MASTER TRIED
ALAN LE MONTAGNER
SIMON PALMA
Advisor ANASTASE CHARANTONIS

PROJET LONG :

Deep learning adversarial model for oceanographic images semantic inpainting

Introduction

Oceans cover 70% of the earth's surface. Sea surface temperature (SST) provides crucial information on the global climate. SST is essential for weather prediction and atmospheric model simulations, as well as the study of the marine ecosystems.
SST helps in a wide range of applications, such as climate monitoring, forecasting, military operations, marine species tracking, maritime commerce, among many others. To measure the SST, we have to rely on a serie of devices ranging from satellites to marine telemetry.
In order to have a complete measurement of the SST, one sensor is not enough to count on. Instead, certain measurements of different devices are taken and treated in order to have a full reconstruction that accurately describes the natural phenomenon of the SST (in this specific case). There exist several projects that dedicate to these tasks, and such is the case of the GlobColour Project. The GlobColour Project merges and treats data from the NASA's L3 Ocean Colour products to achieve a fully-reconstructed SST measure.
Although we have relied on the feasibility of these reconstruction from different sensors for many years, the task becomes quite slow to implement. This measurement merging turns out to be a heavy task to carry out since it uses methods with complex techniques over massive quantities of data, making the process considerably exhaustive.
It is of our interest to be able to retrieve a fully-reconstructed accurate description of the data in a faster way.
We, thus, present an alternative to this already existing method aiming to perform a competitive reconstruction when compared to the already existing techniques without taking a huge amount of time and computational resources.
We rely on the capability of artificial intelligence techniques that could be useful to this goal. More specifically, the use of deep learning convolutional neural networks that must potentially succeed to inpaint the areas that initially were not possible to measure, such as clouds covering the areas.
Throughout the experience, it will be shown that we made use of an autoencoder in order to semantically retrieve the missing values of the images. Moreover, a discriminator was used, taking as inputs the ground-truth and the inpainted images to determine which of the two is the

real one. The implementation was made through an adversarial network in which the two models get into competition, allowing a reciprocal adjustment to ensure a more realist prediction.

State of the art

Studies have been carried out on the field where we place our work. Different techniques have been implemented over a variety of images that make us arise interest due to the promising results were held. Studies [3] and [5] are the most similar to the one we carried out. In these two it exist the idea of semantic inpainting which refers to the capacity to reconstruct the missing parts of the images by considering the surrounding areas. These studies are based on the reconstruction capacity of the autoencoders to retrieve the initial images after extracting a compact feature representation that is expected to describe the ground-truth image. The missing areas we aim to reconstruct should keep relevant information regarding the missing values since they are spatially very close. The network learns to give special attention on the existent areas close to the non-existent ones which will translate in a better approximation of the reality.

Figure 1. Context encoders reconstruction

There are other methods with different focus. Instead of generating a reconstruction model, an already existent one is used and constantly tuned until having the best model configuration. This clearly allows outperform the previously existing network since it was not optimized, and at the same time provide results that permit to successfully inpaint an specific sort of image. Such is the case of the study made in [6].

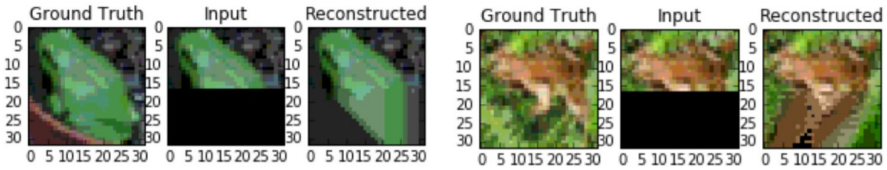


Figure 2. Existing optimized RNN and CNN reconstruction

Objective

Throughout the experience, it will be of our main interest to completely reconstruct thermal oceanographic images from an incomplete image, with as few differences with the original image as we can get. We will need to build an efficient model, with measured errors from the original images as small as possible, with a tolerable amount of time needed to compile.

Dataset

Initially, we had a set of measurements of 92 latitudinal and 107 longitudinal temperature points during a period of 86 days from the winter of 2008. The spatial temperature was measured each 0.083 longitudinal (and latitudinal) degrees equivalent to 9.26 km. Usually, the range of temperature values goes from 18 to 24 degrees Celsius.
The data comes from the GlobColour project and correspond to an oceanographic area located in the north-west of Africa, near Mauritania and Western Sahara.
The GlobColour project has as goal to provide continuous data set of merged Ocean Colors products (of the NASA). This merge of different sensors ensures continuity of data, improves spatial and temporal coverage as well as reduces noise. The measured zones will be represented by their relative latitude and longitude values on the images.

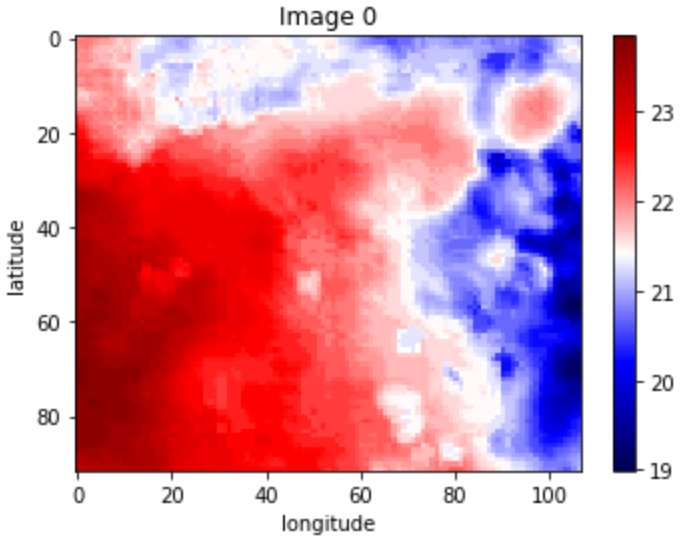


Figure 1. Image representation of the data set.

Pre-processing

To increase the number of samples of our dataset and, more specifically, to have size of data being a power of 2 to ease the generation of CNNs with pooling, is was decided to work with images of size 64x64. To this aim, every image was chopped into four new ones, each of these representing one of the four corners of the original image. Doing so, a total of 344 images were generated.
As we wanted to represent the incompletion of the satellital raw measurement of SST, we artificially created hole in the image that would represent the input of ou model to be reconstructed. These hole generations were made as follows:
From these images, three new sets of data were then created

- One with each image having an artificial square 10x10 hole inside
- The second with multiple square holes of the same size (6x6)
- The last one with multiple square holes of different size

The number of data was also tripled by creating holes in different zones of the same images. It guaranteed different input values to be predicted. Thus we mastered to have a total of 1032 images.
A normalization was also carried out since normalizing tends to make the training process better behaved by improving the numerical condition of the optimization problem and ensuring that various default values involved in initialization and termination are appropriate [1].

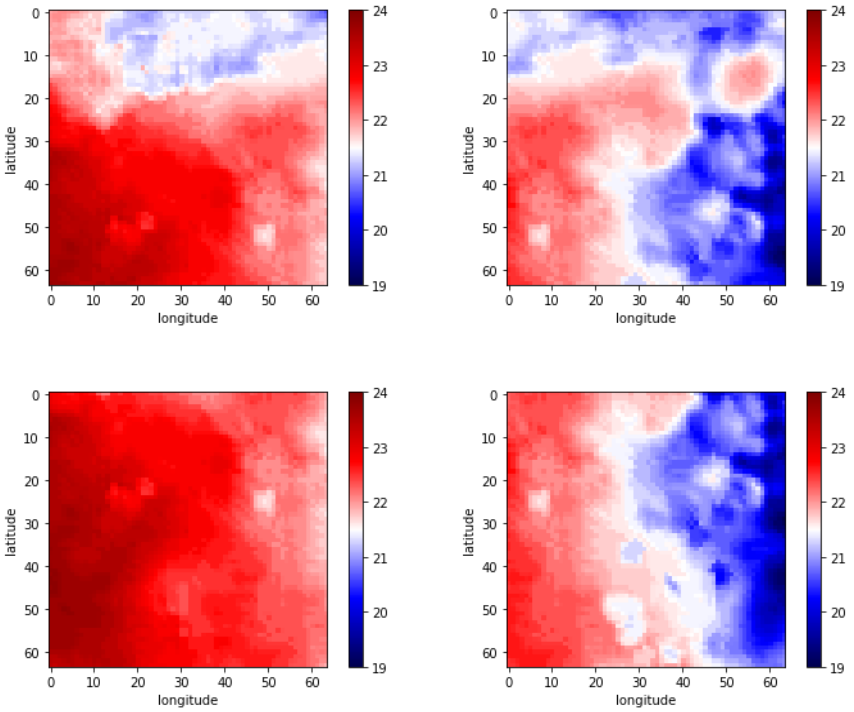


Figure 2. An image split into four sub-images

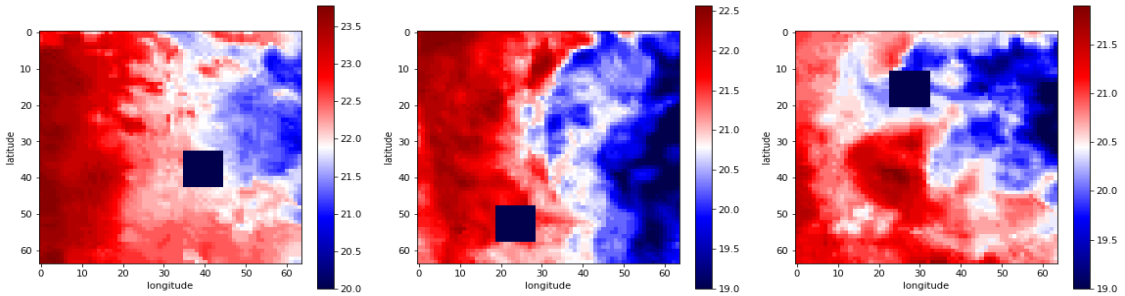


Figure 3. Oceanographic images with a 10x10 hole size

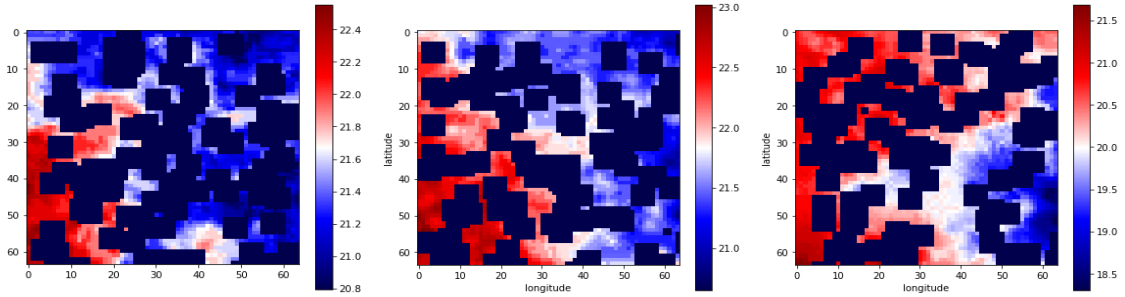


Figure 4. Oceanographic images with multiple holes of fixed 6x6 sizes

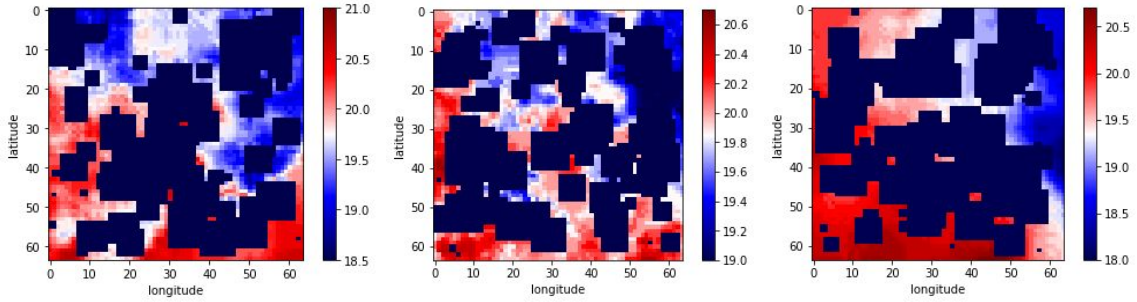


Figure 5. Oceanographic images with multiple holes of variable sizes

Autoencoder

The autoencoder is a variant of neural networks which takes an input image and tries to reconstruct it after passing through a bottleneck layer of lower dimension than the input [2]. It is possible to observe the autoencoder as a concatenation of an encoder and a decoder.

In the encoder, the first layer represents the input to the network and the last one is a bottleneck layer, as mentioned before, which has the aim of extracting a compact feature representation of the input.

On the decoder side, the first layer is the same as the last layer of the encoder and the last one is of the same size as the input of the encoder. Thus, the decoder aims to effectively reconstruct the original image from the compact feature extracted in the first part.

In this experience, after each deconvolution step we made concatenations with the encoder layer having the same shape, since we wanted to keep the small details as much as possible. Without it, the transfer of the details between layers could be lost while reconstructing the image.

Throughout our analysis the autoencoder was slightly modified, since we needed to solve a much more complex task than a simple reconstruction of a whole image. Instead, we need to fill in large missing areas of the image, where it cannot get hints from nearby pixels. This requires a much deeper semantic understanding of the scene, and the ability to synthesize high-level features over large spatial extents [3].

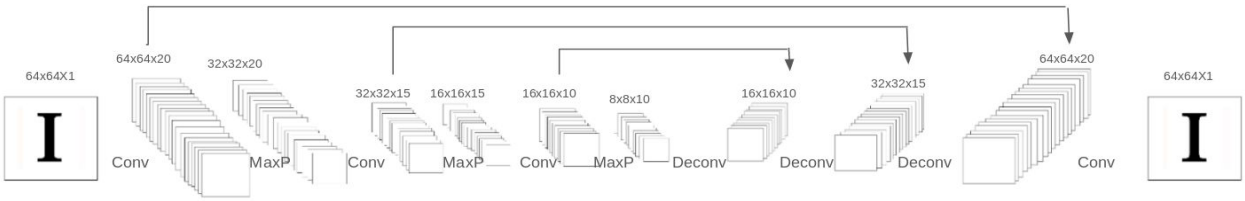


Figure 6. Autoencoder structure

When using a simple autoencoder on our images, we had modest results, which proved us the necessity of using a complete Generative Adversarial Network to improve our results.

Discriminator

In this method, the discriminator is in charge of determining the ground-truth image from two image inputs. To do so, a convolutional neural network was implemented, receiving as input two images, one being the real image and image predicted for the autoencoder. The output layer is composed by a single neuron that determines whether the real image is the first (output equal 0) or the second one (output equal 1).
As expected, when we used our discriminator on our poorly-predicted images obtained before, we had a 100% performance on determining the true or fake images.

Generative Adversarial Network

Generative adversarial networks (GANs) are deep neural network architectures comprised of two nets, competing one against the other [4]. It is of our interest to understand how a generative and a discriminant algorithm work.
In a general scope, one neural network called the *generator* generates new data instances, while the other, the *discriminator*, evaluates them for authenticity.; i.e. the discriminator decides whether each instance of data belongs to the actual training dataset or not. In our experience, the generator was replaced by the autoencoder since we are reconstructing an image from another (incomplete) image, so the input and the output must keep the same shape.

The discriminator network is a standard convolutional network that can categorize the images fed to it, a binomial classifier labeling one of the images as the real one.

The autoencoder is a composition of an encoder and a generator (direct and inverse convolutional network respectively): The encoder model takes an image and downsamples it to produce features, while the generator takes a vector of random noise and upsamples it to an image. The first throws away data through downsampling techniques like maxpooling, and the second generates new data.

Both nets are trying to optimize a different and opposing objective function, or loss function. As the discriminator changes its behavior, so does the generator, and vice versa [4]. Their losses push against each other, and makes them improve their performances with the help of the other one's results

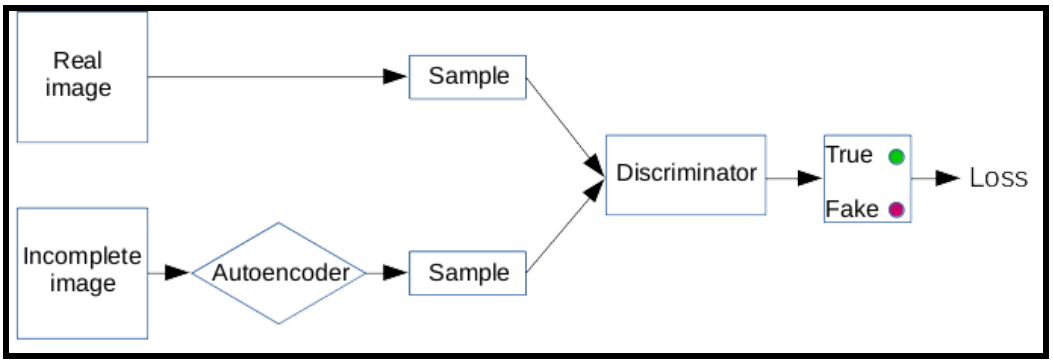


Figure 7. Adversarial network

Loss function and mask

In the aforementioned paragraphs we described the idea of semantic learning which refers to the capacity to learn a given feature from the information of the context (verbal context, image context, etc). Due to previous work in the same type of tasks we are treating, it was noticed that it is important to strongly consider the features that are more semantically related to the results we desire to obtain.
Being a human trying to fulfill this task of a good reconstruction of the missing parts, we must consider the rest of the image, but even more what is around the missing values. This will allow us to have a better intuition of what there really is.
In our case, in order for the model to understand that the surrounding pixels of the missing parts are key to achieve a good reconstruction, a custom loss function was implemented.
Our loss function then uses a mask which considers the surrounding pixels' errors with an even greater weight, forcing the model to reconstruct these pixels and to predict the hole pixels with values that fits their neighbours. With this, it was possible to obtain more soften transitions between the known and unknown locations of the image. This was inspired by the context encoders method from UC Berkeley's Deepak Pathak team [3].

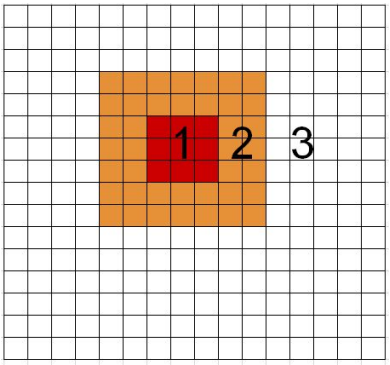


Figure 8. The different areas of our mask applied to an image

In our case, we apply a mask matrix as follows :

- A factor 0.1 to the missing pixels (area 1)
- A factor 1 to the neighbouring pixels, of a parametrable size (area 2)
- A factor 0 to the other pixels (area 3)

We then calculate the loss function with the formula :

$$Loss = \ < RMSE, \ Mask >$$

The factors being hyperparameters, as well as the size of neighbourhood, they can be modified to change the results and vary the context or the precision of our inpainted pixels.

Training

In the encoder part of the autoencoder, there were three convolutional layers, for each of the layers a maxpooling operation was carried out, having 20, 15 and 10 layers in the tensors of the first hidden to the fourth hidden layer respectively. The decoder had an inverse encoder architecture coming from a tensor of 10 layers of size 8x8 to the original 64x64x1 image size. After each deconvolution process, a concatenation was done with the equivalent encoder layer to maintain the high quality details lost in the reconstruction.

The discriminator was built with four convolutional layers quite similar to the layers of the first part of the autoencoder but with an input size allowing two image to input the model, a flatten of the last ConvNet layer and three more hidden layers before the last layer. This last layer had one neuron that would provide 0 or 1 depending of the index of the image that it considered the real one as it was mentioned before.

For the autoencoder we had 14910 parameters and for the discriminator 28208. In total we had 43118 parameters to train.

The network was compiled using the adam (adaptive model estimation) as optimizer of both the binary cross entropy loss function for the discriminant model and the custom semantic-based loss function for the autoencoder model.

From a total of 1032 images, 660 were used as training set, 165 as validation and 207 as test using a proportion of approximately 65%, 15% and 20% respectively.

Early stopping was used in order to determine the optimal training when, in the validation set, the lowest loss value was not reached or surpassed after the 5 subsequent iterations. The number of epochs was set to 100, meaning that if after 100 iterations the validation continued decreasing, the algorithm stops. We used 10 as the batch size value since it provided better results when compared to higher values (25, 35 and 50) and almost the same results than with lower values (1 and 2) and in a shorter time.

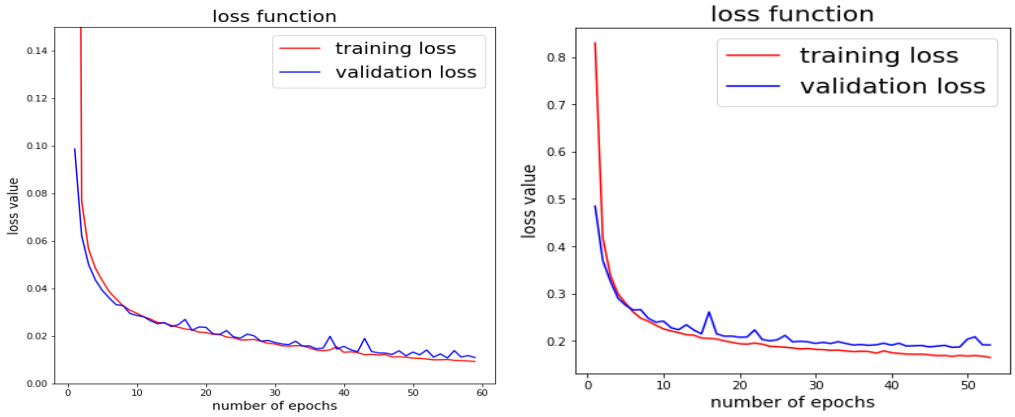


Figure 9. Loss function vs iterations when using one (left) and multiple (right) holes

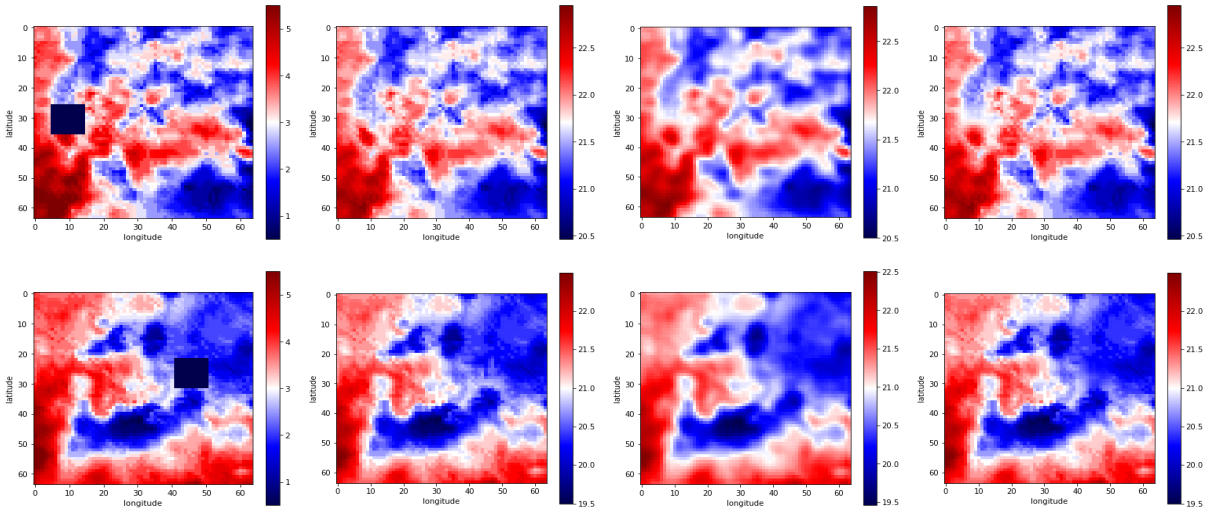
Results

A - Reconstruction

With our different sizes and number of holes, and different type of sizes, we got good results in the image inpainting we aimed. As shown on the figures below, the difference between our reconstruction and the original image is really small, and can sometimes be unseen by the naked eye.

To get a better reconstruction, we also chose to keep only the reconstructed pixels from the missing parts, and to keep the original pixels on the areas containing temperature information. Thus, we got an even better result and the predicted pixels appeared to be very close to the targeted output.

1 - Single hole



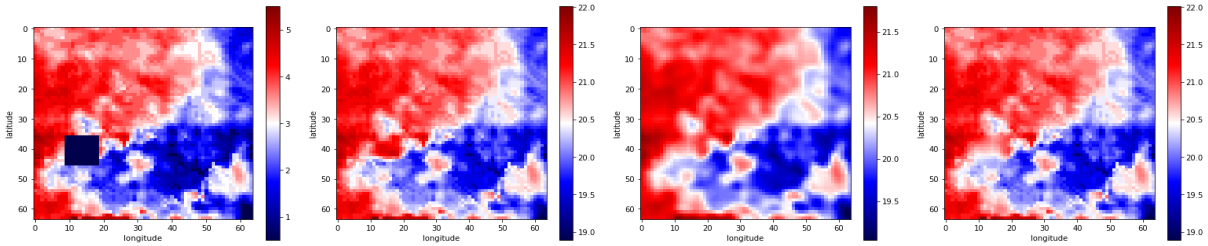


Figure 10. Left to right: Input image, real image, predicted image, prediction only on missing zones

2 - Multiple holes of the same size

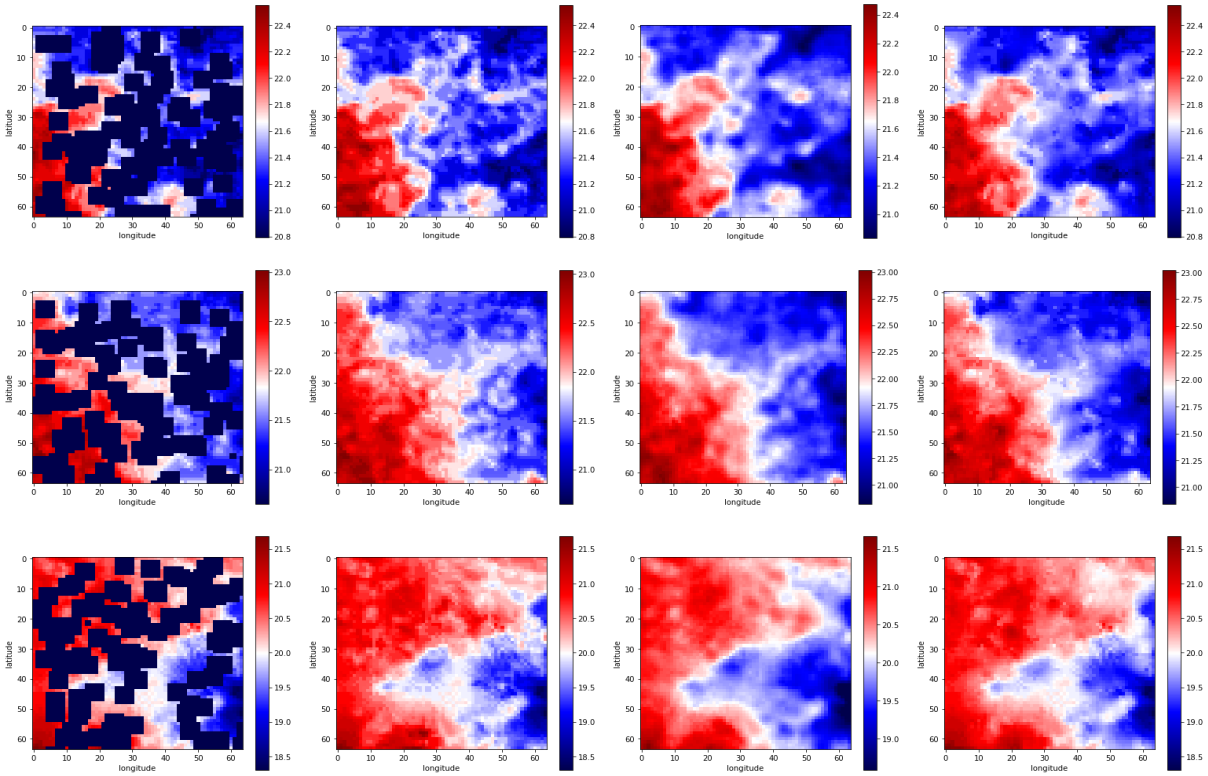
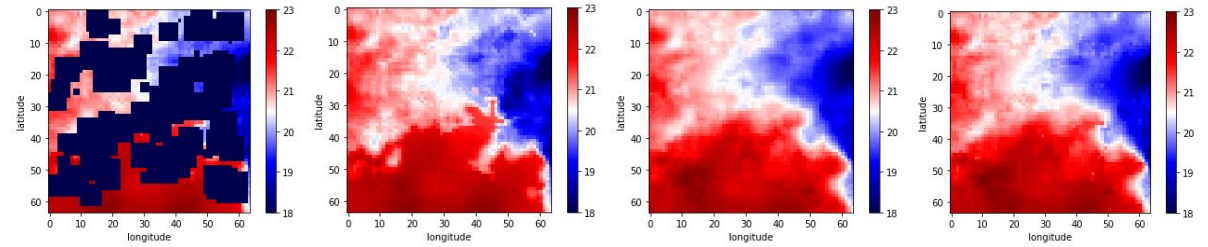


Figure 11. Left to right: Input image, real image, predicted image, prediction only on missing zones

3 - Multiple holes of different sizes



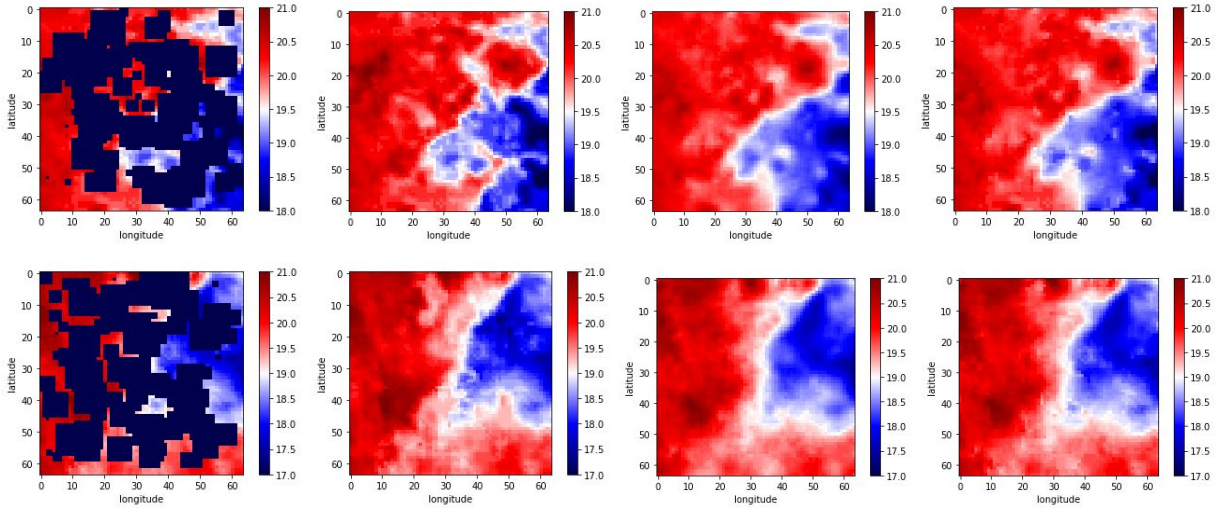


Figure 12. Left to right: Input image, real image, predicted image, prediction only on missing zones

For a single hole it is observed that the reconstruction is very accurate knowing that there is only a small part to be predicted. In this case, ou model is not expected to struggle in the approximation.

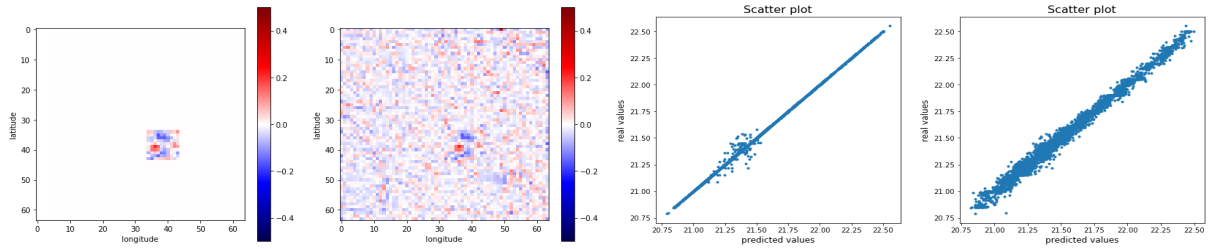
When considering images with multiple holes of the same and different sizes, the model works surprisingly well, even though a big part of the image is covered by holes the model is capable of generating a accurate reconstruction even in zones with traces not easy to determine.

B - Errors

To have a more visual insight of the reconstruction of our model it is of our interest to evaluate the differences between the real and the predicted image. To do so, a subtraction of the real minus predicted image was computed to see the zones where the difference between these two was higher.

A scatter plot was plotted to observe the correlation and the confidence interval between the real and predicted temperature values allowing us to assess even more on the accuracy of the results obtained.

1 - Single holes



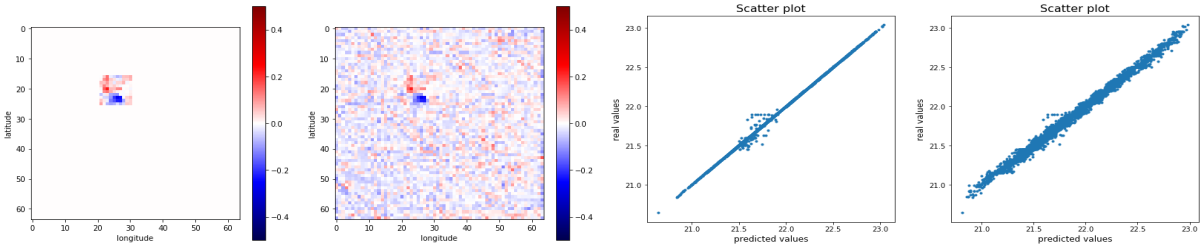


Figure 13. Simple subtraction and scatter plots between predicted and real images.

It is observed that the zone where the input values were inexistent the error is slightly higher as it was expected. Nevertheless it does not diverge too much from the already-known values.
The error does not have any clear relation with the temperature value itself. In figure x, our first example does have a slightly higher dispersion for low temperature values than for high. Nevertheless on the second image this behavior is not kept, not allowing us to accept this true hypothesis beforehand.

2 - Multiple holes of the same size

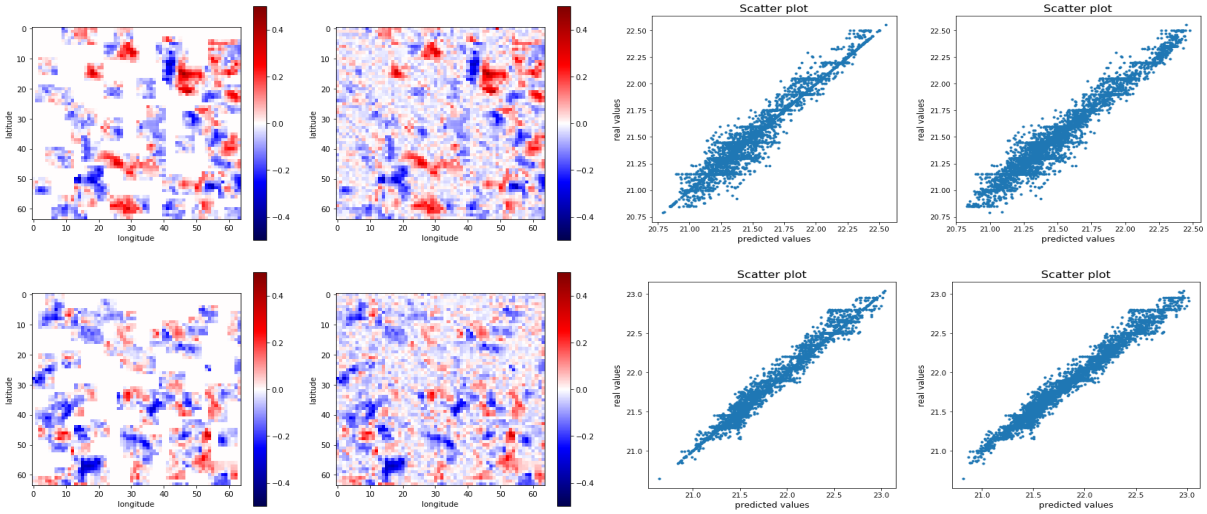


Figure 14. Subtraction and scatter plots between predicted and real images.

We observe different sort of errors on the whole scale of temperature this time. It is observed that the differences between the known and the unknown parts are quite higher than in the single-hole experience. Once more, on the first image of figure x it can be observed a higher variance of the values than for high temperature values. On the second image the variance is stable almost throughout the whole interval (except a small zone at the beginning). Thus since the error appears to be variating likewise regardless of the temperature when we observe different image results, the differences do not seem to be correlated with the temperatures.

3 - Multiple holes of different sizes

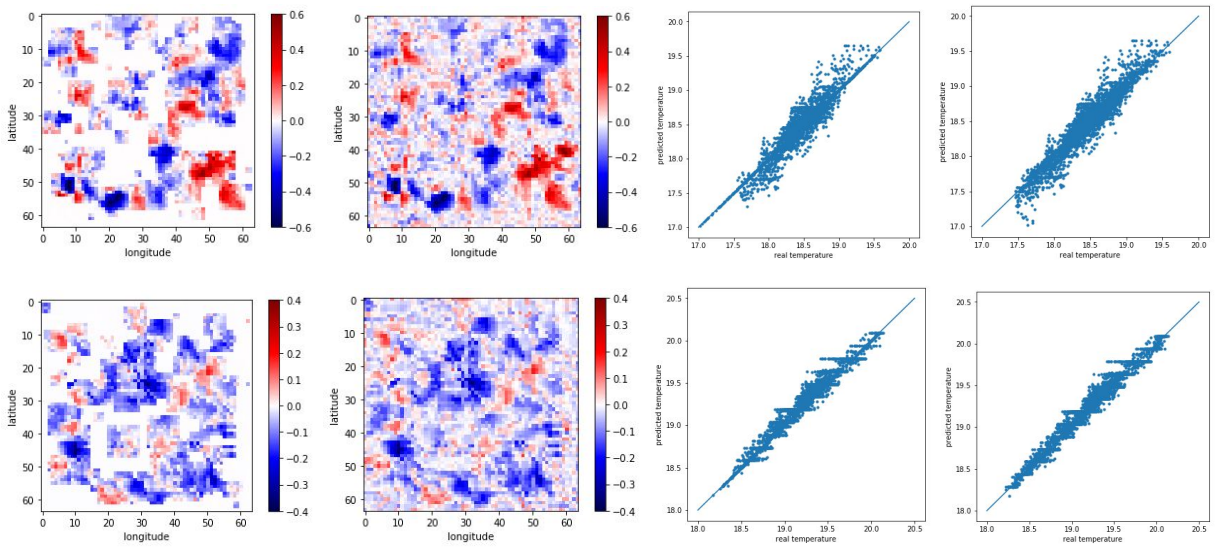


Figure 15. Subtraction and scatter plots between predicted and real images.

In the third experience the results obtained were quite similar to those of the second part since the nature of the problem was, in general, maintained. However, it proved that the variating size of the holes do not influence the shape of errors too much.

C - Quantitative results

With these different datasets of simulated clouds, we had different values of performances and computing time. As it was mentioned in the training part, a maximum of 100 epochs and a batch size comprised between 10 (one hole or multiple holes of same size) and 25 (multiple holes) were some of the configurations used during training.
The different models were trained on our personal machines, with an computing average time of 30 to 45 seconds per epoch, leading to about one hour of processing for the whole model.

Following are summed up the root mean square errors (RMSE) values, considering the whole predicted images and just the inpainted pixels, with the loss values obtained in both train and validation sets for our autoencoder+discriminator model.

Dataset	RMSE (prediction)	RMSE (inpainting)	Loss (train)	Loss (Validation)
One hole	0.05338	0.0376	0.0092	0.0109
Multiple holes	0.0876	0.1223	0.1575	0.1764
Multiple holes & different sizes	0.1351	0.1417	0.1398	0.1472

Conclusion

The results obtained were visually quite impressive since there were very similar to the real ones and quantitatively close to the real images, with low error values (RMSE from 0.0376 to 0.1417 which is considerably low).
The model showed to provide quality prediction even when having a small size of the images and a quantity of them. Some very precise details were rebuilt by the model, which proves its efficiency and adaptability to very small data. With a sufficient amount of pixels, it is even able to rebuild images with a reasonable performance, even with up to 75% missing data.
To compare our results with the other methods, it should be interesting to try our model on the datasets used by those experiences, ensuring even condition to assess on the relative performances of each one of the models.
It could also be interesting to use our model on real data, with missing values coming from actual sensor limitations and not only from our simulations.
An improvement and optimization of this model can still be done. It could be useful to try several configurations of the hyperparameters lying on the help of grid search or random search techniques.
The computing time could have also been reduced by an optimization of our loss function, which is visible in our github listed in the annex, and could also be reduced by the utilization of better machines for training.
However, up to this level the results obtained were meaningful and the model implemented proved to be useful in order to rebuild missing data.

Bibliography and sources

[1] Neural nets : Should I normalize/standardize/rescale the data ?
http://www.faqs.org/faqs/ai-faq/neural-nets/part2/section-16.html

[2] Stanford deep learning tutorial
http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/

[3] Context Encoders: Feature Learning by Inpainting
Deepak Pathak, Philip Krähenbühl, Jeff Donahue, Trevor Darrell & Alexei A. Efros
University of California, Berkeley, United States of America

[4] Generative Adversarial Nets
Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio
Département d'informatique et de recherche opérationnelle, University of Montreal

[5] Semantic image inpainting with deep generative models
Raymond A. Yeh, Chen Chen, Teck Yian Lim, Alexander G. Schwing, Mark Hasegawa-Johnson & Minh N. Do

University of Illinois, Urbana-Champaign, United States of America

[6] DeepPaint: a tool for image inpainting
Kushagr Gupta, Suleman Kazi & Terry Kong
Stanford University, United States of America

[7] Image inpainting for high-resolution textures using CNN texture synthesis
Pascal Laube, Michael Grunwald, Matthias O. Franz & Georg Umlauf
Institute for Optical Systems, University of Applied Sciences Constance, Germany

[8] Keras Adversarial Models
Ben Striner's Github
https://github.com/bstriner/keras-adversarial

[9] Generative Adversarial Networks implementation with Keras
Robin Ricard's website
http://www.rricard.me/machine/learning/generative/adversarial/networks/keras/tensorflow/2017/04/05/gans-part2.html

[10] Globcolour project's datasets
http://www.globcolour.info/

Github

The python code, as well as the data used to run this model, may be found on our Github, which is accessible by the following link: https://github.com/Alanlemontagner/ProjetLong