `pix2pix` format

or several of the following : batch normalization, dropout and ReLU/LeakyReLU activation. The skip connections involve the concatenation of the output channels from layer i in the downsampling block with those of layer $n - i$ in the upsampling block. The idea behind this operation is to allow to some "early" features to flow easily through the complex architecture and not to suffer from too heavy compression. More details can be found on the architecture diagram 3.

B. Discriminator

The discriminator takes both the condition image (e.g. architecture label in the case of Facades dataset) as well as an other candidate image, either the one created by the generator or the real corresponding image. Its architecture first concatenate the channels of both inputs and then operates a succession of convolution and batch normalization operations. The output is a one-channel feature map of size 30x30. The values obtained might be interpreted as the pre-sigmoid probabilities of each cell being a real image. A binary cross entropy loss with logits is used to process to improve the discriminator behavior against the one of the generator.

C. Implementation

We implemented `pix2pix` from scratch on PyTorch.

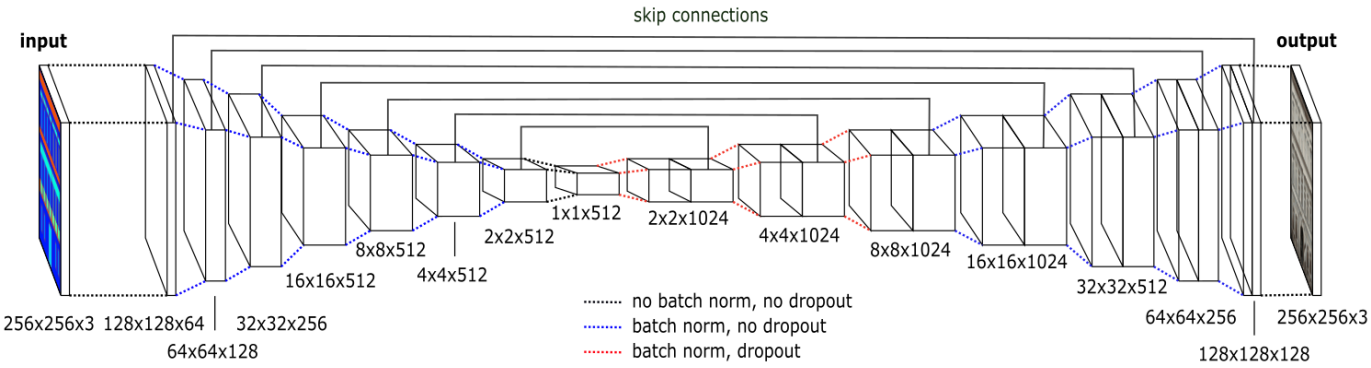


Fig. 3: `pix2pix` generator architecture

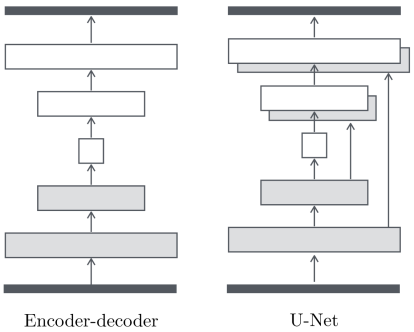


Fig. 4: Encoder-decoder vs U-Net architecture

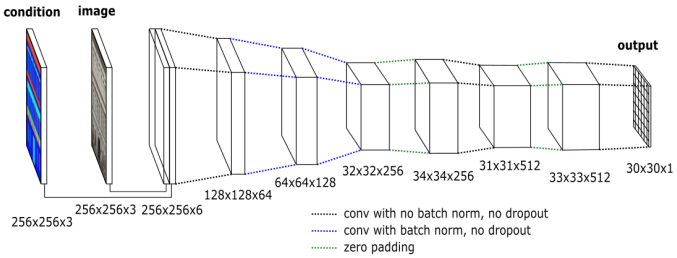


Fig. 5: `pix2pix` discriminator architecture

IV. Training

We perform training on GCP using T4 or A100 NVIDIA GPUs.

A. Objective function

The objective function of a conditional GAN (cGAN loss) can be written as:

$$\mathcal{L}_{cGAN}(G,D) = \mathbb{E}_{x,y}[log(D(x,y))] + \mathbb{E}_{x,z}[1 - log(D(x,G(x,z)))]$$

where G and D are respectively generator and the discriminator, z the random noise, x is the prior information and y thee ground truth. It should be noted that there is no random noise fed to the `pix2pix` model, but it is replaced by adding random dropout in the generator architecture to the first three decoder layers. The way of training the model of cGAN is equivalent of how to train a classic GAN: the generator tries to minimize this function again a discriminator which tries to maximize it.

In the image-to-image translation task, the generated image must be close to the ground truth image. Thus to the cGAN loss is added an *L1* loss in `pix2pix`:

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x,z)\|_1]$$

The *L1* norm is preferred to the *L2* norm because it generates less blurred images.*L1* norm succeeds to extract high frequencies in the image which corresponds to the little details whereas the *L2* norm tends to extract mainly low frequencies (i.e. the general pattern of the image such as the light or building colors).

The overall `pix2pix` optimization problem can be written as:

$$\min_G \max_D \lambda \mathcal{L}_{cGAN}(G,D) + \lambda \mathcal{L}_{L1}(G,D)$$

where λ is an hyper-parameter of the model which controls how close we want the generated image to the ground truth. The higher λ is, the closer the two images are. In the original paper([1]), this parameter is to 100, but in our study, we found that the model was more efficient with a parameter of 200.

B. Hyper-parameters

We present below the hyper-parameters used at training time by our `pix2pix` implementation.

- L1 loss weight coefficient λ = 200
- Batch size = 1
- Learning rate for the ADAM optimizer = 2e-4
- Dropout probability = 0.5
- Momentum coefficient fro ADAM optimizer betas = (0.9, 0.999)
- LeakyRelu negative slope = 0.01
- Weight initialization for convolution and deconvolution layers = $\mathcal{N}(0, .02)$

	Generator	Discriminator	pix2pix
Parameters	54,413,952	2,768,641	57,182,593
Memory (in Mo)	217.7	11.1	228.8

TABLE I: Memory footprint

C. Data Augmentation

As for data augmentation, authors from the paper applied a random jitter to their Facades dataset: they resize each image to 286×286 and then randomly crop them back to 256×256 size. They also use mirroring on all of their datasets. We found that using these data augmentation routines slightly worsened our results – and in fact some images from pix2pix datasets were already jittered –, so we decided not to use them in our final implementation.

D. Model Weights

The following table (I) contains the number of parameters and the memory needed to save the generator, the discriminator and the overall pix2pix model:

V. Results

A. Facades dataset

We trained our adversarial models for 200 epochs on the training subset of Facades. This is a replication of what was done in the original paper [1]. We save model weights every 20 epochs so as to be able to reconstruct generator outputs on the validation set throughout training. These outputs are presented below for a mode without data augmentation on figure 6 and a mode with data augmentation on figure 7. Although we do not have a clear set of metric to assess the quality of the images, we observe that the outputs with no data augmentation tend to get better between 20 and 100 epochs and then struggle to improve. This is an indicator of overfitting on the training set, provoking a weaker ability to generalize on new pictures.

On the other hand, the outputs of the generator consistently improve during training when we incorporate data augmentation, as we can see on figure 7. We observe new details within windows, more credible roofs and more natural colors on synthetic pictures throughout episodes.

In order to have a better picture of final results after all 200 epochs. We plot 16 generated images (condition in validation set) for both modes. The results are presented on figure 8 without augmentation and on figure 9 with data augmentation. We observe a large diversity of reconstructed facades with augmentation in contrast to the mode with no augmentation. In the latter case, the architecture images almost look like impressionist paintings, with colors messing with each other.

Overall, our results on the Facade dataset are similar to the ones obtained in [1]. We obtain seemingly credible output images for a same range of epochs and same training hyperparameters. Our training might be slightly slower, as we do not rescale pixel values between -1 and +1 as in the paper

Fig. 6: Generated images throughout epochs for a same validation condition without data augmentation

but rather between 0 and +1. We believe that an additional 100 epochs would have even more improved our results with data augmentation as we obtain similar training and validation outputs, indicating that we are not facing the overfitting issue yet.

B. Image colorization

For image colorization, we trained pix2pix on 20 epochs. Only a portion of the generated images seem to indeed have realistic coloring, even when taking the same hyperparameters as in the pix2pix paper. This might be explained by the lack of data (30,000 for our implementation vs 1.2 Millions for pix2pix), but also by the lack of power of the model itself. Indeed, the color outputs from the model presented in the original paper are not all impressive (cf Fig. 10), and on the Internet many other users came up with the same conclusion. This is why we decided to enhance our colorization model, inspired by [4] (itself inspired by *Super Resolution* literature https://arxiv.org/abs/1609.04802); the idea is to pre-train the Generator on a supervised task at the beginning of the training.

Firstly, we pre-train the down sampling path of the Generator for classification (in fact, we use a ResNet-18 pre-trained on ImageNet). Then, we then pre-train the whole generator on the task of colorization with L1 loss using our training dataset. After these two stages, we will train our

Fig. 7: Generated images throughout epochs for a same validation condition with data augmentation

Generator and Discriminator as before with the cGAN and L1 loss.

On Fig. 12 are the results for our `pix2pix` replication model, our enhanced `pix2pix` model trained on 10 epochs and our enhanced Pix2Pix model trained on 20 epochs. We see that our enhanced model produces seemingly better results, and giving it more epochs even ameliorate results. At first, we wanted to match the training process of the original model and stop at 10 epochs (original model: even 6 epochs). But given Fig. **??**, we saw that all the losses (except gan_loss)were steeply decreasing, and decided to run the model on 10 other epochs (on the right part of Fig. **??**, all losses start from epoch 10).

Eventually, we can take a look at the color distribution within the generated images and compare them to the ground truth; we take a look at the color spectrum of a validation example on Fig. 14. We see that, indeed, our enhanced 20 epoch-trained `pix2pix` seem to have the color distribution that fit the best to the ground-truth distribution.

Fig. 8: Generated images after training of 200 epochs without data augmentation

VI. Ablation experiments

A. Generator Losses

We first took a look at the generator loss choice impact on the training and outputs. We used different possibilities for the generator loss. We took into account the two initial losses present in `pix2pix` model: the *L1* norm and the cGAN loss. In addition, we incorporated also the *L2* norm loss in order to understand its effect on the outputs. We run the model over 50 epochs with the Facades dataset and used the seven following combinations for the generator loss while keeping the cGAN loss for the discriminator. The combinations are the following.

- *L1* loss
- *L2* loss
- *L1* loss + *L2* loss
- cGAN loss
- *L1* loss and cGAN loss
- *L2* loss and cGAN loss
- *L1* loss + *L2* loss and cGAN loss

We notice on figure 15 that if we train the model using only the *L2* norm for the generator, it produces a model with a low *L1* metric, even lower that if we train the model with *L1* norm as generator loss. In the particular case of image reconstruction from architectural label, the *L2* norm seems to not produce blurred images for the training set. However, looking at the different outputs over the training set (16), the model using *L2* norm loss tends to be a little more unclear than a *L1* model. Nevertheless, it always tries to capture the color of the ground truth image for the training set. It captures low frequency patterns. It generates over-fitting when we use

Fig. 9: Generated images after training of 200 epochs with data augmentation

Fig. 10: Original `pix2pix` results on colorization task; left: generated images, right: ground-truth

the validation set. Because what is relevant here is to define a model able to rebuild an image from a sketch in a general case and not just outputting the correct color only for the training set. The model loses its generality.
Figure 16 was built only with a low number of epochs (50) over the training set in order to avoid over-fitting and being able to identify the true influence of each training method.

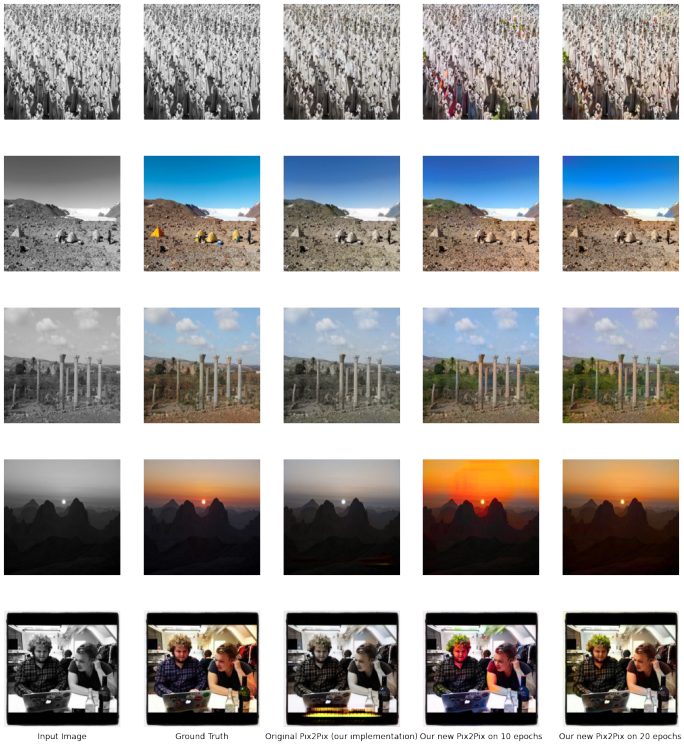


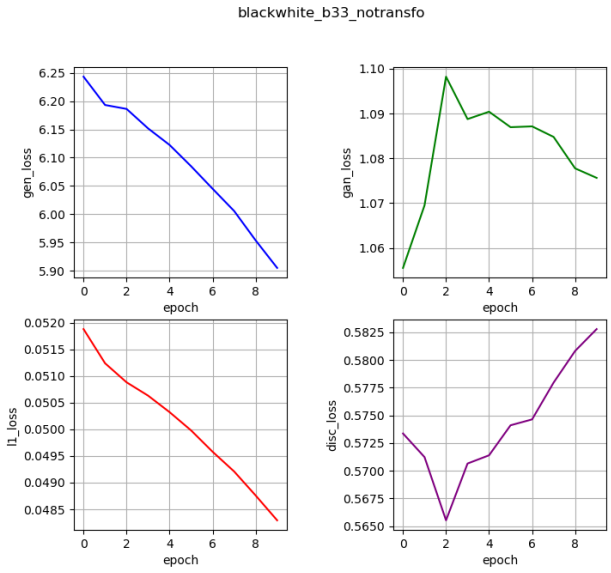
Fig. 11: Different `pix2pix` models' results on colorization task. From left to right: grey input; ground truth; our first `pix2pix` model's results; our enhanced `pix2pix` model's result on 10 epoch; same on 20 epochs

As we could expect, training the model only using the cGAN loss doesn't produce satisfying results(16). In order to achieve visually realistic pictures, the cGAN loss must be mixed with other ones. Still after 50 epochs, we can distinguish in the output the influence of the generator convolution kernels. Lastly, the most realistic models are mixing the cGAN loss with *L1* and/or *L2* norms over the training set. Looking at the outputs for the validation set(17), we can clearly see the difference of which frequencies the two different norm are extracting: the image details produced by the *L1* norm model are realistic but it keeps track of the input color, whereas the *L2* norm model gives realistic colors with a blurred output image.

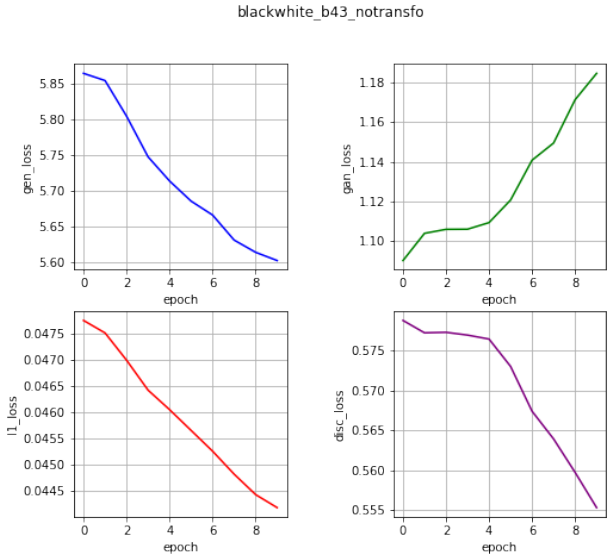
B. Data augmentation

As we were in possession of relatively small datasets (e.g. 400 training images for Facades), our complex model was naturally prone to overfitting. In order to assess the significance of data augmentation, we trained `pix2pix` on Facades for 200 epochs with and without data augmentation using *L1* + cGAN loss. In this part, data augmentation refers to random mirroring and cropping. Namely, we first toss a coin and mirror both the condition image and the real image accordingly, and then we rescale our image to 286x286 and random crop it to

Fig. 12: Loss of our enhanced `pix2pix` model on 10 epochs



(a) Losses on 10 first epochs



(b) Losses from epoch 10 to epoch 20

Fig. 13: Loss of our enhanced `pix2pix` model on 20 epochs for colorization task

256x256. This strategy is used to enrich our dataset so as to prevent overfitting.

On figure 18, we observe the results of our training experiment. The first two rows show the output of the generator on conditions encountered during training. We observe that without data augmentation, the network is trying to replicate the ground truth as much as possible, whereas with data augmentation the output image is more credible as a real image while being more structurally different from the ground truth. This is an indicator of overfitting. Indeed, the lack of data augmentation makes our generator able to perfectly replicate the ground truth given the input condition after a sufficient number of epochs. This is due to the overcomplexity of our model with respect to our dataset diversity. When encountering a new input condition, our model will behave poorly and produce an image likely to be unrealistic. Indeed, this is what we observe on the third vs fourth row of figure 18, exhibiting outputs on the validation set. Model trained with data augmentation produces a credible facade while the other model's output is easy to distinguish from a real facade.

It is also interesting to track the different losses with and without data augmentation. As we can see on figure 19, the generator loss decreases extremely rapidly without augmentation. This is due to our model believing that it fully understands the dataset underlying distribution and producing output very close to ground truth but with poor generalization capability. Indeed, we observe that the training *L1* loss depleates a lot faster without augmentation than with augmentation.

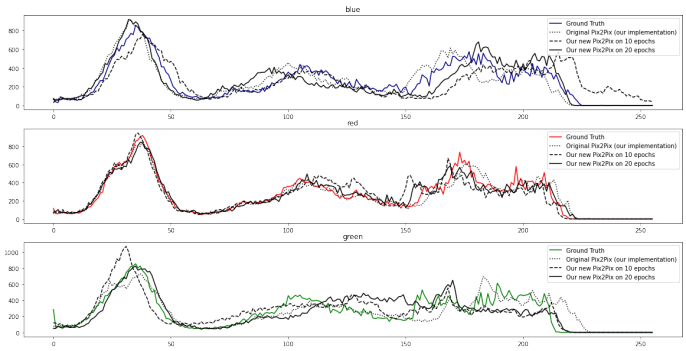
C. Batch size

We conducted training for 100 epochs on Facade with a batch size of 32 vs 1. Our results are shown in figure 20 and 21 respectively for 1 and 32 as batch size. We observe that the behavior of the loss is more erratic with a larger batch size, leading to difficulties for the cGAN loss to decrease.

VII. Conclusion

This project sheds into light the efficiency and the generability of `pix2pix` to a range of applications. From building a picture from architectural labels to colorizing black and white images, this conditional GAN performs well on a wide array of image-to-image translation tasks. The choice of loss expoited by the generator affects the rendered output and make the

(a) Generated images by our various models



(b) Color distributions on blue, red, green channels

Fig. 14: Comparison of the color distributions of the outputs of several models and the ground-truth image. In color: the ground-truth distribution, in dotted line: prediction from our replication of `pix2pix`, in dashed line: prediction from our enhanced `pix2pix` on 10 epochs, in black full line: prediction from our enhanced `pix2pix` on 20 epochs

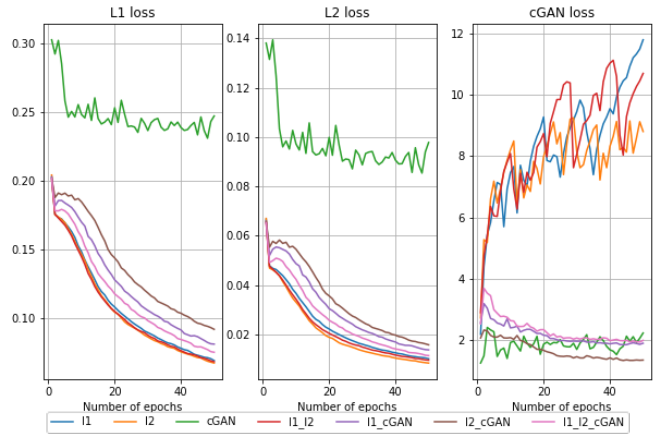


Fig. 15: Evolutions of the three main metrics for the generator while training it with different losses over 50 epochs.

image more or less credible to the eye. As in the original paper [1] experiments, the *U-Net* architecture proved to be more convincing at test time when decreasing our batch size. We also brought out the crucial importance of data augmentation in preventing fast overfitting on the training set.

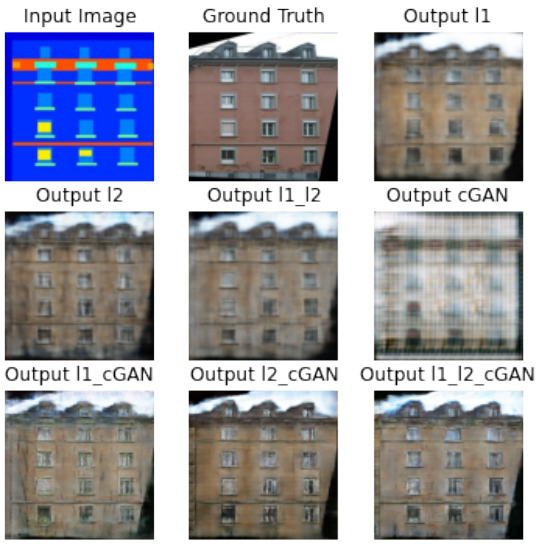


Fig. 16: Output of the model trained for 50 epochs using different losses. Training set.

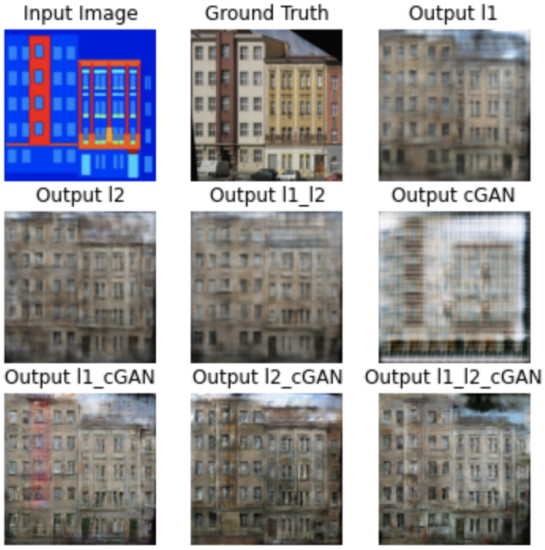


Fig. 17: Output of the model trained for 50 epochs using different losses. Validation set.

References

[1] Philip Isola and Jun-Yan Zhu, Tinghui Zhou and Alexei A. Efros, Image-to-Image Translation with Conditional Adversarial Networks, arvix: https://arxiv.org/abs/1611.07004, doi: 10.48550/ARXIV.1611.07004

[2] `pix2pix` datasets: http://efrosgans.eecs.berkeley.edu/`pix2pix`/datasets

[3] Country211 dataset: https://github.com/openai/CLIP/blob/main/data/country211.md

[4] Colorizing black white images with U-Net and conditional GAN https://towardsdatascience.com/colorizing-black-white-images-with-u-net-and-conditional-gan-a-tutorial-81b2df111cd8

[5] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, Wenzhe Shi, Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network, arxiv: https://arxiv.org/abs/1609.04802, doi: 10.48550/arXiv.1609.04802

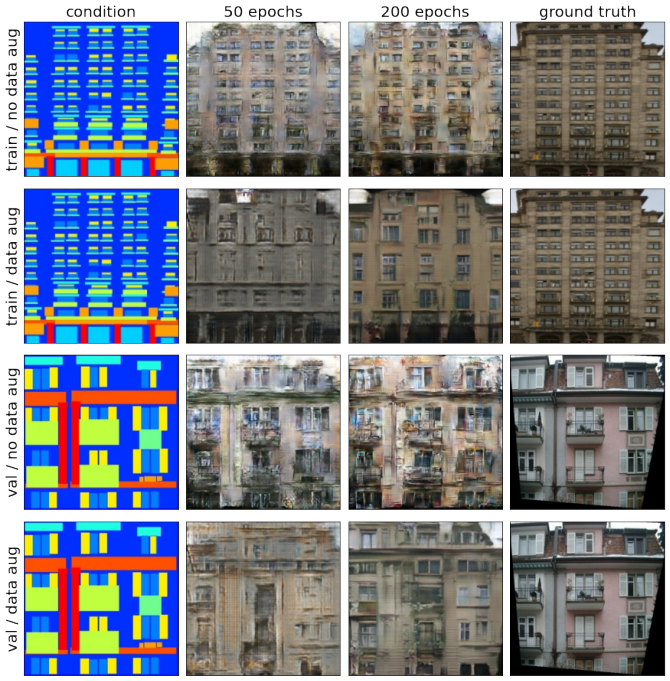


Fig. 18: Outputs of the generator with input with training or validation set with or without data augmentation (random mirroring and cropping)

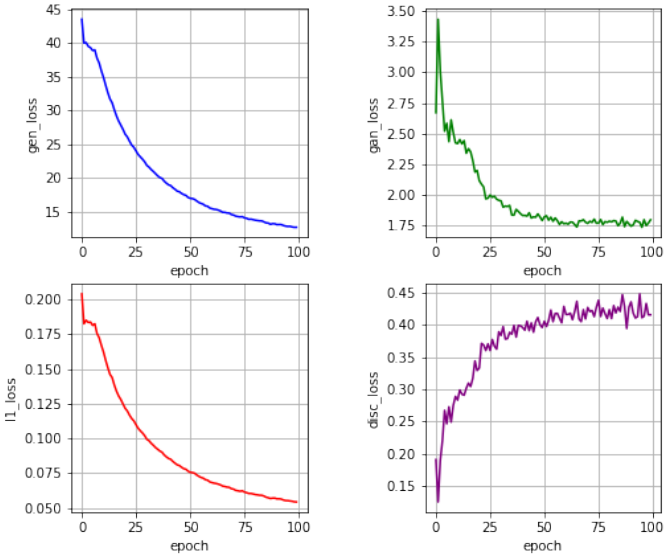


Fig. 20: Evolution of losses throughout training on Facades with batch size of 1

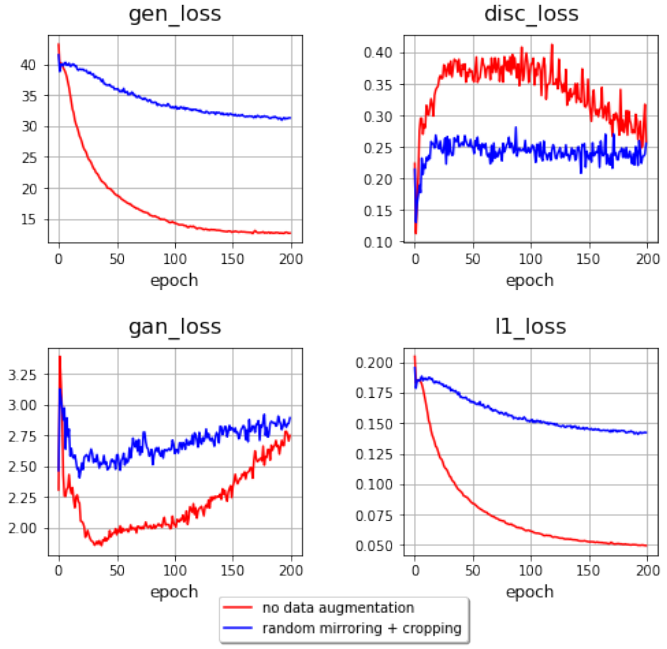


Fig. 19: Evolution of losses throughout training on Facades with or without data augmentation (random mirroring and cropping)

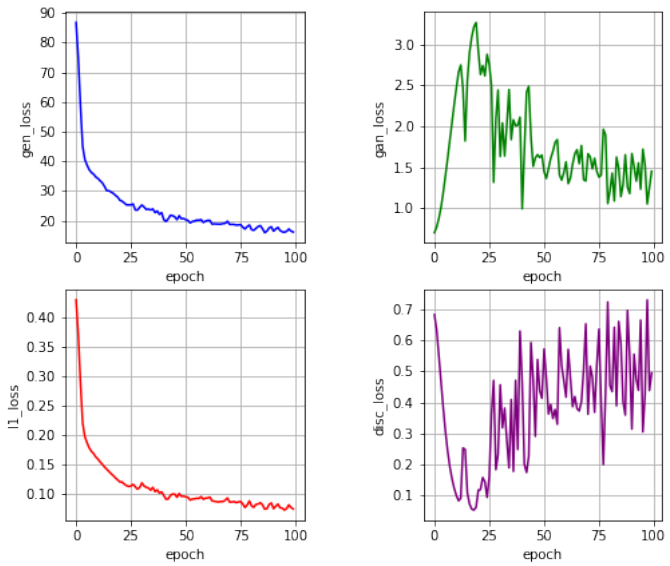


Fig. 21: Evolution of losses throughout training on Facades with batch size of 32