Sat2Dron 3: Spectral Superresolution TFG report

Daniel Rojas Pérez
Tutor: Dr. Felipe Lumbreras

12th October 2020

1 Preliminary information

Multispectral (MS) remote sensing sensors are proven to acquire useful information in order to perform object detection, classification and other applications. These sensors acquire information within limited broad spectral bands, usually between 3 and 10 number of bands. This low number of bands become a drawback when performing more detailed analysis of earth's surface. Hyperspectral (HS) sensors are instruments designed to have a considerably greater number of narrow spectral bands compared to MS sensors. Hence, this type of sensor becomes the best option for distinguishing spectral resembling material of the earth's surface. However, availability of immediate HS data is currently lacking.

Most of the work involving superresolution has been in the spatial and temporal modalities in which Deep Learning has achieved state-of-the-art results [1, 2]. Nevertheless, little study has been involved with increasing the spectral information of an image with Deep Learning [3], most of it is using other techniques [4].

2 Objectives

This TFG proposes transforming multispectral data to hyperspectral data using Deep Learning techniques in order to increase the spectral information from the MS data by more than 10 times —from less than 10 number of bands to over a 100 bands—.

2.1 Main objectives

1. Moderate spectral superresolution: increase the number of spectral bands in multispectral data by a factor of 2.
2. Extreme spectral superresolution: increase the number of spectral bands in multispectral data by a factor of 10.

3. Achieve domain adaptation and transfer the learning to other satellites using zero-shot learning techniques.
4. Generate architectures that are innovative and competitive in a research area that does not have a large quantity of literature or resources.
5. Evaluate our model comparing the results obtained with current architectures.

2.2 Side objectives

1. Propose a novel architecture that achieves state-of-the-art performance.
2. Combine all the modalities of superresolution (spatial, temporal and spectral) into one architecture [5].

3 Learning outcomes

1. Master deep learning techniques applied to computer vision.
2. Master the manipulation and processing of satellite remote sensing imagery.
3. Learn the state-of-the-art of superresolution and deep learning in different modalities and be able to use the knowledge to propose novel architectures.
4. Learn by performing a large quantity of computational experiments.
5. Be introduced to the research field.
6. Describe the overall conclusions from all the experiments performed into an article document format.
7. Decompose a big challenge into different tasks and milestones.
8. Be challenged with a project in which there is little literature.

4 Methodology

The management of a project is of a great degree of importance to meet the final objectives and deliver them on time. In order to choose the best alternative available, these next points have been taken into consideration:

- Individual project: the work of the project is realized by a single person. Therefore, there is no need of daily communication and coordination with colleagues, neither of a strict schedule.

- Weekly meetings with supervisor: weekly meetings will be held with my supervisor to show the progress and consult any doubts. This gives me the aim to set weekly minor goals and work from the start of the project.

- Poor literature in the area: the research topic has not been deeply studied by other researchers using the techniques that are being planned to be utilized. Thus, there is a level of uncertainty on the overall project, which makes the timelines not as clear. However, there is some literature on related lines of research, such as performing data fusion of MS and HS [6] or single hyperspectral image super-resolution [7] (which will be analyzed and studied to get a solid understanding of the handling of similar data as ours).

- Novelty: although computer vision and deep learning have already been introduced to me, remote sensing and the satellite scene is a novelty, as well as other tools that will be used. Therefore, the mastering of these tools will be required in order to achieve the goals and it could affect the timelines.

The time management methodology chosen is Kanban[1] due to the fact that it is a methodology characterized for improving the speed and quality of work. It will cause a desire for finishing the current work-in-progress set, as multi-tasking is limited. Moreover, it enables a visible display of the project and work in progress which will be of a great use when communicating with the supervisor. Finally, weekly minor goals will be set in order to increase the motivation and performance.

5 Work plan

1. Documentation (Weeks: 1st-2nd): study the literature, actual and old, of superresolution and deep learning. We will emphasize on the work related to our field and desired satellites, but we will also explore other alternatives. Moreover, the study of the different characteristics of satellites and their instruments will also be realized. This phase is emphasized in the initial part of the project but will remain active throughout the whole project due to its importance and need.

2. Introduction of new tools (Weeks: 2nd-5th): learning to use new tools such as GDAL [9], QGIS [10], Earth Engine [11], among others, and the downloading and processing of satellite imagery. This phase will be the base to get my further work done.

3. Reproduction of state-of-the-art (Weeks: 3rd-7th): in order to achieve a good understanding of the matter and increase the familiarity with the

[1] Trello [8] is the desktop web application chosen for managing the project with Kanban methodology.

new tools and methods, a reproduction of the state-of-the-art scene [3] (no code available) will be produced. This phase overlaps with the previous phase due to the fact that this phase is a way of improving the skills needed and achieve a solid learning on the matter.

4. Proposing a novel architecture (Weeks: 7th-13th): more study and computational experiments will be done in order to achieve a novel architecture that stands out from others. This phase is the most exploratory one, it will be based on research among different areas, learning what can be done in order to improve a solution and thinking outside the box. Getting a very broad knowledge, design architectures based on the characteristics that will get the most out of it and performing trial-and-error will be the objectives of this phase. The timeline for this phase starts from when a good practical understanding is achieved on the problem and it ends soon enough to be able to propose new approaches, such as the zero-shot learning.

5. Zero-shot learning approach (Weeks: 13th-17th): the last main goal of the project is that the learning of the network can be applied to other satellites and instruments different than the one used for training. Using very little or none set of data, we hope to achieve satisfactory results. This phase remains as the last technical phase, the project could diverge to some different specific goal if we find a viable solution to an unsolved problem.

6. Elaborating a final report (Weeks: 18th-22nd): a final article report will be written up explaining our approach and solution to the problem, specifying the processing of data, the design of the architecture, decisions taken, etc.

6 Work progress

6.1 Update I

6.1.1 Summary

The work done till the date of this first update report of this project has been in accordance with the work plan established in a previous report. The deadlines and outcomes of the established plan has been achieved with great results and will be presented in the next sections.
This first update report contains the learning methods so that a deep understanding of the matter is achieved. Moreover, the detailed process done will be described in order to achieve the objective set as well as the methods used and decisions taken when the original paper did not provide enough detailed descriptions of their experiments.

6.1.2 Learning resources and tools

Prior to developing the experiments of the mentioned paper, it was desirable to get the skills needed for it. The major knowledge gap involving this project development was about working with geospatial data and remote sensing.
A large quantity of resources have been read and studied involving these topics although the most influential ones are based on the libraries, frameworks and servers used. GDAL, Google Earth Engine, QGIS and USGS EROS Archive are some tools that were utilized and therefore learnt to acquire experience using them.

GDAL is a software library for reading and writing raster and vector geospatial data formats. This library is mostly used in my experiments involving reading and preprocessing tasks of the satellite imagery. A general learning of the usage and capabilities of this library was obtained using its official documentation, resources and some online tutorials. Regarding some more specific tasks with the library, Stack Overflow has been a great source of information.

Google Earth Engine was the first choice as the server used for obtaining the satellite imagery and therefore some guides about obtaining and manipulating Earth Engine data have been visited. However, I finally decided not to use Earth Engine as our server used as the original paper obtained its data from a different source and I wanted to reproduce and obtain the exact same data.

QGIS is an open-source cross-platform desktop geographic information system (GIS) application that was helpful for visualizing our obtained data and making sure that our imagery from different satellites are co-registered.

USGS EROS Archive is the chosen server for obtaining the desired satellite imagery. The process to obtain the images is very trivial as it is served with a GUI.

PyTorch is the software library chosen regarding deep learning experiments due to the exponential increase of its usage in research and academia. The usage of such has not become a challenge since I already have some experience with it.

6.1.3 Activities and progress

The major work defined by the plan was to develop a reproduction of the state-of-the-art of the specific problem of increasing the spectral bands of a multi-spectral image (6 bands) to a quasi-hyperspectral image (170 bands). In order to develop this, the implementation is divided in 3 parts: obtaining the satellite imagery of the area of interest where the experiments will be executed, the preprocessing of this imagery and building the deep learning model designed to increase the spectral bands.

6.1.3.1 Obtaining satellite scenes

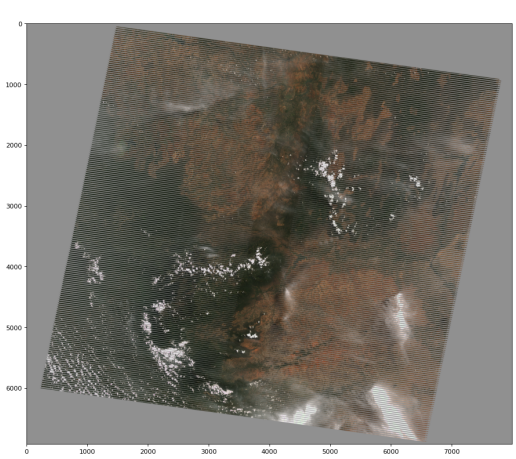
The desired scenes have been located and downloaded through a software that I am currently developing which gives easy access to different servers through APIs and provides methods for obtaining and processing scenes from the main earth observing satellites. The area of interest selected has been the Karnataka state of India according to the paper [3]. The satellite for the MS data is the Landsat 7 ETM+ [12] and the EO-1 Hyperion [13] for the HS data. The obtained scenes consist of an image over the area of interest for each desired satellite in the same date.

6.1.3.2 Preprocessing of images

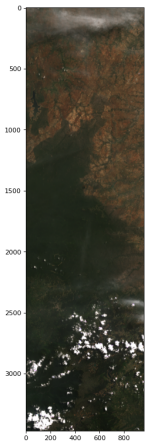
Preprocessing of the scenes is needed so that they meet the requirements for being well introduced to the deep learning model built later. Each satellite image has different preprocessing needs:

- Landsat 7 preprocessing: the first technique applied consists of a 3x3 mean filter gap filling of the scene due to an error in the sensor of the satellite [14] that corrupts the image and does not provide pixel information in continual thin diagonal lines. Then, I clip the raster to fit the rectangular shape of the Hyperion scene, this makes both scene to be co-registered and they now have the same size and cover the same area. Next, I extract 5x5 window size patches for every common pixel with the Hyperion data. Finally, the data gets normalized in values in the range of [0-1] and divided randomly in 2 different subsamples: 50% of pixels for the training set and 50% for the test set. Results of the clipping and gap filling are shown in Figure 1.

- Hyperion preprocessing: Hyperion data will be normalized and split randomly in half for generating the training and test sets (maintaining correspondence with Landsat 7 pixels).



(a) Before preprocessing



(b) After preprocessing

Figure 1: Preprocessing on Landsat 7 ETM+

6.1.3.3 Deep learning model

The architecture of the deep learning model has also been designed according to the state-of-the-art.
In the proposed model, the Hyperion spectral bands are the target variables and the six bands of Landsat 7 are the predictor variables. The Landsat 7 bands will be fed to the model as patches of 5x5 centered for every single pixel. The paper does not specify if the borders of the image (in which some pixels of the patch would not have a value) are also used so I decided not to use it since I did not want to include misinformation on the dataset.

The neural network architecture consists of 3 convolutional layers with 3x3 kernel size followed by ReLU activation functions. The 6-band pixel patch (6@5x5) will increase its feature maps to 32, then down to 16 and finally to 8 performing convolutional operations. Next, once the features of the data have been obtained, the data will be flattened so we get a single long feature vector. Finally, we perform two linear operations to the long vector, first reducing its size from 200 to 170 (Hyperion quantity of bands) and then performing Dropout in order to reduce overfitting as it is a regularization technique for preventing complex adaptations on training data.

The model is using Adam as the optimizer and Mean Squared Error as the loss function. Since the batch size and learning rate are not specified in the paper, I have executed a grid search in order to find the best combination with a considerable execution time. A batch size of 512 and a learning rate of 0.0002 will be set for the model.

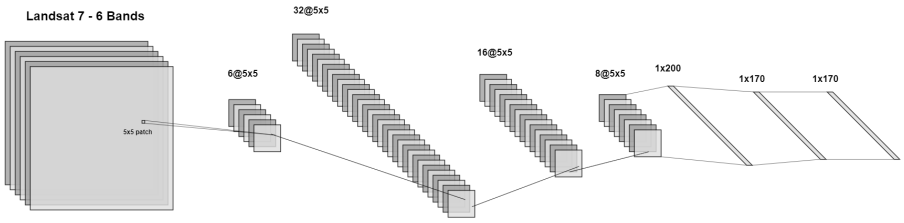
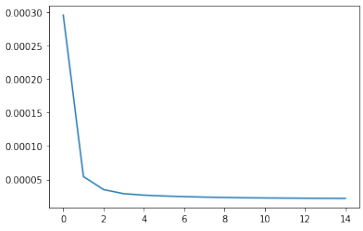


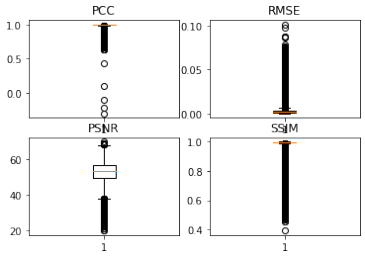
Figure 2: Neural network architecture

6.1.4 Outputs and evaluation

The training set of the data has been processed by the network and the loss, computed with the MSE metric, has been measured along the whole training, which is shown in Figure 3a. In order to evaluate the quality of the output data, four statistical metrics are going to be measured comparing the generated test data with the ground-truth Hyperion test data. The metrics selected are: Pearson's correlation coefficient, root-mean-square error, peak signal-to-noise ratio and structural similarity index. The results measured are shown in different boxplots in Figure 3b.



(a) Training loss curve.



(b) Statistical metrics.

Figure 3: Testing of the network

6.1.5 Risks, issues and next challenges

Despite the great results shown in the report, more study will be done in the methods of the original paper and our reproduction to build it as close as possible so that a more representative understanding of the processes can be extracted from these experiments.
Next steps on this project involve study and experimentation. A study of cutting-edge architectures will be conducted focused on the objective of coming up with a different approach to the same problem that could be focused on the effect of time and zone variation in the spectral response.

6.2 Update II

6.2.1 Summary

Experimentation and model tweaking has been the emphasis during this second project progression in order to achieve a solid model that is able to increase the spectral information of a desired image. The ability to adapt to different content distribution such as having different bit placements, correcting corrupted information and ignoring clouds has also been a main focus so that we minimize the effects of subtle exceptions that could worsen the desired output.

6.2.2 Advanced preprocessing of images

One of the main points that have been deducted from the ongoing making of this project is that preprocessing of the data affects in great manner the results of the experiments. Since we want to achieve the best results possible, and as adaptable as possible, we have conducted some more intelligent preprocessing to our images. Some of the techniques that we have developed are listed below:

- Pixel error correction: satellite sensors sometimes deliver a bad behaviour by missing information in some pixels. During a study of our data, we noticed some spikes on the maximum values of different bands from a single image scene caused by these corrupted pixels. We then decided to correct them by changing its value to the median of the same pixel from the neighbour bands.

  (will insert graph)

- Adaptive normalization: for satellite imaging purposes, an 8-bit display range is not sufficient to differentiate the subtle differences between some spectral responses. The data that we are handling is stored as a 16-bit integer (both signed and unsigned). The numerical range that each band and each satellite sensor covers in this storage differs from one another (for instance [0-2000]). The objective of normalizing the data is to have the whole distribution of the image between the float values [0-1] so that the model can process computational operations using low numbers. The previous way of data normalization that we were doing was dividing all values by their maximum possible value, for instance 65535 for uint16 data. This resulted in very low values. Due to this, we believed that developing an adaptive system of normalization would be the best decision to represent the data so that it does not get stuck in one portion of the whole valid data range. A new approach to normalizing data was developed, it was done by computing the histogram of each band, identifying the most frequent data and linearly stretching the data between the 5% and 95% histogram values.

  (will insert graph)

- Evading clouds: while performing experiments and visualizing the data, we noticed how frequent the image had either clear clouds or a subtle white

gradient due to clouds. This clearly results in our model potentially doing a bad interpretation of the relationships of the data and will definitely damage the results. Moreover, the scene from one satellite and the one from the other are done in a different time (although at the same day), and have different clouds. The Landsat 7 ETM+ satellite imagery product includes a cloud mask, although we used our own technique due to our desire to test this project with different satellites too. Our method is based on performing multiple thresholds to filter image patches that satisfy the requirements set or not. This requirements are based on the mean value, the standard deviation and removing patches that contain limit values.

(will insert image of random image patches evading clouds and not doing it)

- RAM capacity: overcoming the problem of lacking enough RAM memory is an ongoing situation and causes setbacks for freely performing experiments. We are dealing with a huge amount of image patches and this causes our limited RAM to exceed its storage capacity.

6.2.3 Generative deep learning model

Through the research of the best deep learning models suited to our task, we became interested in generative neural networks. These are deep learning models that generate outputs that resemble a target given an input. For our case, we thought that experimenting with Pix2Pix would be a great choice because it is specialized in image-to-image translation.
The Pix2Pix model is a conditional Generative Neural Network that learns a function to map the representation of a multispectral scene into a hyperspectral one, thus increasing the number of spectral bands during the whole process.
Our neural network is compounded of two main pieces: the discriminator and the generator. Our generator will transform our input image to get an output image while the discriminator will measure the similarity of the input image to an unknown image (either a target image from our dataset or an output image from our generator) and will try to guess if it was produced by the generator.
The generator consists of an encoder-decoder architecture in which a convolution is applied after the last layer in the decoder to map the number of output channels, 170 channels in our case.
The discriminator will decide whether the unknown image was generated from the generator or not. The architecture is also called a PatchGAN and will decide if patches from the sub-image patches are real or not.

(include more specifications about model)

The data we are dealing with now is in the format of sub-image patches of a given window size (contrary to the 5x5 input but single pixel output of the past CNN model), these are extracted from the main scene and selected through our filters in order for it to accomplish the requirements set (to avoid clouds, no-data values, etc).

6.2.4 Outputs and evaluation

A large quantity of experiments have been realized with the purpose of designing a neural network suited for our task and data, as well as its parameters.
Varying of the intrinsic architecture involving layers and neurons has been done so that a more stable training is realized by the model. Furthermore, tweaking of the size of the sub-image patches has been done to extract conclusions and take decisions about the best input image size for our neural network. Also, we have performed experiments based on image augmentation by overlapping patches with each other and thus generating more images from a single scene. In the next visualizations, we present our results involving the lasts mentioned testing experiments:

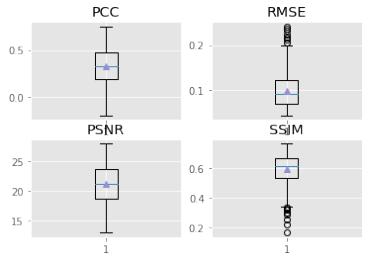


Figure 4: Patches: 32x32, Step: 16.

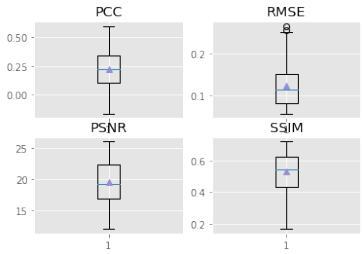


Figure 5: Patches: 32x32, Step: 32.

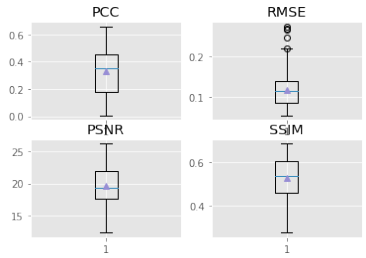


Figure 6: Patches: 64x64, Step: 24.

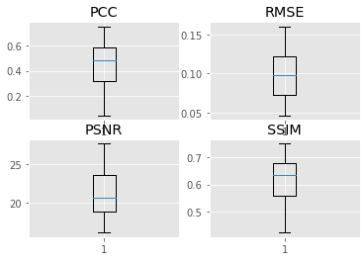


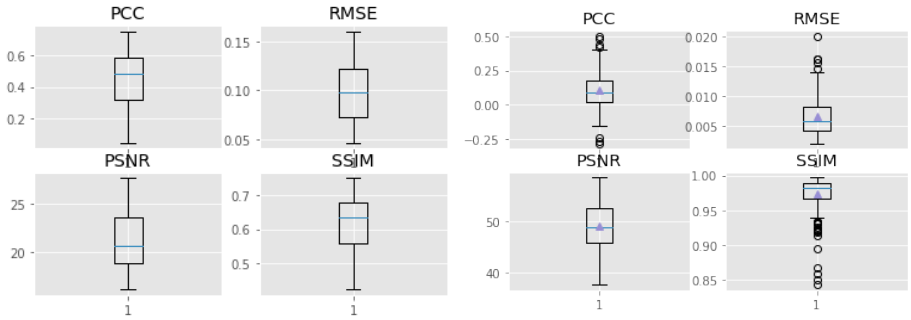
Figure 7: Patches: 64x64, Step: 32.

Moreover, analysis of the normalized data compared to non-normalized raw data has also been performed:

The metrics used for this evaluation are based on similarity measured between two images or arrays.

6.2.5 Ongoing conclusions and next work

The results obtained are now useful in order to take basic decisions such as the size of image patches and quantity of overlapping over patches, this is why the graphs presented show these experiments. We believe that the results could be



(a) Normalized data

(b) Non-normalized data

Figure 8: Comparison on the effects of normalizing data

largely enhanced if we perform experiments with more data (from a few scenes in the same are but different time) and less overlapping. Pix2Pix architectures work better with large amounts of data and we are not currently providing enough, so conclusions from the final model can not be extracted till we perform such experiments.

Apart from the benefits of having a larger dataset, we also noticed that normalizing the data and masking out the clouds does very well for improving the results of our tests. Clouds affected the outputs by providing useless and harmful data that our network was extracting conclusions from.

We also believe that a better hardware equipment would benefit the overall performance due to having the ability to deal with a larger quantity of data and larger batches, making the experiments faster too.

References

1 Zhang L. Nie J., Wei W. et al. (2020). "Deep Blind Hyperspectral Image Super-Resolution". In: *IEEE TNNLS*.

2 Jiang J. Sun H., Liu X. et al. (2020). "Learning Spatial-Spectral Prior for Super-Resolution of Hyperspectral Imagery". In: *IEEE TCI*.

3 Subir P., Nagesh K. (2020). "Transformation of Multispectral Data to Quasi Hyperspectral Data Using Convolutional Neural Network Regression". In: *IEEE TGRS*.

4 Yi C. Zhao Y., Cheung-Wai J. (2019). "Spectral Super-Resolution for Multispectral Image Based on Spectral Improvement Strategy and Spatial Preservation Strategy". In: *IEEE TGRS*.

5 Shen H.; Meng X., Zhang L. (2016). "An Integrated Framework for the Spatio–Temporal–Spectral Fusion of Remote Sensing Images". In: *IEEE TGRS*.

6 Yokoya N. Grohnfeldt C., Chanussot J. (2020). "Hyperspectral and Multispectral Data Fusion: A Comparative Review". In: *IEEE GRSM*.

7 J., Jiang (2020). *Hyperspectral-Image-Super-Resolution-Benchmark*. URL: https://github.com/junjun-jiang/Hyperspectral-Image-Super-Resolution-Benchmark.
8 Atlassian (2011). Trello. URL: https://trello.com/.
9 GDAL/OGR contributors (2020). *GDAL/OGR Geospatial Data Abstraction software Library*. Open Source Geospatial Foundation. URL: https://gdal.org.
10 QGIS.org (2020). GIS Geographic Information System. Open Source Geospatial Foundation Project. URL: http://qgis.org.
11 Gorelick, Noel, Matt Hancher, Mike Dixon, Simon Ilyushchenko, David Thau, and Rebecca Moore (2017). "Google Earth Engine: Planetary-scale geospatial analysis for everyone". In: *Remote Sensing of Environment*. DOI: 10.1016/j.rse.2017.06.031. URL: https://doi.org/10.1016/j.rse.2017.06.031.
12 USGS (2003a). USGS EROS Archive. URL: https://www.usgs.gov/core-science-systems/nli/landsat/landsat-7?qt-science_support_page_related_con=0#qt-science_support_page_related_con.
13 — (2000). USGS EROS Archive. URL: https://www.usgs.gov/centers/eros/science/usgs-eros-archive-earth-observing-one-eo-1-hyperion?qt-science_center_objects=0#qt-science_center_objects.
14 — (2003b). USGS EROS Archive. URL: https://www.usgs.gov/faqs/what-landsat-7-etm-slc-data?qt-news_science_products=0#qt-news_science_products.