UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona

telecos
BCN

# Pan-sharpening of WorldView-2 images with deep learning

A Degree Thesis
Submitted to the Faculty of the
Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona
Universitat Politècnica de Catalunya
by

Miquel Jové Caballero

In partial fulfilment
of the requirements for the degree in
**TELECOMMUNICATION TECHNOLOGIES AND SERVICES ENGINEERING**

Advisors:
Verónica Vilaplana Besler
Luis Fernando Salgueiro

Barcelona, June 2021

# Acknowledgements

I would like to express my sincere gratitude to my both supervisors, Verónica Vilaplana and Luis Salgueiro for their constant dedication and interest on the project evolution, for guiding me and having patience on critical moments. I would also like to appreciate the opportunity that was given to me by Verónica to work in a project that I was interested in.

In addition, I would like to extend my gratitude to Manel Davins and Sauc Abadal for the companionship and the suggestions given through the development of the project.

Moreover, I have to express my thanks to the Universitat Politècnica de Catalunya, the Instituto de Oceanografía y Cambio Global de Universidad de Las Palmas de Gran Canaria and the European Space Agency for giving me the tools to participate in this project.

# Abstract

In recent years the exponential growth on Deep Learning interest has had a huge impact on improving the resolution of images. In particular, enhancing the quality of remote sensing imagery is a field where many models have been proposed by different researchers.

One of this approaches is pan-sharpening, which takes advantage from the satellites imagery pairs in order to raise the resolution of multispectral or hyperspectral images.

In this project, a model from the literature will be adapted for WorldView-2 satellite imagery and modified to improve the current stated results from the model. Experiments results will be compared between the adapted model and the modified one so the adjustments effectiveness can be proven.

# Resumen

En los últimos años el incremento exponencial del interés por el Aprendizaje Profundo ha tenido un gran impacto en la mejora de la resolución de imágenes. En particular, enriquecer la calidad de las imágenes captadas con teledetección es un campo donde distintos investigadores han propuesto varios modelos.

Uno de estos enfoques es el pan-sharpening, que aprovecha los pares de imágenes de los satélites para incrementar la resolución de imágenes multiespectrales o hiperespectrales.

En este proyecto, un modelo de la literatura se adaptará para imágenes del satélite WorldView-2 y será modificado para mejorar los resultados establecidos en la actualidad por el modelo. Los resultados de los experimentos se compararán entre el modelo adaptado y el modelo modificado para verificar que los cambios realizados son efectivos.

# Resum

En els darrers anys l'increment exponencial de l'interés per l'Aprenentatge Profund ha tingut un gran impacte en la millora de la resolució d'imatges. En particular, enriquir la qualitat de les imatges captades per teledetecció és un camp diferents investigadors han proposat diversos models.

Un d'aquests enfocaments és el pan-sharpening, que aprofita els parells d'imatges dels satèl·lits per incrementar la resolució d'imatges multiespectrals o hiperespectrals.

En aquest projecte, un model de la literatura s'adaptarà per a imatges del satèl·lit WorldView-2 i serà modificat per millorar els resultats establerts pel model actualment. Els resultats dels experiments es compararan entre el model adaptat i el model modificat per tal de verificar l'efectivitat del canvis realitzats.

# Contents

List of Figures 7

List of Tables 7

1 Introduction 11
1.1 Statement of purpose . . . . . 11
1.2 Requirements and specifications . . . . . 11
1.3 Methods and procedures . . . . . 12
1.4 Work Plan . . . . . 12
1.5 Deviations and incidences . . . . . 13

2 State of the Art 14

3 Preliminaries 15
3.1 Deep Learning . . . . . 15
3.2 Generative Adversarial Network . . . . . 15
3.3 WorldView2 and its characteristics . . . . . 15

4 Methodology 18
4.1 Datasets . . . . . 18
4.2 System architecture . . . . . 20
4.2.1 Blocks architectures . . . . . 20
4.2.2 Networks architectures . . . . . 22

5 Experiments 28
5.1 Initialization . . . . . 28
5.1.1 Generative network . . . . . 28
5.1.2 Reconstructor network . . . . . 31
5.1.3 Discriminative network . . . . . 31
5.2 Training results . . . . . 31
5.3 Test results . . . . . 34

6 Budget 40

7 Conclusions and future development 42

References 43

Appendices 46

A Dataset Images 46

# Listings

# List of Figures

1 Updated GANTT diagram. . . . . 13
2 WorldView-2. (image credit: European Space Agency) . . . . . 15
3 Satellite plane view from different tilted perspectives. (image credit: James Dietrich [1]) 16
4 Spectral Response of the WorldView2 panchromatic and multispectral imagery. (image credit: DigitalGlobe) . . . . . 17
5 Blending together image fragments with ENVI . . . . . 19
6 Cropping images with SNAP . . . . . 19
7 ReLu and LeakyReLu activation functions. . . . . . 21
8 Residual Dense Block architecture. . . . . . 21
9 Residual in Residual Dense Block architecture. . . . . . 22
10 Generator residual extraction model from PercepPan [2]. . . . . . 23
11 Generator HR MS image computation from PercepPan [2]. . . . . . 23
12 WorldView-2 band reproduction from Figure 4. . . . . . 24
13 Reconstructor network from PercepPan [2] . . . . . 25
14 VGG-style network from PercepPan [2] without the classifier. . . . . . 25
15 Generator model from PercepPan modification 1. . . . . . 26
16 Generator model from PercepPan modification 2. . . . . . 27
17 Pretraining on model from Section 4.2.2.1 evolution. . . . . . 29
18 Pretraining of model described in Section 4.2.2.1 image results. . . . . . 30
19 Training on model of Section 4.2.2.3 image results before divergence is reached. . . . . 32
20 Image color prediction difference because of dynamic range expansion. . . . . . 33
21 Examples of image results taken from Section 4.2.2.1 model test. . . . . . 35
22 Test images at true color comparison between models. . . . . . 37
23 Test images comparison between models depending on $\alpha$ value. . . . . . 39
24 Teide (Spain) image. . . . . . 46
25 Dublin (Ireland) image. . . . . . 47
26 Wolverhampton (UK) first image. . . . . . 48
27 Wolverhampton (UK) second image. . . . . . 49
28 Riga (Latvia) image. . . . . . 50

# List of Tables

1 Dataset images general information . . . . . 18
2 Reconstructor network measured coeficients. . . . . . 24
3 Results of testing the generative network from model described in Section 4.2.2.1 pretrained in function of input patches size and batch size. . . . . . 29
4 Results of testing all pretrained models at input size 64×64 and batch size 4. . . . . . 30
5 Results of testing all models at best training state accomplished. . . . . . 36
6 Results of testing all equalized models for different values of $\alpha$. Best models and best results for each metric are marked on red. . . . . . 38

7 Imagery cost computation . . . . . . . . . . . . . . . . . . . . . . . . . . . . 40
8 Total cost computation. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 41

# Abbreviations

**AdaIN** Adaptative Instance Normalization

**CNN** Convolutional Neural Network

**DL** Deep Learning

**ESRGAN** Enhanced Super-Resolution Generative Adversarial Network

**GAN** Generative Adversarial Network

**GPU** Graphics Processing Unit

**HR** High Resolution

**LeakyReLu** Leaky Rectified Linear Unit

**LR** Low Resolution

**MS** Multispectral

**NIR** Near Infra-Red

**PAN** Panchromatic

**PSNR** Peak Signal-to-Noise Ratio

**QNR** Quality Without Reference

**RDB** Residual Dense Block

**ReLu** Rectified Linear Unit

**SAM** Spectral Angle Mapper

**SSIM** Structural Similarity

**STD** Standard Deviation

**WV2** WorldView-2

# Revision history and approval record

| Revision | Date | Purpose |
|---|---|---|
| 0 | 01/06/2021 | Document creation |
| 1 | 18/06/2021 | Document revision |
| 2 | 21/06/2021 | Document revision |

DOCUMENT DISTRIBUTION LIST

| Name | e-mail |
|---|---|
| Miquel Jové Caballero | miquel.jove.caballero@estudiantat.upc.edu |
| Verónica Vilaplana Besler | veronica.vilaplana@upc.edu |
| Luis Fernando Salgueiro Romero | luis.fernando.salgueiro@upc.edu |

| Written by: | | Reviewed and approved by: | |
|---|---|---|---|
| Date | 21/06/2021 | Date | 21/06/2021 |
| Name | Miquel Jové Caballero | Name | Verónica Vilaplana Besler |
| Position | Project Author | Position | Project Supervisor |

# 1 Introduction

Remote sensing is based on detecting physical characteristics of an area by measuring its reflected and emitted radiation at a certain distance [3].

Spatial and spectral resolution are two concepts that define the quality of a remote sensing image. Spatial resolution stands for “the size of the smallest feature that can be detected by a satellite sensor”, while spectral resolution refers to the ability of a satellite sensor to measure specific wavelengths of the electromagnetic spectrum” [4]. In our case, using WorldView-2 (see Section 3.3) imagery entails that one of the pair of images obtained is a multi-spectral image, with 8 bands covering a narrow range of the electromagnetic spectrum (high spectral resolution) but with a lower spatial resolution. The other image from the pair is called the panchromatic image, which is a high spatial resolution image that covers a wider range of the electromagnetic spectrum (lower spectral resolution).

Nevertheless, satellites sensors do not have enough resolution to detect small-scale objects properly. Multispectral images are able to solve this issue if having higher resolutions. In order to improve MS image resolution, techniques like pan-sharpening are used. Pan-sharpening is an image processing technique that makes use of the panchromatic band spatial information to enhance the details and leverage the resolution of the multi-spectral bands creating a multi-spectral high-resolution image. Deep Learning methods are getting attention in the remote sensing community for the great performance in computer vision tasks, being pan-sharpening one of the main research topics tackled.

## 1.1 Statement of purpose

This project is proposed by the Universitat Politècnica de Catalunya (UPC) to use Deep Learning techniques to exploit spatial and spectral correlations of remote sensing images to generate high resolution images from low resolution counterparts.

The project aims at reproducing some pan-sharpening techniques, where the finally proposed neural network will be based on different models in literature and ultimately compared with State of the Art results to validate the functionality of the proposal. In addition, this project also intends that the author familiarizes with remote sensing, Deep Learning and pan-sharpening concepts in order to apply the knowledge acquired to the project development tasks and in future works.

## 1.2 Requirements and specifications

The main objectives of this project are:

- Analyze the State of the Art in Deep Learning of pan-sharpening for super resolution of remote sensing images.
- Get familiar with Python language and Pytorch Library for developing and implementation of the Convolutional Neural Networks.
- Propose, train and test a CNN models to tackle pan-sharpening.

In order to accomplish these objectives, some specifications may also be defined:

- Achieve a comprehensive State of the Art knowledge about pan-sharpening in favor of supporting the background of the project.
- Prepare and use a database of WorldView-2 images with a reasonable amount of samples compared to the literature.
- Implement an alternative model that manages to perform pan-sharpening of input images successfully.
- Experiments results will be compared with a model based on one of the literature proposals so as to have quality performance reference for our own alternative model.

## 1.3 Methods and procedures

This project is based on a modification from [2], adapting the available code to be able to work with our database and using its structure to build our own models. In addition, test modifications are done to the original code to fit our needs, such as showing image results and save metrics. Modifications applied are based on [5] input and output CNN data treatment.

## 1.4 Work Plan

The project organization was done in first place taking into account the limited experience and knowledge in Deep Learning and pan-sharpening concepts. Due to this, first weeks of work where exclusively assigned to acquire knowledge and complete some Deep Learning courses. Afterwards, most part of the work time was planned to be spent on some previous models related to pan-sharpening and design of our own model to compare the results with. Finally, once final models are prepared to be tested, some time is attributed to obtain results and elicit comparisons and conclusions.

However, tests on previous models took more time than expected due to inexperience in coding for Deep Learning purposes and unexpected issues while trying a few training evaluations with different characteristics. Nevertheless, planned tasks were completed as initially designed, even with a some changes on timings.

Figure 1 shows the final Time Plan with the tasks as they were finally executed:

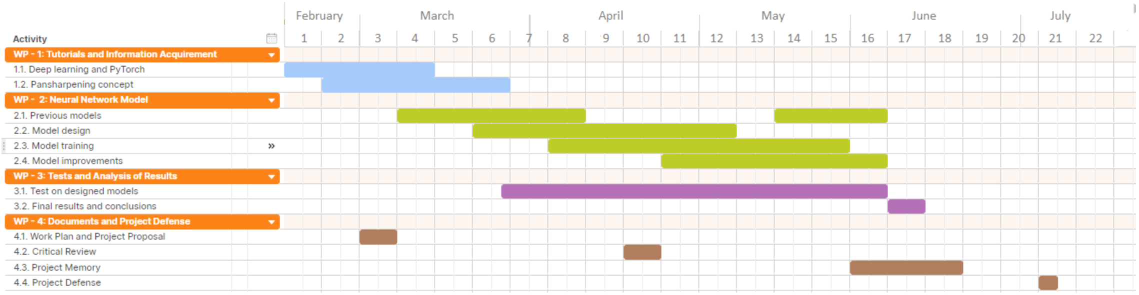


Figure 1: Updated GANTT diagram.

During the development of this project, some milestones are achieved:

- Week 6: Extensive State of the Art knowledge about project's field of research.
- Week 8: Base model fully analyzed and working properly.
- Week 12: Functional designed model.
- Week 12: Extensive and heterogeneous database.
- Week 17: Final results and conclusions.

## 1.5 Deviations and incidences

A few incidences occurred during the development of the project. Most of them were caused by coding issues that stopped some running executions and delayed results analysis. In addition, limited resources caused to also interrupt executions because of full disk usage or GPU memory exceeded. In order to solve this issues, more storage allocation was requested to save the database and output results, as well as the reduction of the input image sizes to fit the hardware requirements.

Moreover, delays on drafting the designed model caused the final tests to postpone too.

# 2 State of the Art

In the recent years, pan-sharpening techniques have been evolving due to Deep Learning methods variety growing.

Nevertheless, classical pan-sharpening methods such as component substitution [6, 7], variational optimization [8, 9] and multi-resolution analysis [10, 11] already existed before DL exponential increasing usage. This methods are based on MS and PAN images fusion using data transformations, filtering and algorithms that exploit spectral and spatial resolution characteristics. However, Deep Learning techniques have ensured better results over past years than classical methods.

Many architectures have been put forward by different researchers from different starting points, such as converting a previously proposed CNN for super-resolution to pan-sharpening [12], or a two branches network that extracts spectral and spatial features from MS and PAN images respectively and subsequently fuses them [13]. Other approaches like [5] aim to preserve spectral and spatial information by adding up-sampled MS images to the network output and training the network parameters in the high-pass filtering domain instead of the image domain. Furthermore, [14] uses a generative adversarial network to deal with the pan-sharpening concern that handles a two-stream generator designed to receive MS and PAN images simultaneously.

Despite all this improvements, pan-sharpening has always had to deal with the fact that generated high-resolution MS images cannot be tested against ground truth images of the same size, since in most cases they do not exist. To avoid this issue and create the pair of LR-HR images to train the network, the MS and PAN images are downsampled to a lower resolution so that the generated image can be compared with its original versions. Some researchers have proposed a model such as [2] that avoids this issue by using a GAN with an auxiliary reconstructor network that is able to generate the MS and PAN pair images from the generator network output employing some features extracted from the satellite characteristics.

Taking into account this last approach, we aim at improving [2] results by adapting their model for a different satellite (WorldView-2 for our case) and develop some modifications on the model considering other approaches.

# 3 Preliminaries

For the purpose of understanding the content of this Memory, some important general concepts and necessary information are described bellow.

## 3.1 Deep Learning

Deep Learning is a field of computer science that "exploits many layers of non-linear information processing for feature extraction and transformation, and for pattern analysis and classification" [15]. It can be supervised, where the algorithm trains using labeled datasets; or as for our case, unlabeled, where the datasets are not classified and the network tries to come upon hidden patterns without human intervention.

## 3.2 Generative Adversarial Network

Generative Adversarial Networks are composed by two neural networks that contest each other. One of the networks is the generative network, whose function is to generate samples that are similar to the target distribution., while the other network, named the discriminator, evaluates the samples coming from the generator and compares with the samples from the target distribution. The generative network learns from the input data in order to generate better results, while the discriminative network tries to distinguish between generative network's output data from the original data. In the training phase, the generator is expected to produce generated samples similar to the target distribution, while the discriminator becomes more skilled in determining which samples come from a generated distribution and which samples come from a target distribution. Nevertheless, they are very sensitive to variations on hyper-parameters.

## 3.3 WorldView2 and its characteristics

WorldView-2 is a satellite used for environmental imaging and monitoring.

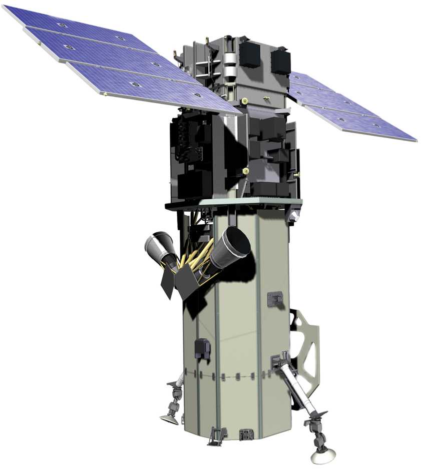


Figure 2: WorldView-2. (image credit: European Space Agency)

It provides commercially available imagery, constituted by panchromatic images of 0.46 m resolution and eight-band multispectral images of 1.84 m resolution [16]. This resolution corresponds to the off-nadir resolution, which is the resolution of the images taken from the satellite aiming perpendicularly to the surface and from where it reaches a swath width of 16km. In addition, WorldView-2 has the capacity of tilting in order to reach other areas that are away from its nadir view, but with a counterpart that implies less resolution for this images, as shown in Figure 3:

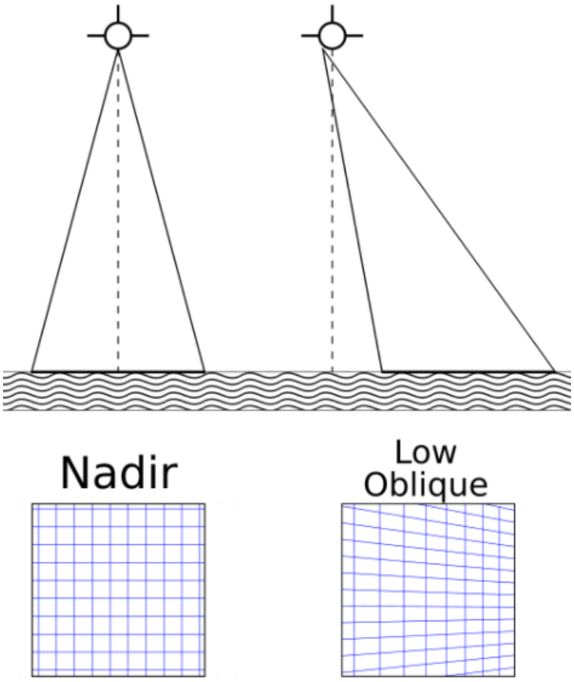


Figure 3: Satellite plane view from different tilted perspectives. (image credit: James Dietrich [1])

To perform remote sensing tasks, the satellite carries on-board two sensing instruments, a multispectral sensor able to generate several spectral bands with high spatial resolution and a panchromatic sensor that generates a very-high spatial resolution, and with a wide spectral range that covers most of the multispectral bands. Besides the 4 most frequently used bands on remote sensing satellites (Blue, Green, Red and NIR), WorldView2 has a shorter wavelength blue band called Coastal, which is used for water color studies; a Yellow band that allows more color accuracy on the visible spectrum; a Red Edge band able to perceive a high reflectivity portion of the vegetation response; and a second NIR band with longer wavelength that is sensitive to atmospheric water vapor. [17] Figure 4 shows the spectral response of WV2 noramalized bands response:

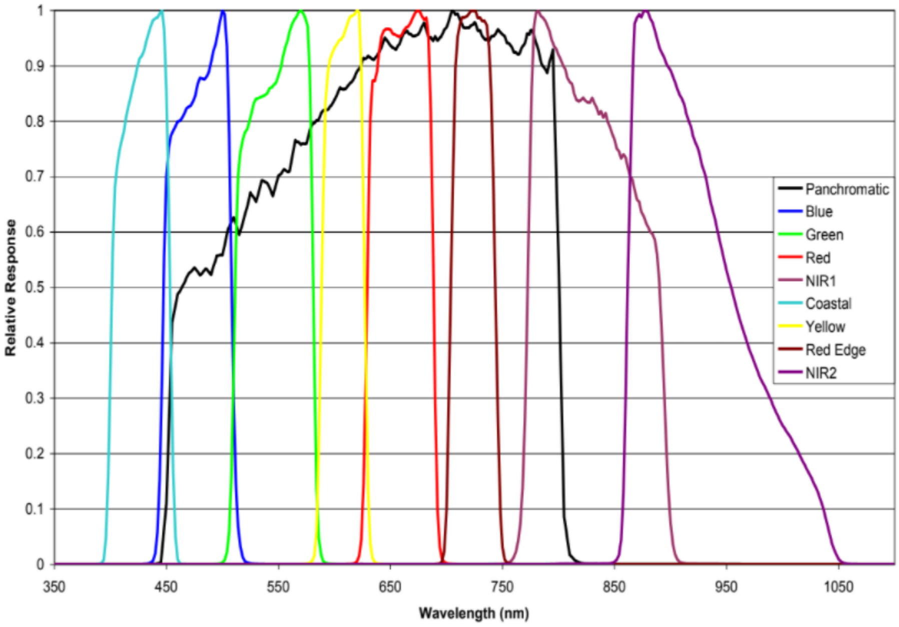


Figure 4: Spectral Response of the WorldView2 panchromatic and multispectral imagery. (image credit: DigitalGlobe)

# 4 Methodology

To set the project basis, the dataset that will be used to train and test our models has to be defined. Then, different models should be designed and their performances must be tested once trained.

## 4.1 Datasets

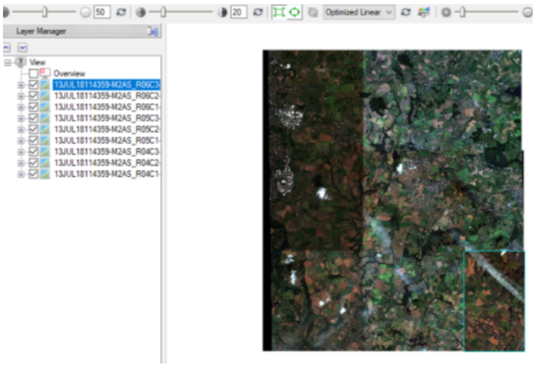
The dataset that has been used to train the models in our experiments is constituted by 1 pair of MS and PAN images from [18] whose MS image has a size of 4,096×4,096 per band and a 1.3 m resolution and 4 pairs from [19] whose MS images have a size between 9,868×10,727 and 16,384×16,384 pixels per band and a 1.6 m resolution. Respective PAN images have a size 4 times higher for each dimension in respect to MS bands images size. The images that we use have the following specifications:

Table 1: Dataset images general information

| Image | Location | Date | Time | Pixel resolution | Size (MS image) |
|---|---|---|---|---|---|
| **1** | Teide (Spain) | 13/06/2017 | 12 : 16 | $1.6m$ | $4,096 \times 4,096$ |
| **2** | Dublin (Ireland) | 21/04/2015 | 12 : 08 | $1.6m$ | $11,838 \times 12,279$ |
| **3** | Wolverhampton rural area (UK) | 18/07/2013 | 11 : 43 | $1.6m$ | $12,288 \times 10,145$ |
| **4** | Wolverhampton urban area (UK) | 18/07/2013 | 11 : 43 | $1.6m$ | $12,288 \times 10,065$ |
| **5** | Riga (Latvia) | 02/05/2012 | 09 : 52 | $1.6m$ | $9,868 \times 10,727$ |

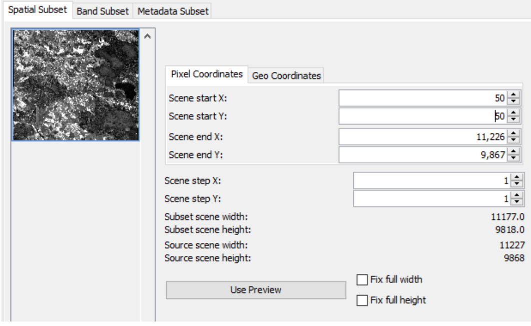
Images may be viewed on Appendix A.

Some images presented black areas on contours or where divided into smaller images. Thus, to create this database we have used software such as ENVI [20] and SNAP [21] to blend together image fragments by making image mosaicing and cropping images to avoid black out-of-bounds pixels and make rectangular images. The mosaicing process also modifies the fragment images colors (homogeinizing image luminance) so that the final result does not have sudden color changes between fragments.



View
□ Overview
☑ 13JUL18114359-M2AS_R06C3
☑ 13JUL18114359-M2AS_R06C2
☑ 13JUL18114359-M2AS_R06C1
☑ 13JUL18114359-M2AS_R05C3
☑ 13JUL18114359-M2AS_R05C2
☑ 13JUL18114359-M2AS_R05C1
☑ 13JUL18114359-M2AS_R04C3
☑ 13JUL18114359-M2AS_R04C2
☑ 13JUL18114359-M2AS_R04C1

Figure 5: Blending together image fragments with ENVI



Scene start X: 50
Scene start Y: 50
Scene end X: 11,226
Scene end Y: 9,867
Scene step X: 1
Scene step Y: 1
Subset scene width: 11177.0
Subset scene height: 9818.0
Source scene width: 11227
Source scene height: 9868
□ Fix full width
□ Fix full height

Figure 6: Cropping images with SNAP

In order to split our dataset for training, validation and test phases, we generate 10,300 patches of 128×128 pixels from MS images and 512×512 pixels from PAN images. The number of patches extracted from each image proportionally depends on the size of that image, so bigger images generate more patches. Afterwards, this database is divided to 8,250 patches (80%) for training and 1,250 patches (10%) for validation and the remaining 1,250 patches (10%) for test. Each patch is generated by selecting a random pixel from the MS and the corresponding pixel from the PAN image. To ensure data variability, we apply a data augmentation method, which consists on flipping vertically and/or horizontally with a 50% probability for each flip and a rotation equiprobably to left or right, with a probability of a 50% to being rotated to any side. Due to hardware restrictions (GPU memory), patches were reduced to 64×64 pixels for MS images and 256×256 pixels for PAN images. In this case, the indices of the pixel from where the patch is generated are preserved inside a dictionary so the CNN output patch can be compared with the original patch by using the same procedure and the saved indices. Finally, this patches are independently normalized by channels using the mean and the standard deviation of each channel, following this procedure:

$$X_{N_c} = \frac{X_c - MEAN(X_c)}{STD(X_c)} \tag{1}$$

where $X_c$ corresponds to a channel of the MS patch or to the PAN patch and $X_{N_c}$ corresponds to a normalized channel of the MS patch or to the PAN patch. In order to be able to denormalize the model output images, the mean and the standard deviation of each channel of every patch are kept inside a dictionary too with the purpose that denormalized images can be obtained by:

$$X_c = X_{N_c} \cdot STD(X_c) + MEAN(X_c) \tag{2}$$

## 4.2 System architecture

This system uses 3 different models. All of them are composed by networks that are build by a concatenation of layers. We will refer to a layer or a group of layers as blocks. Before going into the models structure, some important blocks must be commented.

### 4.2.1 Blocks architectures

Before setting up the global structure of our models, some basic blocks that are used may be described.

#### 4.2.1.1 Convolutional Block

Convolutional layers are the major building blocks used in convolutional neural networks. Convolutional layers perform a convolutional operation between an array of input data and the kernel, which is a bi-dimensional array of weights that the network modifies to improve its results. This kernel affects every channel of the input data independently, using different weights for each. For our case, we use a kernel of a size 1×1, 3×3 and 4×4 pixels depending on the position of the block inside the network. In addition, convolution procedure is done using zero padding when needed, which involves filling with the input data array borders with pixels of value 0 so that the output array can be the same size as the input array. Afterwards, the output can be normalized using batch normalization. This is used to stabilize the learning process and reduce the amount of epochs required to train the network. Nevertheless, some experiments like [22] conclude that batch normalization does not help deblurring images but deteriorates the results because of features normalization. Finally, an activation function is used in order to increase the non-linearity in the output. For our case, the LeakyReLu Block is used when needed.

#### 4.2.1.2 ReLu and LeakyReLu Blocks

As previously mentioned, activation functions are used to add non-linearity and therefore allow the network to learn. In particular, a Leaky Rectified Linear Unit layer performs a threshold operation where any input value less than zero is multiplied by a fixed scalar (0.2 for our experiments). That also means that the block parameters/weights are fixed through the entire training phase. The usage of LeakyReLu grants an extended output range than using the ReLu block, in addition to hiking the flexibility of the model, since ReLu block converts the negative values to 0. ReLu usage is reserved for upsampling blocks.

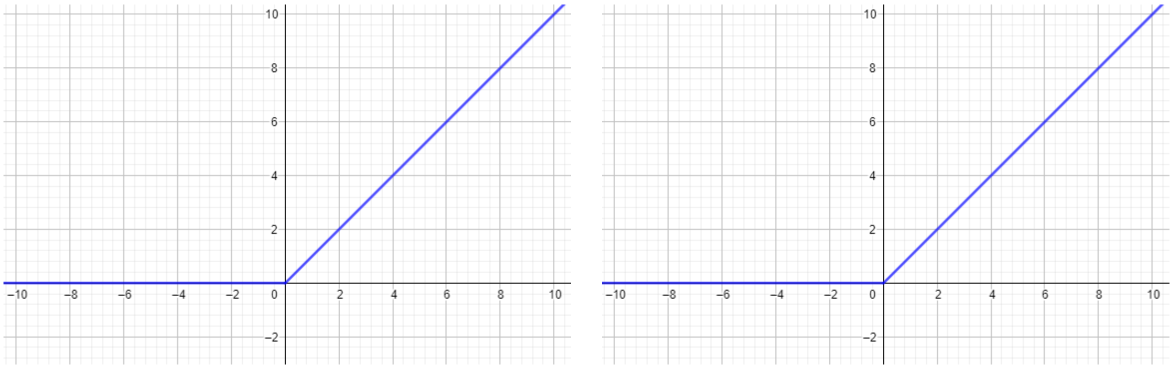


Figure 7: ReLu and LeakyReLu activation functions.

#### 4.2.1.3 Upsampling Blocks

An upsampling layer is used to perform a feature map expansion from the input to the output by increasing the size of the input array. The interpolation that we use to upsample inside our models is the nearest neighbour, since based on [23] this approach accomplishes better results. Moreover, a convolutional block with zero padding and a ReLu activation function are added before the upsampling block to improve the results of the upsampling method. This block allows the network to generate images of bigger size at the output than the input images size. Since the latter layer is expected to have the size of the PAN image, the height and the width of the MS image has to be increased by 4. The upsampling block does an upsampling by 2, so a pair of block must be concatenated to accomplish the objective.

#### 4.2.1.4 Residual Dense Blocks

Based on [24], Residual Dense Block extracts abundant local features via dense connected convolutional layers. The RDB block is used on our models as a basic unit block. This is formed by the following structure:

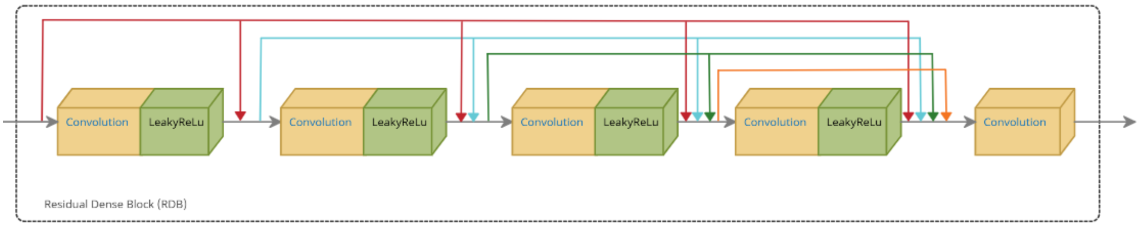


Figure 8: Residual Dense Block architecture.

#### 4.2.1.5 Residual in Residual Dense Blocks

Based on [25], Residual Dense Block extracts abundant local features via dense connected convolutional layers. The RDB block is used on our models as a basic unit block. This is formed by the following structure:

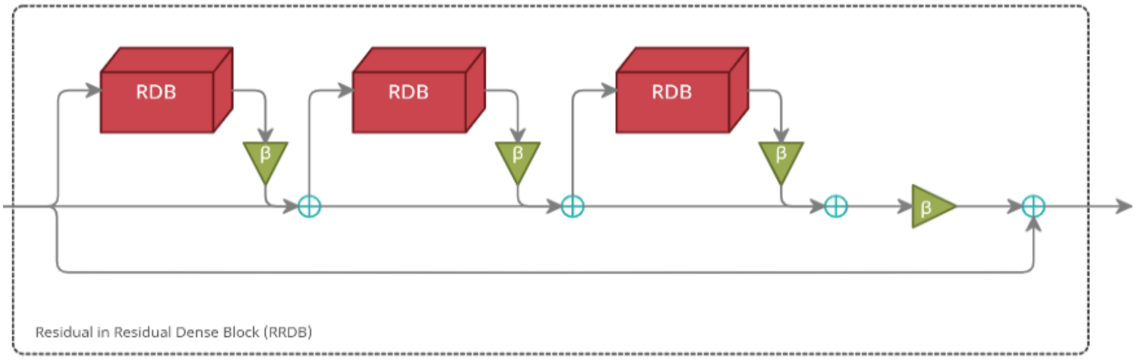


Figure 9: Residual in Residual Dense Block architecture.

### 4.2.2 Networks architectures

The models that we designed for our experiments are based on the PercepPan model used for pan-sharpening on IKONOS and QuickBird [2], satellites with less resolution than WorldView-2 that collect PAN imagery and MS imagery of 4 bands.

#### 4.2.2.1 PercepPan

Taking into account PercepPan model, we adapted it to our input data, maintaining the rest of the network blocks and parameters unchanged. The PercepPan is based on a GAN model with a generator and a discriminator as general GAN network do, as well as a reconstructor network. The generative network is based on ESRGAN model [25] and it is adapted for remote sensing images. This network takes the MS image as input and extracts learning residual details, which are denoted as $\sigma_x$ and $\mu_x$. Inspired by AdaIN [26], the MS input image is treated as the style image from where the style features $\sigma_x$ and $\mu_x$ are extracted using the ESRGAN-style generative network, and the respective PAN image is treated as the content image. Content features $\sigma_p$ and $\mu_p$ corresponding to PAN image are not computed and are assigned as an identity matrix and as a zero matrix respectively. Thus, the final output of the generative network is computed from the following operation:

$$y = \sigma_x \cdot p + \mu_x \tag{3}$$

where $y$ corresponds to the output generated high resolution MS image and $p$ corresponds to the original PAN image. This operation is done channel-wise, since both style features have the dimensions of the high resolution MS image and the PAN image influence is equally applied on each channel. The generative network has the following structure:

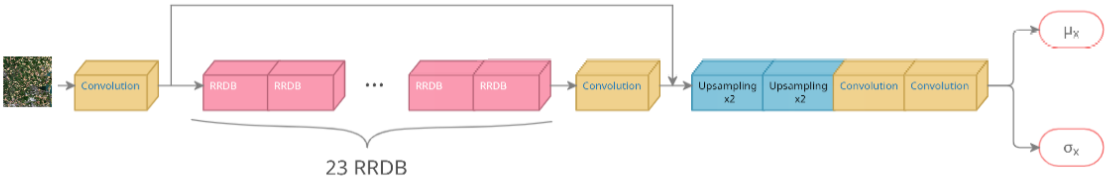


Figure 10: Generator residual extraction model from PercepPan [2].

As previously mentioned, the out residuals will be used to compute the high resolution image following the next scheme:

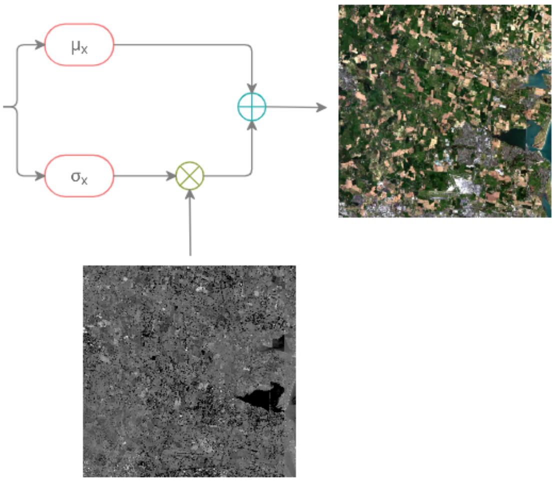


Figure 11: Generator HR MS image computation from PercepPan [2].

The reconstructor network from [2] is also adapted to WorldView2 imagery. This network aims at generating a prediction of the original pair of images from the generative network output. In order to recover the LR MS image, a combination of a blurring and downsampling is applied. Blurring is done using filters with cutoff frequencies just as in [27]. For the reconstruction of the PAN image, the method is more complex. In general, the PAN image covers all the wavelengths of the MS image spectral bands, so the PAN image can be approximated by a linear combination of the HR MS image bands. The weight of each band is computed taking into account MS band distributions over the spectrum in respect to the PAN frequencies. Therefore, Figure 4 is reproduced approximating its values since they are not published because they are approximations too. Although seeming suboptimal to reproduce these approximated values, with an appropriate sampling frequency we accomplish a good estimation.

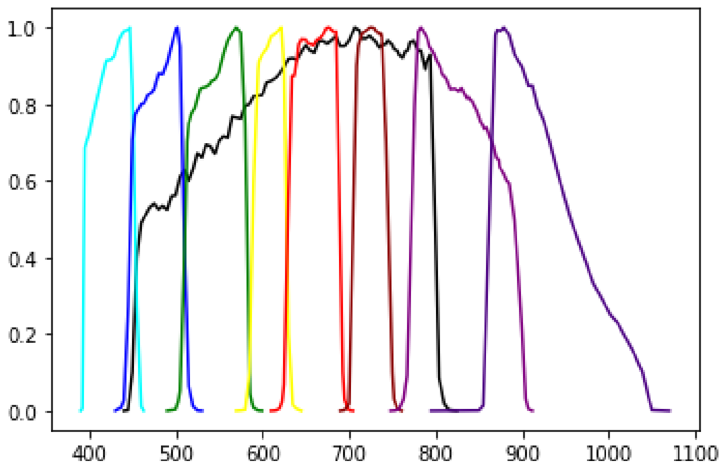


Figure 12: WorldView-2 band reproduction from Figure 4.

This reproduction allows us to compute intersection areas between the PAN image and the MS image bands and hence measure the weight of each band. The measured coefficients are shown in Table 2:

Table 2: Reconstructor network measured coeficients.

| **Bands** | **Weight** |
|---|---|
| **Costal** | 0.008988 |
| **Blue** | 0.132015 |
| **Green** | 0.213481 |
| **Yellow** | 0.142277 |
| **Red** | 0.234538 |
| **Red Edge** | 0.160831 |
| **NIR1** | 0.107839 |
| **NIR2** | 0.000031 |

The linear combination applying the weights is done using a 1×1 convolutional block without normalization neither activation function. Despite being considered a network, it is initialized with values that will be fixed during the entire training phase. This values are the cutoff frequencies of the filters used for blurring and the weights of every band in respect to the PAN image. The output of the reconstructor, which are the reconstructed images, are compared with the original images in order to compute the Pixel Loss measuring the sum of the L1Loss between MS original image and the MS reconstructed image and the L1Loss between PAN original image and the PAN reconstructed image.

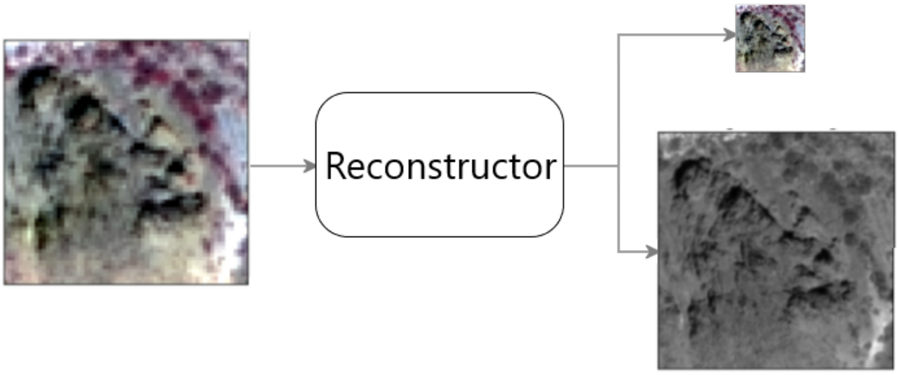


Figure 13: Reconstructor network from PercepPan [2]

The discriminative network takes the reconstructed pair of images as input and uses a feature extractor for the MS reconstructed image and a different feature extractor for the PAN reconstructed image. The MS image feature extractor is formed by 2 concatenated convolutional blocks, both with LeakyReLu activation function and the last of them using batch normalization. The PAN image feature extractor is formed by 4 concatenated convolutional blocks that use batch normalization and have a LeakyReLu activation function. In addition, The PAN image is downsampled to the MS reconstructed image size so the output of both feature extractor will have the same size. Subsequently, the output of both extractor is concatenated and features are then compared to the features extracted when the input of the feature extractors are the original images instead of the reconstructed images. The L1Loss between features extracted from reconstructed images and original images leads to the computation of the Feature Loss, that will be added to the already measured Pixel Loss. Afterwards, a VGG-style network as in [28] receives the computed features in order to rate the probability of the input features being from real data rather that generated data. The output of the VGG-style network is a scalar that corresponds to this probability. This VGG-style network is composed by a concatenation of convolutions as shown in Figure 14:

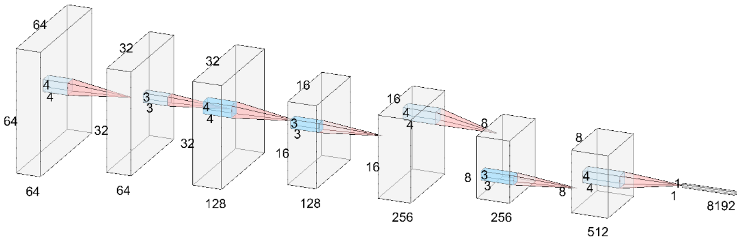


Figure 14: VGG-style network from PercepPan [2] without the classifier.

Then this vector is introduced into the classifier of the VGG-style network, which is the one extracting the probability value. The classifier architecture is conformed by a Linear Block, which transforms the 8,192 size vector input into a 100 output size vector; a LeakyReLu Block that reduces the linearity of the data; and another Linear Block that transform the vector into a single scalar that corresponds to the mentioned probability. This probability is compared with the probability obtained by taking the original images features as the VGG-style input instead of the reconstructed images features. This is used to compute the GAN Loss.

#### 4.2.2.2 PercepPan modification 1

In order to design a model that may increase the original PercePan model performance, we propose a set of modification that may be done to the generative network. The reconstructor and the discriminative network remain unchanged so they will not be mentioned in this section, since they are already explained on Section 4.2.2.1. First of all, the residual at the output have been modified for a generated image. This way, we avoid the PAN multiplication by the $\sigma_x$ residual and the ensuing sum with the $\mu_x$ residual. To counteract this modification, we perform a bicubic interpolation to the MS input image so that it has the same size as the PAN image. Then we add the PAN image to the MS image as it was a $9^{th}$ band. This allows the network to learn about the PAN influence to the MS image instead of performing those stated computations. In addition, the upsampling blocks inside the generative network are removed. That implies that the generative networks stands with the following architecture:

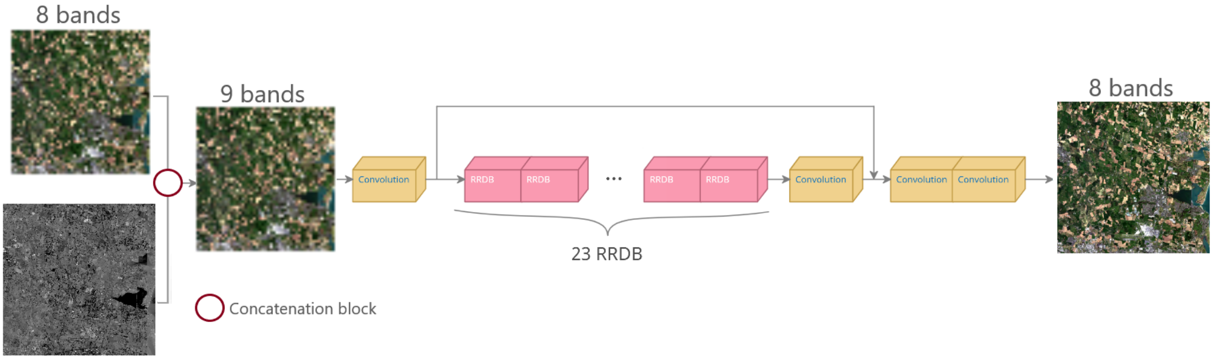


Figure 15: Generator model from PercepPan modification 1.

This network does not learn to upsample the original MS image with good resolution using the residuals but to improve the quality of an already upsampled MS image with the help of the PAN image.

#### 4.2.2.3 PercepPan modification 2

Despite being a little modification, we decided to implement another change to the already modified PercepPan model. This modification is based on [5] and consists in adding the input MS image of the model explained in Section 4.2.2.2 to the output of the generative network, which is adding the 8 bands of the upsampled MS image. That allows the network to easily preserve spectral and spatial information and avoid divergence while training. The modifications implemented on Section 4.2.2.2 are also implemented on this adaptation, while the reconstructor and the discriminative network remain unchanged too, so as on previous Section, they are the ones described in Section 4.2.2.1. Therefore, the generative network has the following structure:

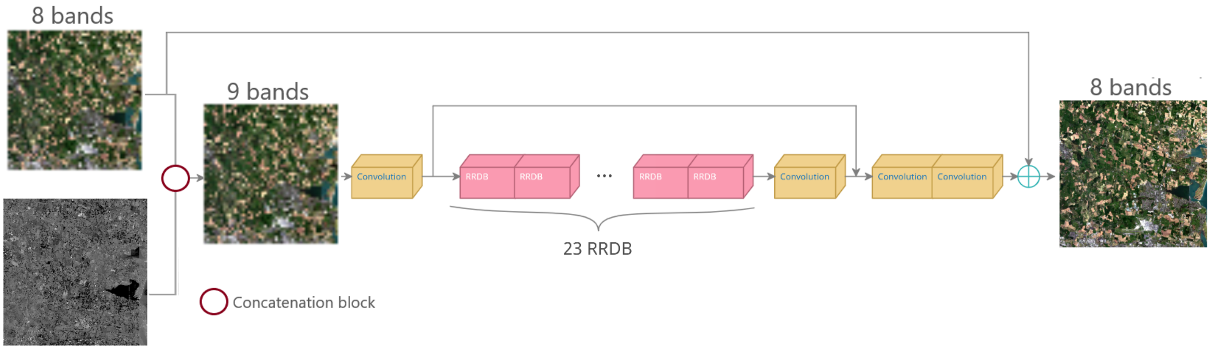


Figure 16: Generator model from PercepPan modification 2.

# 5 Experiments

## 5.1 Initialization

In our experiments, all the models are initialized with certain parameters.

### 5.1.1 Generative network

The generator is initialized from a pretrained model. This model is previously trained using the same dataset as we use for our experiments. Nevertheless, the model is only composed by the generative network of each respective used model in Section 4.2.2. Using only the generative networks instead of the GAN model involves same effects as mentioned in Section 2. This means that no ground truth of the same size as the output generator images is available. In consequence, images must be downsampled to be used as input images and the network output must be directly compared to the original image to compute the pixel loss. For this training, different parameters were tested, such as batch size and downsampled input image size. In addition, just the first image of the database (Figure 24) was used to create the patches for this model since the rest of the dataset was not available when these trainings were performed. In order to evaluate the models, some metrics were used based on the reference model that we have implemented [2]. Given that I corresponds to the original image and Î corresponds to the predicted image:

- PSNR, which measure the quality of the image and will be the main metric to decide which will be the final pretained model to use in each case:

$$PSNR(\hat{\mathrm{I}}, \mathrm{I}) = 10log_{10}\left(\frac{MAX_I}{RMSE(\hat{\mathrm{I}}, \mathrm{I})}\right), \tag{4}$$

  where $MAX_I$ is the maximum possible pixel value of I and RMSE(Î,I) corresponds to the root of mean squared error between Î and I.

- SSIM, which is used for measuring quality assessment:

$$SSIM(\hat{\mathrm{I}}, \mathrm{I}) = \tfrac{1}{C}\textstyle\sum_{c=1}^{C}\tfrac{1}{B}\sum_{i=1}^{B}\frac{(2\mu(\hat{\mathrm{p}}_c{}^{(i)})\mu(\mathrm{p}_c{}^{(i)})+c_1)\cdot(2\sigma(\hat{\mathrm{p}}_c{}^{(i)})\sigma(\mathrm{p}_c{}^{(i)})+c_2)\cdot(Cov(\hat{\mathrm{p}}_c{}^{(i)},\mathrm{p}_c{}^{(i)})+c_3)}{(\mu^2(\hat{\mathrm{p}}_c{}^{(i)})+\mu^2(\mathrm{p}_c{}^{(i)})+c_1)\cdot(\sigma^2(\hat{\mathrm{p}}_c{}^{(i)})+\sigma^2(\mathrm{p}_c{}^{(i)})+c_2)\cdot(\sigma(\hat{\mathrm{p}}_c{}^{(i)})+\sigma(\mathrm{p}_c{}^{(i)})+c_3)}, \tag{5}$$

  where C is the number of channels of the MS image (c referring to the $c^{th}$ channel). $\hat{\mathrm{I}}_c$ and $I_c$ are divided into b patches pairs $(\hat{\mathrm{p}}_c{}^{(i)}, \mathrm{p}_c{}^{(i)})$ up to B and $c_1 = (0.01MAX_I)^2$, $c_2 = (0.03MAX_I)^2$, and $c_3 = c_2/2$. [2]

- SAM, which measures the spectral distortion:

$$SAM(\hat{\mathrm{I}}, \mathrm{I}) = \frac{1}{HW}\sum_{i=1}^{H}\sum_{j=1}^{W} arccos\frac{\langle\hat{\mathrm{I}}_{i,j}, \mathrm{I}_{i,j}\rangle}{\|\hat{\mathrm{I}}_{i,j}\|\|\mathrm{I}_{i,j}\|}, \tag{6}$$

  where H and W correspond to the image height and width respectively and $\langle\cdot,\cdot\rangle$ is the inner product operator.

Models are pretrained for at most 100 epochs until the learning stabilizes. This stage is reached between epoch 20 and 80, depending on the model and the parameters. Figure 17 shows an example of the metrics and loss evolution through the whole training:

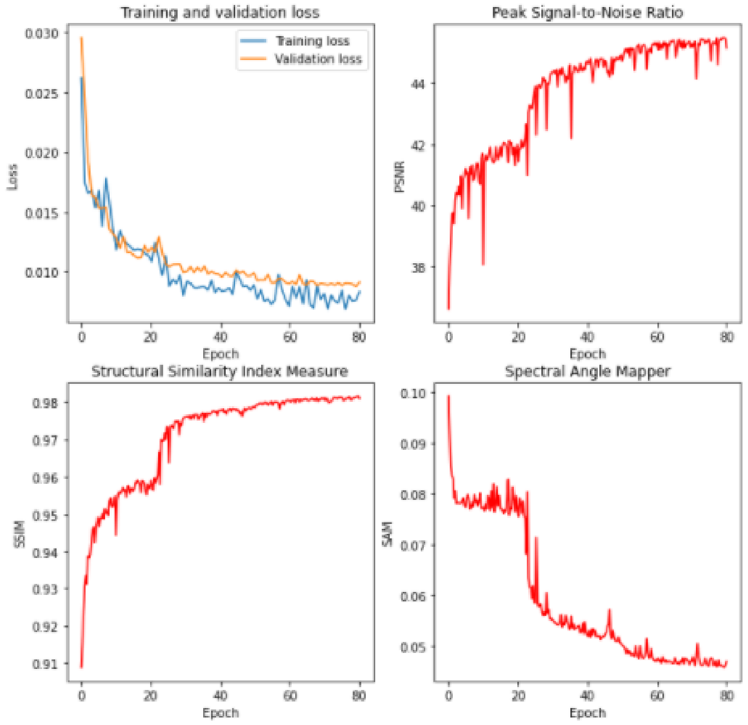


Figure 17: Pretraining on model from Section 4.2.2.1 evolution.

For all the models explained in Section 4.2.2, results had a similar pattern even with different results.

Table 3: Results of testing the generative network from model described in Section 4.2.2.1 pretrained in function of input patches size and batch size.

| **Input size** | **Batch size** | **PSNR ($\infty$)** | **SSIM (1)** | **SAM (0)** |
|---|---|---|---|---|
| 32x32 | 4 | 44.958 | 0.978 | 0.047 |
| 32x32 | 8 | 43.981 | 0.971 | 0.050 |
| **64x64** | **4** | **45.354** | **0.982** | **0.046** |
| 64x64 | 8 | 44.609 | 0.974 | 0.047 |
| 96x96 | 4 | 45.045 | 0.980 | 0.046 |
| 96x96 | 8 | 44.675 | 0.975 | 0.048 |

As it can be seen in Table 3, 64x64 MS downsampled input patches and a batch size of 4 obtain best performance for all 3 measures. That tendency is also observed for models described in Sections 4.2.2.2 and 4.2.2.3 Therefore, the 3 designed models used these parameters for the pretraining, obtaining the following results:

Table 4: Results of testing all pretrained models at input size 64×64 and batch size 4.

| **Model** | **PSNR ($\infty$)** | **SSIM (1)** | **SAM (0)** |
|---|---|---|---|
| **PercepPan** | 45.354 | 0.982 | 0.046 |
| **PercepPan mod. 1** | 48.873 | 0.987 | 0.043 |
| **PercepPan mod. 2** | 50.192 | 0.993 | 0.039 |

This results show that the modification implemented to [2] model allow for a better performance, at least for this pretraining. Nevertheless, it is important to mention that ”PercepPan modification 1” from Section 4.2.2.2 model diverged on early training, despite accomplishing good performance. Although measures differ from one model to another, visual results are similar. This pretrained models generate upsampled images with and added blurring, which is supposed to be corrected when using the GAN models. Figure 18 shows some examples of images generated by pretrained models:

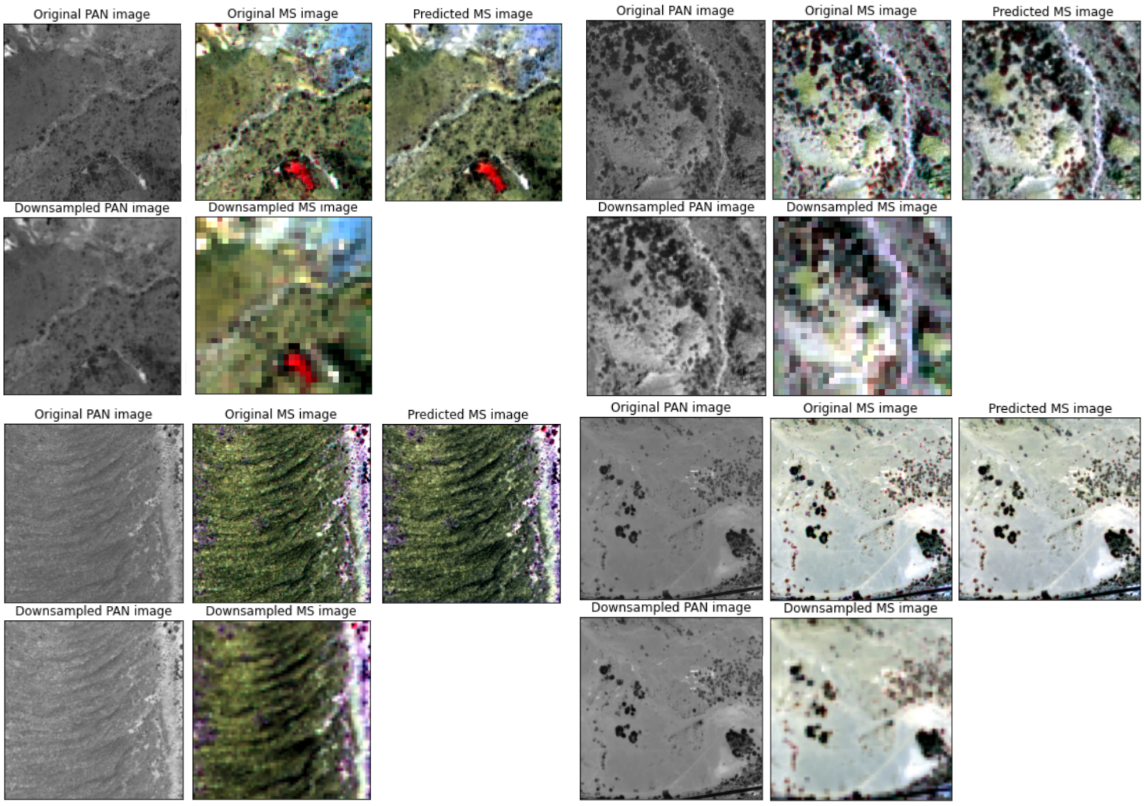


Figure 18: Pretraining of model described in Section 4.2.2.1 image results.

As it can be observed in Figure 18, downsampled images correspond to the network inputs, the predicted image is the output of the network and the original MS image is used as ground truth. In these results, the presence of blurring implies the loss of details, such as contours and tiny objects. The colors of the generated images are also paler. Training with our mentioned full models should improve this results, correcting blurring and improving the details.

### 5.1.2 Reconstructor network

As the 3 models of Section 4.2.2 use the same reconstructor network, all of them are initialized the same way. Network parameters are obtained through the method explained in Section 4.2.2.1 and parameter values are shown in Table 2.

### 5.1.3 Discriminative network

No parameters are previously computed to initialize the discriminative network. Therefore, it is initialized using Kaiming initialization, since it shows better performance than random initialization [29].

## 5.2 Training results

Each model is initialized using its respective generator model parameters mentioned in Section 5.1.1 and with reconstructor and discriminative networks as described in Sections 5.1.2 and 5.1.3 respectively.

For all trainings, full dataset with 64x64 of size patches are used. Nonetheless, it must be noted from Sections 4.2.2.2 and 4.2.2.3 that both models use upsampled 256×256 size patches as input even parting from the same dataset patches.

Since predicted MS images do not have the same size as original MS images, other metrics that allow comparing images of different size have to be used. Based on [2], these metrics are the following:

- $D_\lambda$, a spectral distortion index:

$$D_\lambda = \left( \frac{2}{C(C-1)} \sum_{c=1}^{C} \sum_{c^{'}>c}^{C} |Q({I_c}^{HRMS}, I_{c^{'}}^{HRMS}) - Q({I_c}^{LRMS}, I_{c^{'}}^{LRMS})|^u \right)^{\frac{1}{u}}, \qquad (7)$$

- $D_s$, a spatial distortion index:

$$D_s = \left( \frac{1}{C} \sum_{c=1}^{C} |Q({I_c}^{HRMS}, {I_c}^{PAN}) - Q({I_c}^{LRMS}, {I_c}^{LRPAN})|^v \right)^{\frac{1}{v}}, \qquad (8)$$

- QNR, a no-reference metric for image quality assessment:

$$QNR = (1 - D_\lambda)^a \cdot (1 - D_s)^b, \qquad (9)$$

where based on [2], we use u=v=1 and a=b=1 for our experiments in all cases. C is the number of channels of the MS image, while ${I_c}^{LRMS}$ corresponds to each channel of the original MS image, ${I_c}^{HRMS}$ corresponds to each channel of the predicted MS image, $I^{PAN}$ is the original PAN image and $I^{LRPAN}$ is a degraded version of the original PAN image. Q(.,.) stands for the Q-index metric, which gathers image contrast, luminance and structure for quality assessment:

$$Q(\hat{\mathrm{I}},\, \mathrm{I}) = \tfrac{1}{C}\sum_{c=1}^{C} \tfrac{1}{B}\sum_{i=1}^{B} \frac{(2\mu(\hat{\mathrm{p}}_c{}^{(i)})\mu(\mathrm{p}_c{}^{(i)}))\cdot(2\sigma(\hat{\mathrm{p}}_c{}^{(i)})\sigma(\mathrm{p}_c{}^{(i)}))\cdot(Cov(\hat{\mathrm{p}}_c{}^{(i)},\mathrm{p}_c{}^{(i)}))}{(\mu^2(\hat{\mathrm{p}}_c{}^{(i)})+\mu^2(\mathrm{p}_c{}^{(i)}))\cdot(\sigma^2(\hat{\mathrm{p}}_c{}^{(i)})+\sigma^2(\mathrm{p}_c{}^{(i)}))\cdot(\sigma(\hat{\mathrm{p}}_c{}^{(i)})+\sigma(\mathrm{p}_c{}^{(i)}))}'' \qquad (10)$$

where parameters have the same meaning as for equation 6.

During the training, the models achieve similar results as well, being able to wipe out most of the blurring that appeared on the previous pretraining phase described in Section 5.1.1. Although, some unexpected grid artifacts appear from the beginning of the training. Most of it disappears as the training progresses, but in image areas that contain less detailed objects these artifacts are still present.

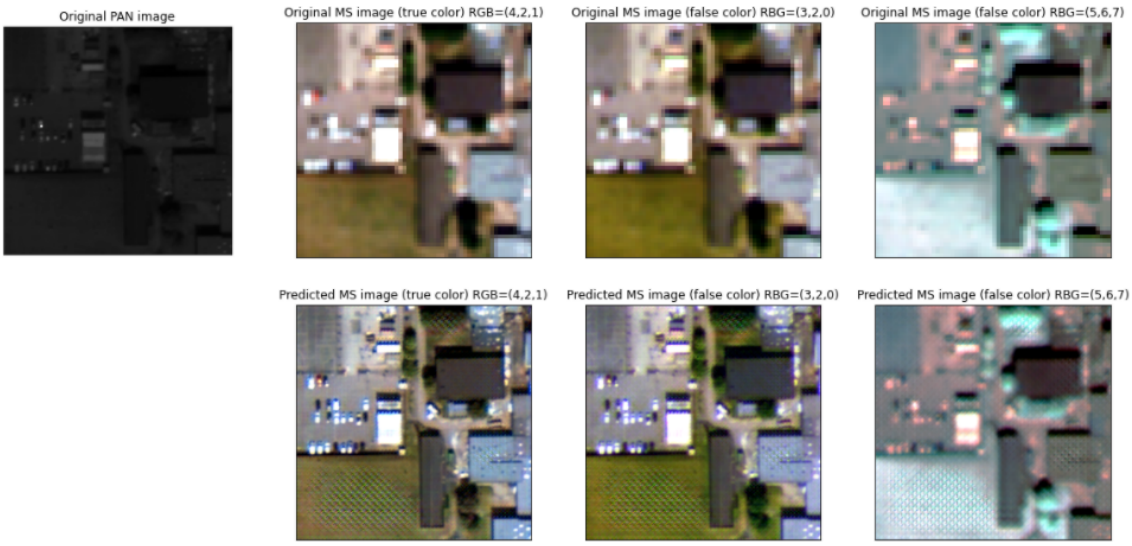

Original PAN image
Original MS image (true color) RGB=(4,2,1)
Original MS image (false color) RBG=(3,2,0)
Original MS image (false color) RBG=(5,6,7)
Predicted MS image (true color) RGB=(4,2,1)
Predicted MS image (false color) RBG=(3,2,0)
Predicted MS image (false color) RBG=(5,6,7)

Figure 19: Training on model of Section 4.2.2.3 image results before divergence is reached.

Figure 19 shows this event. Since MS image has 8 bands, all of them have been showed lumped together using groups of three (shown as RGB=(x,y,z), where x,y,z correspond to WV2 bands, interpreting Costal band as band 0). This is a method that will be used for the rest of Section 5. This results also show that details are improved but can be still enhanced.

Even though, most of the trainings reach divergence before results can be acceptable for our expectations. In most cases, this divergence is caused because pixels with higher values (white ones) tend to grow in value on the predicted image as the training progresses. Afterwards, the difference between these white pixels and the rest is so large that original colors are completely lost and the image starts distortioning as the iterations advance. Loss of color can be seen in Figure 20, where the dynamic range of the predicted image increases and triggers the color difference observed. The original colors cannot be recovered by normalizing the predicted image values since histograms peaks would be displaced and therefore we would generate lots of incorrect pixel values.

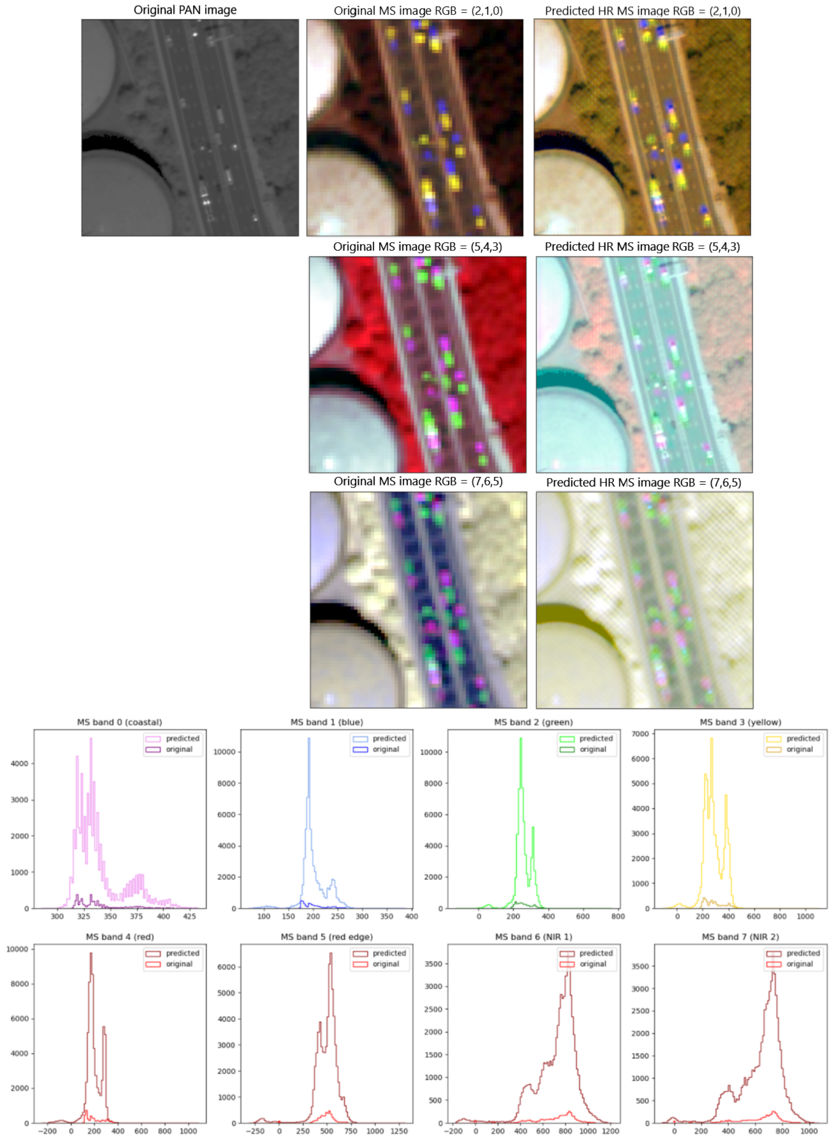


Figure 20: Image color prediction difference because of dynamic range expansion.

In order to solve this, many variations on training parameters have been tested. In first place, MultiStepLR scheduler [30] used in [2] was substituted for CosineAnnealingLR scheduler [31] using Tmax parameter at 1 epoch (corresponding to learning rate period) and ReduceLROnPlateau scheduler [32] with patience parameter at 2 epochs (that corresponds to the amount of epochs with no improvements after which the learning rate will be reduced). Traning without scheduler was also tested. CosineAnnealingLR scheduler accomplished better results because of reaching diverged slightly after the rest and having better metrics before that happens.

Learning rate was also reduced, but even reaching further epochs, results at that number of epochs were the same as for the previous trainings. Batch size could not be increased to 4 because of hardware limitations, and decreasing it to 1 did not get any good result.

As commented in Section 5.1.1, model of Section 4.2.2.2 diverged faster than the rest while pretraining the model. When performing this experiments, this model diverged even before eliminating blurring from pretraining and did not succeed on generating any comparable results to the other 2 models.

Despite the unexpected results on training, test where done to compare visual results and metrics between each model best performance.

## 5.3 Test results

All models have been tested, obtaining different image results and metrics. As stated in section 5.2, every model has started to diverge rapidly while training so this test results have been applied to the model states previous to the divergence. Some image results can be seen in Figure 21:

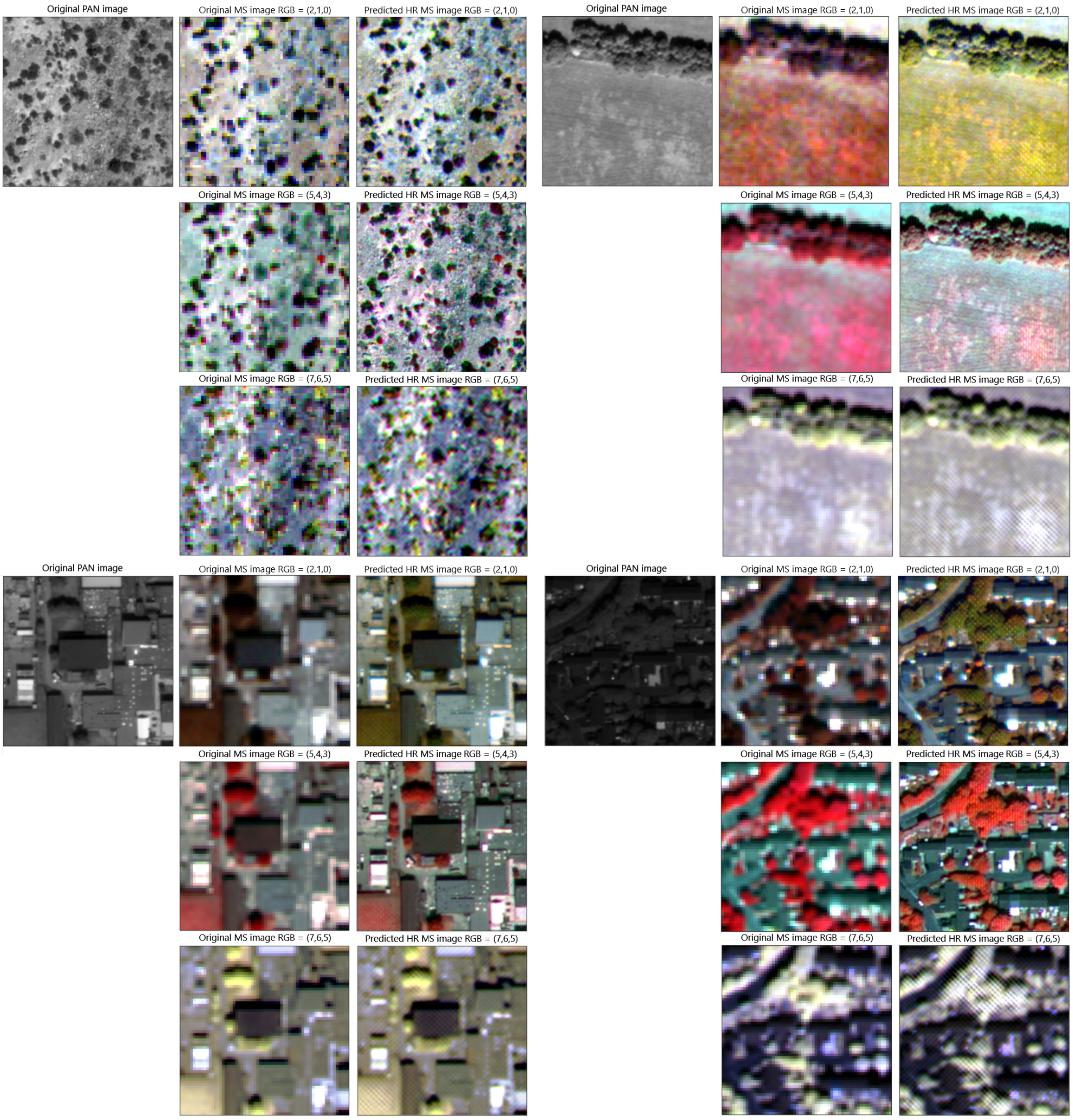

Original PAN image
Original MS image RGB = (2,1,0)
Predicted HR MS image RGB = (2,1,0)
Original PAN image
Original MS image RGB = (2,1,0)
Predicted HR MS image RGB = (2,1,0)
Original MS image RGB = (5,4,3)
Predicted HR MS image RGB = (5,4,3)
Original MS image RGB = (5,4,3)
Predicted HR MS image RGB = (5,4,3)
Original MS image RGB = (7,6,5)
Predicted HR MS image RGB = (7,6,5)
Original MS image RGB = (7,6,5)
Predicted HR MS image RGB = (7,6,5)
Original PAN image
Original MS image RGB = (2,1,0)
Predicted HR MS image RGB = (2,1,0)
Original PAN image
Original MS image RGB = (2,1,0)
Predicted HR MS image RGB = (2,1,0)
Original MS image RGB = (5,4,3)
Predicted HR MS image RGB = (5,4,3)
Original MS image RGB = (5,4,3)
Predicted HR MS image RGB = (5,4,3)
Original MS image RGB = (7,6,5)
Predicted HR MS image RGB = (7,6,5)
Original MS image RGB = (7,6,5)
Predicted HR MS image RGB = (7,6,5)

Figure 21: Examples of image results taken from Section 4.2.2.1 model test.

As it can be seen in Figure 21, first bands perceptual view is better than in last bands, this can be due to perceptual loss only taking into account 3 channels than can be perceived by the human eye. For our case, feature extractors work on Red, Green and Blue bands (which for WorldView-2 correspond to bands 4, 2 and 1 respectively if we count the Coastal band as band 0). Yellow is excluded so the chosen bands seem more likely to the bands chosen by other researchers from the literature, since other satellites do not provide a Yellow band.

All images contain the grid-style artifacts that appeared during the training phase. This kind of effect are common when using GAN [33, 34]. In addition, results seems to perform worse on bleached images or around white pixels. This are symptoms previously analyzed on Section 5.2, so the best training states saved for each model, despite having best results, they have some evidences of distortion because of divergence inception.

Depending on the model, metrics obtained by these tests are the following:

Table 5: Results of testing all models at best training state accomplished.

| **Model** | **D$_\lambda$ (0)** | **D$_s$ (0)** | **QNR (1)** |
|---|---|---|---|
| **PercepPan** | 0.101 | 0.122 | 0.791 |
| **PercepPan mod. 1** | 0.307 | 0.342 | 0.457 |
| **PercepPan mod. 2** | 0.086 | 0.105 | 0.819 |

As is can be seen in Table 5, models from Sections 4.2.2.1 (PercepPan) and 4.2.2.3 (PercepPan mod. 2) outperform Section 4.2.2.2 (PercepPan mod. 1) model on metrics too. Moreover, our own designed model accomplishes slightly better results than the model adapted from [2], despite no visual difference can be perceived.

The measured metrics also support the visual results acquired from each model:

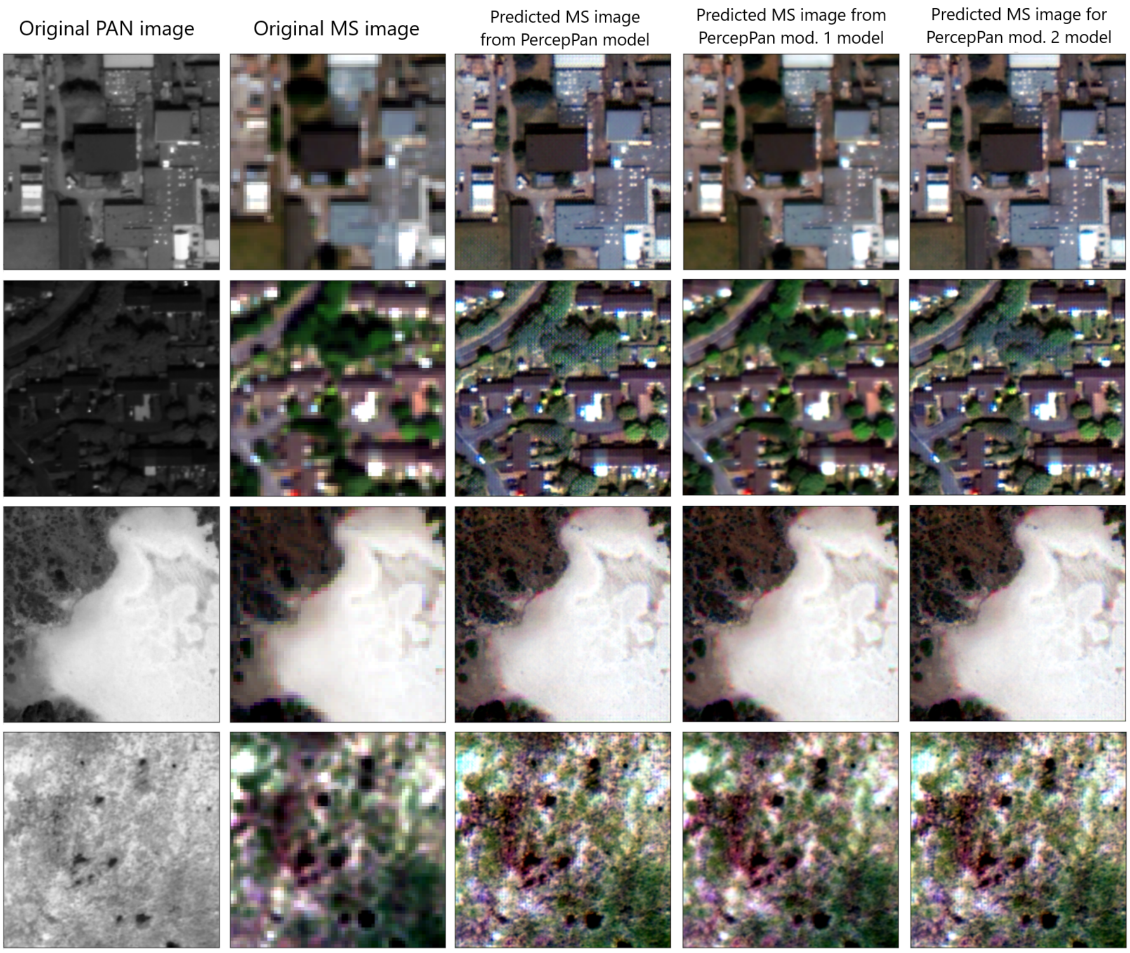

Original PAN image
Original MS image
Predicted MS image from PercepPan model
Predicted MS image from PercepPan mod. 1 model
Predicted MS image for PercepPan mod. 2 model

Figure 22: Test images at true color comparison between models.

According to Figure 22, models that obtain better metrics also achieve better perceptual results.

In addition, following [25] methodology, an equalized model has been designed as a linear combination of each pretrained model and its respective adversarial training model. This linear combination is not done on the image results but on the model parameters following next equation:

$$M_{eq} = M_G \cdot \alpha + M_{GAN} \cdot (1 - \alpha), \tag{11}$$

where $M_{eq}$ is the equalized model created from the linear combination of parameters from pretrained model $M_G$ and adversarial trained model $M_{GAN}$. The results are calculated from $\alpha$=0 to $\alpha$=1 in steps of 0.1.

Each equalized model is also tested to compute its own metrics.

Table 6: Results of testing all equalized models for different values of $\alpha$. Best models and best results for each metric are marked on red.

| $\alpha$ | **Model** | **D$_\lambda$ (0)** | **D$_s$ (0)** | **QNR (1)** |
|---|---|---|---|---|
| | **PercepPan** | 0.101 | 0.122 | 0.791 |
| **0.0** | **PercepPan mod. 1** | 0.307 | 0.342 | 0.457 |
| | **PercepPan mod. 2** | 0.086 | 0.105 | 0.819 |
| | **PercepPan** | 0.098 | 0.122 | 0.793 |
| **0.1** | **PercepPan mod. 1** | 0.256 | 0.295 | 0.525 |
| | **PercepPan mod. 2** | 0.084 | 0.099 | 0.825 |
| | **PercepPan** | 0.095 | 0.121 | 0.797 |
| **0.2** | **PercepPan mod. 1** | 0.239 | 0.287 | 0.543 |
| | **PercepPan mod. 2** | 0.084 | 0.097 | 0.828 |
| | **PercepPan** | 0.0926 | 0.119 | 0.801 |
| **0.3** | **PercepPan mod. 1** | 0.205 | 0.216 | 0.623 |
| | **PercepPan mod. 2** | 0.080 | 0.094 | 0.834 |
| | **PercepPan** | 0.090 | 0.114 | 0.808 |
| **0.4** | **PercepPan mod. 1** | 0.186 | 0.194 | 0.657 |
| | **PercepPan mod. 2** | 0.078 | 0.091 | 0.838 |
| | **PercepPan** | 0.086 | 0.107 | 0.817 |
| **0.5** | **PercepPan mod. 1** | 0.164 | 0.172 | 0.692 |
| | **PercepPan mod. 2** | 0.075 | 0.088 | 0.844 |
| | **PercepPan** | 0.082 | 0.099 | 0.829 |
| **0.6** | **PercepPan mod. 1** | 0.142 | 0.166 | 0.716 |
| | **PercepPan mod. 2** | 0.073 | 0.085 | 0.848 |
| | **PercepPan** | 0.079 | 0.091 | 0.839 |
| **0.7** | **PercepPan mod. 1** | 0.120 | 0.153 | 0.745 |
| | **PercepPan mod. 2** | 0.071 | 0.080 | 0.855 |
| | **PercepPan** | 0.078 | 0.091 | 0.840 |
| **0.8** | **PercepPan mod. 1** | 0.118 | 0.149 | 0.751 |
| | **PercepPan mod. 2** | 0.072 | 0.078 | 0.856 |
| | **PercepPan** | 0.077 | 0.101 | 0.831 |
| **0.9** | **PercepPan mod. 1** | 0.113 | 0.149 | 0.755 |
| | **PercepPan mod. 2** | 0.076 | 0.083 | 0.847 |
| | **PercepPan** | 0.078 | 0.111 | 0.822 |
| **1.0** | **PercepPan mod. 1** | 0.111 | 0.143 | 0.762 |
| | **PercepPan mod. 2** | 0.080 | 0.085 | 0.842 |

The 3 models proposed follow a similar tendency on metrics as $\alpha$ varies. Best results are accomplished around $\alpha$=0.8 for ”PercepPan” model and ”PercepPan modified 2” models, while ”PercepPan modified 1” model achieves best results when only using the pretrained model. This means that the influence of the pretrained model is higher than the adversarial model when acquiring best metrics. Our own design model stands as the model with best results accomplished during all the experiments that were executed during this project.

These results can also be analyzed visually using generated image by each equalized model:

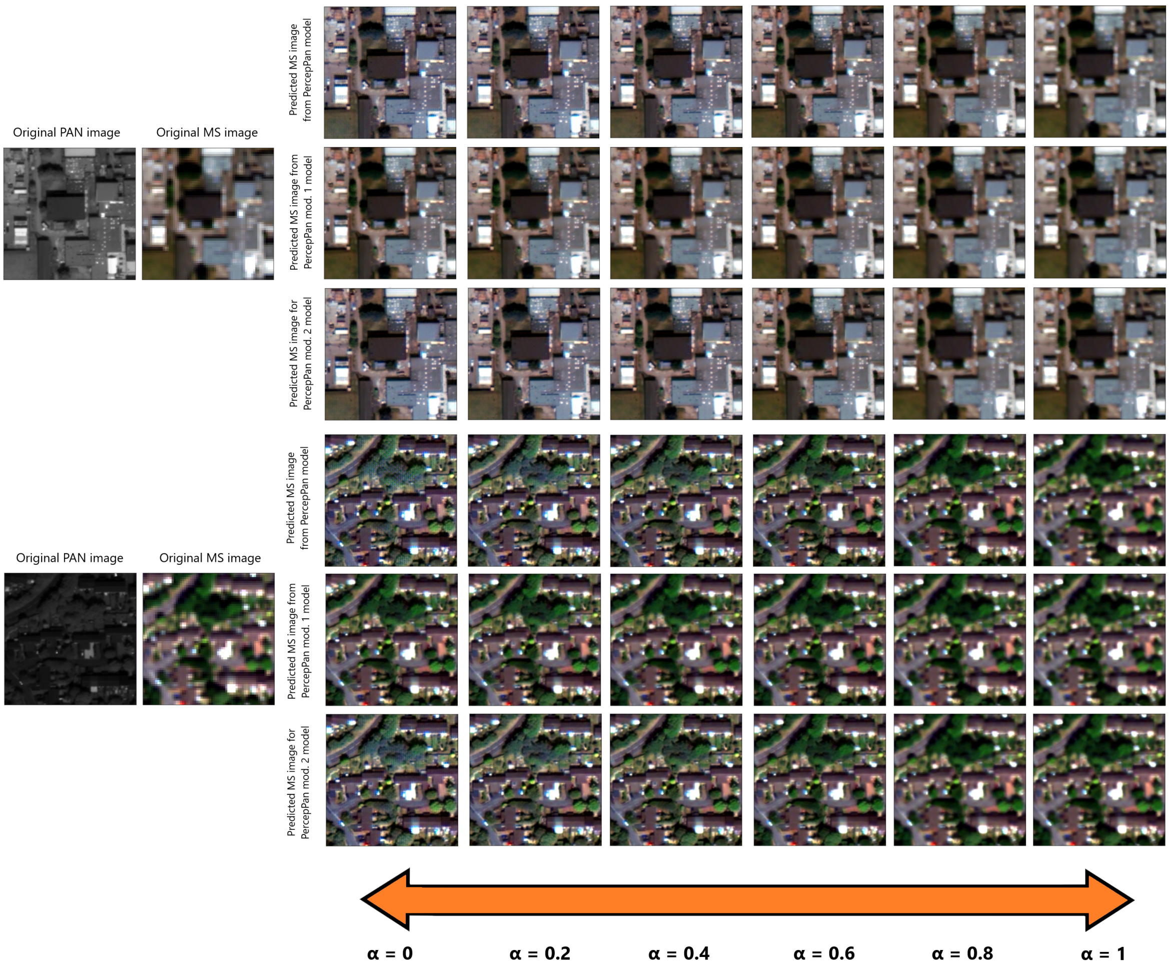

Original PAN image
Original MS image
Predicted MS image from PercepPan model
Predicted MS image from PercepPan mod. 1 model
Predicted MS image for PercepPan mod. 2 model
Original PAN image
Original MS image
Predicted MS image from PercepPan model
Predicted MS image from PercepPan mod. 1 model
Predicted MS image for PercepPan mod. 2 model
α = 0
α = 0.2
α = 0.4
α = 0.6
α = 0.8
α = 1

Figure 23: Test images comparison between models depending on $\alpha$ value.

In this case, models with best metrics do not achieve the best perceptual results. Equalized models around $\alpha$=0.4 are the ones accomplishing best visual results. Just as in [25], the implementation of equalization on trained models was able to erase those artifacts that were appearing while training. Since the pretrained model is a PSNR-oriented model that is specialized on accomplishing better results on parts of the image with less contours, and the GAN-based trained model is a perceptual-driven model specialized on performing better on image details while worsen the rest, this results were expected to happen. This results are not still as good as the results shown by different researchers from the literature such as [2, 5], but the elimination of those artifacts implies that the results are much more acceptable than the ones shown on Section 5.2.

# 6 Budget

To develop this project, multiple hardware and software have been used. Despite not being able to know the exact cost of the material and the salary of the personnel that collaborated during the evolution of the project, an estimated cost can be measured. First of all, in order to do some tasks locally, a personal computer has been used. Its cost has been estimated to 12 €/month, since it has an approximate initial cost of 1000 € and an expected life-time of 7 years. Moreover, most of the computational tasks where done remotely. Since the cost of the server usage can't be estimated properly, a 50 €/month has been assigned to its maintenance. Other fungibles like electricity or Ethernet have been given a total cost of 10 €/month. Besides that, some of the software that has been used also have costs because of the licenses of usage. Thus, although SNAP is a free-to-use software, ENVI has a cost of 220€ annual license fee. The dataset that has been used also has a cost, even though we have utilized it for free since all imagery was given at no cost. Taking into account [35], a WorldView2 pair of MS and PAN images has a price of 24 €/km2 (30% price reduction for academic usage is not discounted). The database contains a total amount of:

Table 7: Imagery cost computation

| Image | Pixel resolution | Pixel area | Number of pixels | Image area | Price |
|---|---|---|---|---|---|
| **Teide** | $1.3m$ | $1.69m^2$ | $16,777,216$ | $28.35km^2$ | 680.40 € |
| **Dublin** | $1.6m$ | $2.56m^2$ | $129,717,720$ | $332.08km^2$ | 7,969.92 € |
| **W'ton rural** | $1.6m$ | $2.56m^2$ | $145,358,802$ | $372.12km^2$ | 8,930.88 € |
| **W'ton urban** | $1.6m$ | $2.56m^2$ | $124,661,760$ | $319.13km^2$ | 7,659.12 € |
| **Riga** | $1.6m$ | $2.56m^2$ | $105,854,036$ | $270.99km^2$ | 6,503.76 € |
| **Total** | | | | | **31,744.08 €** |

In last place, the salary of the personnel involved in the project is estimated from the standard salary of a junior engineer (15 €/hour), a senior engineer (20 €/hour) and a technical advisor (30 €/hour). Since the project has a total duration of 4 months, the budget of the project is estimated as follows:

Table 8: Total cost computation.

| **Item** | **Cost** | **Time of usage** | **Total cost** |
|---|---|---|---|
| Personal computer | 12 €/month | 4 months | 48.00 € |
| Server hardware | 50 €/month | 4 months | 200.00 € |
| Fungibles | 10 €/month | 4 months | 10.00 € |
| ENVI license | 200 €/year | 1 year | 220.00 € |
| Database | 31,744.08 € | – | 31,744.08 € |
| Junior engineer | 15 €/hour | 4 months (15 hours/week) | 3,600.00 € |
| Senior engineer | 20 €/hour | 4 months (5 hours/week) | 1,600.00 € |
| Technical advisor | 30 €/hour | 4 months (3 hours/week) | 1,440.00 € |
| **Total** | | | **38,892.08 €** |

As Table 8 shows, the total budget of the project is about 32,512.08 €

# 7 Conclusions and future development

Final results did not accomplish our initial expectations, since the generated images by all 3 models had not enough quality compared to the results from literature. It must be said that the pretraining phase did achieve our expectations, not only generating the expected results according to literature but getting over the stated metrics on Section 2 by [2], despite the differences on the imagery characteristics (8 bands instead of 4). Nevertheless, this pretraining results did not help enough to overcome literature experiments. Equalization served as a good solution to solve issues presented on the training phase though, taking advantage of the pretraining results as well.

Hardware limitations, large training periods and lack of knowledge on the earliest phases of the project, which delayed the rest of the tasks, did not benefit to accomplish better results.

Despite the results, it has to be said that I, personally, did learn a lot from mistakes taken during the project development, and will be able to avoid them in future projects. I enjoyed the experience of working on a project of that kind, which field I was interested on but did not have a clear idea of how working on a Deep Learning, or more precisely, on a pan-sharpening project was.

For future development, models of Sections 4.2.2.1 and 4.2.2.3 still have the possibility of reaching the expected results since the literature covered in Section 2 demonstrates that our models are based on concepts that can produce good perceptual results, even if not being able to get over the rest of results presented in the literature. To do so, they need more time to be trained using a bigger diversity of parameters, because GANs, as mentioned in Section 3.2, are very sensitive to that kind of changes. Therefore, taking advantage of what was tested on our experiments and applying a few more changes or improvements, this models will accomplish their purpose for sure. To help in future development, code used for the experiments is available at author's Github homepage (`https://github.com/miqueljc/Pan-sharpening_experiments`)

# References

[1] James Dietrich. New off nadir resolution calculator. Available online: `http://adv-geo-research.blogspot.com/2016/08/,`. (accessed on 19 June 2021).

[2] Changsheng Zhou, Jiangshe Zhang, Junmin Liu, Chunxia Zhang, Rongrong Fei, and Shuang Xu. Perceppan: Towards unsupervised pan-sharpening based on perceptual loss. *Remote Sensing*, 12(14), 2020.

[3] USGS. What is remote sensing and what is it used for? Available online: `https://www.usgs.gov/faqs/what-remote-sensing-and-what-it-used?qt-news_science_products=0#qt-news_science_products`. (accessed on 15 June 2021).

[4] CIMSS. Remote sensing - spatial analysis. Available online: `https://cimss.ssec.wisc.edu/sage/remote_sensing/lesson3/concepts.html`. (accessed on 18 June 2021).

[5] Junfeng Yang, Xueyang Fu, Yuwen Hu, Yue Huang, Xinghao Ding, and John Paisley. Pannet: A deep network architecture for pan-sharpening. pages 1753–1761, 10 2017.

[6] Alan R Gillespie, Anne B Kahle, and Richard E Walker. Color enhancement of highly correlated images. ii. channel ratio and “chromaticity” transformation techniques. *Remote Sensing of Environment*, 22(3):343–365, 1987.

[7] Nikos Koutsias, Michael Karteris, and Emilio Chuvieco. The use of intensity-hue-saturation transformation of landsat5 thematic mapper data for burned land mapping. *Photogrammetric Engineering and Remote Sensing*, 66:829–839, 07 2000.

[8] Min Guo, Hongyan Zhang, Jiayi Li, Liangpei Zhang, and Huanfeng Shen. An online coupled dictionary learning approach for remote sensing image fusion. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(4):1284–1294, 2014.

[9] Xiao Xiang Zhu and Richard Bamler. A sparse image fusion algorithm with application to pan-sharpening. *IEEE Transactions on Geoscience and Remote Sensing*, 51(5):2827–2836, 2013.

[10] X. Otazu, M. Gonzalez-Audicana, O. Fors, and J. Nunez. Introduction of sensor spectral response into image fusion methods. application to wavelet-based methods. *IEEE Transactions on Geoscience and Remote Sensing*, 43(10):2376–2385, 2005.

[11] Muhammad Murtaza Khan, Jocelyn Chanussot, Laurent Condat, and Annick Montanvert. Indusion: Fusion of multispectral and panchromatic images using the induction scaling technique. *IEEE Geoscience and Remote Sensing Letters*, 5(1):98–102, 2008.

[12] Giuseppe Masi, Davide Cozzolino, Luisa Verdoliva, and Giuseppe Scarpa. Pansharpening by convolutional neural networks. *Remote Sensing*, 8(7), 2016.

[13] Zhenfeng Shao and Jiajun Cai. Remote sensing image fusion with deep convolutional neural network. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 11(5):1656–1669, 2018.

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[15] Li Deng and Dong Yu. Deep learning: Methods and applications. Technical Report MSR-TR-2014-21, Microsoft, May 2014.

[16] European Space Agency. Worldview2. Available online: `https://earth.esa.int/eogateway/missions/worldview-2`. (accessed on 18 June 2021).

[17] DigitalGlobe. Spectral response for digitalglobe earth imaging instruments. Available online: `https://dg-cms-uploads-production.s3.amazonaws.com`. (accessed on 18 June 2021).

[18] Instituto de Oceanografía y Cambio Global de la Universidad de Las Palmas de Gran Canaria (ULPGC). Proyecto artemisat. Available online: `https://earth.esa.int/eogateway/missions/worldview-2`. (accessed on 18 June 2021).

[19] European Space Agency. Worldview-2 european cities. Available online: `https://earth.esa.int/eogateway/catalog/worldview-2-european-cities`. (accessed on 18 June 2021).

[20] L3Harris Geospatial. Envi. Available online: `https://www.l3harrisgeospatial.com/Software-Technology/ENVI`. (accessed on 16 June 2021).

[21] European Space Agency. Snap. Available online: `https://step.esa.int/main/toolboxes/snap/`. (accessed on 16 June 2021).

[22] Seungjun Nah, Tae Kim, and Kyoung Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. 12 2016.

[23] Distill. Deconvolution and checkerboard artifacts. Available online: `https://distill.pub/2016/deconv-checkerboard/`. (accessed on 18 June 2021).

[24] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2472–2481, 2018.

[25] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Chen Change Loy, Yu Qiao, and Xiaoou Tang. Esrgan: Enhanced super-resolution generative adversarial networks, 2018.

[26] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization, 2017.

[27] Junmin Liu, Jing Ma, Rongrong Fei, Huirong Li, and Jiangshe Zhang. Enhanced back-projection as postprocessing for pansharpening. *Remote Sensing*, 11(6), 2019.

[28] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015.

[30] Catalist. Source code for torch.optim.lr_scheduler. Available online: `https://catalyst-team.github.io/catalyst/v20.07/_modules/torch/optim/lr_scheduler.html`. (accessed on 17 June 2021).

[31] Huosheng Xie and Zesen Wu. A robust fabric defect detection method based on improved refinedet. *Sensors*, 20:4260, 07 2020.

[32] Keras. Reducelronplateau. Available online: `https://keras.io/api/callbacks/reduce_lr_on_plateau/,`. (accessed on 17 June 2021).

[33] Francesco Marra, Diego Gragnaniello, Luisa Verdoliva, and Giovanni Poggi. Do gans leave artificial fingerprints?, 2018.

[34] Xu Zhang, Svebor Karaman, and Shih-Fu Chang. Detecting and simulating artifacts in gan fake images. In *2019 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6, 2019.

[35] Woldwide Mapping LLC. Satellite imagery pricing, 5 2016. Available online: `http://www.ict-agri.eu/system/files/dow/LANDINFO%20Satellite%20Imagery%20Pricing.pdf`. (accessed on 12 June 2021).

# Appendices

## A Dataset Images

### Image 1

Location: Teide (Spain)
Geographical central coordinates: 28°12'42.5"N 16°37'17.0"W
Date: $13^{th} June 2017$ at $12:16$
Image resolution: 1.3 m
Image size: $4,096 \times 4,096$

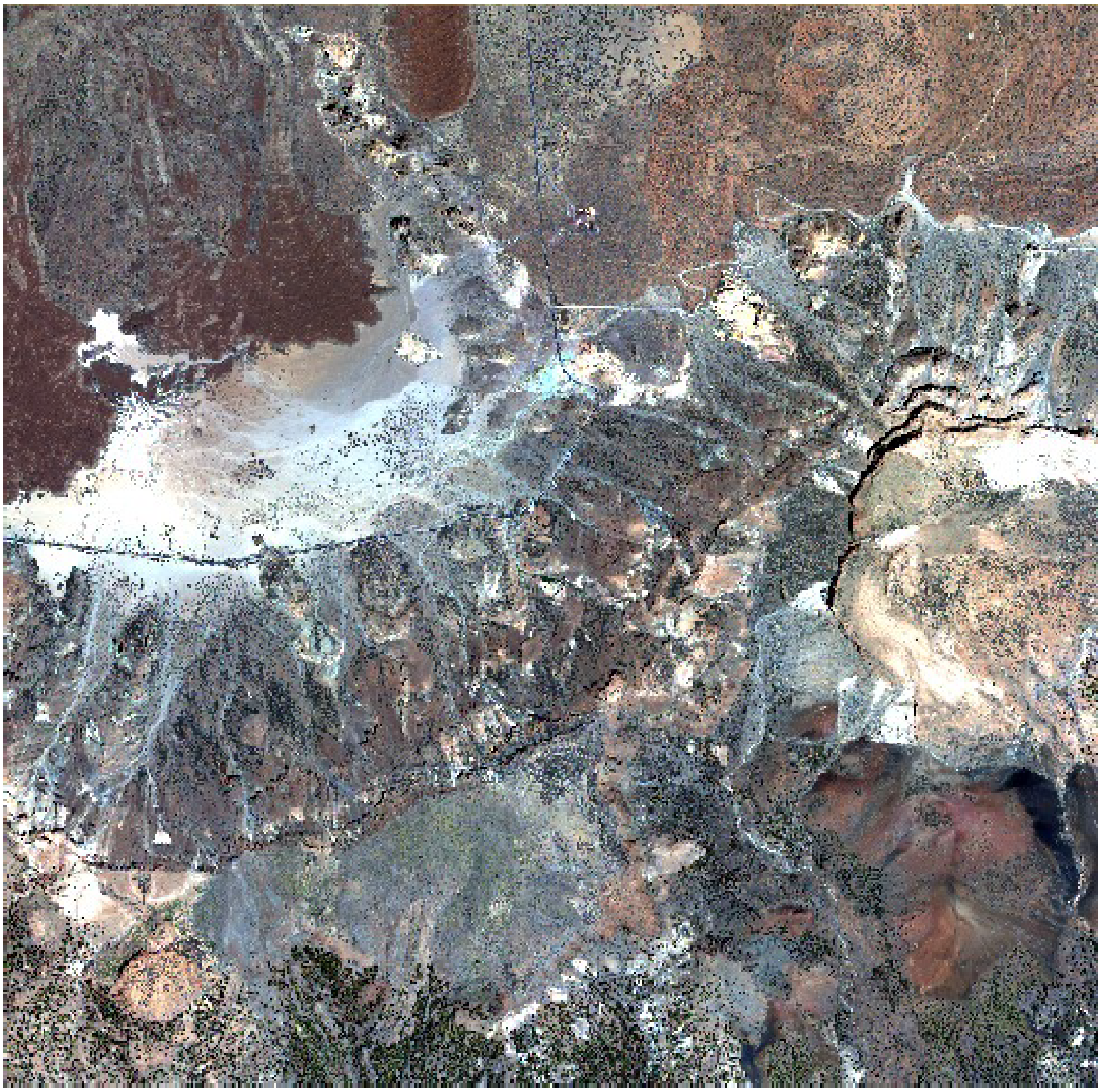


Figure 24: Teide (Spain) image.

## Image 2

Location: Dublin (Ireland)<br>
Geographical central coordinates: 53°29'20.4"N 6°16'12.0"W<br>
Date: $21^{th} April 2015$ at $12:08$<br>
Image resolution: 1.6 m<br>
Image size: $11,838 \times 12,279$

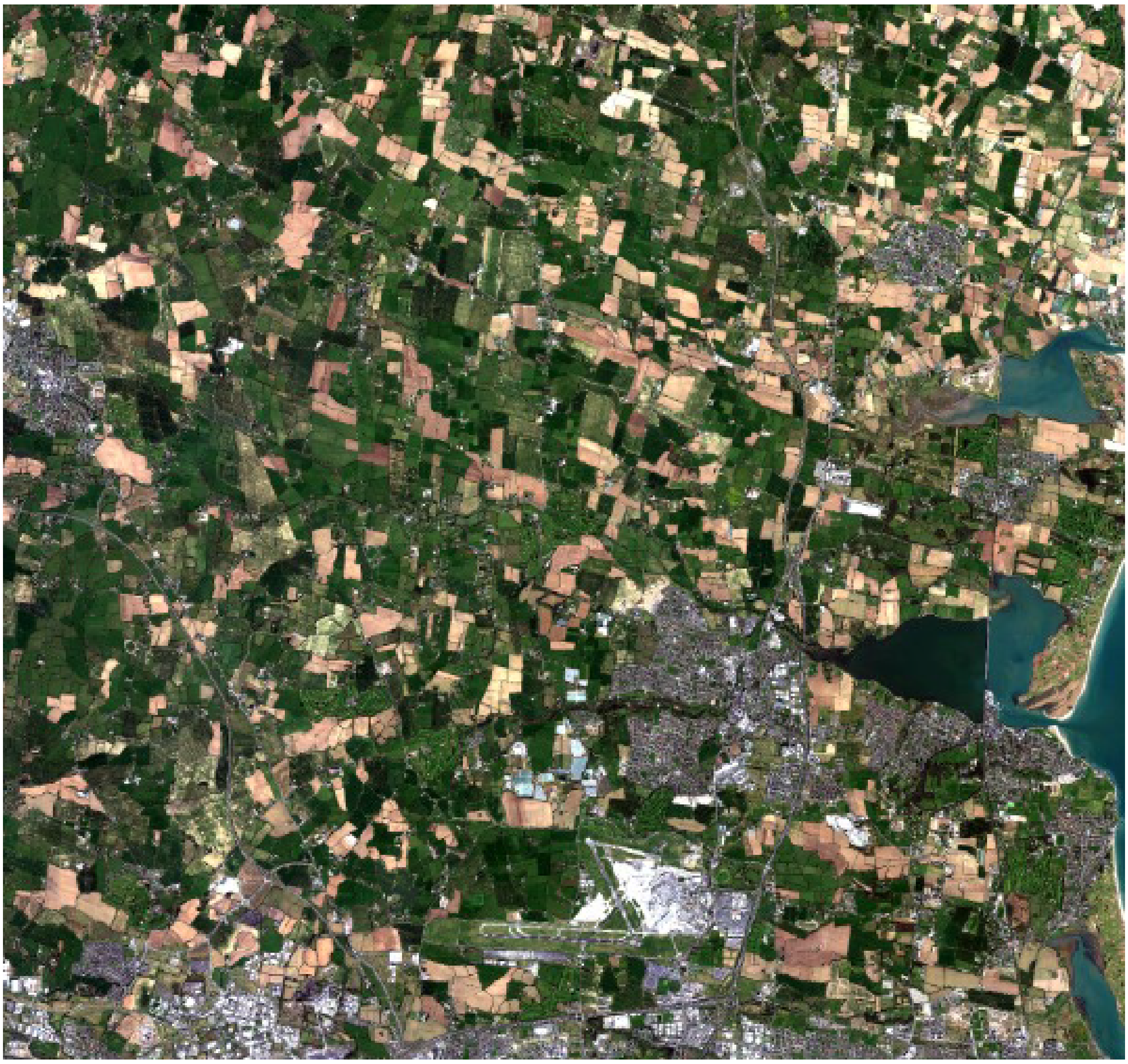


Figure 25: Dublin (Ireland) image.

## Image 3

Location: Wolverhampton (United Kingdom)
Geographical central coordinates: 52°37'33.6"N 2°19'19.2"W
Date: $18^{th} July 2013$ at $11 : 43$
Image resolution: 1.6 m
Image size: $12,288 \times 10,145$

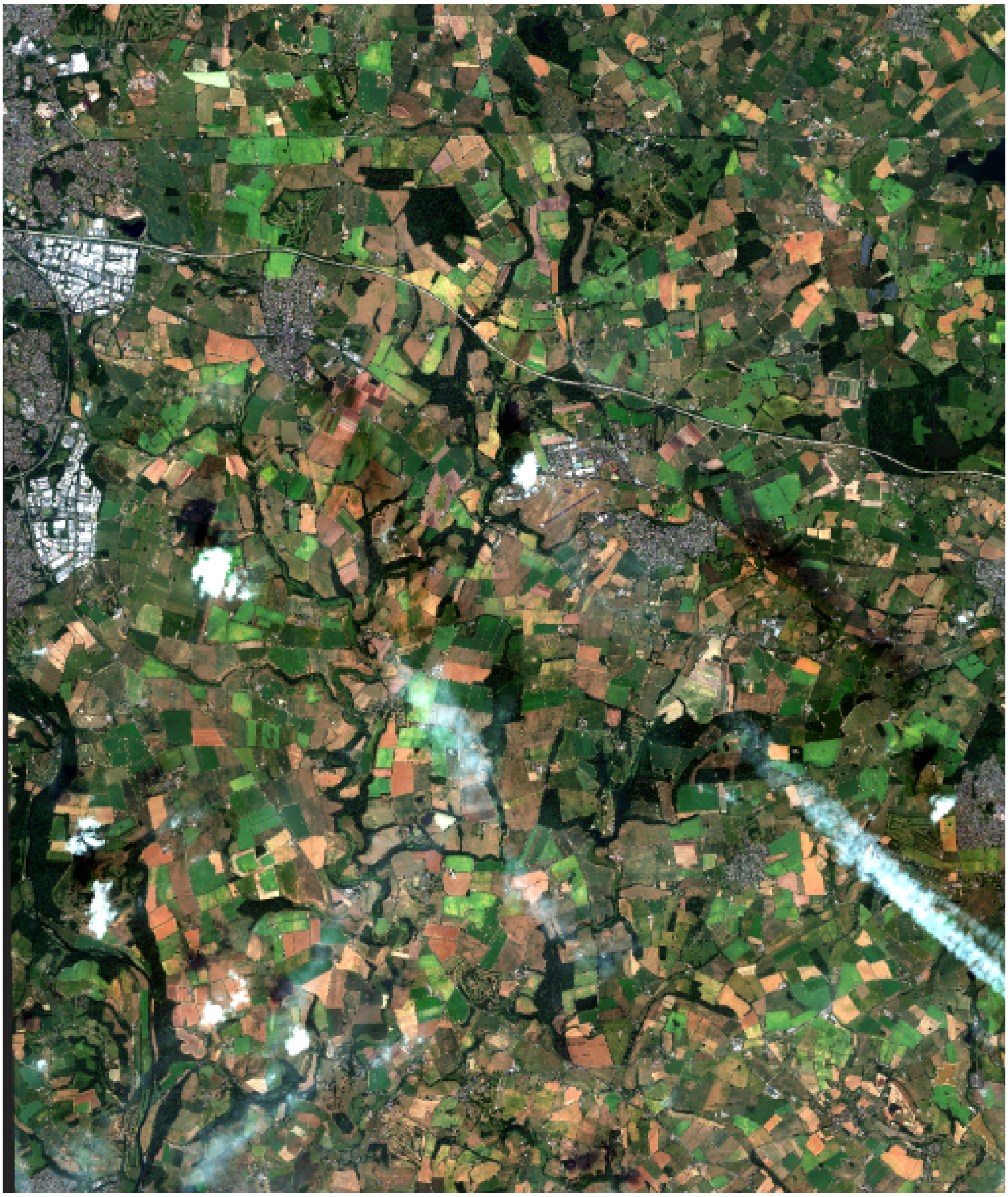


Figure 26: Wolverhampton (UK) first image.

## Image 4

Location: Wolverhampton (United Kingdom)
Geographical central coordinates: 52°38’13.2”N 2°05’09.6”W
Date: $18^{th} July 2013$ at $11:43$
Image resolution: 1.6 m
Image size: $12,288 \times 10,065$

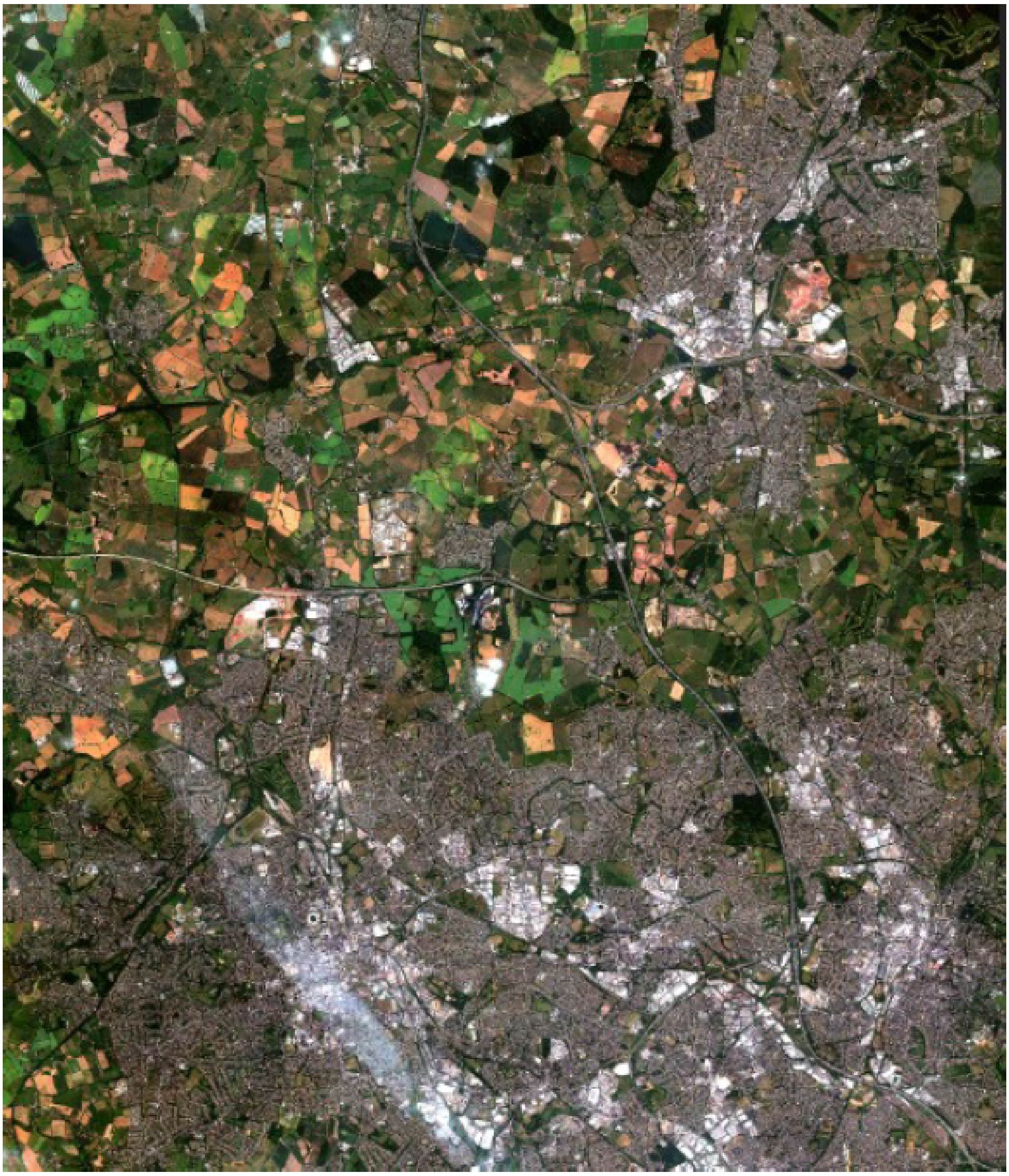


Figure 27: Wolverhampton (UK) second image.

## Image 5

Location: Riga (Latvia)
Geographical central coordinates: 56°43’30.0”N 25°09’21.6”E
Date: $2^{nd} May 2012$ at $09:52$
Image resolution: 1.6 m
Image size: $9,868 \times 10,727$

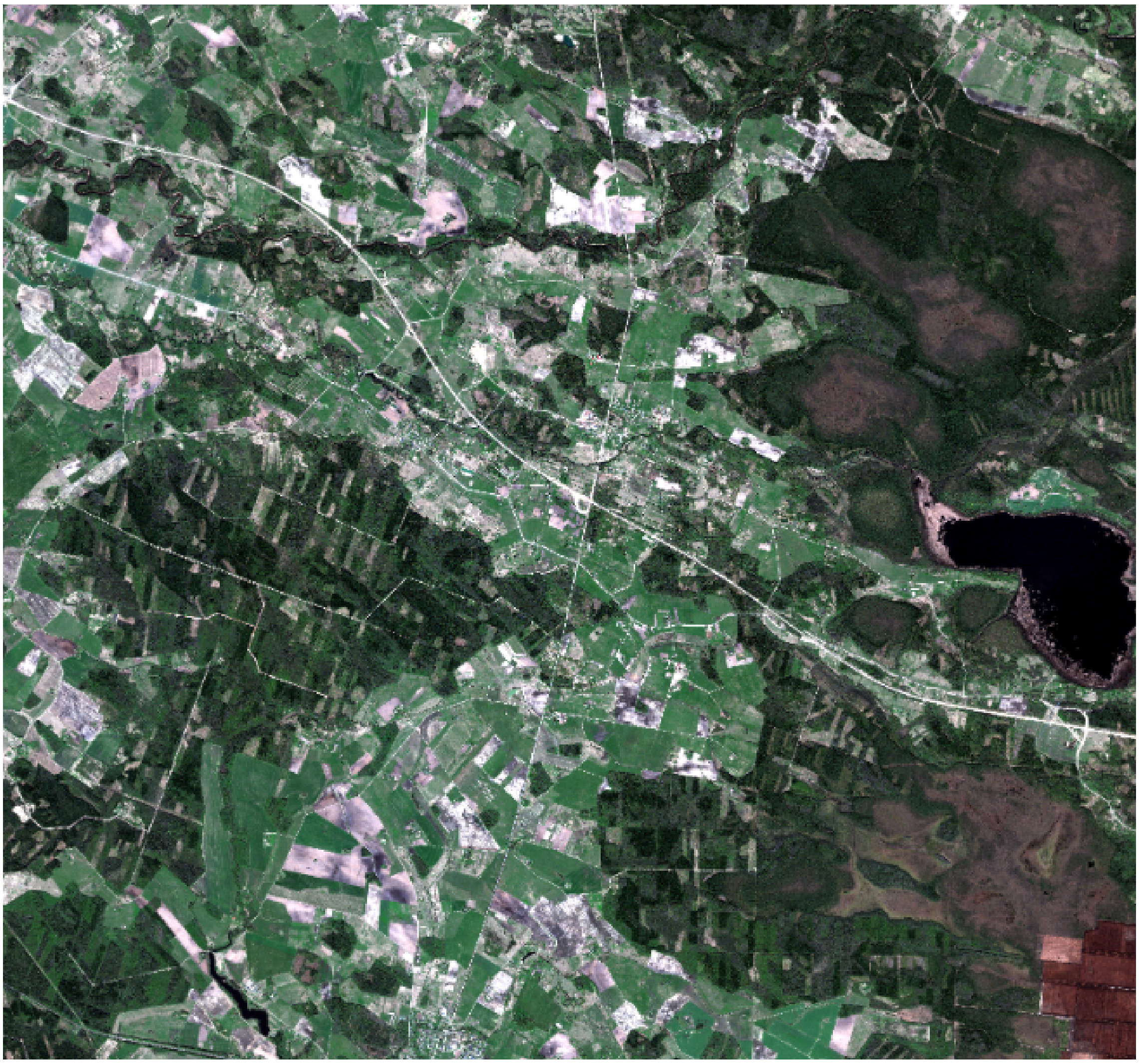


Figure 28: Riga (Latvia) image.