Communication Systems
Semester Project
Bachelor Semester 6 - Spring 2020

Aerial Photography Vectorization

Student: Niels Escarfail

Supervised by: Nils Hamel
Prof. Dr. Frédéric Kaplan

July 30, 2022

Digital Humanities Laboratory
EPFL

Acknowledgement

I would like to extend my sincere thanks and gratitude to Nils Hamel and the entirety of the Digital Humanities Laboratory for their precious and kind help throughout the completion of this project, which has been a delightful experience despite the complicated world situation this beginning of the year.

I am incredibly grateful to have had the opportunity to work with you and wish the entire team great success in their future projects.

Abstract

Aerial photography vectorisation aims to faithfully reproduce a location's plans from its aerial photograph. The world is changing nowadays quicker than ever, may it be from human development or natural changes, with a continuous augmentation in constructions, deforestation and geographical changes due to climate change. Such events create much more applications and a genuine need for real-time maps development, not mentioning the already existing possibilities such technologies could offer.

The objective of this project is to get familiarised with the Pix2Pix Conditional Adversarial Network through the specific task of aerial photography vectorisation, in order to attain an accurate and viable maps depiction, while analysing the potentialities and limits of this network. Methodologies for processing and analysis of the acquired network outputs will be developed in order to quantify the obtained results. The data acquired through this project will hopefully begin to clear a path for future researches involving image-to-image translation problems related to this subject.

Contents

1 Introduction

The word 'map' comes from the Latin word, *Mappa Mundi*, which had for literal translation "sheet of the world." The earliest known world maps date to classical antiquity, since then, knowledge of the approximate size of the Earth allowed cartographers to estimate the extent of their geographical knowledge, and to indicate parts of the planet known to exist but not yet explored as *terra incognita*. As we came closer to the modern age, world maps became increasingly accurate; covering almost the entirety of the world. In the modern era, big tech companies have developed plans numerically, and we have witnessed the birth of Google Maps, Bing Maps, Mapbox and others. However, creating these maps is a costly and lengthy process involving multiple actors and techniques. For instance, Google has appealed to both governmental and independent organisations to improve its geographical data. Nevertheless, even with all these factors, there still exists maps which are not an actual depiction of reality, especially in emerging countries where construction is continuous, and interest from these tech giants is minor. No complete up-to-date vectorised map of the world exists as of May 2020.

The goal of this project is therefore to explore the possibilities offered by the relatively new Pix2Pix Conditional Adversarial Network in aerial photography vectorisation, defined by the translation of cadastral aerial images into cadastral plans [1].

In this project, we gain insight on Conditional Adversarial Networks' (in particular Pix2Pix's) potential for aerial photography vectorisation. This project aims to define the limits of Pix2Pix's potential for this particular purpose, for example, by defining if a network trained on a particular scale and region can generalise to multiple scales and other regions. It also aims to define the optimal image data configurations, regarding the image zoom, for instance, in order to obtain the most efficient algorithm. Throughout the advancement of this project, we brought particular attention to the training data parameters and their influence on the results of their respective trained Pix2Pix instances.

For this purpose, we created various training data through MapBox Studio's API, then used it to train the Pix2Pix network on default parameters. Finally, we used the Python Imaging Library in order to analyse test output images and quantify the network's performances in function of its initial training data. The rest of the report follows the development process of this project analysed in the corresponding sections :

1. State-of-the-art of Pix2Pix's usage and particularly in aerial photography vectorisation.
2. The data collection process and its intentions.
3. The neural network training.
4. Describing the post-processing of recorded testing data and interpreting it in Pix2Pix's performance for aerial photography vectorisation.

2 Pix2Pix overview

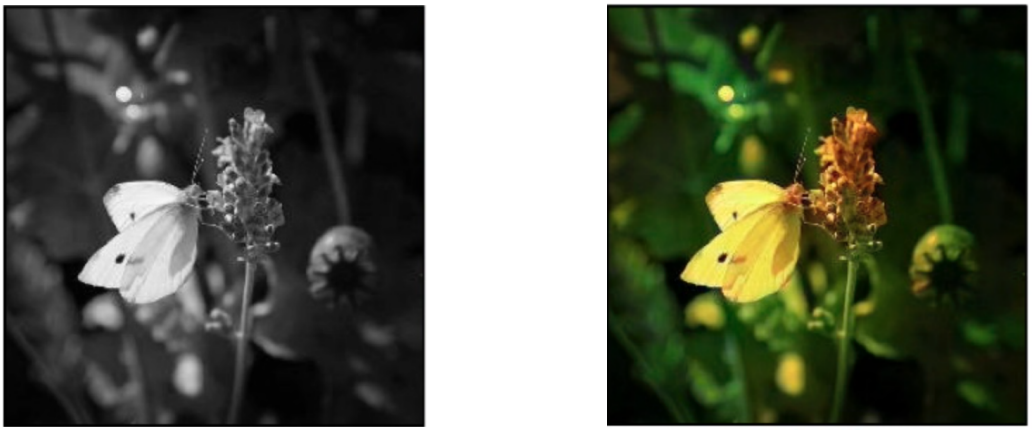
2.1 How it works

The Pix2Pix paper *"Image-to-Image Translation with Conditional Adversarial Networks"* comes up with a general solution to tackle image-to-image translation problems, defined by the task of translating one possible representation of a scene into another. The paper published in November 2016 quickly became one of the best-known computer vision papers.

The benefit of the Pix2Pix model is that compared to other Generative Adversarial Networks for conditional image generation, it is relatively simple and capable of generating large high-quality images across a variety of image translation tasks. Thus, the paper provides a one-size-fits-all solution to rule them all. The particularity of GANs is that these networks not only learn the mapping from an input image to output image but also learn a loss function to train this mapping. This property makes it possible to apply the same generic approach to problems that traditionally would require very different loss formulations. Indeed, this broad applicability and ease of adoption are without the need for parameter tweaking. The creators of Pix2Pix have released multiple versions of the network on the internet, allowing its users to unleash the network's capacity freely.

2.2 Pix2pix current usages

Our imagination only limits the applications of such a versatile network. Pix2Pix's creators have shared multiple implementations of the model, as well as comprehensive guides which allow almost anyone to explore the model's capacity. In the first place, they shared a few examples of the usages of the network, trained on small (about 300 images) and large (50000+ images) datasets. For instance, the model can be used to translate images of daytime to nighttime, from black & white to coloured (cf. Figure 1), aerial to map, or sketches to pictures, and vice versa.



(a) Input (b) Output

Figure 1: Black & white to colour

Since then, other researchers have found many practical usages of the network. For instance, segmentation like researchers of IEEE, in *Image-to-Images Translation for Multi-Task Organ Segmentation and Bone Suppression in Chest X-Ray Radiography* [2], forecasting a promising future for Pix2Pix and GANs in general.

2.3 Pix2pix in aerial photography vectorisation

As mentioned before, Pix2Pix's creators notably used it in order to obtain map data from aerial images. The original paper presents the results of Pix2Pix trained on a dataset of 1096 images (cf. Figure 2). Please note that this particular example seems to be the best result in the entirety of the test set, and the presented network did not seem to be able to generalise complex shapes or distinguish landscapes accurately.

(a) Input (b) Output

Figure 2: Example given in the Pix2Pix paper

2.3.1 Pix2Pix in this project

An aim of this project was, therefore, to improve the datasets used to train the network, compared to the original paper. By doing so, we will be able to determine whether Pix2Pix has a potential in future research in order to determine efficiently and precisely a map associated with an aerial image.

Please note that no changes have been made to the default Pix2Pix network model's parameters, the most relevant of which are listed below :

- Number of generator filters in the last convolutional layer.
- Number of discriminator filters in the first convolutional layer.
- Specify discriminator architecture (basic, n_layers, pixel). The basic model is a 70x70 PatchGAN and n_layers allows you to specify the layers in the discriminator.
- Specify generator architecture (resnet_9blocks, resnet_6blocks, unet_256, unet_128).
- Use instance normalization or batch normalization (instance, batch or none).
- Network initialization (normal, xavier, kaiming, orthogonal), with a defined scaling factor (default is 0.02).
- Add no dropout for the generator.

3 Data Collection

The data collection process represented a significant part in the development of this project. In total, we constructed 32 different datasets, mostly composed of 1000 satellite/map images pairs.

In order to obtain the map view required to train the algorithm, we considered the following providers: Google Maps, Bing Maps, and Mapbox. The final choice went to Mapbox's Static Images API, mainly because they provided users with a detailed and modular interface in order to create Map Styles: Mapbox Studio [4]. Mapbox Studio is a Mapbox application allowing us to design custom map styles. This feature permitted us to select which elements we want in our data. Using Mapbox Studio, a map style removing all text labels, buildings, and arbitrary zone limitations visible on "usual" maps. This feature left us with a map consisting mainly of only green, blue and beige, representing respectively green lands/parks, water and land (cf. Figure 3b). The choice to leave out building labels was intentional, as the data provided to represent these buildings was limited. As we will see through datasets containing such labels, this scarce data leads to inconsistencies in the Pix2Pix network's results (cf. Figure 5b, Figure 17).

(a) Satellite view (b) Main style used (c) With buildings

Figure 3: Different map styles obtainable with Mapbox Studio

We collect the images through Mapbox's StaticStyle Python API. We wrote a python script which, from various `JSON` lists of world cities and their coordinates (depending on the desired dataset), requested the desired features from Mapbox's server. The parameters of a request are the following :

- The style id, from a style created in Mapbox Studio. This parameter can select the view from satellite to any custom-made style.
- The coordinates (longitude & latitude) of the desired image.
- The zoom value of the desired image.

This python script allows us to collect Satellite and Maps images separately in `.jpg` format, which we then append through the `combine_A_and_B.py` script provided with Pix2Pix (cf. Figure 4).

3.1 Initial training sets

In the first place, we based our approach on the general assumption that autoencoders generally work well with a significant amount of data. Thus, we start by creating 2 "brute force" datasets, composed of 6000 image pairs, whose coordinates were taken from a `JSON` file listing the 10000 biggest cities in the world. The chosen styles were the main map style described in the **Data Collection**'s section introduction, and the zoom was 13.

(a) Valid pair (b) Invalid pair

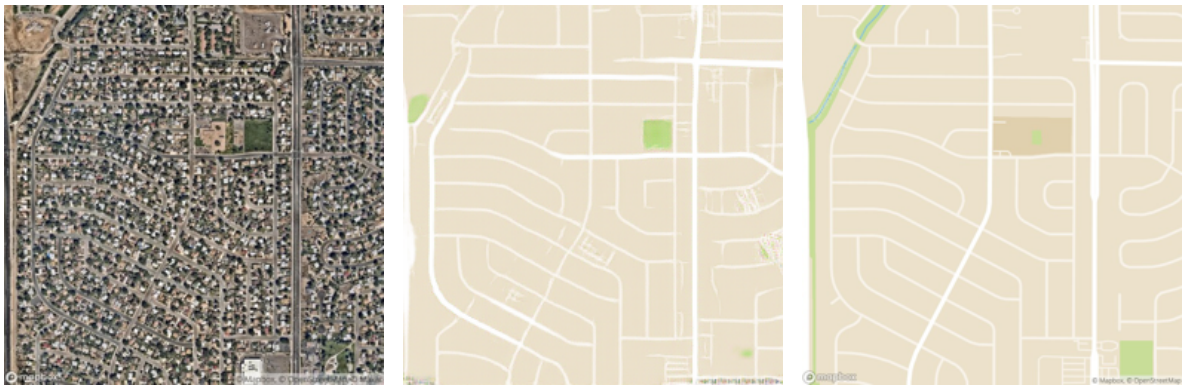
Figure 4: Example pairs from the first dataset

These datasets quickly turned out to be unusable, as many coordinates were inaccurate, satellite images turned out to be obscured by clouds, even sometimes blurred (cf. Figure 5b). We will later discuss this further in the **5.3 Results discussion** section.

(a) Satellite view (b) Network output (c) Expected output

Figure 5: First two datasets output examples

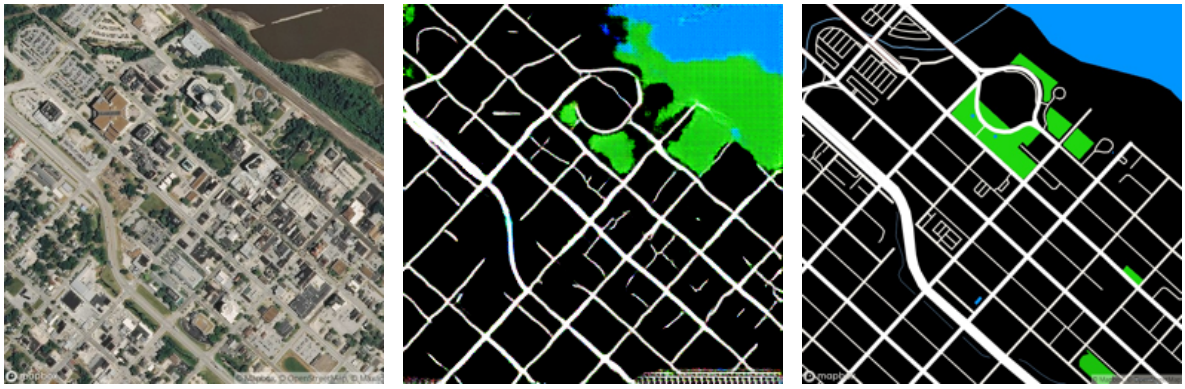
Our next goal was to reach a network with correct performance. To do so, we created a new dataset composed of 1000 images of the 1000 biggest cities in the United States of America. Taking their coordinates from a new `JSON` file, we also decided to augment the zoom to 15, as the previous zoom value seemed too remote compared to the working example offered in the Pix2Pix paper. This dataset finally showed promising results (cf. Figure 6), and from then on, we decided to push the algorithm in order to define its boundaries.



(a) Satellite view (b) Network output (c) Expected output

Figure 6: Example output for set 3 (United States, Zoom 15)

After these initial trials, we wanted to verify whether the contrast in an image would influence the performance of Pix2Pix's network. To do so, we created a dataset made up of the same images as the previous dataset (USA, zoom 15), but with a much more contrasted look (cf. Figure 7c), created with Mapbox Studio.



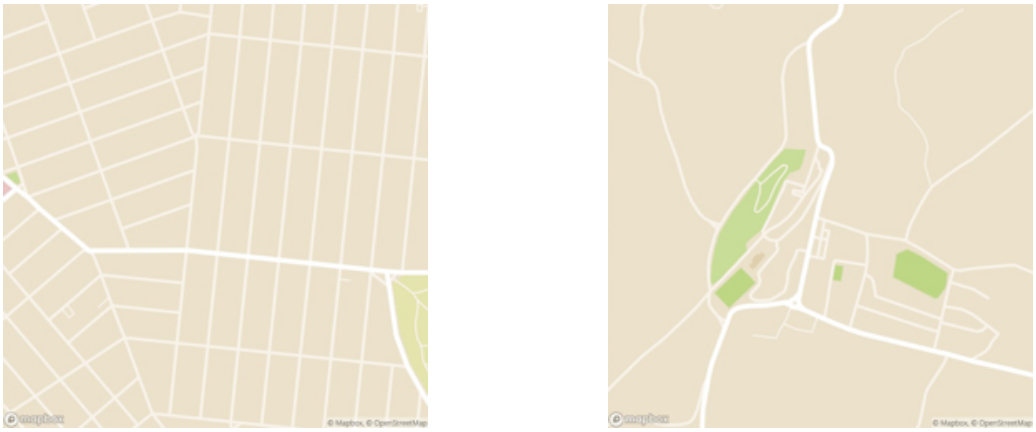
(a) Satellite view (b) Network output (c) Expected output

Figure 7: Example output for set 4 (United States, Zoom 15, Contrasted)

Given the similar results as the previous dataset, we preferred not to continue with such a style, as it was less intuitive and less pleasant to use in comparison to conventional map aspects.

3.2 Cities comparison datasets

A particular aspect of Pix2Pix we wanted to explore through was whether the network would work better on some areas of the world and specific city layouts. Urban planning is very disparate throughout the world, with the United States of America mainly hosting grid cities like New York (cf. Figure 8a), and other regions hosting unsystematically developed cities, like Sevilla, for instance. In order to first compare the USA cities dataset results with other regions, we also created a dataset composed only of European cities with the same parameters (cf. Figure 8b).



(a) Map view of New York

(b) Map view of an italian city

Figure 8: Different cities configuration

Furthermore, in order to verify the network's performance related to the region on which we train and test it, we built the next datasets from five different cities throughout the world with clearly distinct architectures: New-York, Paris, Rio de Janeiro, Beijing and Jakarta. The zoom choice for these datasets was 16, in order not exceed the cities' limits by too much while maintaining a dataset size of 1000 images (cf. Figure 9).

(a) Paris

(b) Rio de Janeiro

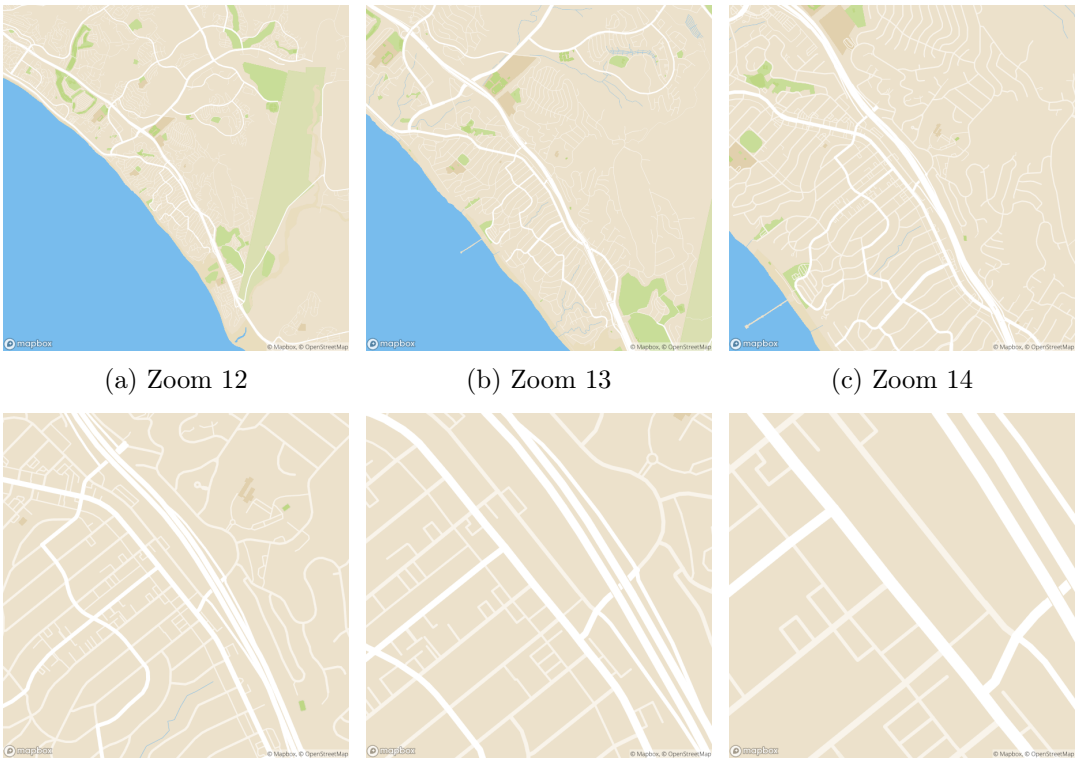
(c) Jakarta

(d) New York

Figure 9: Dataset examples for various cities

3.3 Zoom comparison datasets

As mentioned in the introduction, this project also aims to define optimal data parameters in terms of training, in order to achieve optimal results at testing time. The main parameter we can change when creating datasets is the zoom value. For this purpose, we observed which regions had good performances in previous training, and decided to create for each of these regions multiple datasets made up of the same coordinates while varying only the zoom value. In total, we created 20 different datasets for this section, six datasets for United States cities (cf. Figure 10), six datasets for Rio de Janeiro (ranging from zoom 12 to zoom 17), four datasets for New-York (cf. Figure 14) and four datasets for Beijing (ranging from zoom 13 to zoom 16).



(a) Zoom 12 (b) Zoom 13 (c) Zoom 14

(d) Zoom 15 (e) Zoom 16 (f) Zoom 17

Figure 10: Example of map images from the US cities datasets

An issue which is important to mention when constructing these zoomed out datasets is that there partly overlapping in the training images, leading us to sometimes reduce the number of training images to 500 in cities datasets. Note that this is not the case for the US cities dataset, as every image is one of a different city.

4 Neural network training and testing procedure

The Pix2Pix network was trained and tested on EPFL's School of Computer Science's cluster. We decided to use the PyTorch implementation of Pix2Pix; this version is under active development and can produce results comparable to or better than other published versions of the network.

4.1 Neural network training

We describe the datasets used for training in the above Data Collection section. We performed most of the training in this project on datasets of 900 images (800 training images, 100 validation images). We note that for datasets of this size, training took about three hours on a dual Maxwell TITAN X GPU.

We remind the reader once again that we made no changes to the default Pix2Pix parameters. We describe optional changes to the network's parameters in the section **2.3.1 Pix2Pix in this project**.

4.2 Neural network testing

We describe the datasets used for testing in the above **Data Collection** section. We performed most of the testing done in this project on datasets of 100 images. Testing usually took a few seconds in order to complete. After testing, the algorithm outputs three different `.png` files, for instance, `0_fake_B.png`, `0_real_A.png`, `0_real_B.png`. In order to view these 3-way pairs, the testing algorithm also generates an `index.html` file, aligning the original satellite image, the network's output and the expected maps output as in the example below (cf. Figure 11).

Figure 11: Example output of the testing procedure (US cities, Zoom 15)

5 Data post-processing and Analysis

5.1 Data

The testing of all trained Pix2Pix networks left us with 100 3-way-pairs of test images for each instance, as described above. As mentioned in the original Pix2Pix paper, evaluating the quality of synthesised images is an open and challenging problem which traditional metrics such as per-pixel mean-squared error do not measure correctly. Consequently, the human observer is mostly the best way to evaluate results. However, in order to quantise our obtained results, multiple metrics are used and usually reflect the presupposed thoughts regarding the performances of the algorithm relative to its training data.

5.2 Post-processing and analysis methods

This subsection describes our multiple approaches in order to measure our results. These metrics take into account pixel, colour and structure similarities between the expected map output value and the network's output, given satellite input data. As we will see in the next sections, there are many limits to these procedures and work could still be made in this regard, using more complicated and time demanding approaches such as perceptual studies on Amazon Mechanical Turk (AMT), for instance [5]. We performed calculations on the testing data through a single Python script, using the combination of the `PIL`, `NumPy` and `XlsxWriter` modules in order to analyse the images and store the results in a summary `.xlsx` file [6] [7] [8].

5.2.1 Pixel-per-pixel comparison

In order to compare the relative performance of each trained Pix2Pix networks, we wrote a pixel-per-pixel comparator in Python. This algorithm iterates over all test output pairs made up of the expected map and the generated map, and applies the following transformations :

- Transform them from RGB images to Palettes made up of only five dominant colours from our initial datasets. This conversion allows us to consider close shades of green as the same, for example.
- Convert these palettes back into RGB images, then NumPy arrays in order to subtract them absolutely, leaving equal pixels to be equal to 0 in the result.
- Finally, we simply count the proportion of 0 pixels compared to the total number of pixels in the image.

This comparator is especially challenging to implement rigorously, as the palette choice is up to the programmer. For this reason, we presented two different results based on two different palettes, one with four colours and one with six colours (cf. Figure 16).

Please note that this comparator does not account for the joint statistics of the result, and therefore does not measure the very structure that structured losses aim to capture. It also does not detect cases where the network predicted a road correctly, yet in a broader manner than that of Mapbox's "correct" map.

5.2.2 Mean Squared Error

The Mean Squared Error (MSE) metric relates to our previous Pixel-per-pixel comparator but accounting for the difference between the pixels, not exclusively counting the equal pixels. As we are dealing with colours, the MSE can sometimes be inaccurate, which is why we did not consider it as a primary metric in this project.

5.2.3 Structural Similarity Index

The Structural Similarity Index (SSIM), sometimes perceived as an improvement to the MSE, is mostly used to define image quality degradation. In this project, we use the SSIM to determine the clarity of the output image given by the model, as some outputs produced correct MSE results while being blurry to the human eye.

5.2.4 Perceptual validation

As mentioned before, the best way to determine the performances of a particular network is usually to look at test outputs directly. In this project, we looked at and visually compared the outputs; however, it would have been a significant advancement to produce an AMT experiment with plenty of test subjects, then to statistically define the proportion of participants fooled by the outputs of the network.

5.3 Results discussion

The metrics results for each dataset, computed as the mean of the metrics for each training pairs, are available in the summary table below. These values, along with visual analysis, help us reach certain conclusions regarding the behaviour of the Pix2Pix algorithm when working with maps and satellite imagery.

Overall, Pix2Pix shows promising results and manages to capture the typology of roads, buildings, water and green areas.

An essential factor to take into account when training the network is the dependence of Pix2Pix to its datasets. As we have witnessed through the project, inaccurate datasets such as the first datasets we created turned out to ruin the performance of the algorithm altogether. Even in the cases in which the network successfully managed to try to draw shapes, we quickly noticed the absence of specific land types in datasets, as the algorithm ignored these land types completely. In order to build an efficient network, one would thus need to train it on a sharply defined dataset, tailored especially for its use. We will come back to this aspect when reviewing the latter sections.

5.3.1 Training and testing on different types of cities

5.3.1.1 Comparison between cities and regions

This section specifies the results obtained with the section **3.2 Cities comparison datasets**. As we can see visually as well as in ..., there is a disparity between the turnout of the Pix2Pix network between different regions or cities, even with the same zoom value. Globally, the network performed very well in the United States of America, especially New-York. This outcome was an expected result, as, as we previously mentioned it, the United States is known for having grid-like cities, and New-York is one of them. We infer that Pix2Pix has an easier time learning the topology of well-defined and ordered cities. However, when looking at the results of Rio de Janeiro, for example, we also notice significant results in some cases, with the network managing to generalise well, probably since Rio de Janeiro is an affluent city in terms of landscapes, going from skyscrapers to favelas, all the while being complex. The city of Paris also properly managed to reconstruct parks, as they are abundant in the French capital.

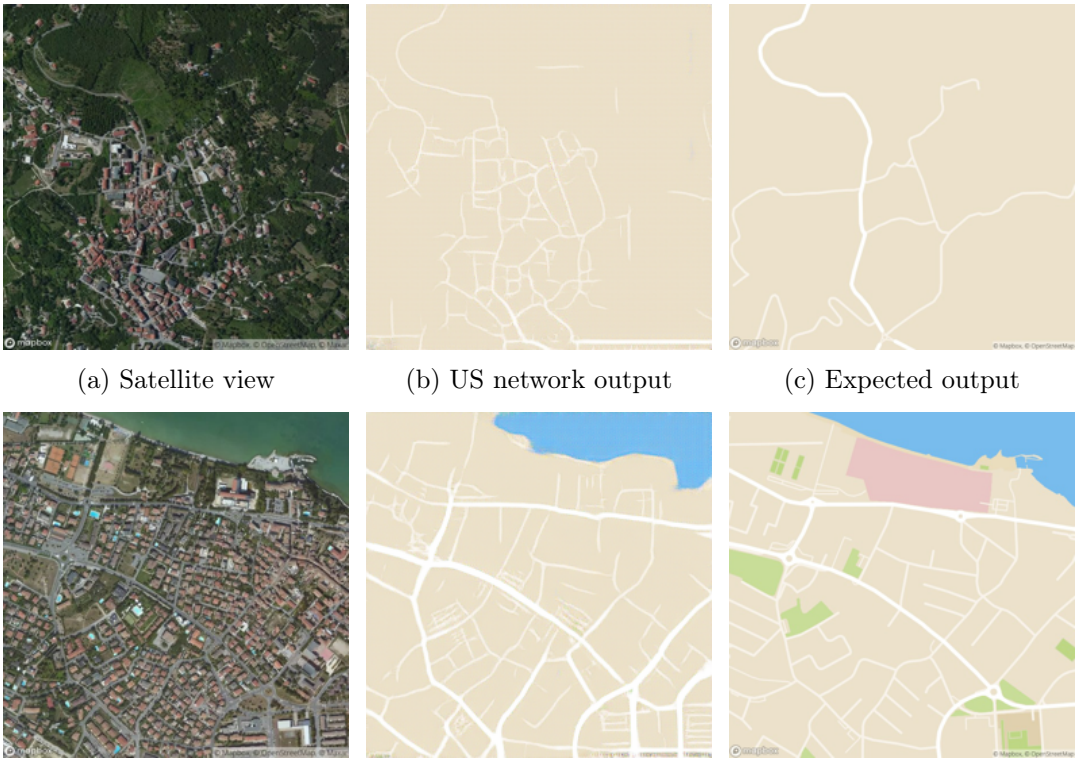
On the other hand, the sets which performed poorly were Beijing, Jakarta, and the European cities' dataset. Jakarta's failure can be explained by the complexity of the city, and by the imprecision of its maps.

One thing we must not neglige when viewing these results is that maps from the United States are most frequently updated, and the most precise. When viewing results from other regions of the world, we quickly notice incomplete mapping, for instance, on the European dataset,

where rural areas data solely consisted of roads instead of fields. These observations lead us to believe the maps data quality is more of a factor than the data topology when training a Pix2Pix network.

5.3.1.2 Testing a sole network in different regions

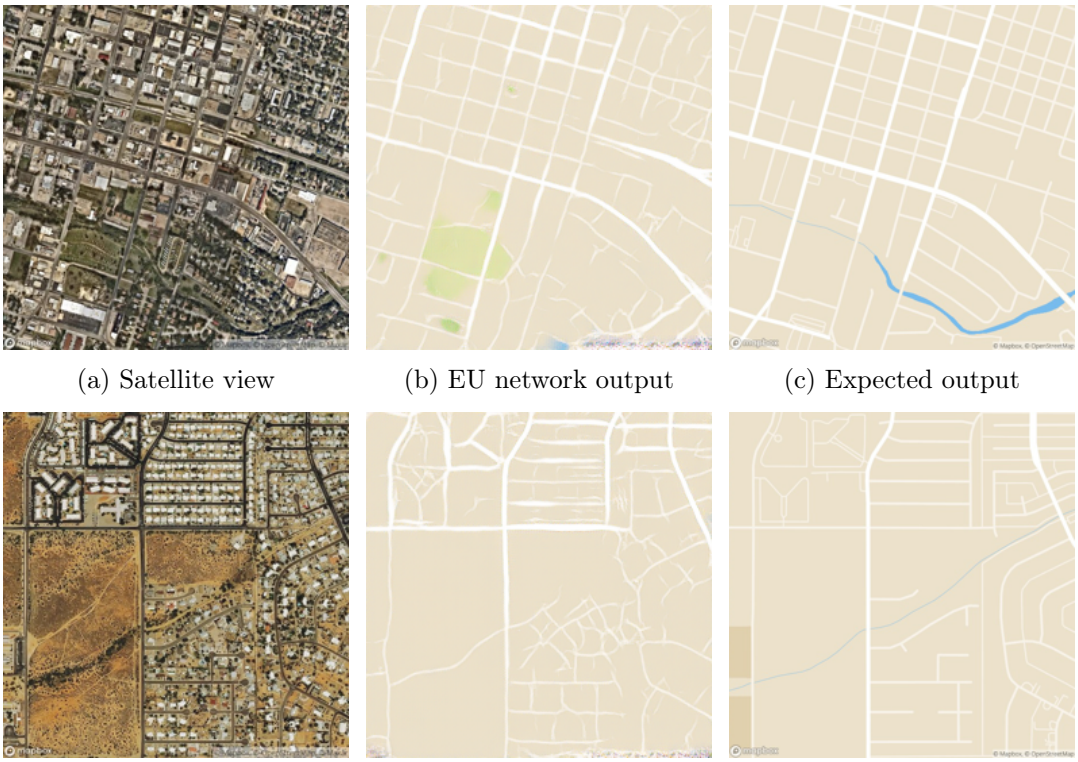
One question we asked ourselves during the project was whether it would be possible to create a single network capable of producing satisfactory results in various regions of the world, on different landscapes. In order to find a solution to this question, we tried running a network trained on a specific region in another region, for example, we tested the Europe-trained algorithm on US cities, and vice-versa. Doing this produces surprising outputs as if someone was reading a text written in a foreign language with an accent. As if there was a particular grammar to each region of the world. The network trained on the United States wanted to find lots of vivid shapes in simple European roads, and the network trained in Europe wanted to "curve" the United States' roads when mapping them (cf. Figure 13). Most surprising indeed is the performance of the United States' trained algorithm on European data: being well trained on various landscapes (as each image corresponded to a different city), it adapted incredibly well to Europe's complex shapes, rendering complex and realistic results (cf. Figure 12).



(a) Satellite view (b) US network output (c) Expected output

(d) Satellite view (e) US network output (f) Expected output

Figure 12: Testing the US-trained network on European data



(a) Satellite view (b) EU network output (c) Expected output

(d) Satellite view (e) EU network output (f) Expected output

Figure 13: Testing the EU-trained network on US data

It would have also been interesting to train a network on a mixed dataset comprised of two different regions to see if it could distinguish and appropriately adapt its behaviour based on the location. We could not explore this approach in this project.

5.3.2 Finding an optimal data zoom configuration

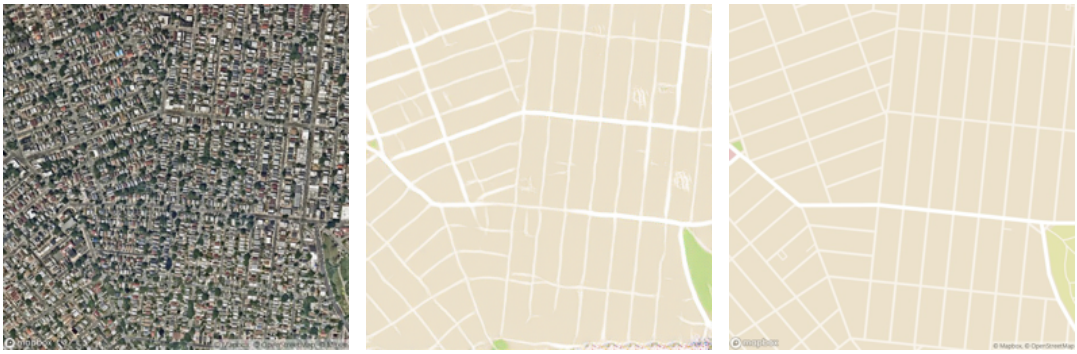
This section specifies the results obtained with the section **3.3 Zoom comparison datasets**. One aim of this project was to define an optimal image zoom configuration in order to obtain the most performing network possible. In order to do so, we created datasets of the same maps with different zoom levels. We notice a difference in the network's performance based on the zoom, visually and metrically.

First of all, it seems that a network with a smaller zoom can get a more comprehensive view of the environment, and therefore can straightforwardly capture the location's landscape and geography, in particular, colours. We directly link this effect to the dependence of the network on the datasets, as giving a broader image of a region as training data ensures the network captures all types of land.

Furthermore, in shape and roads detection itself, the network seems to have an optimal value dependant on the scale of the city itself. Please note this effect is much more apparent on visible results, and metrics may mislead us to believe otherwise. For instance, the optimal zoom value for the city of New-York is 16, whereas Beijing performs much better on a zoom value of 13. This result is particularly interesting, as it will significantly influence our choices when designing future datasets for this kind of network.

(a) Zoom 13 : Satellite view (b) Network output (c) Expected value

(d) Zoom 14 : Satellite view (e) Network output (f) Expected value

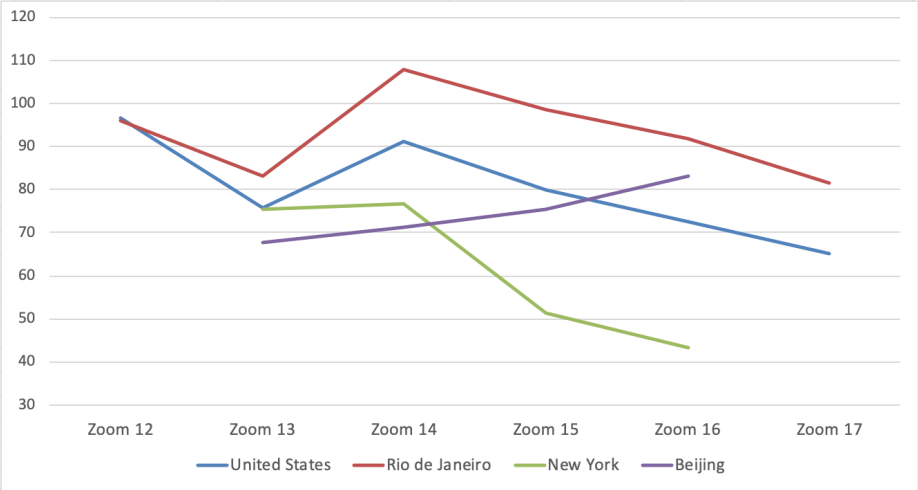


(g) Zoom 15 : Satellite view (h) Network output (i) Expected value

(j) Zoom 16 : Satellite view (k) Network output (l) Expected value

Figure 14: Comparison of New York's network performance in function of the zoom

The following graph represents the testing sets' mean squared error (cf. Figure 15). Primarily, as expected, there is a negative correlation between the MSE and the pixel-per-pixel comparator (see figure...). Furthermore, Beijing's results suggest the hypothesis posed in the previous section that the network seems to have an optimal value dependant on the scale of the city itself since the Beijing-trained Pix2Pix network functions better on a broader scale.



(a) Mean Squared Error in function of the datasets' zoom

	Zoom 12	Zoom 13	Zoom 14	Zoom 15	Zoom 16	Zoom 17
United States	96,694	75,833	91,094	79,992	72,412	65,090
Rio de Janeiro	95,866	83,192	107,821	98,463	91,962	81,655
New York		75,517	76,559	51,461	43,278	
Beijing		67,644	71,125	75,352	83,010	

(b) Mean Squared Error data

Figure 15: Comparison of the MSE of different datasets in function of their zoom value

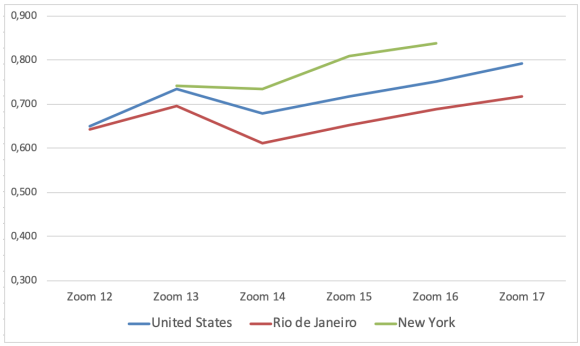
The two following graphs represent the proportion of pixels similar in the image, computed using the pixel-per-pixel comparator (cf. Figure 16). The only difference between their computations is that the first graph used a simpler colour palette (four colours), thus putting less attention to the network's performance regarding shades, whereas the second one used a more sophisticated colour palette (six colours), distinguishing certain shades of beige and green in part.

We can explain the difference between these results as follows:

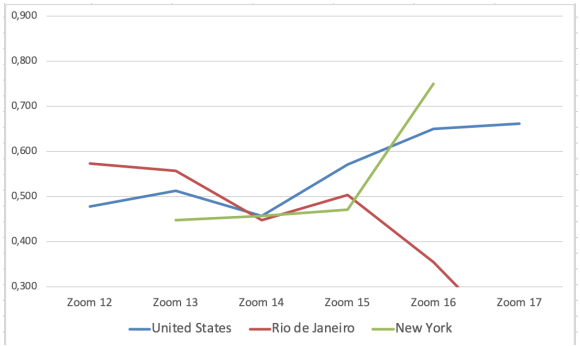
- When we zoom into the map data, the network is exposed to fewer colours during training, making it harder for it to distinguish them. Thus, since Rio de Janeiro's test set will also be composed of forest-like landscapes, it will not be able to distinguish those at test time.

- We can also explain this outcome by the fact that Mapbox fewer documents Rio de Janeiro and its surroundings, and therefore green areas and buildings are scarce in the training data, rendering the network unable to recognise them.

These results, considered individually, also expose some minimum performance of almost every network (at zoom 14). Furthermore, throughout the project, the zoom 15/16 values notably produced excellent results. We could thus, through more training and testing, define the optimal zoom value in order to most efficiently train Pix2Pix for this map creation purpose.



(a) Proportion of equal pixels on a 4 colour palette



(b) Proportion of equal pixels on a 6 colour palette

	Zoom 12	Zoom 13	Zoom 14	Zoom 15	Zoom 16	Zoom 17
United States	0,650	0,735	0,678	0,718	0,752	0,793
Rio de Janeiro	0,642	0,696	0,612	0,653	0,688	0,717
New York		0,742	0,735	0,810	0,838	

(c) 4 colour palette data

	Zoom 12	Zoom 13	Zoom 14	Zoom 15	Zoom 16	Zoom 17
United States	0,478	0,512	0,456	0,570	0,649	0,661
Rio de Janeiro	0,573	0,557	0,447	0,504	0,353	0,158
New York		0,447	0,456	0,472	0,750	

(d) 6 colour palette data

Figure 16: Pixel-per-pixel comparison of different datasets in function of their zoom value

6 Conclusion

In this project, we began by understanding Pix2Pix's versatility and its various applications by the scientific community. Our initial outline was to reproduce and enhance the experiments conducted in the original Pix2Pix paper regarding the conversion of aerial photography to map views. Having brought attention to Pix2Pix's inner functioning, we discovered the network did not need particular thought on its parameters in order to produce complete and comprehensive results. Therefore, we chose to focus mainly on the datasets we provided the network and their parameters such as colouration and zoom levels, leading us to try to determine the new possibilities offered by Pix2Pix regarding aerial photography vectorisation.

However, it is crucial to bring explicit attention to the limits of this method to generate maps: in order to obtain usable outputs, one must draw particular attention to the cleanliness of the datasets used in order to train this network. The datasets must reflect the entirety of the desired output, and few outliers are sufficient to exhaust Pix2Pix's capacity.

By using the data collected throughout this project, one can evolve its strategy when working on a Generative Adversarial Network solution to this particular problem. Alongside some data post-processing, one should thus be able to define realistic and accurate maps, fitting their needs in terms of style and precision.

This project only focused on particular aspects of Pix2Pix and its possibilities. By further researching its behaviour, creating datasets composed only of certain map features and using the network to extract them separately, for instance, one could then reassemble a complete depiction of maps from multiple layers. There are varied paths to explore through this project, for instance, training with precise buildings data. However, due to lack of proper data and time, we could not go further than simple training for this direction (cf. Figure 18). Even though we did not tackle it in this project, Pix2Pix can operate both ways; thus, using the same datasets as we have created in this project, we could recreate realistic aerial views of cities from maps. This particular instance proved to achieve even better results than going from cities to maps in the original Pix2Pix paper. To conclusively summarise this paper, the Pix2Pix network still has much to offer, yet its future potential is dependent on whether its users bring meticulous attention to their data sources. Moreover, we may not yet have thought about the prospects of this technology.

References

[1] P.Isola, J.Zhu, T.Zhou, A.Efros *Image-to-Image Translation with Conditional Adversarial Networks* Published Nov 2016, Revised Nov 2018, Berkeley AI Research (BAIR) Laboratory, UC Berkeley
`https://phillipi.github.io/pix2pix/`
`https://arxiv.org/pdf/1611.07004.pdf`

[2] IEEE: *Image-to-Images Translation for Multi-Task Organ Segmentation and Bone Suppression in Chest X-Ray Radiography*

[3] *Google geographical data upload program.*
`https://support.google.com/mapcontentpartners/answer/9359574`

[4] *Mapbox Static API & Mapbox Studio*
`https://github.com/mapbox/mapbox-sdk-py`
`https://docs.mapbox.com/api/legacy/static-classic/`
`https://studio.mapbox.com`

[5] R.Zhang, P.Isola, A.A.Efros. ECCV: *Colorful image colorization* 2016
`https://arxiv.org/pdf/1603.08511.pdf`

[6] Python Imaging Library documentation
`https://pillow.readthedocs.io/en/stable/`

[7] NumPy module documentation
`https://numpy.org/doc/`

[8] XlsxWriter Python module documentation
`https://xlsxwriter.readthedocs.io/`

A Appendix

https://github.com/Nielsesca/aerial-photography-vectorization

No.	Location	Zoom	MSE	SSIM	MSE (greyscale)	SSIM (greyscale)	Proportion of equal pixels
3	United States	15	105,850	0,737	141,645	0,667	0,549
4	United States (Contrast)	15	116,249	0,529	117,713	0,546	0,623
5	United States	17	74,615	0,787	117,549	0,697	0,670
6	United States	16	83,926	0,761	118,731	0,686	0,636
7	Europe	15	56,380	0,843	78,472	0,772	0,737
8	Paris	16	120,380	0,729	113,519	0,701	0,586
9	New York	16	64,301	0,852	65,607	0,831	0,750
10	Jakarta	16	162,665	0,517	176,155	0,488	0,392
11	Rio de Janeiro	16	107,496	0,754	124,647	0,693	0,551
12	Beijing	16	114,121	0,734	161,873	0,630	0,495
13	New York	13	138,110	0,664	127,372	0,687	0,621
14	New York	14	116,752	0,695	128,866	0,678	0,595
15	New York	15	71,566	0,823	76,237	0,794	0,729
16	Beijing	13	103,884	0,710	129,709	0,651	0,577
17	Beijing	14	103,801	0,724	153,331	0,645	0,532
18	Beijing	15	101,281	0,757	141,965	0,690	0,556
19	Rio de Janeiro	12	139,777	0,569	116,899	0,602	0,573
20	Rio de Janeiro	13	143,479	0,546	132,516	0,571	0,557
21	Rio de Janeiro	14	143,030	0,602	165,965	0,574	0,447
22	Rio de Janeiro	15	116,397	0,702	147,169	0,652	0,504
23	Rio de Janeiro	16	165,557	0,705	140,882	0,612	0,353
24	Rio de Janeiro	17	147,436	0,775	232,888	0,647	0,158
25	United States	12	152,485	0,523	153,656	0,520	0,478
26	United States	13	129,769	0,616	154,342	0,576	0,512
27	United States	14	138,236	0,611	175,183	0,578	0,456
28	United States	15	104,826	0,730	131,465	0,684	0,570
29	United States	16	90,931	0,755	106,416	0,713	0,649
30	United States	17	77,001	0,793	111,263	0,703	0,661
31	United States (w. buildings)	16	127,186	0,703	126,677	0,654	0,472

Figure 17: Complete data analysis results

(a) Satellite view (b) Network output (c) Expected output

Figure 18: Example output for set 31 (United States, Zoom 16 with buildings)