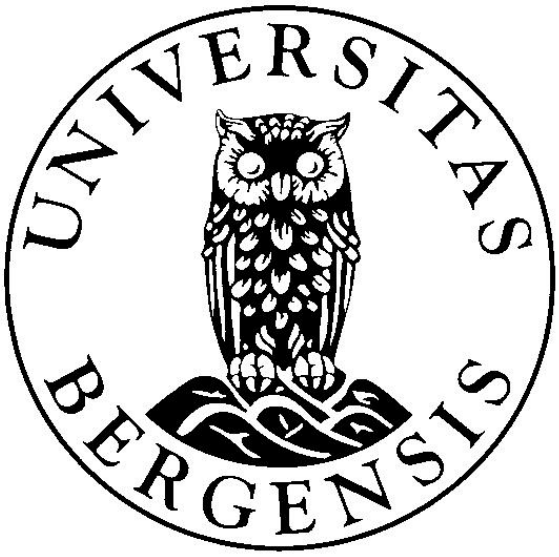
Deep learning-based cross-sensor super resolution of satellite images

Multispectral-to-panchromatic single-image super resolution of GeoEye-1 satellite images using an ESRGAN deep learning model trained exclusively on WorldView-2 images

Master's thesis in Statistics

Øystein Helle Nordberg



Supervisor

Hans Karlsen

Department of Mathematics

University of Bergen

November 2021

Results on the GeoEye-1 test set

The images are intended to be viewed on a high resolution monitor.

Multispectral bands (model input)
$128 \times 128 \times 4$

ESRGAN (model output)
$512 \times 512 \times 1$

Panchromatic band (ground truth)
$512 \times 512 \times 1$

Figure 1: La Spezia 2013-07-18, *Satellite image © 2021 Maxar Technologies*

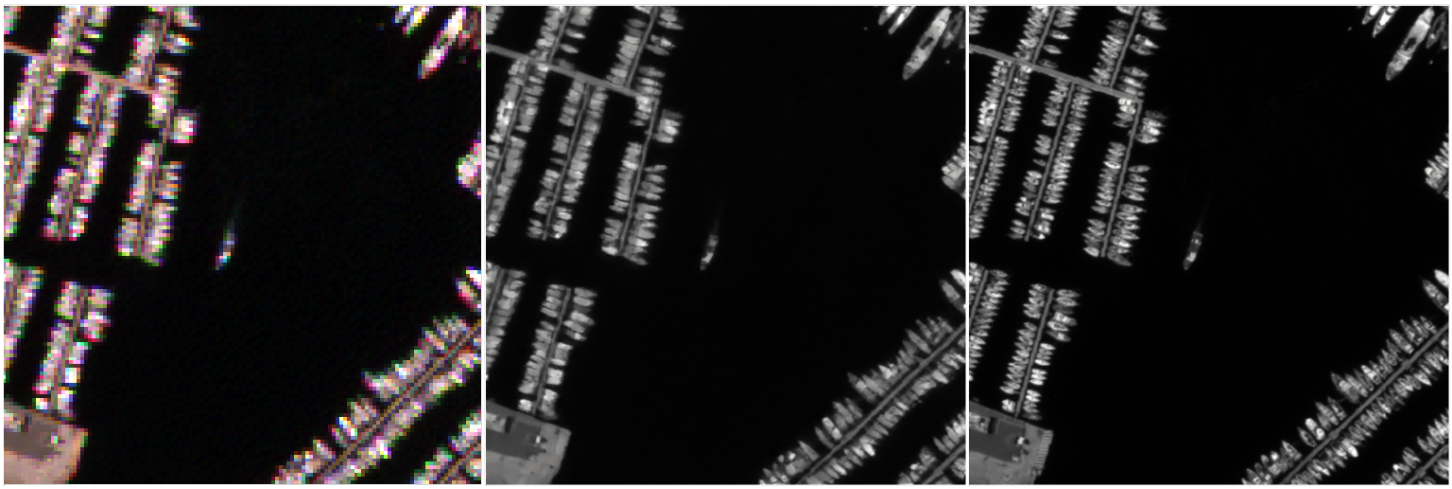


Figure 2: Toulon 2010-06-08, *Satellite image © 2021 Maxar Technologies*

Figure 3: Toulon 2017-09-05, *Satellite image © 2021 Maxar Technologies*

See Appendix A and B for additional images.

Abstract

Today, easy and abundant access to high resolution satellite imagery is taken for granted by consumers and businesses. Many remote sensing applications require optical images with a spatial resolution of 0.5 meters ground sampling distance (GSD) or less, but satellites that capture such high resolution images require heavy optical instruments, and are thus expensive to manufacture and launch. Consequently there are only a handful of such commercial satellites in orbit. WorldView-2 and GeoEye-1 are two of them. They both capture multispectral (MS) bands with a GSD of approximately 2 meters, as well as a matching panchromatic (PAN) band with a 4× higher resolution, a GSD of about 0.5 meters.

Miniaturization have enabled cheaper satellites, and has made it commercially viable to launch and maintain large constellations of nanosatellites. While plentiful, their sensors are not as capable as their larger counterparts. Their MS bands typically have a GSD of around 3-5 meters, and they do not capture a PAN band whatsoever. This limits their applications. The question then arises: Can we increase the spatial resolution of the nanosatellites through post-processing of the images? Single image super-resolution models, tasked to recover a high resolution (HR) image from a single lower resolution (LR) image, are designed to do this.

We modify and apply one of the highest performing deep learning SISR models, ESRGAN, to estimate an HR PAN band from a set of LR MS bands (4× increase in resolution). The model is trained on images taken by WorldView-2 and evaluated on images taken by both WorldView-2 and, most interestingly, GeoEye-1, a different satellite. We thus demonstrate an ability to construct an artificial HR PAN band from the MS bands of a satellite, without training on images from that particular satellite, i.e., a cross-sensor application of SISR. This opens up the possibility to construct an artificial HR PAN band for the aforementioned nanosatellites, and we suggest this topic as an area for further research.

An added benefit of the MS-to-PAN design is that we avoid having to downsample (degrade) HR images into LR images as a preprocessing step, since the MS/PAN image pair is already an LR/HR image pair. Consequently, our model performance is not reliant on any particular downsampling method.

Acknowledgements

I would like to thank my supervisor Hans Karlsen for his continuous support, advice, encouragement, and for giving me the freedom to explore a topic relevant to my personal interests and professional background. In addition, his support in providing me with the necessary compute hardware to run my experiments was essential. I would also like to thank Sondre Hølleland for his excellent technical assistance with setting up the necessary compute environment.

A special thanks to Aksel Wilhelm Wold Eide, Ingebjørg Kåsen, Eilif Solberg and Ole A. Øverland at the Norwegian Defence Research Establishment (FFI), for both suggesting the topic of the thesis, providing me with the satellite imagery data and for their continuous support and advice throughout the process. Also, a thank you to Ingrid Byre in the Norwegian Armed Forces for assisting me with copyright matters.

My passion for satellite imagery was sparked during my time in the Norwegian Armed Forces. Thank you to my former colleagues and teachers who helped spark this interest, and provided me with the necessary foundational knowledge of satellite imagery.

Finally, a very special thank you to my wonderful partner and role model, Hege, and my joyful and inspiring son, Eirik. Hege's professional and personal support with both the thesis and the Master's degree in general has been invaluable.

Contents

1 Introduction **9**

1.1 Multispectral satellite images 9

1.2 Single image super-resolution (SISR) and the perception-distortion plane 10

1.3 SISR applied to satellite images 12

1.4 Research questions 12

1.5 Method 13

1.6 Outline of thesis 13

I Background **15**

2 Multispectral satellite imagery **17**

2.1 WorldView-2 and GeoEye-1 18

2.1.1 The sun-synchronous orbit 20

2.2 Four types of resolution 21

2.2.1 Spatial resolution 21

2.2.2 Spectral resolution 23

2.2.3 Temporal resolution 24

2.2.4 Radiometric resolution 25

2.3 Processing of satellite images 25

3 Deep learning **27**

3.1 A machine learning model 27

3.1.1 The task, T 28

3.1.2 The experience, E 28

3.1.3 The performance measure, P 28

3.2 A single neuron 29

3.3 Densely connected networks 30

3.4 The forward pass 31

3.5 The activation function . . . 32
3.6 The loss function . . . 33
3.7 Training the network . . . 34
3.8 Back-propagation: Computing the gradient . . . 35
3.9 Stochastic Gradient Descent: Updating the weights . . . 36
3.9.1 The Adam optimizer: Adaptive moments . . . 37
3.10 Convolutional layers . . . 38
3.10.1 Zero padding . . . 40
3.11 Building and representing a network . . . 40
3.11.1 Building a network in TensorFlow . . . 42
3.12 Generative Adversarial Networks (GAN) . . . 43

4 Single image super-resolution (SISR) 47
4.1 Perceptual quality and the perception-distortion plane . . . 48
4.2 SISR performance metrics . . . 48
4.2.1 Peak Signal-to-Noise Ratio (PSNR) . . . 50
4.2.2 Natural Image Quality Evaluator (NIQE) . . . 52
4.2.3 Ma et al. . . . 52
4.2.4 Perceptual Index (PI) . . . 53
4.3 ESRGAN . . . 53
4.3.1 ESRGAN loss functions . . . 55
4.3.2 ESRGAN Generator . . . 57

II Experiments on satellite imagery 59

5 Experimental design 61
5.1 The data: Two Mediterranean towns . . . 62
5.1.1 Introducing Toulon and La Spezia . . . 63
5.1.2 Data partition: training, validation and test sets . . . 65
5.1.3 Temporal correlations . . . 66
5.2 The experiments . . . 67
5.2.1 E1. The baseline experiment . . . 67
5.2.2 E2. The regularization experiment . . . 68
5.2.3 E3. The final evaluation . . . 69
5.3 Training, logging and evaluation . . . 69

5.3.1 Computing NIQE, Ma et al., and Perceptual Index (PI) metrics 70
5.4 Adapting ESRGAN to the MS-to-PAN task 71
5.4.1 Changes to the network architecture 71
5.4.2 Changes to the training configurations 72

6 Data pipeline **75**
6.1 Image patches and the fully-convolutional neural network 76
6.2 Step 1: Patch allocation 77
6.3 Step 2: Patch extraction 78
6.3.1 Border pixels 78
6.3.2 Cloud and sea classifier 79
6.4 Step 3: Patch pipeline 80
6.4.1 Normalization 81

7 Results **83**
7.1 E1. The baseline experiment 83
7.1.1 GeoEye-1 performance 85
7.1.2 Are we overfitting? 86
7.2 E2. The regularization experiment 88
7.2.1 The best models 90
7.3 E3. The final evaluation – Test set 91

8 Conclusion **95**
8.1 Ideas for future research 95
8.1.1 Apply SISR to satellite images without a PAN band 96
8.1.2 Develop alternative performance metrics 96
8.1.3 Generalize beyond two towns and a temporal dataset 97
8.1.4 Train on less processed images 97

Bibliography **97**

A Random patches from the GeoEye-1 test set **107**

B Random patches from the WorldView-2 test set **113**

C Satellite image metadata **119**

Nomenclature

Terms, abbreviations and acronyms

Adam	An SGD-like optimizer (Kingma & Ba, 2014)
BN	Batch Normalization
CCD	Charge-Coupled Device
CNN	Convolutional Neural Network, used interchangeably with ConvNet
ConvNet	Convolutional Neural Network, used interchangeably with CNN
DCGAN	Deep Convolutional Generative Adversarial Network, a GAN model by Radford et al., 2016
DigitalGlobe	A US satellite imagery company, acquired by Maxar in 2017
ECCV	European Conference on Computer Vision
ERTS	Earth Resource Technology Satellite, a 1972 NASA satellite later renamed Landsat 1
ESA	European Space Agency
ESRGAN	Enhanced Super-Resolution Generative Adversarial Network, a deep learning SISR model by X. Wang, Yu, Wu, et al., 2018
FFI	Norwegian Defence Research Establishment
FR	Full-Reference, a type of IQA
GAN	Generative Adversarial Network
GE01	GeoEye-1 satellite
GeoTIFF	TIFF with additional geospatial metadata
GSD	Ground Sample Distance
HR	High Resolution
IQA	Image Quality Assessment
L1	L1 loss
L2	L2 loss
Landsat	A joint NASA/USGS satellite program
LR	Low Resolution

Ma et al.	A SISR specific IQA algorithm. The name refers to the authors of the paper that introduced the algorithm (Ma et al., 2017)
MAE	Mean Absolute Error
MATLAB	A programming language and compute environment
Maxar	Maxar Technologies, a US space technology company
MOS	Mean Opinion Score
MS	Multispectral
MSE	Mean Squared Error
MSS	Multispectral Scanner, an instrument carried by Landsat 1-5
Nadir	The direction pointing directly below a particular location
NASA	National Aeronautics and Space Administration
NGA	US National Geospatial-Intelligence Agency
NIQE	Natural Image Quality Evaluator, an IQA algorithm (Mittal et al., 2013)
NIR	Near-infrared, sub-division of the electromagnetic spectrum.
NR	No-Reference, a type of IQA
PAN	Panchromatic
PI	Perceptual Index, an IQA metric
PIRM	Perceptual Image Restoration and Manipulation, a workshop in ECCV
Planet	Planet Labs, a US space technology company, strongly associated with nanosatellite imagery
PSNR	Peak Signal-to-Noise Ratio
RaGAN	Relativistic average GAN (Jolicoeur-Martineau, 2018)
ReLU	Rectified Linear Unit, a common activation function
RGB	Red, Green, Blue, a color model
RRDB	Residual-in-Residual Dense Block
Sentinel	A family of ESA satellites in the Copernicus program
SGD	Stochastic Gradient Descent
Sigmoid	Sigmoid function, an S-shaped function commonly used as activation function
SISR	Single-Image Super Resolution
SR	Super Resolution
SRCNN	Super Resolution Convolutional Neural Network, a deep learning SISR model by Dong et al., 2016
SRGAN	Super-Resolution Generative Adversarial Network, a deep learning SISR model by Ledig et al., 2017

SSO	Sun-Synchronous Orbit
tanh	Hyperbolic tangent function, an S-shaped function commonly used as activation function
Tensor	In this thesis defined to be a multidimensional array, i.e., a generalization of matrices to higher orders (Bi et al., 2021; Kolda & Bader, 2009)
TensorFlow	Open source machine learning library
TIFF	Tag Image File Format
ULA	United Launch Alliance, a US spacecraft launch service provider
USGS	United States Geological Survey
VGG19	Visual Geometry Group (19 layer version), a deep convolutional image classifier by Simonyan and Zisserman, 2015
WV02	WorldView-2 satellite

Notation

a	A scalar
$\mathbf{a}$	A vector
$\mathbf{A}$	A tensor or matrix. See *Tensor* above for definition.
$a_{i,j}$	Row i, column j of a matrix $\mathbf{A}$
$f(\cdot)$	A function
$\mathbb{R}$	The real numbers
$\boldsymbol{a}^\top$	The transpose of $\boldsymbol{a}$
$\|\mathbf{a}\|_p$	p-norm of a vector $\mathbf{a}$
$\mathbf{X}$	Input features tensor or matrix, i.e., model input Also denoted $\boldsymbol{x}$ if vector shaped
$\boldsymbol{y}$	Ground truth Also denoted y or $\mathbf{Y}$ depending on its shape
$\boldsymbol{e}$	Residuals
$\hat{\boldsymbol{y}}$	Estimate of the ground truth $\boldsymbol{y}$, i.e., model output Also denoted y or $\hat{\mathbf{Y}}$ depending on its shape
$\mathbf{W}$	Trainable weights and biases in a neural network
$\mathbf{W}^{(k)}$	The weights and biases of layer k in a neural network
$L(\hat{\boldsymbol{y}}, \boldsymbol{y})$	Loss function, often simply denoted L
$\nabla_{\mathbf{W}} L$	Gradient of the loss function, L, with respect to the weights, $\mathbf{W}$
$\hat{\boldsymbol{g}}$	Gradient estimate

η	Learning rate
$g(\cdot)$	Activation function
T	The *Task* of a machine learning algorithm
E	The *Experience* of a machine learning algorithm, i.e., how the model learns from data
P	The *Performance measure* of a machine learning algorithm
G	The Generator in a GAN
D	The Discriminator in a GAN
$\mathbf{X}_{LR}$	One or multiple low resolution (LR) images. Satellite imagery context: $\mathbf{X}_{LR} = \mathbf{X}_{MS}$, the multispectral bands. Either a 3D or 4D tensor: $H_{LR} \times W_{LR} \times C$ (3D) $N \times H_{LR} \times W_{LR} \times C$ (4D)
$\mathbf{X}_{HR}$	One or multiple high resolution (HR) images. Satellite imagery context: $\mathbf{X}_{HR} = \mathbf{X}_{PAN}$, the panchromatic band. Either a 3D or 4D tensor: $H_{HR} \times W_{HR} \times C$ (3D) $N \times H_{HR} \times W_{HR} \times C$ (4D) If the HR image is the panchromatic band, then $C = 1$
$\mathbf{X}_{SR}$	One or multiple super-resoluted (SR) images. Estimate of $\mathbf{X}_{HR}$ Either a 3D or 4D tensor: $H_{HR} \times W_{HR} \times C$ (3D) $N \times H_{HR} \times W_{HR} \times C$ (4D) If the SR image is the panchromatic band, then $C = 1$

List of Figures

1.1 WorldView-2 multispectral and panchromatic bands . . . 9
1.2 The perception-distortion plane . . . 11

2.1 1976 M satellite image of the Bergen, Norway area . . . 17
2.2 WorldView-2: Schematic overview of the main components . . . 19
2.3 A satellite capturing both nadir and off-nadir images (Maxar, 2019a, 2019c) . . . 20
2.4 A syn-synchronous orbit . . . 20
2.5 A push broom scanner . . . 22
2.6 Focal plane layout of WorldView-2 . . . 22
2.7 Relative spectral radiance response for WorldView-2 and GeoEye-1 satellites . . . 24

3.1 The use of training, validation and test sets during model selection and the final estimation of model performance. . . . 29
3.2 A single artificial neuron . . . 29
3.3 A densely connected feedforward neural network . . . 30
3.4 Comparison of some of the most common activation functions . . . 32
3.5 Backpropagation in a densely connected feed-forward neural network . . . 35
3.6 SGD with and without momentum . . . 37
3.7 2D convolutions . . . 40
3.8 Zero padding . . . 41
3.9 A small convolutional neural network . . . 42
3.10 Standard GAN architecture . . . 44

4.1 Overview of relevant SISR performance metrics . . . 49
4.2 Scatter plots comparing PSNR and Ma et al. with MOS . . . 50
4.3 The main steps of the Ma et al. performance measure . . . 52
4.4 A standard GAN architecture applied to SISR . . . 53
4.5 GAN training of ESRGAN on the MS-to-PAN task . . . 54
4.6 Deep feature extraction from a trained VGG19 network . . . 55

4.7 The ESRGAN Discriminator Network with RaGAN 56
4.8 The ESRGAN Generator Network 57

5.1 Experimental design 61
5.2 Satellite images of Toulon and La Spezia 63
5.3 Distribution of image sizes 64
5.4 Image patches of the same location extracted from multiple images 66
5.5 Flips and 90 degree rotations applied to an MS patch 68
5.6 Image patches at different stages of training 70

6.1 Overview of the data pipeline 75
6.2 Extraction of paired MS and PAN image patches from larger satellite images . . 76
6.3 Overview of patch extraction process 78
6.4 Density maps of sampled patches 79
6.5 Effect of different pipeline optimization techniques 81

7.1 Baseline experiment results summarized on the perception-distortion plane 83
7.2 Comparison of baseline models with different number of MS bands 84
7.3 Scatter plot of m4 individual image patches 85
7.4 Learning curves: PSNR and NIQE plotted against training iterations in E1 . . . 87
7.5 Regularization experiment results summarized on the perception-distortion plane 88
7.6 Learning curves: PSNR and NIQE plotted against training iterations in E2 . . . 89
7.7 Scatter plot of individual image patches, comparing the baseline m4 with regularized version 90
7.8 Scatter plot of individual image patches, comparing the regularized m4-os-aug model on both validation and test set 91
7.9 GeoEye-1 test set: Comparison between an MS, an ESRGAN estimated and a PAN ground truth image patch 92

List of Tables

2.1 Selection and description of some common satellite imagery processing levels . . . 26

5.1 Contingency table with number of images by areas and satellite sensor 64

5.2 Contingency table with number of images in train, validation and test sets across town and satellite sensor 65

5.3 Models and band combinations in the baseline experiment (E1). 67

5.4 Configuration and hyperparameter settings 72

6.1 Different patch sizes for different partitions. C varies across experiments. 77

List of Algorithms

3.1 Training a feedforward neural network 34
3.2 Stochastic gradient descent (SGD) 36
3.3 Adam optimizer 38
3.4 GAN 45

Chapter 1

Introduction

1.1. Multispectral satellite images

The design and construction of a camera is a result of many engineering trade-offs. For instance, we might want our camera to capture images with high spatial resolution and dynamic range, yet we also want the camera to be small, cheap and robust. Probably nowhere are these trade-offs more prominent than on optical remote sensing instruments, i.e., satellite-mounted cameras. WorldView-2 is a so-called *very high resolution* optical multispectral imagery satellite. It captures eight multispectral (MS) bands with a spatial resolution of about 2 meters, and one panchromatic (PAN) band with a four times higher spatial resolution, about 0.5 meters. This is all done from an altitude of 770 km and a velocity high enough to orbit the Earth in 100 minutes. Optical requirements are on the extreme end of the spectrum. (Maxar, 2019c)

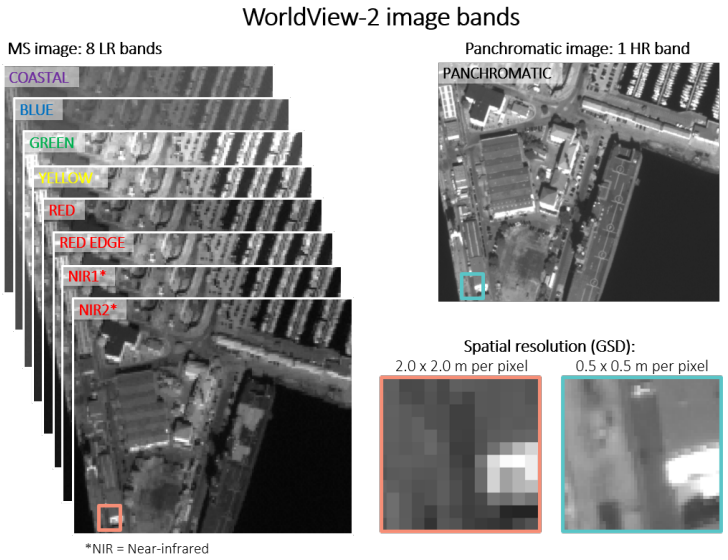


Figure 1.1: WorldView-2 image with eight MS bands that capture different wavelength ranges across the electromagnetic spectrum, and one higher-resolution PAN band that captures a single, wider range of wavelengths. *Satellite image © 2021 Maxar Technologies*

WorldView-2 launched into orbit in 2009. Still, there are only a handful of commercial satellites with similar specifications in orbit today. As a consequence, image capturing is prioritized over areas where customer demand is high. Thus, the *temporal resolution*, a measure of how frequently images are captured over a specific area, is low across large portions of the Earth. Years may pass between each time satellites like WorldView-2 capture an image of a low demand area. (E. O. P. ESA, 2021b; Maxar, 2019b)

Miniature satellites have become increasingly popular. Their optical instruments are not as capable as their heavier counterparts, but unit costs are orders of magnitude smaller. Consequently, companies can launch and maintain large constellations of these smaller satellites. Planet Labs, the largest player in the miniature optical imagery satellite space, maintains a constellation of over 200 satellites with a goal to capture the entire surface of the Earth every day. Their Dove nanosatellite is about as big as a shoe box and weigh around 4 kilograms. The trade-off is a comparably lower spatial resolution, about 3-5 meters for its four MS bands, and no PAN band whatsoever. (Planet, 2021)

Users of commercial satellite imagery are thus often left with a trade-off between spatial and temporal resolution. Can this trade-off be reduced through post-processing of images? Can the spatial resolution of satellite images be increased after the image has been captured?

1.2. Single image super-resolution (SISR) and the perception-distortion plane

Single image super-resolution (SISR), a classic computer vision problem, is the task of estimating a high-resolution (HR) image from a single lower-resolution (LR) image. It is inherently difficult, and considered an ill-posed, inverse problem, since for every LR image input there exists multiple HR image solutions. Ever since the pivotal work of Dong et al., 2016 and their SRCNN model, deep learning methods have dominated SISR. Dong et al. showed that a convolutional neural network (CNN) was equivalent to several of the leading SISR methods at the time. (Ledig et al., 2017)

The next big advancement in deep learning-based SISR came with SRGAN (Ledig et al., 2017). Until then, most SISR models had focused on minimizing the distortion between the SR image and the ground truth HR image. Distortion between two images are usually measured with the peak-signal-to-noise (PSNR) metric, a derivative of the well-known mean squared error (MSE). SR images produced by these models were blurry and easily distinguishable from the ground truth by humans. The images had low *perceptual* quality. Ledig et al., 2017 combined and implemented a few different techniques, including the use a generative adversarial network (GAN) design (I. J. Goodfellow et al., 2014), to motivate the model into producing more photo-

realistic outputs, i.e., SR images with higher perceptual quality.

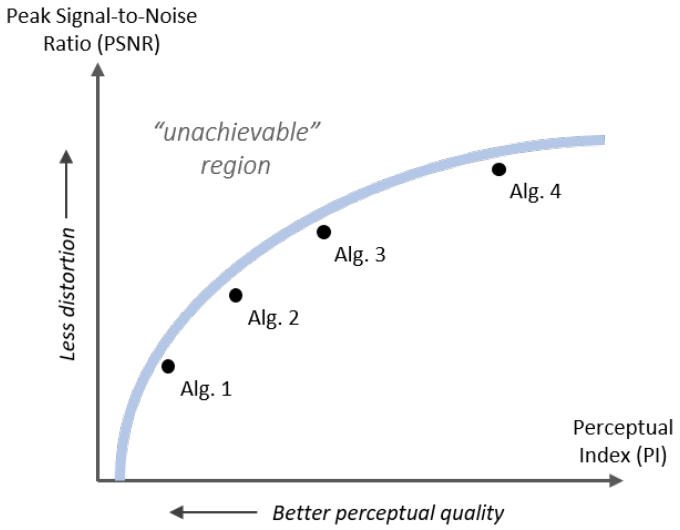


Figure 1.2: The perception-distortion plane illustrating the trade-off between perceptual quality and distortion.

It was however evident that a trade-off between high perceptual quality and low distortion existed. SR images could not both have high perceptual quality and low distortion, and we may depict this as a boundary on a perception-distortion plane (see Figure 1.2). Real images often consist of sharp edges and high-frequency details. Models that favor high perceptual quality have to be more aggressive in predicting such details, often resulting in penalties by distortion-type metrics like PSNR. We may illustrate this with checkerboard image example. A model that favors high perceptual quality will try to estimate sharp edges between black and white squares. Still, if the model misses with just one pixel in any direction it will receive a low PSNR score. Lots of pitch black pixels should have been chalk white, yet humans will probably not notice this one-pixel shift. (Blau & Michaeli, 2018; Vasu et al., 2018)

In 2018, a competition, the PIRM Challenge on Perceptual Super-Resolution (Blau et al., 2019), challenged participants to push the boundaries of the perception-distortion trade-off. Enhanced SRGAN (ESRGAN), a modified and evidently improved version of SRGAN, emerged as one of the winners.[1] Naturally, better performing models have been published since 2018 (Blau et al., 2019; Ma et al., 2020; Soh et al., 2019). However, amongst models that favor high perceptual quality over distortion, GAN-based models still dominate and most share many similarities with SRGAN and ESRGAN.

[1] In the challenge, the perception-distortion plane was divided into three regions. ESRGAN won first place in the high-perceptual-quality region.

1.3. SISR applied to satellite images

There has been some research into SISR on satellite images. Many have focused on super-resoluting images beyond their native spatial resolution. HR images are downsampled into LR, and models are trained on the resulting LR-HR image pairs. After training, the models are subsequently fed HR images instead of LR images, outputting SR images with a higher-than-native spatial resolution. Shermeyer and Van Etten, 2019 showed that object detection models performed better on super-resoluted 15 cm imagery than on the native 30 cm imagery. Additionally, Maxar have recently productized this approach with their *Maxar HD Technology* (Gleason, 2020).

Others have focused on the multispectral aspects of satellite imagery. Lanaras et al., 2017 apply SISR to Sentinel-2 images. Images taken by the Sentinel-2 satellite has MS bands with varying spatial resolution, and their SISR model super-resolve all bands equal to the highest resolution band: 10 meters GSD. Müller et al., 2020 takes advantage of the relationship between the lower resolution MS bands and the higher resolution PAN band. Pan-sharpening, a widely used deterministic technique to fuse MS and PAN bands into a single HR image, is used to create HR versions of the MS images. They subsequently train different SISR models on pairs of LR MS images and 4× HR pan-sharpened images.

To our knowledge there is almost no publicly available research into training a SISR model on images from one satellite and applying it to images from another satellite. A notable exception is the work done by Pouliot et al., 2018. They demonstrate the ability to train SISR models on images from Sentinel-2 and apply the model to images taken by Landsat-5 and Landsat-8. However, they limit their research to SRCNN

Evidently, there is a general lack of research into super-resolving images from one satellite by training on images from another. This is an area that warrants more attention, especially due to the potential benefits from increasing the spatial resolution of nanosatellites.

1.4. Research questions

The overall goal in this thesis is to determine whether a SISR model can be trained to estimate a higher-resolution PAN band from lower-resolution MS bands. Furthermore, we want to explore whether this model can be used to super-resolve images taken by a different satellite. With this background we formulate our research questions. Using ESRGAN as our SISR model we explore the following topics:

R1: To what extent can the higher-resolution 0.5 m GSD PAN band be reconstructed from the

lower-resolution 2.0 m GSD MS bands?

R2: To what extent can the model trained on images from satellite A be used to super-resolute images from a similar satellite B?

R3: Can we increase performance by introducing regularization, in the form of data augmentation and over-sampling of patches from the satellite images? If so, by how much?

1.5. Method

We run a set of experiments on satellite images of the French town of Toulon and the Italian town of La Spezia. The images have been captured by Maxar satellites WorldView-2 and GeoEye-1, two satellites with somewhat similar image characteristics. Both capture a number of MS bands, eight in the case of WorldView-2 and four in the case of GeoEye-1, with a similar spatial resolution of around 2 meters GSD. Both also capture a matching PAN band with a spatial resolution of around 0.5 meters GSD, a 4× increase in resolution compared to the MS bands. Still, the satellites are different. In fact they were designed and operated by two different companies prior to a merger between GeoEye and DigitalGlobe (now Maxar) in 2013.

Why GeoEye-1 as satellite B? Why not choose a satellite with no PAN band? That would directly demonstrate the utility of our approach. The answer is simply that with the lack of prior research on cross-sensor SISR of satellite images, we see a need to first evaluate performance on a test set that has a ground truth PAN band. While it is possible to use so-called no-reference image quality metrics to assess the quality of a super-resolved image, without a reference ground truth image, it is much harder to reach a conclusion based on such metrics. In Chapter 8 we suggest building on the findings of this thesis and apply SISR to a satellite without the PAN band.

1.6. Outline of thesis

The thesis is divided into two main parts: Background and Experiments. Background starts with an introduction to multispectral satellite imagery. We then proceed with a chapter on deep learning and how neural networks are trained, covering topics such as loss functions, backpropagation and optimizers to name a few. Background is concluded with a chapter on SISR covering the ESRGAN network and the performance measures used to evaluate models.

In the Experiments part we present the data and methods used to answer the research questions. We also dedicate a chapter to the custom data pipeline developed to support efficient training. Finally, experiment results are presented, discussed, and the thesis concludes with pointing to ideas for future research.

We also encourage the reader to check out the project's GitHub repository.[2] A large portion of the work behind this thesis have gone towards modifying and implementing the ESRGAN model in TensorFlow 2, implementing a performant TensorFlow data pipeline decoding satellite images and feeding the model with appropriately shaped tensors, and finally getting the whole GAN machinery to train consistently, without loss divergence.

In addition, there are four appendices. Appendix A and Appendix B contain results from randomly sampled image patches from the test set. They provide an unfiltered view of actual image results on unseen data, and we actually recommend taking a look at these now before proceeding with the rest of the thesis. Finally in Appendix C we provide a full list of satellite images used.

[2] https://github.com/onordberg/multispectral-super-resolution

Part I

Background

Chapter 2

Multispectral satellite imagery

Multispectral satellite imagery was introduced to the research community with the launch of the NASA Earth Resource Technology Satellite (ERTS) in 1972, later renamed Landsat 1. ERTS carried two sensing instruments, of which the Multispectral Scanner (MSS) became the primary instrument. It captured four spectral bands with a spatial resolution of 80 meters and a range of 6 bits per pixel. An example of such an image can be seen in Figure 2.1. ERTS and MSS were spectacularly successful, exceeding expectations on both utility and operating lifetime. (Baghdadi & Zribi, 2016; Mika, 1997)

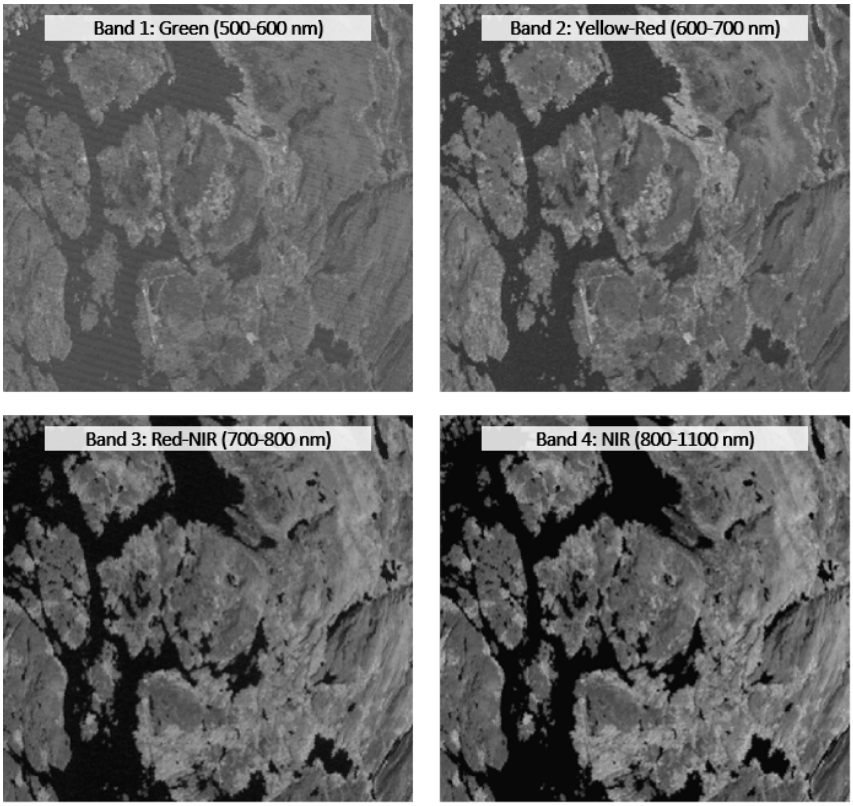


Figure 2.1: 1976 MS satellite image of the Bergen, Norway area captured by the Landsat 2 MSS sensor (identical to the Landsat 1 MSS sensor). (USGS, 1976) *Landsat 2 image courtesy of the U.S. Geological Survey (USGS)*

Unsurprisingly then, the Landsat program is still going strong, with Landsat 7, 8 and 9 currently in operation. Naturally, their capabilities have improved dramatically since the 1970s, and together with the European Space Agency's (ESA) Sentinel satellites, Landsat is the backbone of free and publicly available satellite imagery. Satellite imagery from these two programs serve a wide variety of use cases. Examples include mapping, climate research, land cover classification, forest management, agricultural analyses and disaster response. Yet, the 10-60 meter spatial resolution (varies between spectral bands) of the Landsat and Sentinel satellites excludes plenty of use cases, for instance many related to urban analysis and most satellite web map use cases. Today's web map users expect a spatial resolution of 0.5 meters or better when they toggle *imagery* on in their web map application. This is where satellites like WorldView-2 and GeoEye-1 from commercial actors like Maxar come into play. (NASA, 2018, 2021b)

Books on multispectral satellite images typically start with a chapter on radiometry, the science of measuring radiation, covering topics like the electromagnetic spectrum, reflectance, absorption, polarization, atmospheric distortions, calibrations and corrections. We will touch upon some radiometric topics throughout this chapter, yet with a focus on specifics related to the WorldView-2 and GeoEye-1 satellites, as well as relationships between the lower-resolution MS bands and the higher-resolution PAN band. For a more systematic introduction to radiometry in the context of multispectral satellite imagery, we refer you to Baghdadi and Zribi, 2016, pp. 1–56.

2.1. WorldView-2 and GeoEye-1

WorldView-2 can trace its commercial origins back to a contract between Maxar, then DigitalGlobe, and the US National Geospatial-Intelligence Agency (NGA), in 2003. Manufacturing of the actual satellite started in 2006 and it was finally launched into orbit on a United Launch Alliance (ULA) Delta-2 rocket in 2009. Full operational capability was reached in January 2010 and it was then the first commercial satellite to carry a very high resolution 8-band MS sensor. WorldView-2 is still, as of October 2021, operating nominally, collecting up to 1 million square kilometers of imagery per day. (E. O. P. ESA, 2021b; Maxar, 2019c)

Similarly, the commercial origins of the 4-band multispectral satellite GeoEye-1 are also a result of a US government contract. GeoEye, the company[1], was awarded a contract by NGA in 2004. In addition, the company signed a deal with Google, giving the map services behemoth web map exclusivity rights to use GeoEye-1 imagery in their Google Maps and Google Earth services. The satellite was manufactured by General Dynamics C4 Systems and launched on a ULA Delta-2 rocket in 2008, becoming fully operational in February 2009. Just like WorldView-2,

[1] GeoEye was later, in 2013, acquired by and merged into Maxar, then DigitalGlobe.

GeoEye-1 is still as of October 2021, operating nominally.

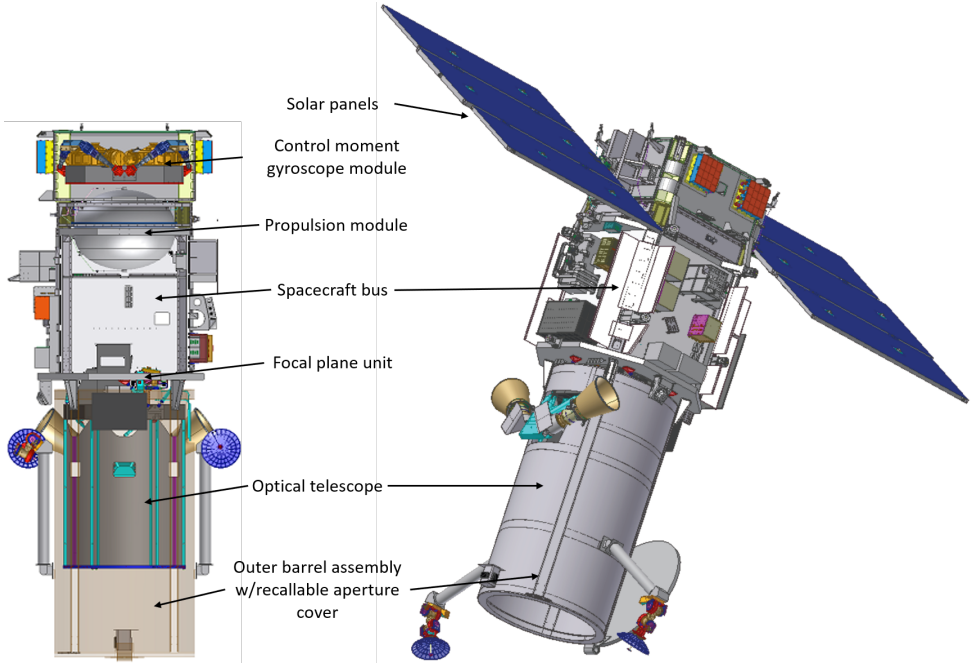


Figure 2.2: WorldView-2: Schematic overview of the main components. © 2021 Maxar Technologies. Reprinted/adapted with permission. (DigitalGlobe, 2009)

The main components of WorldView-2 are depicted in Figure 2.2. GeoEye-1 has a similar design. Looking at the lower part of the figure, WorldView-2 would not be an imagery satellite without its optical telescope. The telescope captures sunlight reflected off the earth surface and channels it to the focal plane unit, where the charge-coupled device (CCD) image sensor is located. We will revisit these core components in a short while. At the center of the figure we notice the spacecraft bus, the backbone that connects everything on the satellite together and provides central services like data storage and transmission. (DigitalGlobe, 2009; E. O. P. ESA, 2021b)

At the top of the figure are components used to physically control the satellite. The propulsion module contains propellant that is burnt in order to control and maintain the ideal orbit, while the gyroscope controls the direction of the whole satellite body, including the optical telescope. It enables the satellite to capture images at an off-nadir angle, for instance to the right or left of its orbit, consequently increasing its range. A nadir image is taken vertically, with the telescope pointing straight down on the Earth's surface. An off-nadir image, on the other hand, is captured at an angle, revealing details of vertical surfaces, like the walls of buildings. (DigitalGlobe, 2009; E. O. P. ESA, 2021b)

Figure 2.3 illustrates how satellites like WorldView-2 and GeoEye-1 are capable of capturing images of the same Earth surface from different angles in one pass. Keep in mind that the

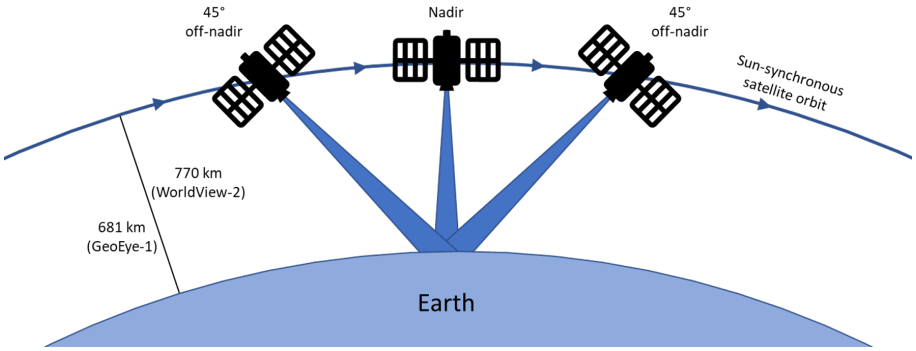


Figure 2.3: A satellite capturing both nadir and off-nadir images (Maxar, 2019a, 2019c)

satellite travels at a ground speed of almost 7 kilometers per second. Consequently, the gyroscope module must be both quick and precise. When a satellite captures two off-nadir images of the same Earth surface we get so-called stereoscopic imagery, as illustrated in Figure 2.3. A use case of such images is stereophotogrammetry, the construction of a 3D model from two of more 2D images. (Maxar, 2019c)

2.1.1. The sun-synchronous orbit

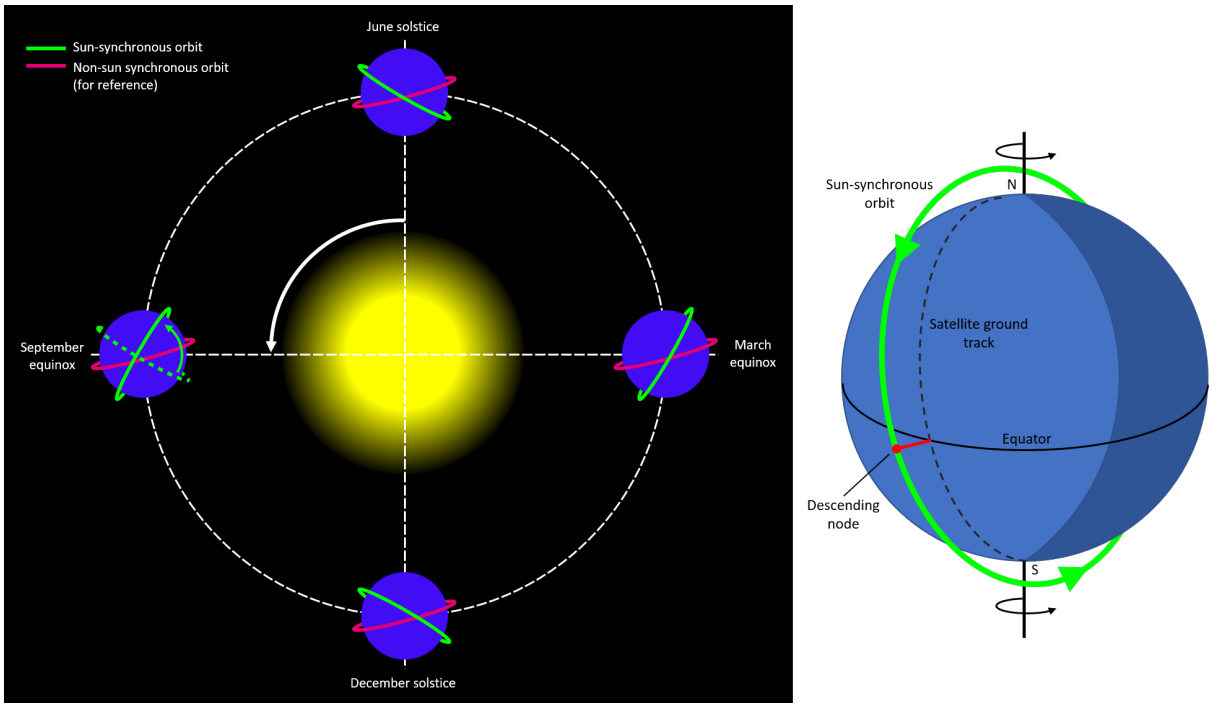


Figure 2.4: A syn-synchronous orbit. Left: As Earth orbits around the Sun the satellite orbit is fixed with reference to the Sun. (Brandir & XZise, 2018) Right: A syn-synchronous orbit is a polar orbit. Its descending node is fixed at the same local time every orbit. (Beck, 2016) *Both figures are adaptations of originals, in accordance with CC BY 3.0.*

WorldView-2 and GeoEye-1 follow so-called sun-synchronous orbits (SSO). These low-altitude,

fast orbits are a particular kind of polar orbits, synchronized to always be in the same fixed position relative to the Sun. This is achieved through careful calibration of the velocity, angle and altitude of the satellite. Satellites in SSO will always visit the same spot at the same local time. For imagery satellites, this is a very attractive property. Every time the satellite is overhead, the surface illumination is nearly the same, enabling consistent conditions for monitoring an area over time. Satellites in SSO also benefit from the fact that SSO is a subtype of polar orbits. Satellites in polar orbits covers the entire surface of the Earth, since the Earth is rotating beneath the satellite in an almost perpendicular plane to the satellite's orbit. SSOs are relatively low-altitude and high speed. WordView-2, for instance, has an altitude of 770 kilometers and an orbital period of 100 minutes. For comparison, a satellite in geostationary orbit has an altitude of approximately 35786 kilometers and an orbital period of 24 hours. (ESA, 2020; Maxar, 2019c)

2.2. Four types of resolution

People often think of spatial resolution, when the term image resolution pop up. However, within the field of remote sensing we typically speak of four types of image resolution: spatial, spectral, temporal and radiometric. Understanding these types give us an insight into how multispectral satellite imagery works and how this relates to the overall MS-to-PAN SISR task we introduced in Chapter 1. Emphasis is put on spatial and spectral resolution, since these concepts lie at the core of the MS-to-PAN SISR task. (Baghdadi & Zribi, 2016, pp. 68–74)

2.2.1. Spatial resolution

In the digital era where images are represented by arrays of pixels, spatial resolution refers to the size of each pixel in the image, or the distance between each measurement pixel center point. For satellite images, we typically report spatial resolution with a *meters per pixel* metric (0.5 meters/pixel), or simply report the size of a pixel in meters (0.5×0.5 meters). Alternatively, we may report the ground sample distance (GSD) of an image. GSD is the distance between pixel center points in an image. Given square pixels, e.g., 0.5×0.5 meters, GSD and the aforementioned ways of reporting spatial resolution is equivalent. We use the terms interchangeably throughout this thesis. (Baghdadi & Zribi, 2016, pp. 68–70)

Spatial resolution is at the core of super-resolution, and in Figure 1.1 we already introduced how WorldView-2 captures eight lower resolution MS bands (approximately 2 meters GSD) and a single higher resolution PAN band (approximately 0.5 meters GSD). Why the difference, and how are images actually captured by a multispectral satellite? To better understand this we need to take a closer look at the actual image sensor. (Maxar, 2019c)

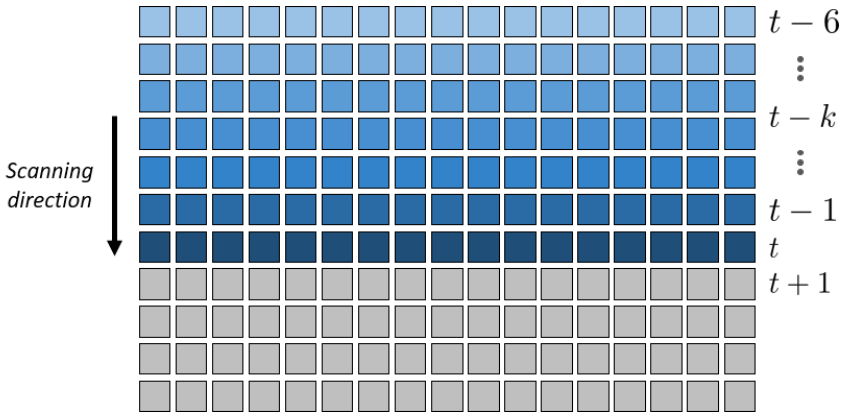


Figure 2.5: A push broom scanner scanning the surface of the Earth sequentially one line at the time.

WorldView-2 and GeoEye-1 uses a push broom scanning technique to capture images. This technique is comparable to how a regular photo-copier line scanner works: scanning is done line by line in one direction. As the satellite passes over the Earth, its sensors scan the surface one pixel line at a time. This sequential process is depicted in Figure 2.5, and we notice the time differences within a single satellite image taken by a push broom scanner. Push broom scanning is not the only method used by imagery satellites. For instance, the Dove satellites operated by PlanetLabs, which we introduced in Chapter 1, capture images similarly to how everyday digital cameras work. A complete 2D image is captured simultaneously by a frame CCD image sensor. (E. O. P. ESA, 2021a, 2021b; Planet, 2021; Updike & Comp, 2010)

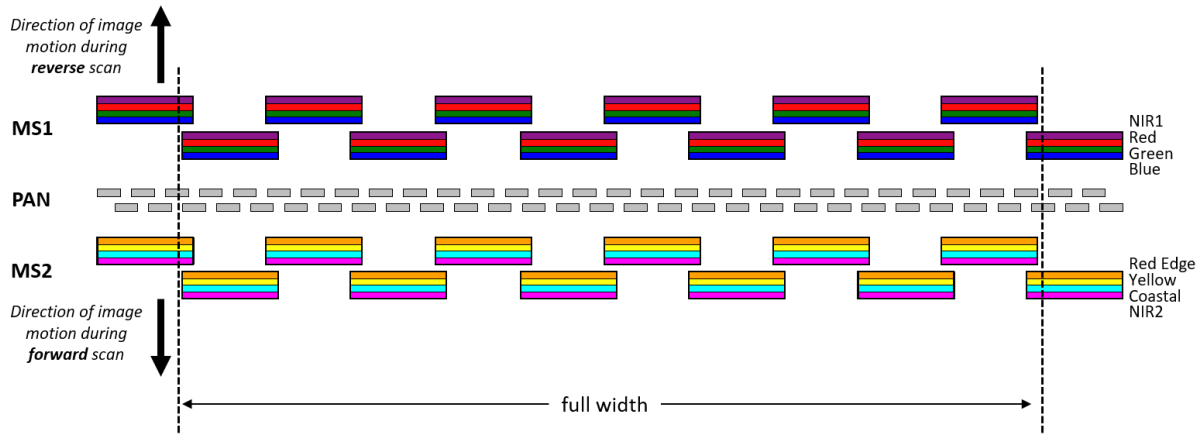


Figure 2.6: Focal plane (image plane) layout of WorldView-2 with multiple staggered CCDs. The figure is conceptual and not drawn to scale. © 2021 Maxar Technologies. Reprinted/adapted with permission. (Updike & Comp, 2010)

Returning to push broom scanners, their task is to transform optical signals (photons) into electrical signals (electrons). This is achieved through the use of multiple one-dimensional CCDs organized in a staggered, overlapping fashion. The concept is illustrated in Figure 2.6, where we

in the specific case of WorldView-2 notice three separate arrays of staggered CCDs: PAN, MS1 and MS2. The PAN array consists of fifty staggered, overlapping CCD sub-arrays each capturing a smaller number of pixels. Subsequently, outputs from the individual sub-arrays are combined into a single $35,420$ pixel-wide line, equivalent to about 17 kilometers of the Earth's surface. (Maxar, 2019c; E. O. P. ESA, 2021b)

The MS arrays uses different color filters to split light into separate spectral bands. In the next section we will explore this further in the context of spectral resolution, but the color filtering has direct implications on spatial resolution as well. Any filter applied before an image sensor will reduce the number of photons that passes through it, and thus reduce the spatial resolution of the image sensor compared to a sensor that is directly exposed to the same photons. In the case of WorldView-2 the MS1 array, consisting of 10 staggered MS CCD sub-arrays, is capable of producing lines with a width of $8,881$ pixels, approximately four times less than its PAN counterpart. However, in contrast to the PAN array, the MS1 array produces four such lines, one for each spectral band. (Baghdadi & Zribi, 2016, p. 74; Updike & Comp, 2010; E. O. P. ESA, 2021b)

2.2.2. Spectral resolution

Spectral bands have been referred to multiple times already, and the concept is central to the main topic in this thesis. In Figure 2.6 we saw that MS images are produced as a result of photons or radiance being separated into different spectral bands by color filters. We lose spatial resolution, but gain the ability to split the electromagnetic spectrum into bands: spectral bands. Simply put, with MS sensors we trade away spatial resolution for spectral resolution. (Baghdadi & Zribi, 2016, pp. 70–72, 74)

Why spectral bands? Remember that for optical satellite images, the principal source of illumination is the Sun and our sensors measure the intensity of sunlight reflected from the Earth's surface. Different surface materials (water, sand, snow, asphalt etc.) absorb and reflect different wavelengths to a varying degree. Surfaces have spectral profiles, and by measuring the light intensity in different spectral bands we are better able to analyze and distinguish different surface materials. A banal example may illustrate the point: A red car may be distinguished from a gray car by measuring the difference in intensity of the red spectral band. (Baghdadi & Zribi, 2016, pp. 70–72)

A high spectral resolution is usually correlated with more spectral bands and it is fair to say that the eight-band WorldView-2 images have higher spectral resolution than their four-band GeoEye-1 counterparts. However, spectral resolution is also related to how well the spectral

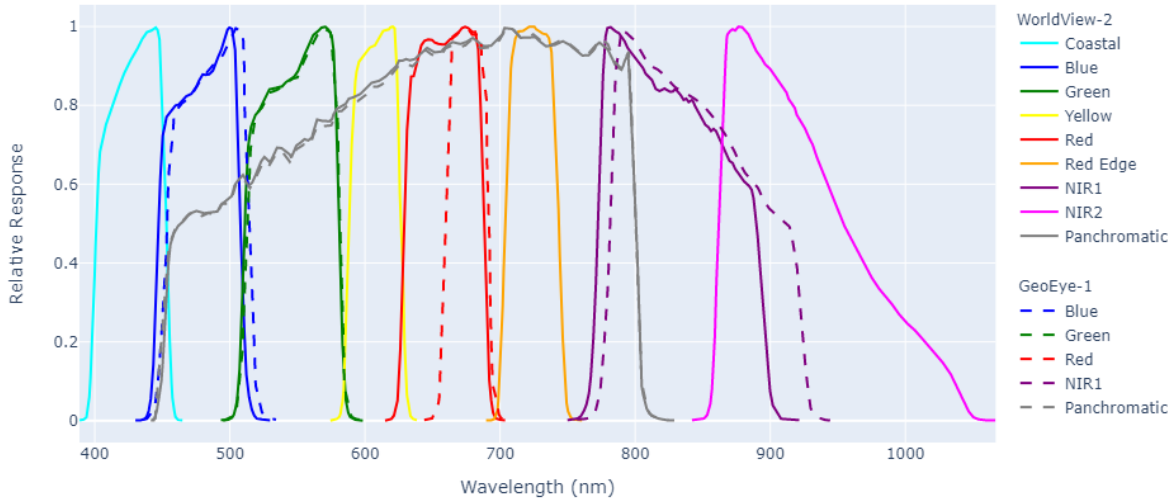


Figure 2.7: Relative spectral radiance response for WorldView-2 and GeoEye-1 satellites. The figure is reproduced from Maxar's (then DigitalGlobe) official spectral response data. (DigitalGlobe, 2014)

bands fit a particular use case. Taken to the extreme, for a *red car detector*, high spectral resolution may mean having one single, well-defined and calibrated spectral band centered around the wavelength humans perceive as *pure red*. (Baghdadi & Zribi, 2016, pp. 70–72, 74)

In Figure 2.7 we see the relative spectral response of all WorldView-2 and GeoEye-1 spectral bands on the same plot. We notice that most of the four bands present in both satellites overlap quite well, except for the Red and Near-infrared 1 (NIR1) bands. Such a systematic difference between the two sensors is a challenge that a model trained on one satellite and tested on another will have to overcome. Notice also how the PAN band covers a large portion of the electromagnetic spectrum, but overlaps only with six out of the eight WorldView-2 MS bands. We will revisit this topic in Chapter 7. (DigitalGlobe, 2014)

2.2.3. Temporal resolution

The temporal[2] resolution of a an imagery satellite is a measure of how frequently the satellite revisits a specific site. As such, temporal frequency is not only related to the sensor itself, but also the orbital characteristics of the satellite. (Baghdadi & Zribi, 2016, pp. 72–73)

There are two ways to define temporal frequency. The first and traditional approach is to require the satellite to have the exact same image acquisition conditions, including both position

[2]Temporal is a word that can be traced back to the Latin word *temporālis*, meaning *of time*, or *belonging to time*. (Lexico, 2021)

and angle. Alternatively, one can allow different image acquisition conditions, as long as an image can be taken of the same site. This definition may make more sense for agile satellites like WorldView-2 and GeoEye-1 that constantly use their gyroscopes to capture off-nadir images in all directions relative to its path. Yet, it is important to note that images taken from different off-nadir angles can present strong disparities. Revisit times for both satellites are a few days, depending on the latitude (more frequent towards the poles), spatial resolution, and maximum off-nadir angles accepted. (Maxar, 2019a, 2019c; Baghdadi & Zribi, 2016, pp. 72–73)

We will get more hands-on experience with the temporal aspect of satellite imagery in the Experiments part of the thesis (see Chapter 5 and beyond). Many of the images in our dataset are of the same area, but taken under diverse image acquisition conditions.

2.2.4. Radiometric resolution

Radiometric resolution refers to the sensitivity of a sensor within the same spectral band. A sensor with higher radiometric resolution is able to capture a wider range of radiance intensity. We say that the sensor has higher dynamic range, since the range of possible intensity values are higher. In digital images, the bit depth of the pixels serves as an indication of radiometric resolution. A common bit depth for images is 8 bits. In 8 bit images every pixel can take one of $2^8 = 256$ discrete values. WorldView-2 and GeoEye-1 have higher dynamic range. Their sensors capture 11-bit images, with pixels then being able to take one of $2^{11} = 2048$ discrete values. (Baghdadi & Zribi, 2016, pp. 73–74; Stathaki, 2011, p. 394)

2.3. Processing of satellite images

The raw images captured by a push broom scanner such as the one in Figure 2.5 are seldom used directly by any customers of satellite images. Instead, images are sent through a processing pipeline. Customers are typically able to request images processed at different levels of the pipeline. Expert imagery analysts may prefer close-to-raw images, while web map users want fully processed satellite images tuned for maximum aesthetic qualities. There is a semi-standardized hierarchy that many satellite image providers use to market and communicate their imagery. Some of the most common levels are summarized in Table 2.1. We will only deal with Level 2A imagery in this thesis. Images at this level are typically delivered in a georeferenced raster image format like GeoTIFF, and require very little tuning of basic settings like brightness and contrast to display good looking images. (Maxar, 2020; NASA, 2021a; Steele, 2018)

NASA Levels	Maxar Product	Description
Level 0		Raw, unprocessed instrument data at full resolution, with any and all communications artifacts.
Level 1B	System-Ready (Basic Imagery)	Sensor corrections: Remove known optical distortions, edge effects and artifacts. Radiometric corrections: Calibration of relative radiometric response of and between detectors, conversion to absolute radiometry.
Level 2A	View-Ready (Standard Imagery)	Atmospheric corrections: Remove atmospheric effects (haze, water vapour, particulates, sun reflectance etc.). Geometric corrections: The image is georeferenced and projected onto a coarse digital elevation model.
	Map-Ready (Ortho Imagery)	Orthorectification: The image is projected onto a more detailed digital elevation model to reduce topographic distortions.

Table 2.1: Selection and description of some common satellite imagery processing levels (NASA, 2021a; Steele, 2018)

Chapter 3

Deep learning

The introduction to deep learning in this chapter is brief and only covers the essentials needed to understand how deep learning is applied to the SISR problem. We will go through the core components of a neural network and how it is trained. By the end of the chapter you should be able to dissect and understand the model architecture of ESRGAN (skip forward to Figure 4.8 for a peek).

If you are already familiar with deep learning, this chapter may safely be skipped. If, on the other hand, you are interested in getting a broader and deeper introduction, there are plenty of options. As of 2021 the best theoretical introduction to the field is arguably still *Deep Learning* by I. Goodfellow et al., 2016 (available for free at www.deeplearningbook.org). If you prefer a more applied and hands-on approach the courses available at Andrew Ng's www.deeplearning.ai is a good alternative.

3.1. A machine learning model

Deep learning, as the term is used today, generally refers to the training of deep neural networks, i.e., artificial neural networks with multiple layers of neurons. By contrast and by most definitions, a *shallow* network may at maximum consist of an input layer, an output layer and a couple of single layers in-between, so-called *hidden* layers.[1] The trend has been for networks to become deeper and deeper. In fact, depending on the configuration and how layers are counted, the ESRGAN generator network has between 200 and 400 layers. (Schmidhuber, 2015)

Furthermore, deep learning is by most definitions considered a sub-field of machine learning (L. Deng, 2014; I. Goodfellow et al., 2016). As such it makes sense to apply machine learning terminology to deep learning models. What then is a machine learning model? The definition provided by Mitchell, 1997 is both succinct and widely used: "A computer program is said to

[1] This is a simplified way of discussing the depth of a neural network, and does for instance not take into account recurrent neural network designs. For a more thorough discussion see (Schmidhuber, 2015, pp. 6–7)

learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

Let us explore T, E and P with a simple house price prediction example.

3.1.1. The task, T

A common task T is regression. In a typical regression task we want our model to estimate a scalar value given a vector of inputs, $\hat{y} = f(\boldsymbol{x})$. For instance estimate a house price based on a set of house attributes (location, size, number of bedrooms etc.). A neural network may be employed to perform such a task, and we will revisit the regression task in Section 3.3. (I. Goodfellow et al., 2016, p. 98)

3.1.2. The experience, E

Experience E relates to how machine learning models learn from data. Learning is usually categorized as *supervised*, *unsupervised*, *reinforcement learning* or some combination of the three. Boundaries between the categories are soft. In supervised learning ground truth targets, $\boldsymbol{y}$, is provided. The model is then tasked to estimate individual scalar values $\hat{y}$ from the input vector $\boldsymbol{x}$. In our house price example, $\boldsymbol{y}$ are the actual selling prices.

In unsupervised learning there is no $\boldsymbol{y}$. Yet, it is still possible to learn useful properties from the dataset. We may for instance attempt to learn the probability distribution of the data and use this to generate new, synthetic samples. (I. Goodfellow et al., 2016)

3.1.3. The performance measure, P

In order to evaluate a machine learning model we need some way to measure its performance quantitatively. In our house price regression example one way to measure P is to measure how close our model estimates, $\hat{\boldsymbol{y}}$, are to the actual ground truth house prices, $\boldsymbol{y}$. Mean squared error (MSE) is commonly used for this purpose. It is also frequently used as a so-called loss function, L, during training (see Section 3.6). Yet, keep in mind that the performance measure, P, and the loss function, L, need not be the same. An important distinction between the two is that we are usually interested in P evaluated on data not seen by the model during training. L and P are thus evaluated on different subsets of the data, L on a training set and P on a test set. (I. Goodfellow et al., 2016)

It is established practice in the field of machine learning to partition the data into three sets; training, validation and test sets. The training set is used to estimate the model parameters, the validation set is used for selection of the model hyperparameters and the test set is used

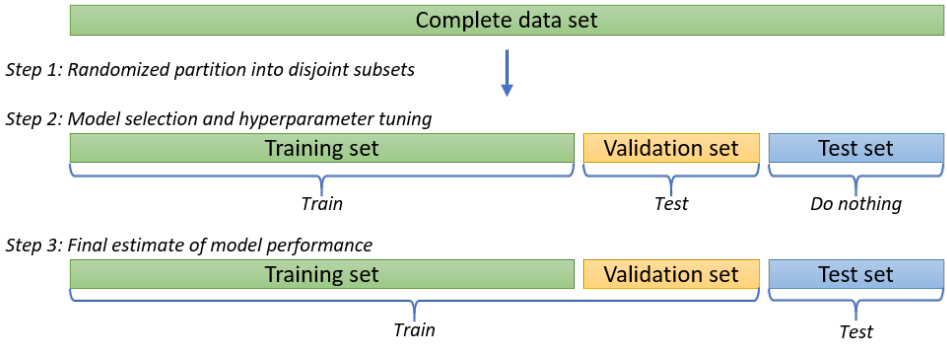


Figure 3.1: The use of training, validation and test sets during model selection and the final estimation of model performance.

to estimate the final model performance P, the generalization error, i.e., how well the model performs on completely unseen data. There are trade-offs when determining the proportional size of each set. More training data is generally associated with better performing models, so we want to maximize the size of the training set. However, we also need the validation and test sets to be large enough to provide low-variance estimates of P and the generalization error. (I. Goodfellow et al., 2016, pp. 117–118; Bishop, 2006, p. 32; Ng, 2018, pp. 13–19)

3.2. A single neuron

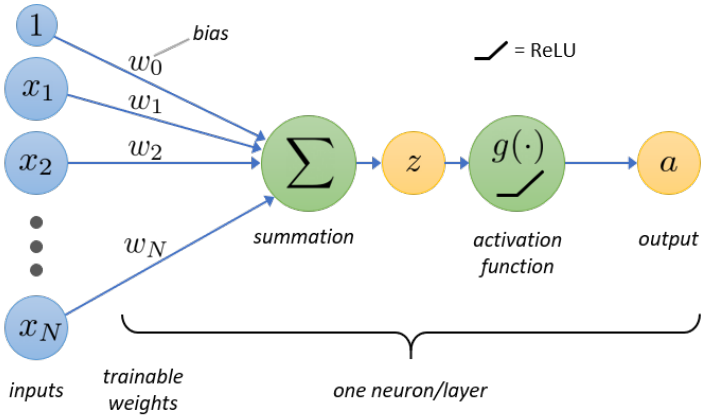


Figure 3.2: The structure of a single artificial neuron

The core component of an artificial neural network is the single artificial neuron, a data structure inspired by the biological neuron. Figure 3.2 depicts how the single neuron accepts N scalar inputs, performs a weighted sum operation and passes this sum z into a so-called activation function $g(\cdot)$ to produce a single scalar output a. We call this operation a *forward pass* and the operation is mathematically expressed in (3.1).

$$
\begin{aligned}
a = g(z) &= g\left(w_0 + \sum_{n=1}^{N} x_n w_n\right) \\
&= g\left(\boldsymbol{x}^\top \boldsymbol{w}\right)
\end{aligned}
\tag{3.1}
$$

The first parameter, w_0, is commonly referred to as the *bias* (analogous to the *intercept* in linear regression) and is sometimes denoted b. The other $\{w_n\}_{n=1}^N$ are usually referred to as the *weights* and denoted w_n. By combining the bias and the weights into a weight vector $\boldsymbol{w}^T = (w_0, w_1, \ldots, w_N)$ and a corresponding input vector $\boldsymbol{x}^T = (1, x_1, \ldots, x_N)$ we are able to express the forward pass compactly with vector notation.[2]

The single artificial neuron is interesting, but its learning ability on its own is very limited. In the next few sections we will cover how the activation function $g(\cdot)$ operates and how the weights $\boldsymbol{w}$ are learned, but to make things a bit more interesting let us first connect neurons together and introduce the artificial neural network.

3.3. Densely connected networks

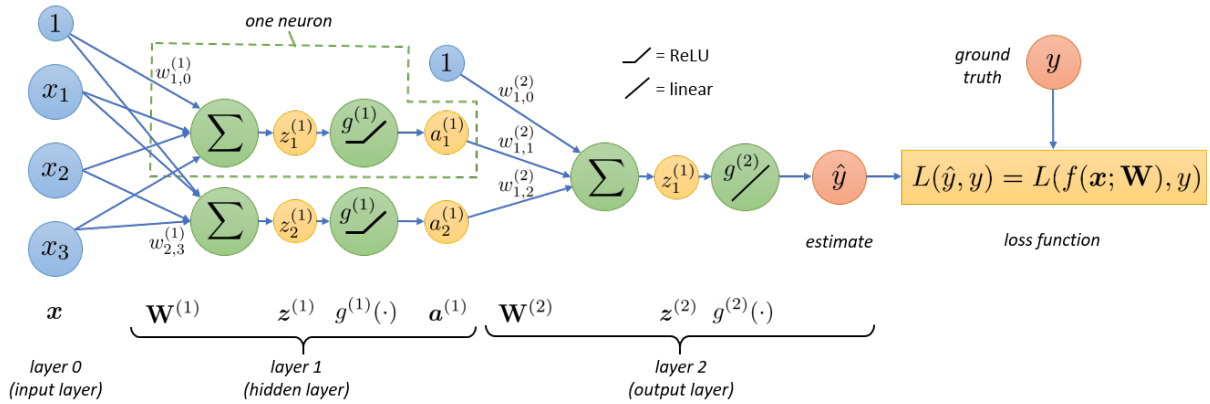


Figure 3.3: A densely connected feedforward neural network with one hidden layer. The network is applied to a supervised scalar regression problem.

When artificial neurons are organized in layers and every neuron in one layer is connected to every neuron in the consecutive layer we have a so-called *densely connected feedforward neural network*.[3] Dense because of the high density of connections and feedforward since no connections form a cycle. In Figure 3.3 we see an example of a very small such network. It is both shallow and narrow, and is by most definitions not considered a deep neural network. The number of

[2]The combination of biases and weights is in some literature referred to as *parameters* and denoted $\boldsymbol{\theta}$. We keep with terminology and notation used by I. Goodfellow et al., 2016: *weights*, $\boldsymbol{w}$ and $\mathbf{W}$

[3]A densely connected feedforward neural network is also commonly referred to as a *multilayer perceptron*.

layers defines its depth, and this network has three layers: an *input layer* (0), a so-called *hidden layer* (1) and an *output layer* (2) predicting a scalar output $\hat{y}$.

Our small network is applied to a supervised scalar regression problem. It is supervised because we know y and intend to use this knowledge to train our network. It is a regression problem since the range of y is unbounded: $y \in \mathbb{R}$. The latter is achieved by using a linear activation function, $g(z) = z$, in the output layer. In the end we notice that a loss function, L, operates on both our prediction $\hat{y}$ and the labelled ground truth y. The goal of L is to reward more correct predictions (low loss), or formulated oppositely: to penalize less correct predictions (high loss).

The activation functions, $g(\cdot)$, in Figure 3.3 operate on scalars. However, since all activation functions within the same layer are equal, it is more efficient to define $g^{(k)}(\cdot)$ as an element-wise activation function that operates on vectors.

3.4. The forward pass

Let us define our dense network mathematically and express the forward pass, from input $\boldsymbol{x}$ to prediction $\hat{y}$. We start by organizing our weights in matrices. In Figure 3.3 we notice that weights $\{w_{i,j}^{(k)}\}$ belong to layer k, counting eight weights in layer 1 and similarly three in layer 2. The weights in layer k may now be structured in a matrix $\mathbf{W}^{(k)}$. In our small network we get the following two matrices:

$$\mathbf{W}^{(1)} = \begin{bmatrix} w_{1,0}^{(1)} & w_{1,1}^{(1)} & w_{1,2}^{(1)} & w_{1,3}^{(1)} \\ w_{2,0}^{(1)} & w_{2,1}^{(1)} & w_{2,2}^{(1)} & w_{2,3}^{(1)} \end{bmatrix}, \quad \mathbf{W}^{(2)} = \begin{bmatrix} w_{1,0}^{(2)} & w_{1,1}^{(2)} & w_{1,2}^{(2)} \end{bmatrix}$$

In addition to the input vector $\boldsymbol{x}$ we also construct activation vectors $\boldsymbol{a}^{(k)}$. In our case we only have $\boldsymbol{a}^{(1)}$:

$$\boldsymbol{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad \boldsymbol{a}^{(1)} = \begin{bmatrix} 1 \\ a_1^{(1)} \\ a_2^{(1)} \end{bmatrix}$$

Putting it all together we can express the forward pass for our small network in four steps:

$$\begin{aligned} \boldsymbol{z}^{(1)} &= \mathbf{W}^{(1)}\boldsymbol{x} \\ \boldsymbol{a}^{(1)} &= g^{(1)}(\boldsymbol{z}^{(1)}) \\ \boldsymbol{z}^{(2)} &= \mathbf{W}^{(2)}\boldsymbol{a}^{(1)} \\ \hat{y} &= g^{(2)}(\boldsymbol{z}^{(2)}) \end{aligned}$$

Or expressed recursively:

$$\hat{y} = g^{(2)}\left(\mathbf{W}^{(2)} g^{(1)}\left(\mathbf{W}^{(1)}\boldsymbol{x}\right)\right) \tag{3.2}$$

Generalizing from our small network into any feedforward network we get the following set of equations that completely describe a forward pass:

$$\begin{aligned} \boldsymbol{a}^{(0)} &= \boldsymbol{x} \\ \boldsymbol{a}^{(k)} &= g^{(k)}\left(\mathbf{W}^{(k)}\boldsymbol{a}^{(k-1)}\right), \quad \text{for } k = 1, \ldots, K \\ \hat{\boldsymbol{y}} &= \boldsymbol{a}^{(K)} \end{aligned} \tag{3.3}$$

3.5. The activation function

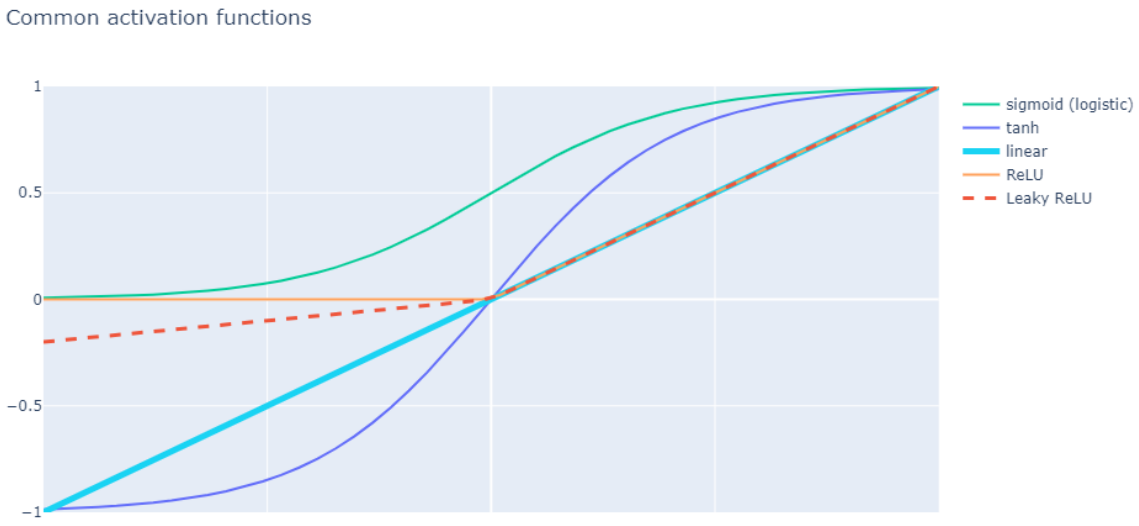


Figure 3.4: Comparison of some of the most common activation functions

The choice of activation functions $g(\cdot)$ is consequential. To illustrate this, let us for a moment revisit our single neuron example in Section 3.2. If the activation function $g(\cdot)$ in (3.1) is the *identity* function we end up with the formula for linear regression. In Figure 3.2 we briefly introduced the *rectified linear unit* (ReLU) as the activation function for our single neuron. ReLU is a piece-wise linear function defined by

$$g(z) = \max(0, z)$$

ReLU is one of the most, if not *the* most used activation function in deep networks today. It was popularized after Glorot et al., 2011 demonstrated that it vastly improved the training

of deeper networks compared to the most widely used activation functions at the time, *sigmoid* and *tanh*. ReLU is more efficient to compute, enabling faster training, and more robust to the *vanishing gradient* problem. On the flip side, ReLU can suffer from the so-called *dead neuron* problem. When $z < 0$ the gradient is 0 and this can permanently disable learning in the neuron.[4]

Several modifications of ReLU have been suggested to mitigate the dead neuron problem. Most involve adding some form of slope when $z < 0$, and thereby avoid the 0 gradient. A straightforward and widely used modification is the *Leaky ReLU* function, depicted in Figure 3.4 and defined by:

$$g(z) = \begin{cases} z, & \text{if } z > 0 \\ \alpha z, & \text{otherwise} \end{cases}$$

Leaky ReLU was introduced by Maas et al., 2013 and they suggested setting $\alpha = 0.01$. The default value of the `tensorflow.keras` implementation of Leaky ReLU on the other hand is much higher, $\alpha = 0.3$ (TensorFlow, 2020). Leaky ReLU is the activation function used in ESRGAN and we will therefore revisit it in Chapter 4.

3.6. The loss function

When training a neural network we need something to optimize for. In Section 3.1.3 we introduced the performance measure P and the loss function L. P is typically optimized indirectly through the minimization of L. In Figure 3.3, L was depicted as a function operating on a single sample: $L(\hat{y}, y) = L(f(\boldsymbol{x}; \mathbf{W}), y)$. This is a simplification. Neural networks are normally trained on so-called *mini-batches* of data samples. The term can be somewhat confusing, especially since we often denote the *mini-batch size* as the *batch size*, B. Let us consider a mini-batch of samples, $\mathbf{X} \in \mathbb{R}^{B \times D}$, $\boldsymbol{y} \in \mathbb{R}^{B}$. B is the batch size and D is the dimension, the number of independent variables, of the input. In our house price example (see Section 3.1) this could for instance be square meters, number of bedrooms, level of standard etc.

The mean absolute error (MAE) and mean square error (MSE) are two widely used loss functions for regression tasks. They are also frequently used in SISR, either directly as loss functions, L, or for model performance evaluation, P. For ESRGAN in particular, MAE is used as the loss function in the *pretraining* phase and is also a component of the loss function in the *GAN training* phase. Let us define the two and discuss how they relate to each other:

[4] The vanishing gradient and dead neuron problems are topics outside the scope of this thesis. See I. Goodfellow et al., 2016, pp. 187–190 and Szandała, 2021 for introductions to the topics.

$$\text{MAE} = \frac{1}{B}\sum_{b=1}^{B}|y_b - \hat{y}_b| = \frac{1}{B}\sum_{b=1}^{B}|y_b - f(\boldsymbol{x}_b)| = \frac{1}{B}\|\boldsymbol{e}\|_1 \tag{3.4}$$

$$\text{MSE} = \frac{1}{B}\sum_{b=1}^{B}(y_b - \hat{y}_b)^2 = \frac{1}{B}\sum_{b=1}^{B}(y_b - f(\boldsymbol{x}_b))^2 = \frac{1}{B}\mathbf{e}^T\boldsymbol{e} = \frac{1}{B}\|\boldsymbol{e}\|_2^2 \tag{3.5}$$

We see from (3.4) that MAE is a scalar multiple of $\|\mathbf{e}\|_1$, the so-called L^1 norm of the residuals vector, $\boldsymbol{e}$. Similarly from (3.5) we note that MSE and $\|\mathbf{e}\|_2$, the L^2 norm of the residuals vector, are closely related. In practice the terms are used interchangeably in the deep learning literature: MAE are often called L1 loss, and MSE called L2 loss.

3.7. Training the network

Training a neural network is all about finding the optimal weights $\mathbf{W}$ so that our model performs best at some performance measure P. Let us for a moment ignore the problem of *overfitting* and just consider our densely connected network in Figure 3.3. Here we have a loss function L, closely related to P, that we want to minimize by adjusting the values of $\mathbf{W}$. How do we do this? Minimizing L directly is intractable for non-trivial machine learning problems.[5] Instead, we take an iterative and example-based approach:

Algorithm 3.1: Training a feedforward neural network

Data: Training set of input-output pairs $\{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^N$

Input: A feedforward neural network m with initial parameters $\boldsymbol{\Theta}$

Input: A loss function L

Input: A *stochastic gradient descent*-like optimizer $\mathbf{W}$

while *stopping criterion not met* **do**

(1) Sample a mini-batch of B input-output pairs $\{\boldsymbol{x}_b, \boldsymbol{y}_b\}_{b=1}^B$ from the training set

(2) Compute $\hat{\boldsymbol{y}}$ with a forward pass through m, see equations (3.3)

(3) Compute the loss $L(\hat{\boldsymbol{y}}, \boldsymbol{y})$

(4) Compute the gradient estimate $\hat{\boldsymbol{g}}$ of the loss L with respect to $\mathbf{W}$ through backpropagation

(5) Update the weights $\mathbf{W}$ with optimizer $\text{SGD}(\mathbf{W}, \hat{\boldsymbol{g}})$

end

We have already covered step (2) and (3) in the preceding text. In the next few sections we

[5] See I. Goodfellow et al., 2016, pp. 268–275 for a discussion of how optimization of a deep neural network differs from pure optimization.

will cover step (4) and (5), but before we do that a few notes on the the mini-batch sampling in step (1) is necessary.

In step (1) of Algorithm 3.1 we sample some number of training examples from the training set and call this a mini-batch. If we instead were to proceed with all training examples in the training set, step (4) and (5) would no longer be stochastic. In step (4) we would compute the gradient $\boldsymbol{g}$, not the estimate $\hat{\boldsymbol{g}} = \nabla_{\mathbf{W}} L$, and in step (5) we would perform a *gradient descent*-like optimization, not SGD. The mini-batch size B is a hyper-parameter that can be tuned and the optimal size is usually determined by characteristics of the computational hardware, e.g., available GPU memory. (I. Goodfellow et al., 2016, pp. 271–275)

3.8. Back-propagation: Computing the gradient

In Section 3.4 we introduced the *forward pass* recursive equations (3.3). Information from the input $\boldsymbol{x}$ was *forward-propagated* through the network predicting $\hat{\boldsymbol{y}}$ (scalar $\hat{y}$ in our Figure 3.3 dense example network) and ending up as a scalar loss L. The *back-propagation* algorithm (Rumelhart et al., 1986) flips the model on its head, so to speak. We let information from the loss flow backwards all the way to our first parameters $\mathbf{W}^{(1)}$, by using the *chain rule* of calculus to compute the gradient of the loss with respect to the weights: $\nabla_{\mathbf{W}} L$. Since it is only based on a mini-batch sample of training data, not all the training data, we call it the gradient *estimate* and denote it $\hat{\boldsymbol{g}}$.

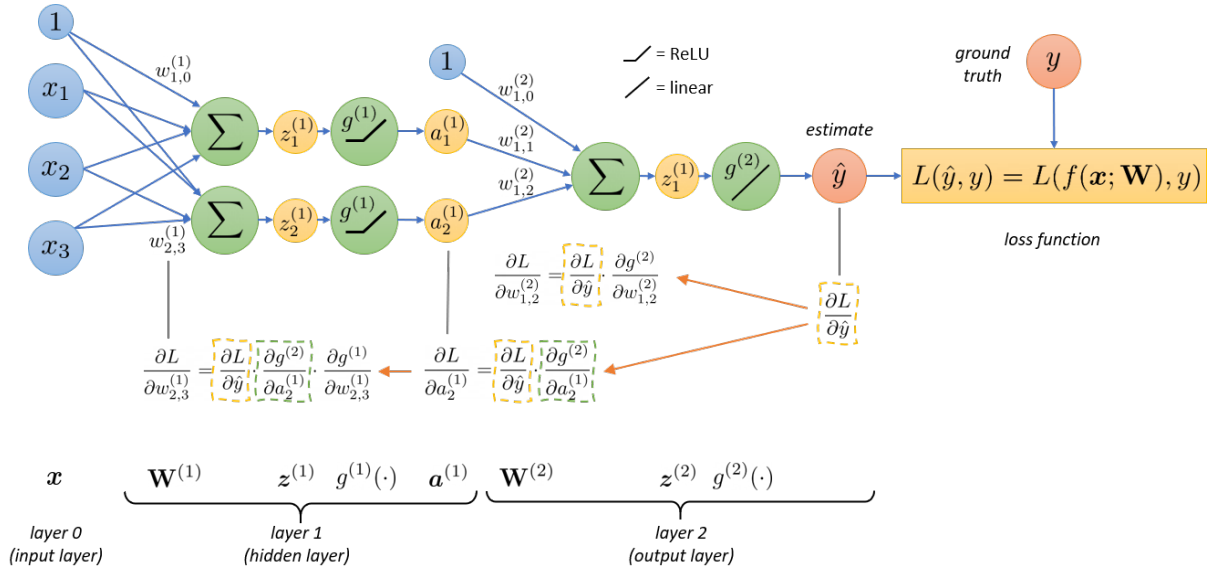


Figure 3.5: Backpropagation in a densely connected feed-forward neural network: Computing partial derivatives with the chain rule

In Figure 3.5 we see backpropagation applied to compute partial derivatives for a few of the

parameters in our densely connected network. We immediately notice the recursive nature of the algorithm: the derivatives in layer $k-1$ is dependent on the derivatives in layer k. Based on the above example, the backpropagation algorithm might seem pretty straight-forward to generalize. For feed-forward networks with a nice and tidy structure this is true, yet it requires a few pages to do it well and is slightly outside the scope of this thesis. Please refer to Chapter 6.5 in *Deep Learning* (I. Goodfellow et al., 2016, pp. 197–217) for a more thorough introduction to the topic, including how *computational graphs* and *automatic differentiation* is related to backpropagation.

What is required going forward is to accept that after completing the backpropagation step (4) in Algorithm 3.1 we have a gradient estimate $\hat{\boldsymbol{g}}$.

3.9. Stochastic Gradient Descent: Updating the weights

In the fifth and final step of Algorithm 3.1 we want to use an optimizer to update the weights of our model so that it, hopefully, performs better. In deep learning, the Stochastic Gradient Descent (SGD) optimizer and its many improved variants, dominate.

Algorithm 3.2: Stochastic gradient descent (SGD) (I. Goodfellow et al., 2016)

Data: Training set of input-output pairs $\{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^{N}$

Input: Learning rate η

Input: Initial parameters $\mathbf{W}$

while *stopping criterion not met* **do**

  Sample a mini-batch of B input-output pairs $\{\boldsymbol{x}_b, \boldsymbol{y}_b\}_{b=1}^{B}$ from the training set

  Compute gradient estimate: $\hat{\boldsymbol{g}} \leftarrow \frac{1}{B}\nabla_{\mathbf{W}} \sum_b L(f(\boldsymbol{x}_b; \mathbf{W}), \boldsymbol{y}_b)$

  Apply update: $\mathbf{W} \leftarrow \mathbf{W} - \eta\hat{\boldsymbol{g}}$

end

Algorithm 3.2 summarizes SGD. The idea behind SGD is to use gradient estimates, $\hat{\boldsymbol{g}}$, to iteratively adjust the weights, $\mathbf{W}$, step-by-step in the direction where the loss, L, is minimized the most. $\hat{\boldsymbol{g}}$ points in the direction where L increases the most. Adversely, $-\hat{\boldsymbol{g}}$ points in the direction where L decreases the most.

On the left hand side of Figure 3.6 we see the SGD stochastic step-wise approach towards a minimum on the actual loss surface. Keep in mind that the actual loss surface is invisible to the optimizer and that gradient estimates may point in wrong directions. The size of the steps, commonly referred to as the learning rate, η, is a crucial hyper-parameter. If steps are too large, SGD could easily miss (jump over) minima. If steps are too small, the algorithm may take forever to converge. Consequently, η should be tuned during training.

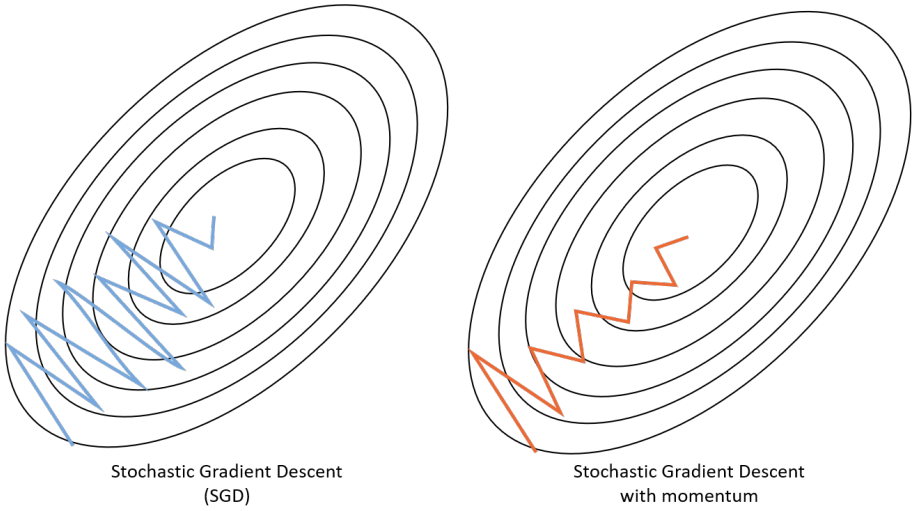


Figure 3.6: SGD with and without momentum. The contour lines represent the actual loss surface. The jagged lines represent the steps taken by SGD algorithms.

3.9.1. The Adam optimizer: Adaptive moments

Ordinary SGDs, like Algorithm 3.2, are sometimes used in deep learning today, but improved variants are the norm. Some variants have incorporated so-called *momentum*. These compute and update a *velocity* variable at every step and use this, instead of the raw gradient estimate, to update the weights. The effect of momentum may be seen on the right hand side of Figure 3.6. After a few initial jagged steps we notice that the algorithm starts taking more focused steps towards the minimum.

Algorithm 3.3: Adam optimizer (Kingma & Ba, 2014)

Data: Training set of input-output pairs $\{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^{N}$

Input: Learning rate η (Suggested default: 0.001)

Input: Exponential decay rates for moment estimates: $\beta_1, \beta_2 \in [0, 1)$

(Suggested defaults: 0.9 and 0.999 respectively)

Input: Small constant ϵ for numerical stabilization (Suggested default: 10^{-8})

Input: Initial weights $\mathbf{W}$

Initialize 1st and 2nd moment variables $\boldsymbol{s} = \mathbf{0}$, $\boldsymbol{r} = \mathbf{0}$

Initialize time step $t = 0$

while *stopping criterion not met* **do**

Sample a mini-batch of B input-output pairs $\{\boldsymbol{x}_b, \boldsymbol{y}_b\}_{b=1}^{B}$ from the training set

Compute gradient estimate: $\hat{\boldsymbol{g}} \leftarrow +\frac{1}{B}\nabla_{\mathbf{W}} \sum_b L(f(\boldsymbol{x}_b; \mathbf{W}), \boldsymbol{y}_b)$

Increment time step: $t \leftarrow t + 1$

Update biased first moment estimate: $\boldsymbol{s} \leftarrow \beta_1 \boldsymbol{s} + (1 - \beta_1)\hat{\boldsymbol{g}}$

Update biased second moment estimate: $\boldsymbol{r} \leftarrow \beta_2 \boldsymbol{r} + (1 - \beta_2)\hat{\boldsymbol{g}} \odot \hat{\boldsymbol{g}}$

Correct bias in first moment: $\hat{\boldsymbol{s}} \leftarrow \frac{\boldsymbol{s}}{1-\beta_1^t}$

Correct bias in second moment: $\hat{\boldsymbol{r}} \leftarrow \frac{\boldsymbol{r}}{1-\beta_2^t}$

Compute update: $\Delta\mathbf{W} = -\eta \frac{\hat{\boldsymbol{s}}}{\sqrt{\hat{\boldsymbol{r}}}-\beta_2^t}$

Apply update: $\mathbf{W} \leftarrow \mathbf{W} + \Delta\mathbf{W}$

end

Other variants have incorporated adaptive learning rates. In these variants, of which AdaGrad (Duchi et al., 2011), RMSProp (Hinton et al., 2012) and Adam (Kingma & Ba, 2014) are the most prominent, learning rates are adapted to the individual weights in the model at each training step. Adam is of particular interest to us, since this is the optimizer used to train ESRGAN. Adam, short for *adaptive moments*, also incorporate momentum. In the Adam optimizer algorithm, Algorithm 3.3, momentum is referred to as the *first moment estimate*, $\boldsymbol{s}$, and adaptive learning rates are incorporated through the computation of the second moment estimate, $\boldsymbol{r}$. Adam is regarded as fairly robust to the choice of hyper-parameters. (I. Goodfellow et al., 2016)

3.10. Convolutional layers

Convolutional neural networks (ConvNets) is a pivotal innovation in neural network design, and arguably *the* most important innovation in the computer vision subfield of deep learning. Introduced with the LeNet digit recognizer (LeCun et al., 1989) and further refined in LeNet-5

(Lecun et al., 1998), the significance of ConvNets really took off when AlexNet (Krizhevsky et al., 2012) won first place in the 2012 ImageNet (J. Deng et al., 2009) competition with a staggering 10.9% margin on the runner-up.

At the core of ConvNets is the convolution[6] function:

$$\mathbf{Y}(h, w) = (\mathbf{X} * \mathbf{K})(h, w) = \sum_i \sum_j \mathbf{X}(h + i, w + j)\mathbf{K}(i, j) \tag{3.6}$$

(3.6) is the 2D variant of convolution. A kernel $\mathbf{K}$ operates on a region of an input image $\mathbf{X}$ to produce an output scalar value $\mathbf{Y}(h, w)$, a pixel value at position (h, w) in the output image $\mathbf{Y}$. The effect of this operation is best understood visually. In Figure 3.7 we see a sharpening kernel, a hand-crafted kernel designed to sharpen images, operate on a 2D image input. In deep learning we let the values of the kernel be trainable parameters and update these just like we update the weights of a densely connected layer: forward pass, backpropagation and an SGD-like optimizer. In this particular case a *stride*, or step-size, of 1 is used. Larger strides may be used to further reduce the height and width of the output matrix/tensor. (I. Goodfellow et al., 2016, pp. 321–361)

In Figure 3.7 we also see convolutions extended to tensors where kernels are replaced by filters. All filters operate on the same input tensor, but outputs at different slices of the output tensor. Keep in mind that convolutions in this case is still 2D: Every slice C of the filter perform a 2D convolution only on slice C in the output tensor.

Why are convolutions so effective? Three closely related aspects are leveraged: *sparse interactions*, *parameter sharing* and *translation invariance*. Sparsity means that not every pixel in the input is connected to every pixel in the output. Sparsity is closely related to parameter sharing, the fact that the same weights are applied to the whole input. Every weight is reused multiple times during a forward pass. This particular form of parameter sharing makes a convolutional layer translation invariant. For instance, a convolutional layer in a ConvNet classifier is able to detect a particular type of feature everywhere in the image. (I. Goodfellow et al., 2016, pp. 321–361)

The kernel or filter $\mathbf{K}$ is usually much smaller than the input, drastically reducing the number of weights that has to be learned in comparison with a densely connected layer. A 3×3 kernel, such as the one in Figure 3.7, only contains $9 + 1$ trainable weights (weights + bias). For comparison, a fully connected layer in-between a 7×7 input and a 5×5 output matrix requires

[6] Actually the *cross-correlation* function, a closely related function to the mathematical convolution function. It has become convention in deep learning literature to refer to both as convolution. Most machine learning libraries actually implement the cross-correlation function and call it convolution.

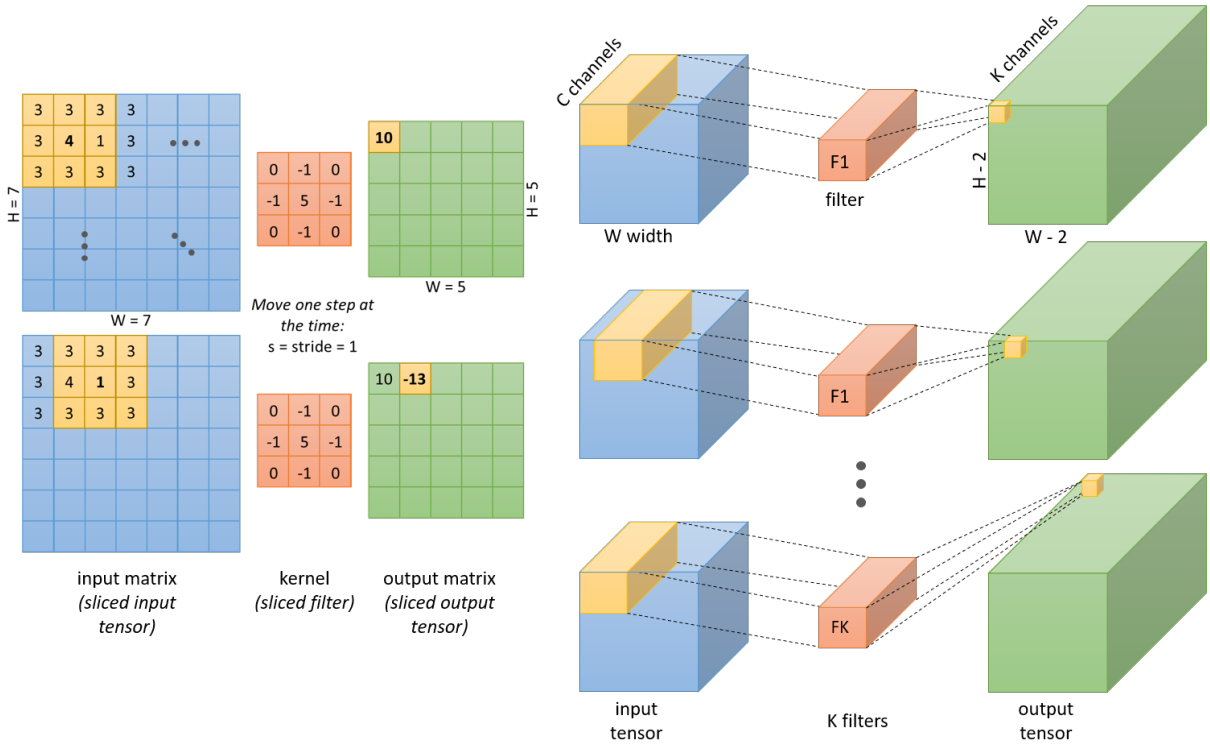


Figure 3.7: 2D convolutions. Left: A 2D convolution kernel operates with a stride of 1 on an input matrix. The values in the yellow part of the input matrix is multiplied element-wise with the kernel values. Right: K 2D convolution filters operate on the input tensor. The depth of the output tensor is determined by the number of filters.

$(7 \cdot 7 + 1) \cdot (5 \cdot 5) = 1250$ trainable weights. The reduction in number of parameters is significant.

3.10.1. Zero padding

Notice in Figure 3.7 how the width and height of the output is reduced compared with the input. This is often undesired, for instance in SISR models. The problem is usually removed through the use of a simple, yet elegant solution: *zero padding*.

The zero paddings in Figure 3.8 are of the *same* type. With same type paddings the number of zeros padded to the input is decided based on the shape of the input and the kernel. Enough are added to create an output with the same size as the input. For instance, a 5×5 kernel would require padding with two zeros in every direction. (I. Goodfellow et al., 2016, pp. 321–361)

3.11. Building and representing a network

One strength of neural networks is that it is relatively straight-forward to piece together a network consisting of various types of layers and connections. Hyper-parameters may need to be updated, but the core principles of Algorithm 3.1 still applies.

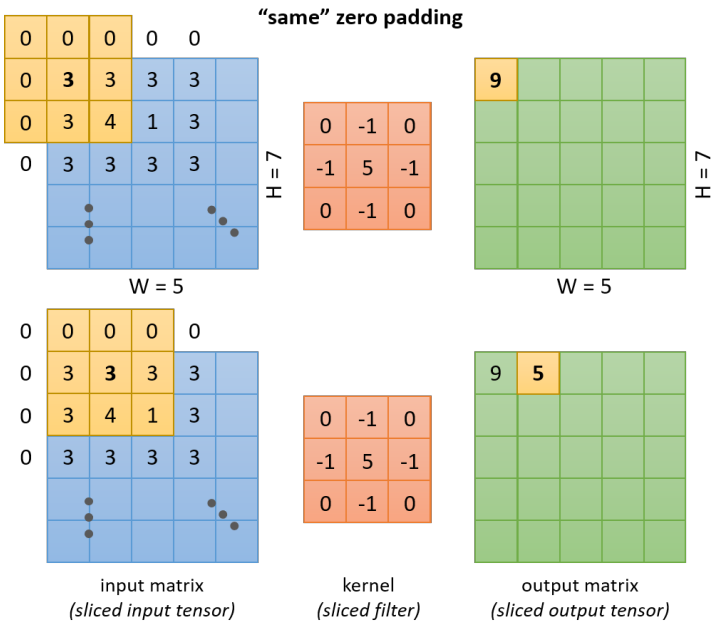


Figure 3.8: "Same" type zero padding used during 2D convolutions. Zeros are added outside the border of the original input matrix in order to let output dimensions equal input dimensions.

A common type of network is the ConvNet image classifier, a network that inputs an image and outputs either a scalar binary variable or a vector of categorical variables, depending on how many classes there are to predict. A simple ConvNet classifier is depicted in Figure 3.9.[7] We notice something that is typical for ConvNets. The width and height of the hidden tensors get smaller while the depth increases. The rationale behind this approach is that we somehow have to reduce the size of the tensors, down to in the end a scalar variable, while at the same time allow the network to pick up on different types of features in the input. This is achieved through increasing the stride and/or configuring the padding

As networks get larger a simplified way of presenting them is required. The minimal representation visible in the lower part of Figure 3.9 is typical for how networks are presented in academic papers today. Hidden tensors and their shapes are removed. Only layers and paths are visible. Such a representation may in the beginning, especially for deep and complex networks, seem a bit daunting to comprehend, yet they include everything necessary to reproduce a network in one of the common neural network libraries, e.g. TensorFlow or PyTorch. Notice how `k5n16s1` in the first 2D convolutional layer exposes the kernel size (5×5) the number of filters (16) and the stride (1). In addition the type of padding used is specified. For the densely

[7] ConvNets usually contain *pooling* layers, of which tasks are to more rapidly reduce dimensions within a network and make the network more invariant to minor changes to the input. Pooling is usually not employed in SISR models and will thus not be covered in this thesis. See (I. Goodfellow et al., 2016, pp. 330–336) for an introduction.

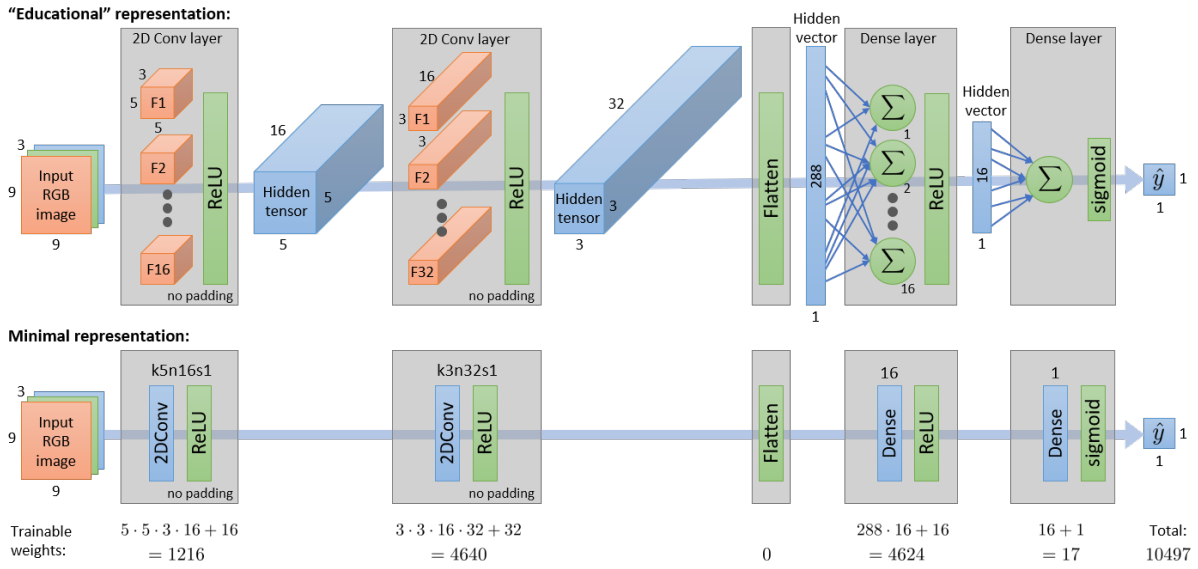


Figure 3.9: A small convolutional neural network represented in an "educational" form (top) and in a minimal form (bottom). The latter, or a similar variant, is what is usually encountered in deep learning papers. Two consecutive convolutional operations are performed on an RGB image input. Then, the hidden tensor is flattened before two densely connected layers condense all information into one final scalar variable, $\hat{y}$.

connected layers it is sufficient to specify the shape of the output (16).

3.11.1. Building a network in TensorFlow

How difficult is it to actually build our Figure 3.9 ConvNet classifier? With libraries like TensorFlow and PyTorch: Not so difficult. TensorFlow 2.3 with the Keras API is used for the experiments in this thesis. We may use the same to build the Figure 3.9 classifier:

```
model = tf.keras.Sequential()
model.add(tf.keras.layers.InputLayer(input_shape=(9,9,3,)))
model.add(tf.keras.layers.Conv2D(filters=16, kernel_size=5, strides=(1,1), activation='relu'))
model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, strides=(1,1), activation='relu'))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(units=16, activation='relu'))
model.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy', metrics='accuracy')
model.summary()
```

The final line of code prints a summary of the model:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 5, 5, 16)          1216
```

```
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 3, 3, 32)          4640
_________________________________________________________________
flatten (Flatten)            (None, 288)               0
_________________________________________________________________
dense (Dense)                (None, 16)                4624
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 17
=================================================================
Total params: 10,497
Trainable params: 10,497
Non-trainable params: 0}
```

The shapes and number of parameters match Figure 3.9 with an important exception: An extra dimension is added to every shape and this dimension is undefined (`None`). This is the mini-batch dimension, a dimension we have skipped over in all our figures above, since it would have introduced difficult-to-visualize 4D tensors. Take note that since it is undefined at build time, we may use the same model to train on, for instance mini-batches of 16 images at the time, yet when using the model for prediction we may use it on a single image.

3.12. Generative Adversarial Networks (GAN)

The networks we have introduced so far in this chapter have been *discriminative* models. This class of models make predictions $\hat{y}$ by estimating the conditional probability $p(y|x)$. Another class of models are the *generative* models. They estimate the joint probability distribution, $p(x, y)$, or just $p(x)$ if there are no labels y. A generative model may then then *generate* samples, $x*, y*$, from this estimated distribution. (Foster, 2019, pp. 1–30)

Deep neural networks are well suited to estimate probability distributions of high-dimensional inputs, such as images. Multiple powerful deep generative model designs, e.g., *Restricted Boltzmann Machines* and *Variational Autoencoders*, have been proposed over the years.[8] Yet today, the generative adversarial network (GAN) (I. J. Goodfellow et al., 2014), with its game-theoretic and intuitive approach to learning, is one of the most prominent frameworks.

In Figure 3.10 we see that a GAN consists of two separate neural networks, a generator, G, and a discriminator, D. The task of G is to generate fake outputs, $\hat{\boldsymbol{x}} = G(\boldsymbol{z})$, in practical applications often images, that are indistinguishable to real outputs, $\boldsymbol{x}$. Subsequently, the task of D is to evaluate whether images are real or fake. It is in other words learning to distinguish

[8] See (I. Goodfellow et al., 2016, pp. 645–710) for an introduction to, and a historical perspective on different deep generative model designs. See (Foster, 2019) for a more applied and hands-on introduction focusing on present day models.

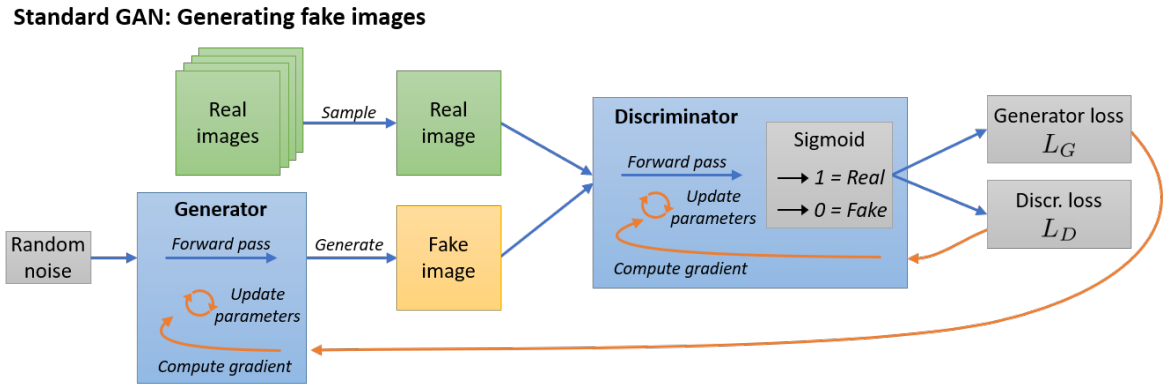


Figure 3.10: A standard GAN architecture applied to the problem of generating fake images

fake images from real images. With the appropriate training configuration and tuning of hyper-parameters, both G and D should improve its performance in tandem.

We notice two separate losses in Figure 3.10: generator loss, L_G, and discriminator loss, L_D. GAN loss functions is a topic of much research interest and the original 2014 versions are rarely used in practice today. However, if we keep to the original for now, it is commonly referred to as the *minimax* loss function[9], derived from the following minimax game:

$$\min_G \max_D L(D,G) = \mathrm{E}\left[\log D\left(\boldsymbol{x}\right)\right] + \mathrm{E}\left[\log\left(1 - D\left(G\left(\boldsymbol{z}\right)\right)\right)\right] \tag{3.7}$$

where $\boldsymbol{x}$ is a stochastic variable sampled from the distribution of training images and $\boldsymbol{z}$ is a random noise variable, often Gaussian. G tries to minimize the function while D tries to maximize it. From (3.7) we may then extract separate empirical loss functions for D and G:

$$\begin{aligned} L_D &= \log D\left(\boldsymbol{x}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}\right)\right)\right) \\ L_G &= \log\left(1 - D\left(G\left(\boldsymbol{z}\right)\right)\right) \end{aligned} \tag{3.8}$$

Both L_D and L_G in (3.8) evaluates a single image and/or noise input. We want our training algorithm to maximize L_D and minimize L_G. The difference between the two stems from the fact that G cannot directly affect the $\log D\left(\boldsymbol{x}\right)$ term. We may now formulate a GAN training algorithm:

[9]Using the term *loss function* is slightly misleading in this context since optimization of the function also involves maximization, and a loss . I. J. Goodfellow et al., 2014 used the term *value function* and *objective function*. Still, for consistency we use the *loss function* terminology.

Algorithm 3.4: GAN (I. J. Goodfellow et al., 2014)

Data: Training set $\{\boldsymbol{x}_n\}_{n=1}^{N}$

Input: Initial weights $\mathbf{W}_D, \mathbf{W}_G$

Input: Noise prior $p_g(\boldsymbol{z})$, e.g., Gaussian or uniform distribution

for *number of training iterations* **do**

  for k *steps* **do**

    Sample a mini-batch of B noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(B)}\}$ from the noise prior $p_g(\boldsymbol{z})$

    Sample a mini-batch of B examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(B)}\}$ from the training set

    Compute the gradient estimate of the discriminator: $\hat{\boldsymbol{g}}_D \leftarrow \nabla_{\mathbf{W}_D} \frac{1}{B} \sum_{b=1}^{B} L_D$

    Update the weights, $\mathbf{W}_D$, by *ascending* its gradient estimate $\hat{\boldsymbol{g}}_D$

  end

  Sample a mini-batch of B noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(B)}\}$ from the noise prior $p_g(\boldsymbol{z})$

  Compute the gradient estimate of the generator: $\hat{\boldsymbol{g}}_G \leftarrow \nabla_{\mathbf{W}_G} \frac{1}{B} \sum_{b=1}^{B} L_G$

  Update the weights, $\mathbf{W}_G$, by *descending* its gradient estimate $\hat{\boldsymbol{g}}_G$

end

In Algorithm 3.4 $k = 1$ is often used, including by the authors of the original GAN paper. We thus get a training algorithm that alternates between training the generator and the discriminator.

With this introduction to the original GAN, we conclude the chapter on deep learning in general. We will revisit GANs at the end of the next chapter when investigating the ESRGAN architecture.

Chapter 4

Single image super-resolution (SISR)

Single image super-resolution (SISR), the task of estimating a high-resolution (HR) image from a single lower-resolution (LR) image, is considered an ill-posed, inverse problem, since for every LR image input there exists multiple HR image solutions. In this chapter we will review prior research, discuss the challenges that present themselves when evaluating a SISR model and finally present the GAN-based and perceptually oriented SISR model ESRGAN in detail.

A typical approach to solving a SISR problem is to restrict the number of possible solutions by adding priors to the SISR model. A straight-forward approach is to use interpolation-based methods, like bilinear or bicubic upsampling, to deterministically generate HR pixel values from neighboring LR pixel values. The result is a smooth estimation of the HR image, usually easily distinguishable from the actual HR image because of the lack of high frequency details and sharp edges.

A more advanced approached is to learn a mapping function between LR and HR images, estimating the parameters of a statistical regression model. Today, deep learning methods dominate this approach to SISR. SRCNN (Dong et al., 2016) showed that some of the leading methods at the time (Yang et al., 2014) were equivalent to a convolutional neural network (CNN). SRCNN's design was simple, with only three convolutional layers and a mean squared error (MSE) loss function, but performed overall better than its peers on established benchmark datasets.

SISR models have since evolved at a rapid pace. Some design elements have been adopted from other deep learning computer vision models, and some are the result of SISR-specific research. Of particular importance has been the introduction of residual networks, ResNets, (He et al., 2016a), skip-connections (He et al., 2016b), the perceptual loss function (Johnson et al., 2016) and GANs (I. J. Goodfellow et al., 2014).

4.1. Perceptual quality and the perception-distortion plane

The pivotal SRGAN model (Ledig et al., 2017) implemented all of the above design elements. It focused on predicting HR images with a high human-perceived image quality, not simply a low MSE. SRGAN had a GAN architecture with a generator, in the form of a ResNet-based model with skip-connections, and a discriminator, based on the DCGAN (Radford et al., 2016) discriminator. *Perceptual loss*[1], an attempt to quantify human-perceived image quality, was implemented with the help of a separate VGG19 (Simonyan & Zisserman, 2015) convolutional image classification model, trained on the ImageNet database of labelled images (J. Deng et al., 2009). The rationale of this approach is that the VGG19 model evaluates details in the image similarly to the human visual perception system when it has learned to classify images.

SRGAN triggered a wave of research into GAN-based, perceptual quality focused SISR. The 2018 PIRM Challenge on Perceptual Image Super-resolution, a competition hosted in conjunction with the European Conference on Computer Vision (ECCV), focused on the tradeoff between minimizing distortion and maximizing perceptual quality. The perception-distortion plane (see Figure 1.2) was divided into three regions and the goal of the competition was to achieve the best perceptual quality within each region.

ESRGAN by X. Wang, Yu, Wu, et al., 2018 achieved best results in Region 3, the high perceptual quality region. As indicated by its name, ESRGAN is heavily inspired by SRGAN, yet with several modifications that both improve perceptual quality and ease training. We will revisit ESRGAN in Section 4.3.

4.2. SISR performance metrics

In Section 3.1.3 we introduced the concept of the performance measure, P, as an absolutely essential part of a machine learning model. The choice of P should motivate the choice of loss function, L. For SISR tasks, the choice of both is one of the most challenging aspects of model development and evaluation.

Evaluating the performance of a SISR model is closely related to performing an image quality assessment (IQA) of the model's predictions $\mathbf{X}_{SR}$. IQA is its own research field with applications beyond SR. Examples include image processing, compression and restoration. Within IQA the

[1] *Perceptual loss* has slightly different meanings in the SRGAN (Ledig et al., 2017) and ESRGAN (X. Wang, Yu, Wu, et al., 2018) papers. In SRGAN, it is the name of the combined loss function, including MSE loss, VGG19 feature extraction loss and adversarial loss from the discriminator. In ESRGAN, perceptual loss is simply the name of the VGG19 feature extraction loss. Since ESRGAN is the topic of this thesis we use the latter definition to avoid confusion.

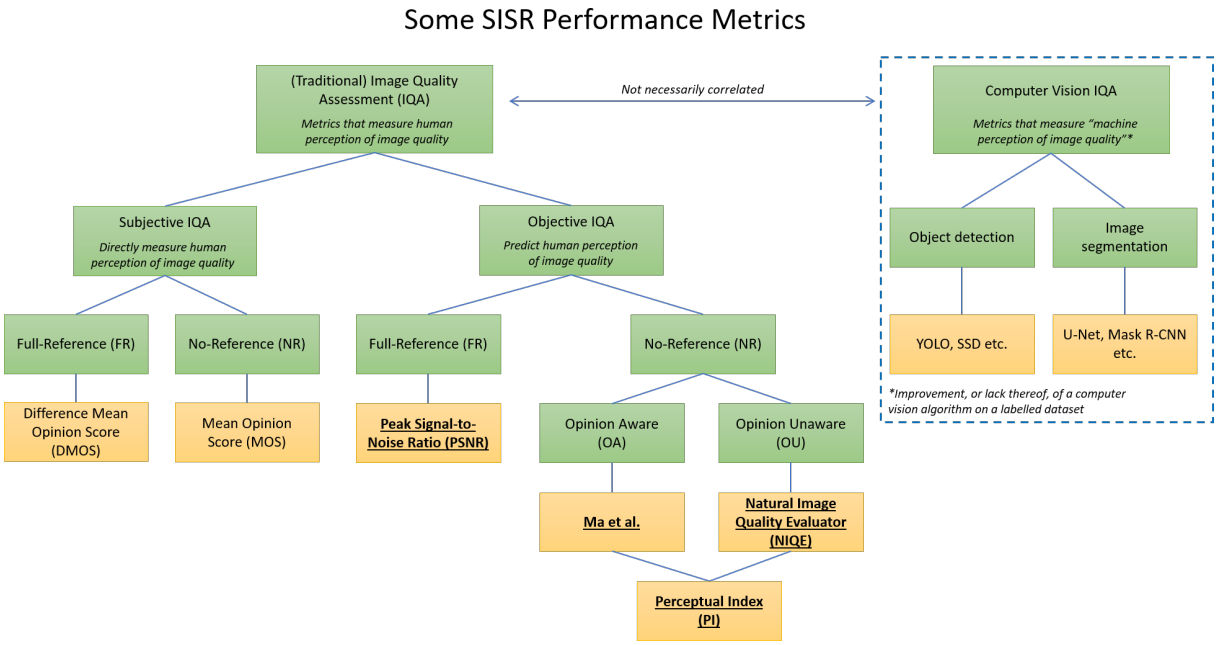


Figure 4.1: Overview of relevant SISR performance metrics organized in an IQA hierarchy. Underlined metrics are used to evaluate model performance in our experiments (see Chapter 7)

term *image quality* is usually defined to mean *perceived* image quality by *human observers* (Athar & Wang, 2019; Zhou Wang et al., 2004). If we keep by this human-centered definition it becomes necessary to define a separate assessment if our observer is non-human, e.g., an object detection algorithm. In Figure 4.1 we call this Computer Vision IQA. There are some references in the literature to the use of object detection performance as a performance metric for SISR models, also for satellite image SISR (Courtrai et al., 2020; Rabbi et al., 2020), but the idea is not well established in the IQA literature.

Computer Vision IQA is not used as a performance metric in this thesis, so while it is very much a relevant topic, we will instead focus on human-centered IQA. On the highest level it is common to divide IQA methods into *subjective* and *objective* IQA. In many ways, subjective IQA is the gold standard. When measuring image quality as perceived by human observers, nothing beats asking human observers directly. The mean opinion score (MOS) is then often used. Human observers are asked to evaluate perceived image quality on a scale, usually 1-4 or 1-5. MOS has some obvious downsides though, the most prominent being that it is expensive and time-consuming. Consequently, there is a need for algorithmic methods that correlate well with subjective IQA methods. We call these objective IQA methods. Their goal is to estimate the human-perceived image quality. (Z. Wang & Bovik, 2006, pp. 1–3; Athar & Wang, 2019, p. 1)

In Figure 4.2 we see how two objective IQA metrics correlate differently with MOS. Based

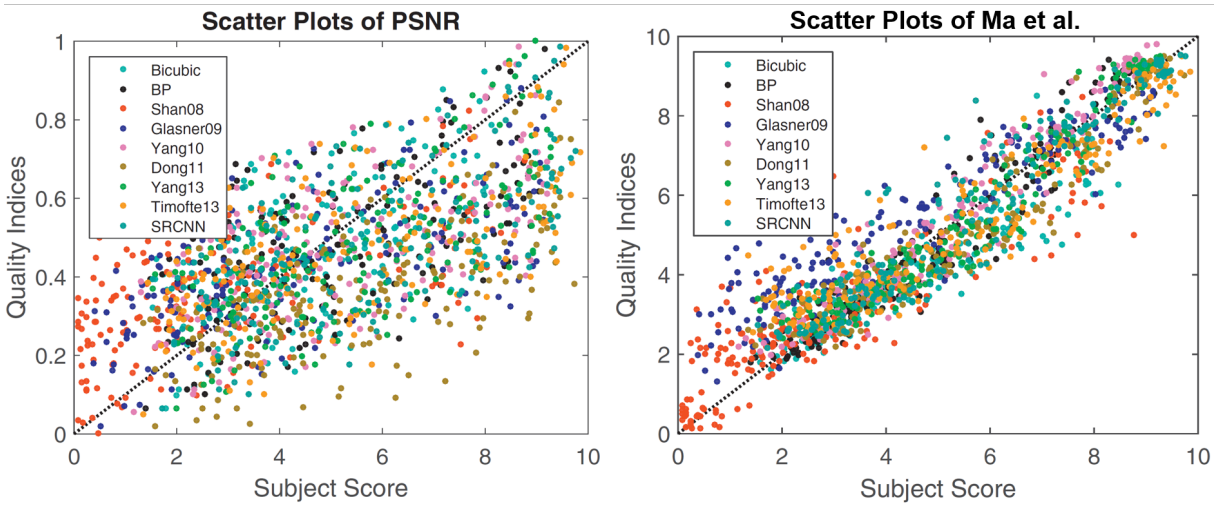


Figure 4.2: Scatter plots comparing the objective IQA methods PSNR and Ma et al. (Y-axes), with the Mean Opinion Score (MOS) (X-axes). Every point is a super-resoluted image, colored by SISR algorithm. Reprinted from Ma et al., 2017. Copyright 2021, with permission from Elsevier

on these scatter plots alone, there seem to be strong evidence in favor of the Ma et al. metric over the PSNR metric.

In the overview of IQA metrics (Figure 4.1) we also notice the terms No-Reference (NR) and Full-Reference (FR). A FR IQA method compares the distorted image, i.e., a SISR estimate, with the ground truth image, the reference image. A FR method is thus reliant on the existence of a reference image, something that is seldom available in real-world applications of IQA. A NR IQA method, on the other hand, is only a function of the distorted image itself. FR and NR has its strengths and weaknesses. In SISR research it is common to report metrics from both categories, as long as a reference image is available. (Athar & Wang, 2019)

4.2.1. Peak Signal-to-Noise Ratio (PSNR)

The Peak Signal-to-Noise Ratio (PSNR), a metric derived from the well-known MSE we introduced in Section 3.6, has historically been the default objective IQA method. A decibel scale is used to increase interpretability compared to a raw MSE metric. The historical standing, combined with its ease of use and good interpretability, is probably the reason why it is still used extensively in SISR today. PSNR is historically defined for one-channel (grayscale) images only. For color images, a color space transformation, i.e., an RGB-to-HSV, is usually performed so that PSNR may be computed only on the value/brightness (V) channel. (Athar & Wang, 2019)

$$\text{PSNR} = 10 \log_{10} \frac{L^2}{\text{MSE}} = 20 \log_{10} \frac{L}{\text{MSE}} \tag{4.1}$$

where L is the dynamic range of pixel intensities. In many image applications $L = 2^8 - 1 =$

255, the dynamic range of an 8-bit image. In the case of 11-bit WorldView-2 and GeoEye-1 satellite images, $L = 2^{11} - 1 = 2047$. L may also be a decimal number, for instance 1.0, a typical value when dealing with preprocessed images in neural networks. The use of L makes PSNR, to a certain extent, invariant to the dynamic range of the image.[2] (Horé & Ziou, 2010; Maxar, 2019a, 2019c)

Despite its status as a default IQA metric, it has been repeatedly shown that it correlates poorly with subjective IQA methods. It relies on assumptions that do not hold when measuring visual perceptual quality. For instance, PSNR is independent of any spatial relationships in the image; it is only a pixel-by-pixel evaluation. Shifting the whole image by one pixel in any direction would lead to a disastrous PSNR score, yet probably no noticeable difference when evaluated by a human observer. We illustrated this with a checkerboard example in Section 1.2. For a comprehensive, and as a matter of fact funny, evaluation of PSNR, see Z. Wang and Bovik, 2009.

A note on MSE and MAE of 2D, 3D and 4D tensors

We introduced MSE (3.4) and MAE (3.5) as functions operating on vectors. In SISR problems we are usually interested in 2D ($H \times W$) or 3D ($H \times W \times C$) images stored in tensors. Additionally, during training the loss is computed over a mini-batch of images, extending the number of dimensions to 4 ($B \times H \times W \times C$). Luckily, the extra dimensions does not add much complexity to the computation of MAE and MSE. Given pairs of SISR predictions $\mathbf{X}_{SR}$ and ground truth HR images $\mathbf{X}_{HR}$ we compute the residuals $\mathbf{E}$

$$\mathbf{E} = \mathbf{X}_{HR} - \mathbf{X}_{SR}, \quad \text{where } \mathbf{E},\ \mathbf{X}_{HR},\ \mathbf{X}_{SR} \in \mathbb{R}^{B \times H_{hr} \times W_{hr} \times C}$$

MAE and MSE are then computed element-wise. This is equivalent to vectorizing, or flattening, $\mathbf{E}$ into $\mathbf{e}$.

$$\mathbf{e} = \text{vec}(\mathbf{E}), \quad \mathbf{e} \in \mathbb{R}^{BH_{hr}W_{hr}C}$$

By doing this we may then use the vector notation in (3.4) and (3.5), with a modified denominator:

$$\text{MAE} = L1 = \frac{1}{BH_{hr}W_{hr}C} \|\mathbf{e}\|_1 \tag{4.2}$$

$$\text{MSE} = L2 = \frac{1}{BH_{hr}W_{hr}C} \mathbf{e}^T\mathbf{e} = \frac{1}{NH_{hr}W_{hr}C} \|\mathbf{e}\|_2^2 \tag{4.3}$$

[2] Rounding errors occur, especially for integer data types.

4.2.2. Natural Image Quality Evaluator (NIQE)

The Natural Image Quality Evaluator (NIQE) is a frequently used so called opinion-unaware no-reference IQA algorithm introduced by Mittal et al., 2013. The metric is on a continuous scale where most images receive a score between 0 and 10, and a lower score is better.

A reference model is created by fitting a multivariate Gaussian model to a training set of natural and pristine images. Let us consider a distorted image. At evaluation time the same model is fitted to the distorted image. The NIQE score is then a function of the distance between the parameters of the two model fits. A shorter distance means that the distorted image is more similar to the training set of natural and pristine images. By this measure one can say that the distorted image is more natural and pristine, i.e., it has a higher perceptual quality. (Mittal et al., 2013)

NIQE is a no-reference metric since the algorithm does not depend on having access to a reference image. It is opinion-unaware because it has not been trained on human-evaluated distorted images. NIQE is only aware of how natural and pristine images look and use this knowledge to estimate image quality. (Athar & Wang, 2019)

4.2.3. Ma et al.

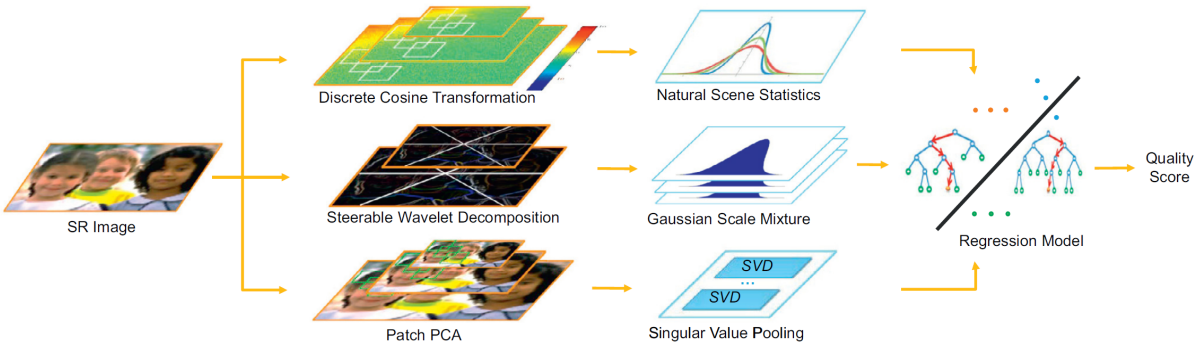


Figure 4.3: The main steps of the Ma et al. performance measure. Reprinted from Ma et al., 2017 Copyright 2021, with permission from Elsevier

Ma et al., 2017 proposed an image quality metric specifically to evaluate SISR algorithms. They did not explicitly name their algorithm, so it is commonly referred to as the *Ma et al. metric* or just *Ma metric* for short. The metric is on a continuous $[0, 10]$ scale where a higher score correlates with higher perceptual quality. Similarly to NIQE it is also a no-reference measure. However contrary to the opinion-***un****aware* NIQE, Ma is opinion-*aware*. A regression model has been fitted to a dataset of human-evaluated super-resoluted images. This also has the added effect of making it optimized to pick-up on typical SR artefacts.

The Ma et al. model exploits three types of statistical properties to quantify artifacts and assess the quality: local frequency variations, global frequency variations and spatial discontinuity. In Figure 4.3 we see how these three types of features are extracted through the use of a quite a few different algorithms. Subsequently, a random regression forest is fitted to each of the three features, before finally all scores are combined with linear regression, outputting a final scalar quality score. (Ma et al., 2017)

4.2.4. Perceptual Index (PI)

ESRGAN was first introduced as a contender in the 2018 PIRM Challenge on Perceptual Image Super-Resolution, where the model won first place in one of the disciplines. The challenge sought to reward both accuracy and perceptual quality and as a consequence they introduced the Perceptual Index (PI) as a measure for perceptual quality. (Blau et al., 2019)

$$\mathrm{PI} = \frac{1}{2}\left((10 - \mathrm{Ma}) + \mathrm{NIQE}\right) \tag{4.4}$$

As evident from (4.4) PI combines two other performance metrics, Ma and NIQE. As described in the preceding few sections these metrics evaluate image quality in two distinctly different ways. PI should therefore be a relatively robust measure of perceived image quality. In fact Blau et al., 2019 validated different measures against human-rated MOS on the 2018 PIRM images. They found that PI correlated well with MOS at least on this particular set of images.

4.3. ESRGAN

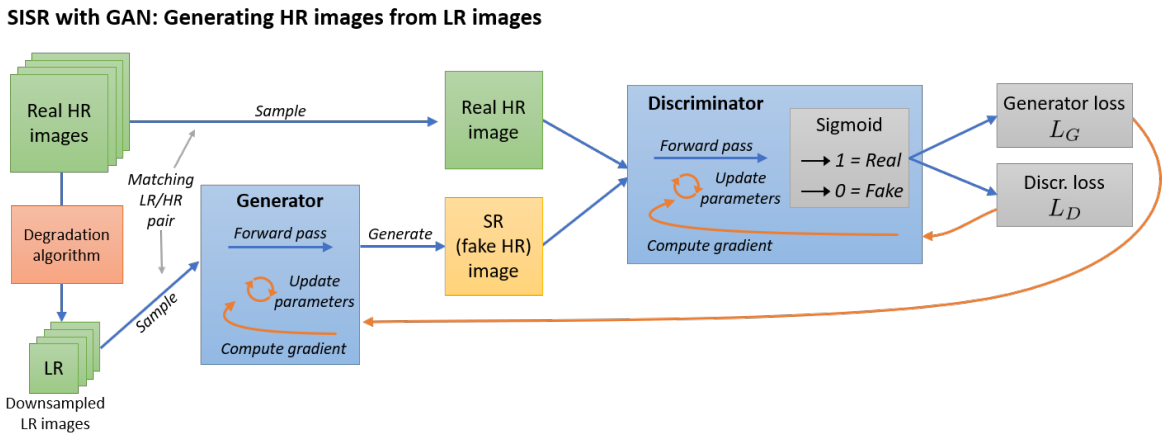


Figure 4.4: A standard GAN architecture applied to SISR where HR images are downsampled/degraded into LR images and presented as an input to the generator.

Before diving into the details of ESRGAN, let us take a bird's eye view of how a GAN can be

applied to the SISR task. In Section 3.12 we introduced GAN applied to the task of generating realistic-looking images from random noise (see Figure 3.10). In Figure 4.4 we have modified the design slightly, replacing random noise with a set of LR images. In theory such a GAN model could work. However, in practice, a number of modifications have been made in both SRGAN and ESRGAN that optimize both performance and training. The high-level modifications are depicted in Figure 4.5.

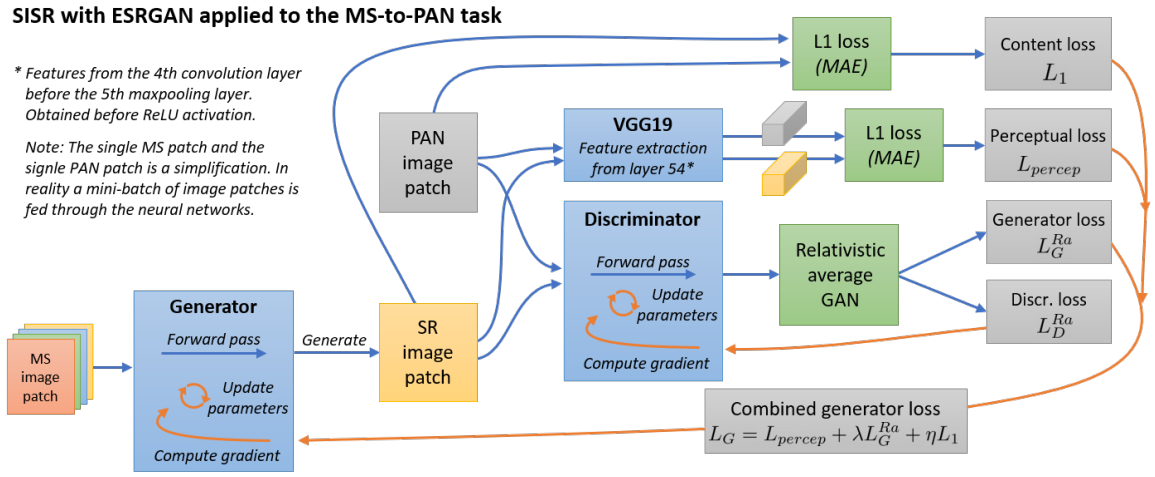


Figure 4.5: GAN training of ESRGAN on the MS-to-PAN task. One training iteration includes forward passes, backpropagation of the loss to compute gradients and the updating of weights in both the generator and the discriminator.

Note that Figure 4.5, and all subsequent illustrations of ESRGAN, include some modifications made for the MS-to-PAN task that is the topic of this thesis. We summarize those in Section 5.4.1.

ESRGAN is trained in two phases: First a pretraining phase where only the L1 content loss (4.2) is used to train the generator, then a GAN training phase where the full set of losses depicted in Figure 4.5 is applied. Pretraining is done in order to produce a reasonable starting point for GAN training. Two reasons for pretraining are mentioned in the ESRGAN paper. Firstly, pretraining helps the generator avoid undesired local optima. Secondly, it helps the generator focus more on texture discrimination and detail from the start. They used the Adam optimizer (see Section 3.9.1) and trained the model for 400k pretraining and 400k GAN training iterations on the DIV2K (Agustsson & Timofte, 2017), Flickr2K (Timofte et al., 2017) and the OutdoorSceneTraining (X. Wang, Yu, Dong, et al., 2018) image datasets. For a full list of ESRGAN configuration hyper-parameters, see Table 5.4.

4.3.1. ESRGAN loss functions

In the GAN training phase the loss function for the generator consists of the L1 content loss, the perceptual loss and the relativistic average GAN (RaGAN) loss:

$$L_G = L_{\text{percep}} + \lambda L_G^{Ra} + \eta L_1 \tag{4.5}$$

λ and η are hyperparameters that adjust the relative significance of each loss component. X. Wang, Yu, Wu, et al., 2018 used $\lambda = 0.005$ and $\eta = 0.01$ for their experiments, and the same values were used for the MS-to-PAN experiments in this thesis.

Perceptual loss

Figure 4.6: Deep feature extraction from a VGG19 network trained on ImageNet. SRGAN and ESRGAN extracts the feature tensor at different locations. The layers to the right of the feature extraction is not used. (Ledig et al., 2017; Simonyan & Zisserman, 2015; X. Wang, Yu, Wu, et al., 2018)

The perceptual loss, L_{percep}, used in ESRGAN builds on the foundations of Johnson et al., 2016 and SRGAN (Ledig et al., 2017). A separate convolutional neural network image classifier, VGG19 (Simonyan & Zisserman, 2015), trained on the ImageNet database of labelled images (J. Deng et al., 2009), is used as a proxy for the human visual system. In Figure 4.6 we see how features are extracted from deep within the VGG19 network. Both the ground truth HR and the SR images are forward propagated through the network and the difference is computed with L1 loss (4.2). A lower loss indicates that the SR image has more similar perceptual characteristics to the HR image.

X. Wang, Yu, Wu, et al. demonstrated that extracting features before an activation layer increases the perceptual quality of SISR outputs. They argue that this is because activation layers turn off a large percentage of neurons, leading to weaker supervision and less information in the gradients.

Relativistic average GAN (RaGAN) loss

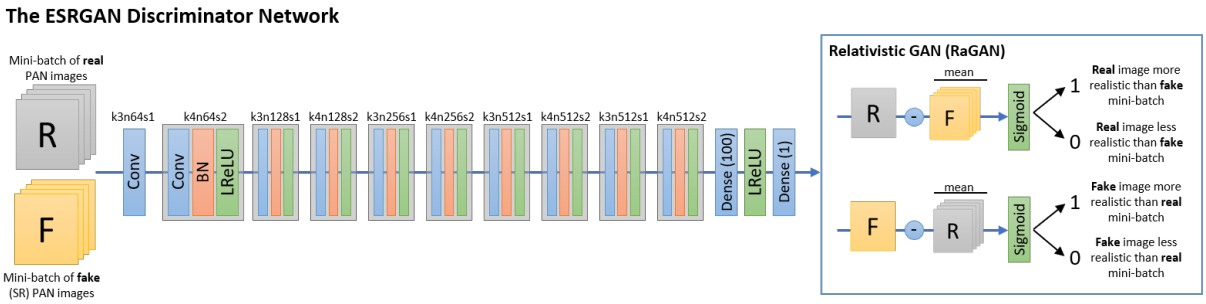


Figure 4.7: The ESRGAN Discriminator Network with the RaGAN configuration at the end. RaGAN is illustrated computed over a single real image and a single fake image. In actual implementations all images in the mini-batch are used in the computation.

SRGAN and ESRGAN employs a discriminator network architecture that builds on the well-established DCGAN (Radford et al., 2016) guidelines. SRGAN uses the standard minimax GAN loss function (3.7) we introduced in Chapter 3. The authors of ESRGAN have employed a different loss function, Relativistic average GAN (RaGAN) (Jolicoeur-Martineau, 2018), that they argue perform better on SISR tasks. Instead of simply estimating the probability of whether an image is fake or real, RaGAN estimates the probability that a real (HR) image is more realistic than the *average* fake (SR) image, with the average (arithmetic mean) computed over the mini-batch of images. Vice versa, it also estimates the probability that a fake (SR) image is more realistic than the average real (HR) image. This concept is depicted on the right side of Figure 4.7.

The mathematical representation of RaGAN is somewhat involved. If we first express the standard discriminator output in the original GAN as $D(\mathbf{X}) = \sigma\left(C(\mathbf{X})\right)$, where $\sigma(\cdot)$ is the sigmoid output activation function and $C(\mathbf{X})$ is the output of the preceding layer in the discriminator, then RaGAN formulated on single HR and SR images, $\mathbf{X}_{\mathrm{HR},b}$ and $\mathbf{X}_{\mathrm{SR},b}$ belonging to a mini-batch of HR-SR image pairs, $\{\mathbf{X}_{\mathrm{HR},b}, \mathbf{X}_{\mathrm{SR},b}\}_{b=1}^{B}$, is:

$$
\begin{aligned}
D_{Ra}\left(\mathbf{X}_{\mathrm{HR},b}, \mathbf{X}_{\mathrm{SR}}\right) &= \sigma\left(C\left(\mathbf{X}_{\mathrm{HR},b}\right) - \frac{1}{B}\sum_{b=1}^{B} C\left(\mathbf{X}_{\mathrm{SR},b}\right)\right) \\
D_{Ra}\left(\mathbf{X}_{\mathrm{SR},b}, \mathbf{X}_{\mathrm{HR}}\right) &= \sigma\left(C\left(\mathbf{X}_{\mathrm{SR},b}\right) - \frac{1}{B}\sum_{b=1}^{B} C\left(\mathbf{X}_{\mathrm{HR},b}\right)\right)
\end{aligned}
\tag{4.6}
$$

We can think of (4.6) as replacing the standard sigmoid output activation function that operates on the HR and SR mini-batches individually, with a stateful sigmoid function that operates on both mini-batches. (4.6) is then used in loss functions similar to the GAN minimax

loss functions (3.8). Formulated as operating on mini-batches of images, and with minus signs introduced so that both L_D^{Ra} and L_G^{Ra} can be minimized, we get the following two loss functions:

$$
\begin{aligned}
L_D^{Ra} &= -\frac{1}{B}\sum_{b=1}^{B}\left(\log D_{\mathrm{Ra}}\left(\mathbf{X}_{\mathrm{HR},b}, \mathbf{X}_{\mathrm{SR}}\right) + \log\left(1 - D_{\mathrm{Ra}}\left(\mathbf{X}_{\mathrm{SR},b}, \mathbf{X}_{\mathrm{HR}}\right)\right)\right) \\
L_G^{Ra} &= -\frac{1}{B}\sum_{b=1}^{B}\left(\log\left(1 - D_{\mathrm{Ra}}\left(\mathbf{X}_{\mathrm{HR},b}, \mathbf{X}_{\mathrm{SR}}\right)\right) + \log D_{\mathrm{Ra}}\left(\mathbf{X}_{\mathrm{SR},b}, \mathbf{X}_{\mathrm{HR}}\right)\right)
\end{aligned}
\tag{4.7}
$$

Notice that the generator loss, L_G^{Ra}, in (4.7) contains both $\mathbf{X}_{\mathrm{SR}}$ and $\mathbf{X}_{\mathrm{HR}}$, something that is not the case for the loss functions in SRGAN. The authors of the ESRGAN paper argue that this improves the ability of the generator to learn sharper edges and more detailed textures. In addition, Jolicoeur-Martineau, 2018 argues for several other favorable properties of RaGAN, including increased stability during training.

4.3.2. ESRGAN Generator

We conclude this chapter on SISR and ESRGAN with a review of the ESRGAN generator network architecture. From a first glance at the generator in Figure 4.8 the jump in complexity from the small ConvNet introduced in Section 3.11, and even the VGG19 (Figure 4.6) and discriminator network (Figure 4.7) seems significant. A more thorough inspection however, reveals that the network consists of the same basic building blocks: convolutional layers, activation layers and skip-connections. It is just a matter of how you put them together.

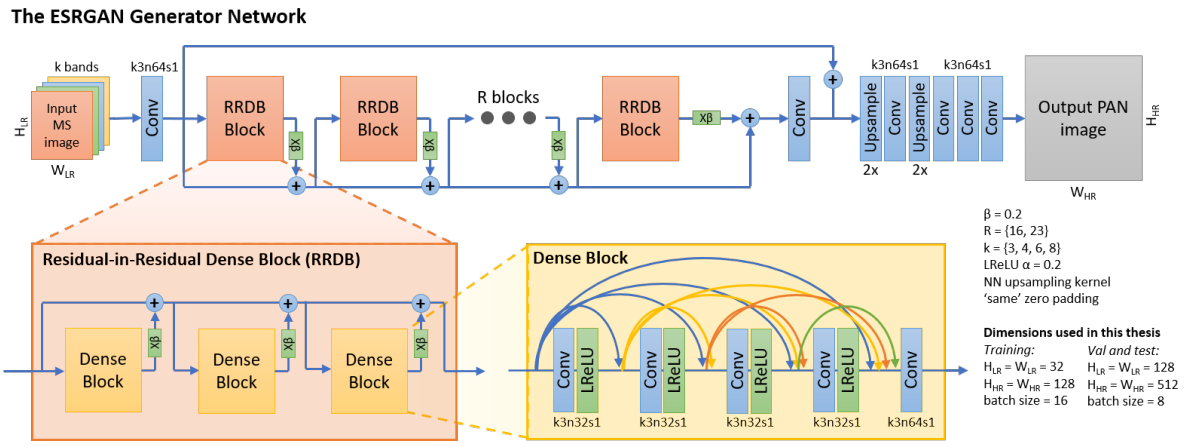


Figure 4.8: The ESRGAN Generator Network with Residual-in-Residual Dense Block and Dense Block architecture. The input and output dimensions of this version is modified from the original so that it can be applied to the satellite MS-to-PAN task.

The design is inspired by the SRGAN (Ledig et al., 2017) generator, which again builds on previous designs like ResNet (He et al., 2016a). A historical review of convolutional image generators is beyond the scope of this thesis. Our focus in the next few paragraphs will thus be on the differences between SRGAN and ESRGAN.

The ESRGAN authors made four changes to the SRGAN generator. First, they removed all so-called *batch normalization* (BN) layers (Ioffe & Szegedy, 2015). BN layers are not covered in this thesis since they are not employed in ESRGAN. Consequently we will not do a deep dive on the topic here, but suffice to say BN layers normalize, re-centers and re-scales a mini-batch inside the network, and the layers are widely used due to their positive effects on training speed and stability. It has however been demonstrated by Lim et al., 2017 that they do not work as well for PSNR-oriented tasks, including SISR. Furthermore it was observed by the authors of ESRGAN that BN layers introduced artifacts, especially for deeper networks trained under a GAN framework.

The second change they made to the SRGAN generator was replacing the straight-forward Residual Block, a block consisting of two convolutional layers, two BN layers and an activation layer, with the novel Residual-in-Residual Dense Block (RRDB) depicted in Figure 4.8. RRDB is inspired by the Residual Dense Block (Zhang et al., 2018) and the Densely Connected Convolutional Network (G. Huang et al., 2017). The RRDB block-within-another-block design increases the depth and capacity of the network substantially. Additionally, all the skip connections ensure that the gradient flows all the way back to the first layers. We thus avoid the vanishing gradient problem (see Section 3.5).

The third and fourth changes are relatively minor. Residual scaling is used to scale down residuals by multiplying with a hyper-parameter, β, between 0 and 1. This prevents instability during training (Lim et al., 2017; Szegedy et al., 2017). Lastly, Leaky ReLU (3.5) is used as activation function throughout the ESRGAN generator. SRGAN uses Parametric ReLU (He et al., 2015), a version of Leaky ReLU where the α is a trainable parameter.

Part II

Experiments on satellite imagery

Chapter 5

Experimental design

In this chapter we will introduce the experiments and the data and see how they relate to the research questions posed in Chapter 1. In addition, we will look at which changes and adaptations we have made to the default ESRGAN architecture and training configuration.

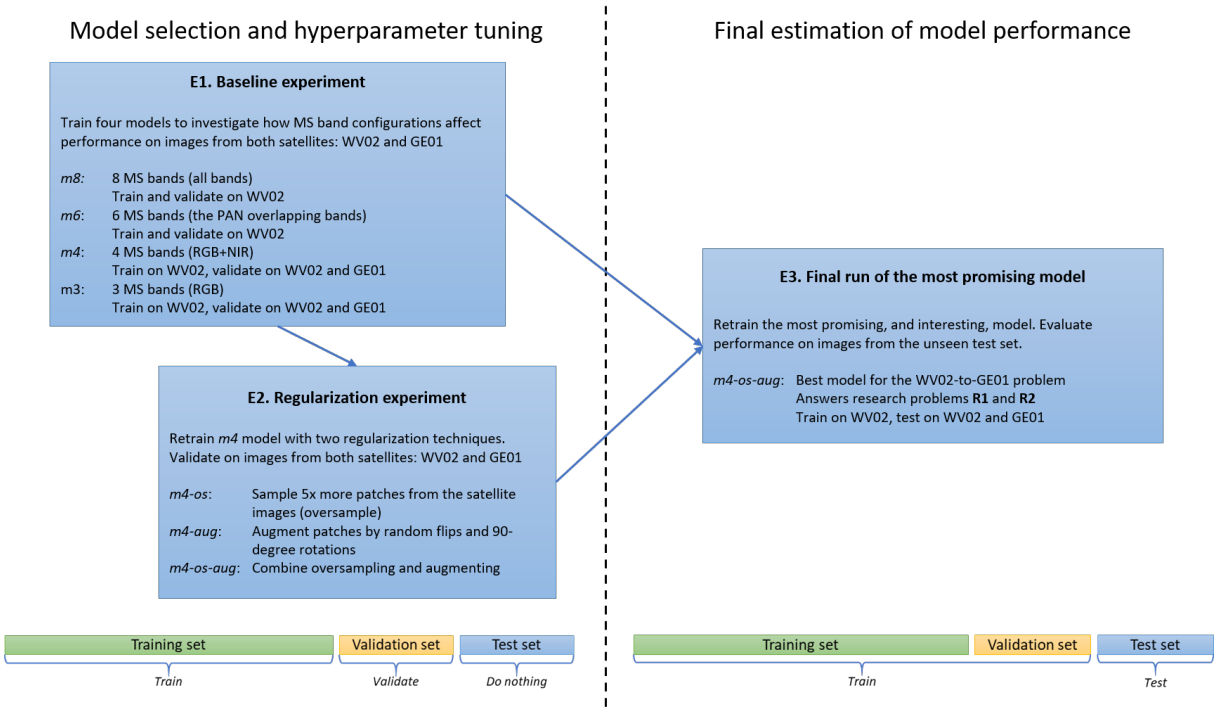


Figure 5.1: Experimental design

For convenience we repeat the three research questions:

R1: To what extent can the higher-resolution 0.5 m GSD PAN band be reconstructed from the lower-resolution 2.0 m GSD MS bands?

R2: To what extent can the model trained on images from satellite A be used to super-resolute images from a similar satellite B?

R3: Can we increase performance by introducing regularization, in the form of data augmentation and over-sampling of patches from the satellite images? If so, by how much?

The first two relate to how well SISR work on the MS-to-PAN problem, either internally on images from the same satellite (R1: training and testing on images from the same satellite A), or across satellites (R2: training on images from satellite A, testing on images from satellite B). WorldView-2 was selected as satellite A and GeoEye-1 selected as satellite B.

As seen in Figure 5.1, we have designed a baseline experiment (E1) that will provide insight into these questions. In the baseline experiment we focus on investigating how different number of MS bands affect performance, instead of absolute performance. Why is this important? WorldView-2 has eight MS bands, while GeoEye-1 has four. While we may use all eight available WorldView-2 bands when researching R1, we are restricted to a maximum of four bands when researching R2. It is therefore of interest to study if and by how much performance is decreased when reducing the number of bands.

Our third research question relates to increasing absolute performance, particularly the ability to generalize well across sensors (R2). We have designed a regularization experiment (E2) where we test a couple of regularization techniques on the otherwise unregularized ESRGAN model.

Ultimately, we run our most interesting and promising model through a final performance evaluation (E3) on completely unseen data from the test set. This reduces the risk of an over-optimistic, overfitted final performance evaluation. (See Section 3.1.3 for more on the topic).

We will revisit the experiments in Section 5.2 after an introduction of the dataset.

5.1. The data: Two Mediterranean towns

The selection and acquisition of the satellite images was done in collaboration with FFI. We had access to search the Maxar satellite image archive. Together we developed some criteria for which areas to acquire images of. We wanted to (a) have a high number of archived satellite images available, (b) train and test on two or more similar and comparable areas, (c) have the images contain a diverse collection of maritime vessels (civilian and military), and finally (d) have the images contain built-up areas (buildings, roads, vehicles etc.).

The criteria are a result of general research interests at FFI as well as a wish to study satellite images with high content variability, i.e., images with diverse details. The latter should increase the generalisability of any findings and frankly make a SISR project more interesting.

It was soon identified that criterion (c) limited the number of potential areas. Most military vessels are located either at sea or at naval bases, and most present-day naval bases are located away from civilian infrastructure and harbors. In Europe there are however some naval bases which for historical reasons are located within the natural boundaries of a town's harbor. The search for potential areas were therefore focused on naval bases in Europe.

Potential naval bases were identified and then vetted against criterion (a) . The number of available images of an area is primarily driven by previous image customer interest. It is mostly the customers of a satellite imaging company that order image collection. The number is also heavily influenced by the typical cloud and sun conditions in the area. Consequently there are orders of magnitude more images of sunny and populous cities like San Diego than an uninhabited patch of land along the dark and rainy coasts of the North Atlantic. We therefore found there to be more images available of naval bases in the Mediterranean since conditions here are drier and sunnier than other coastlines in Europe.

5.1.1. Introducing Toulon and La Spezia

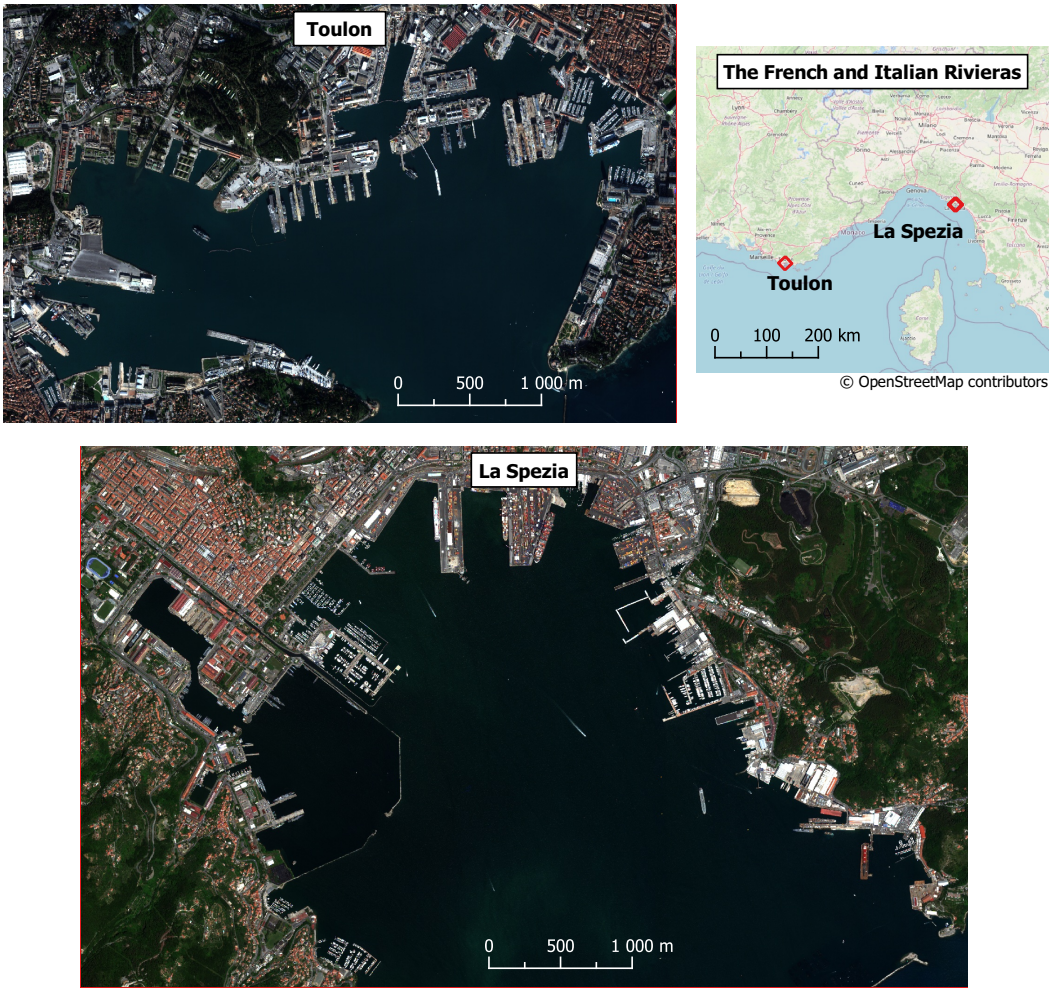


Figure 5.2: Satellite images of Toulon and La Spezia and their location on the French and Italian Rivieras – *Satellite image © 2021 Maxar Technologies*

Toulon on the French Riviera and La Spezia across the border, on the Italian Riviera, are two historical naval towns located only 330 km apart. Both have military and civilian ports as well as a combination of historical and modern architecture and street structure. In Figure 5.2

we see the boundaries of the two areas that images have been acquired from. The acquisition boundaries have been set with a focus on including harbor infrastructure. Consequently the more compact nature of Toulon's harbor has resulted in a smaller acquisition boundary for this town, ca. 13.7 km^2, than for that of La Spezia, ca. 23.5 km^2.

Town \ Satellite	WorldView-2 8 MS + 1 Pan	GeoEye-1 4 MS + 1 Pan	Total
La Spezia (ca. 23.5 km^2)	22	11	35
Toulon (ca. 13.7 km^2)	20	9	38
Total	42	20	62

Table 5.1: Contingency table with number of images by areas and satellite sensor

We also note that large portions of both images consist of sea surface. In addition, some images are cloudy. From a SISR perspective, sea and cloud surfaces are monotonous and uninteresting compared to the rest of the images. The fact that these surface types combined dominate versus all other surface types (buildings, roads, ships, vegetation etc.) means we have a severe data imbalance problem that could be detrimental to the performance of a robust CNN based model (Buda et al., 2018). Initial training of ESRGAN confirmed that this indeed was a problem. We observed overfitting of the model to sea surfaces and mode collapse during GAN training. To correct for this imbalance we trained and implemented a *cloud and sea classifier* and used this to significantly undersample sea and cloud patches. We will revisit this classifier in Section 6.3.2.

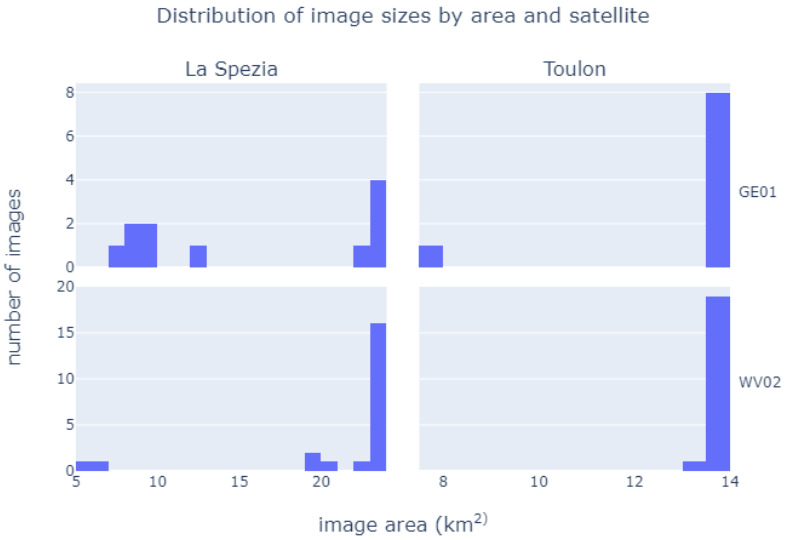


Figure 5.3: Distribution of image sizes in km^2 by town and satellite sensor

FFI acquired 62 images that partly or fully overlap the acquisition boundaries in Figure 5.2.[1] Table 5.1 depicts the distribution and we note a good balance between the number of images from satellites WorldView-2 and GeoEye-1 as well as between the two towns. It is however worth noting that actual images vary in size as some only partly cover the acquisition boundaries. How the image sizes are distributed across the two towns and two satellites can be seen in Figure 5.3. We note that small image sizes are concentrated in La Spezia and most prominently in the set of GeoEye-1 images of La Spezia. This may lead to less representative and robust performance measures of this particular combination of town and sensor, something we are taking into account in the next section.

5.1.2. Data partition: training, validation and test sets

In Section 3.1.3 we introduced the practice of dividing the complete dataset into three disjoint subsets: training, validation and test. We have followed the same practice for the experiments, and the distribution is depicted in Table 5.2. The images were sampled with Python NumPy's pseudorandom number generator. For the WorldView-2 images we used a 50/25/25 split (train/-val/test), with one exception: images that covered less than 50 % of the area in Figure 5.2 were designated to the training set. This was done in order to increase the representativeness of the validation and test set. For the GeoEye-1 images we used a 0/50/50 split since no images from this satellite should be used for training (see the introduction to this chapter).

	Town			Satellite		
Partition	La Spezia	Toulon	All	GE01	WV02	All
Training	12	10	22	0	22	22
Validation	10	9	19	9	10	19
Test	11	10	21	11	10	21
All	33	29	62	20	42	62

Table 5.2: Contingency table with number of images in train, validation and test sets across town and satellite sensor

Finally, a training set size of 22 should at face value raise alarms for anyone familiar with neural networks. We are however boosting this number considerably by sampling smaller patches from the larger satellite images. In our baseline experiment (E1) we are for instance training on almost 130 000 image patches extracted from the 22 large satellite images. We elaborate further on this topic in Section 6.1.

[1] 74 images were originally acquired, 11 of which were taken by the WorldView-3 satellite. These are not included in our experimental design. In addition, one image was found to only consist of opaque clouds and consequently discarded, bringing the total to 62.

5.1.3. Temporal correlations

A consequence of our dataset being concentrated on two towns is that we get multiple overlapping images taken of the same area, but at different times. In Figure 5.4 we see an example of how this looks in practice. It is natural to believe that temporal correlations exist and that these impact the performance on the training set as well as the validation and test set. For instance, in Figure 5.4 we notice a distinct looking roof in the center of the image patches. A neural network may start memorizing this specific roof after some period of training. Depending on the task at hand this could be problematic, since memorization could impact the network's ability to digest new information in a new, previously unseen, MS input image. For instance, an explosion could blow a hole in the roof, large enough to be clearly visible in the MS image. A model relying too much on memorization could then interpolate and thus remove the hole in its SISR prediction.

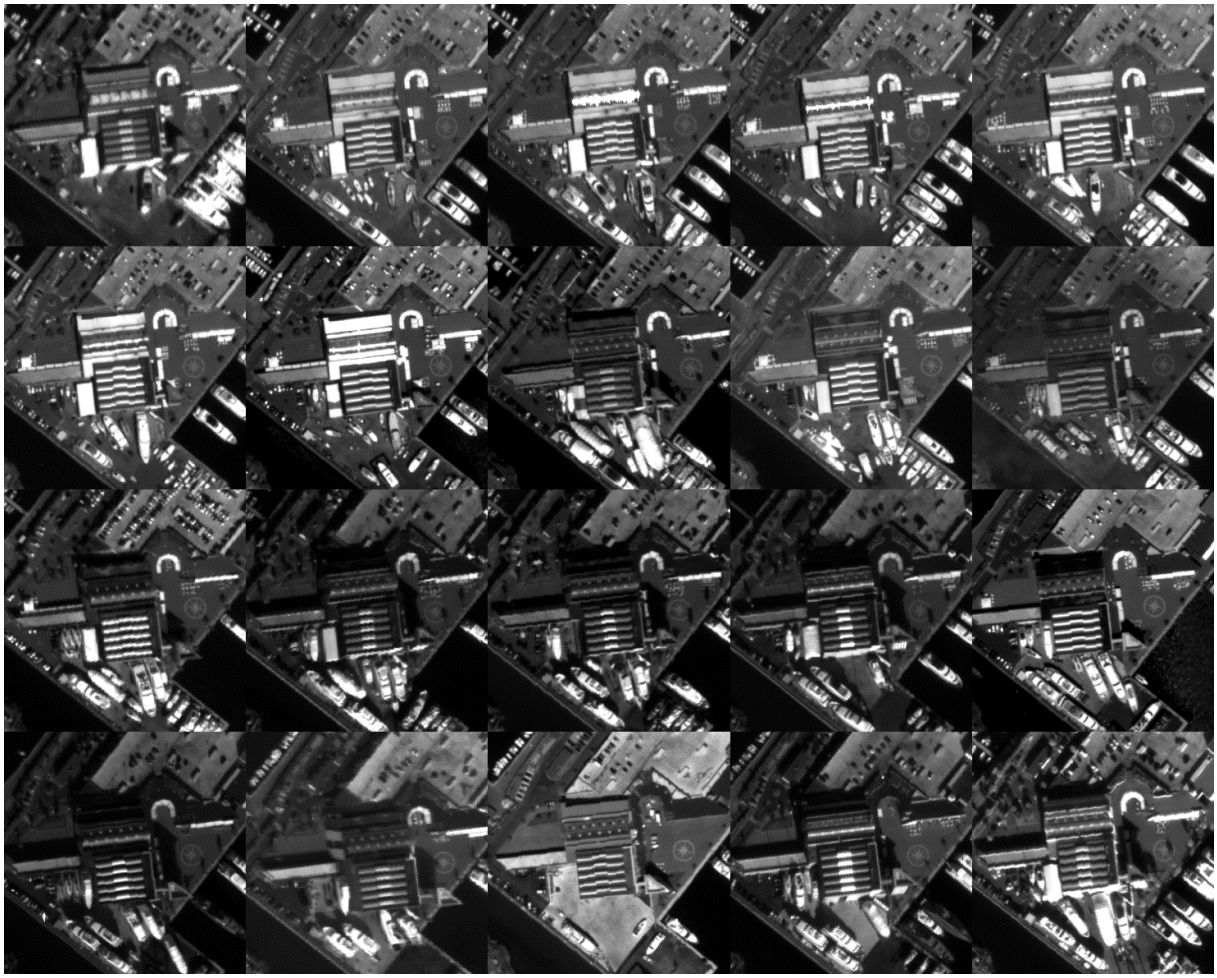


Figure 5.4: Image patches of the same location extracted from multiple satellite images taken at different times
Satellite image © 2021 Maxar Technologies

As we have not investigated the impact of temporal correlations, it is unclear to what extent memorization affects performance. Based on anecdotal inspections of image patches, such as

in Figure 5.4, we could also hypothesize that there is enough variation from image to image, especially at the pixel level, that pure memorization is a non-viable strategy for a neural network anyway. Yes, images are taken at the same location, but angles and light conditions are different. In addition, ground activities like construction, moving vehicles and people, make the content of images taken of the same location at different times quite different. In E2 and E3 we also implement data augmentation (flips and rotates) on the image patches (see Section 5.2.2). This should make it harder for a neural network to memorize specific features across time.

5.2. The experiments

5.2.1. E1. The baseline experiment

As mentioned in the introduction to this chapter, E1 is designed to answer research questions R1 and R2. From this perspective m8 and m4 should be the most interesting models. In m8 we let ESRGAN use all available information, all WorldView-2 MS bands, and hypothesize that this will enable the best possible reconstruction of the WorldView-2 PAN band. In m4 we let ESRGAN use only four of the WorldView-2 MS bands, the four particular bands that are also available in GeoEye-1 images (see Figure 2.7). We are curious about whether training on these WorldView-2 images, without any pre-processing or transformations to imitate GeoEye-1 MS characteristics, is enough to produce acceptable SR performance on the unseen GeoEye-1 images.

	WorldView-2 (WV02)								GeoEye-1 (GE01)			
Model name	0: Coastal	1: Blue	2: Green	3: Yellow	4: Red	5: Red Edge	6: NIR	7: NIR2	0: Blue	1: Green	2: Red	3: NIR
m8	X	X	X	X	X	X	X	X				
m6		X	X	X	X	X	X					
m4		X	X		X		X		X	X	X	X
m3		X	X		X				X	X	X	

Table 5.3: Models and band combinations in the baseline experiment (E1).

In addition, we include models m6 and m3 to check whether simpler models perform equally well, or even better, than m8 and m4 respectively. The rationale behind m6 is that it only includes MS bands that actually overlap with the PAN band on the electromagnetic spectrum (see Figure 2.7). m3 is included to investigate the performance without the NIR band. This is relevant because certain satellites, for instance early versions of Planet's Dove satellites, only capture RGB.

5.2.2. E2. The regularization experiment

In the regularization experiment we investigate whether and to what degree regularization increase the performance of our models. Regularization should improve the generalizability of the models, i.e., reduce the generalization error and improve performance on validation and test sets. After an initial review of the ESRGAN architecture and the data available we hypothesized that a data-centric approach to regularization is effective, seeing as the training set is somewhat limited in size. (More on the sampling of patches in Chapter 6)

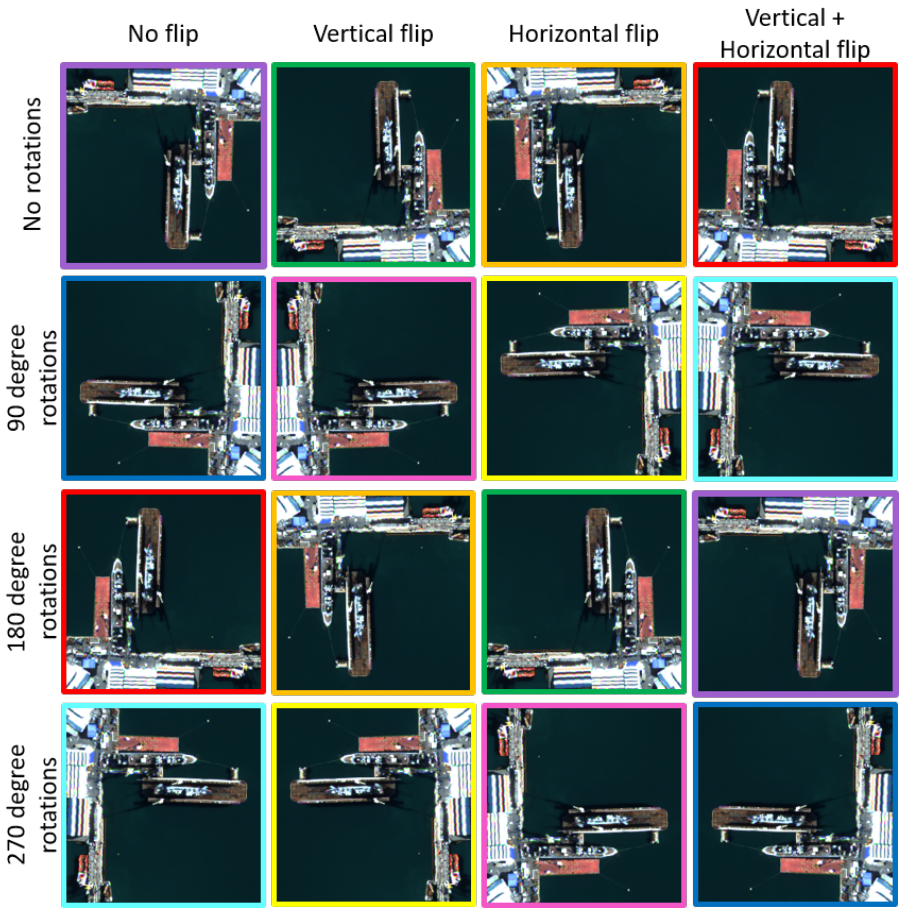


Figure 5.5: Flips and 90 degree rotations applied to an MS patch. There are 16 combinations at face value, but on further inspection we notice that only 8 are unique. *Satellite image © 2021 Maxar Technologies*

In E2 we test two regularization techniques on the m4 model from E1. Why m4? We found m4 to be the most interesting model emerging from E1 (see Chapter 7 for more) as it allows evaluation on both WorldView-2 and GeoEye-1 data (R1 and R2). In addition it performed better than its competitor, m3, in E1. A total of three models were trained and evaluated in E2:

- m4-os: Sampling five times as many patches from the 22 training images than what was done in the baseline (m4). This *oversampling* (os) would result in a large number of overlapping image patches. The expected number of patches covering a unique square

meter is set to 10, producing approximately 650k image patches.

- m4-aug: Randomly augment patches using two data augmentation methods: random flips and random 90 degree rotations. We consider these methods conservative/safe in the context of the MS-to-PAN problem since neither pixel resampling nor any pixel value manipulation that could change the spectral signature of an area is performed. In Figure 5.5 we notice that one image patch is augmented into eight different versions through the combination of the two methods.
- m4-os-aug: Combine oversampling and the random data augmentation described above.

5.2.3. E3. The final evaluation

In E3 we want to perform a final performance evaluation of the most promising and interesting model identified through E1 and E2, m4-os-aug. To avoid effects of hyperparameter overfitting this evaluation should be done on the completely unseen test set (see Figure 3.1 and the accompanying section). The validation set is now included in the training set, adding much needed variation to our training set. If our m4-os-aug model generalize well we should *not* observe a considerable dip in performance across the performance metrics, P, of interest.

5.3. Training, logging and evaluation

All models were trained in two phases: 400k iterations of pretraining with L1 loss and 400k iterations of GAN training with the composite loss function (4.5).[2] Subtracting for time spent on validation, the total training time hovered around four days, with pretraining taking slightly less than two days and GAN training taking slightly more. In Figure 5.6 we see how the SISR output changes as training progresses.

Tensorboard was used to log metrics during experiments. In addition, model weights were saved to disk every 1000 training iterations, enabling time travel and post-experiment evaluation of performance at different iterations of the two training phases.

All logs and some model weights are publicly available under the MIT license in the project's GitHub repository. The satellite images are not publicly available and would have to be acquired from Maxar Technologies. However, metadata about the images is available in the GitHub repository and in Appendix C, enabling ordering of the exact same images and extents by others who would like to either reproduce or build on the experiments in this thesis.

[2] One iteration defined as including the forward pass, the backward pass (backpropagation) and the updating of the model weights.

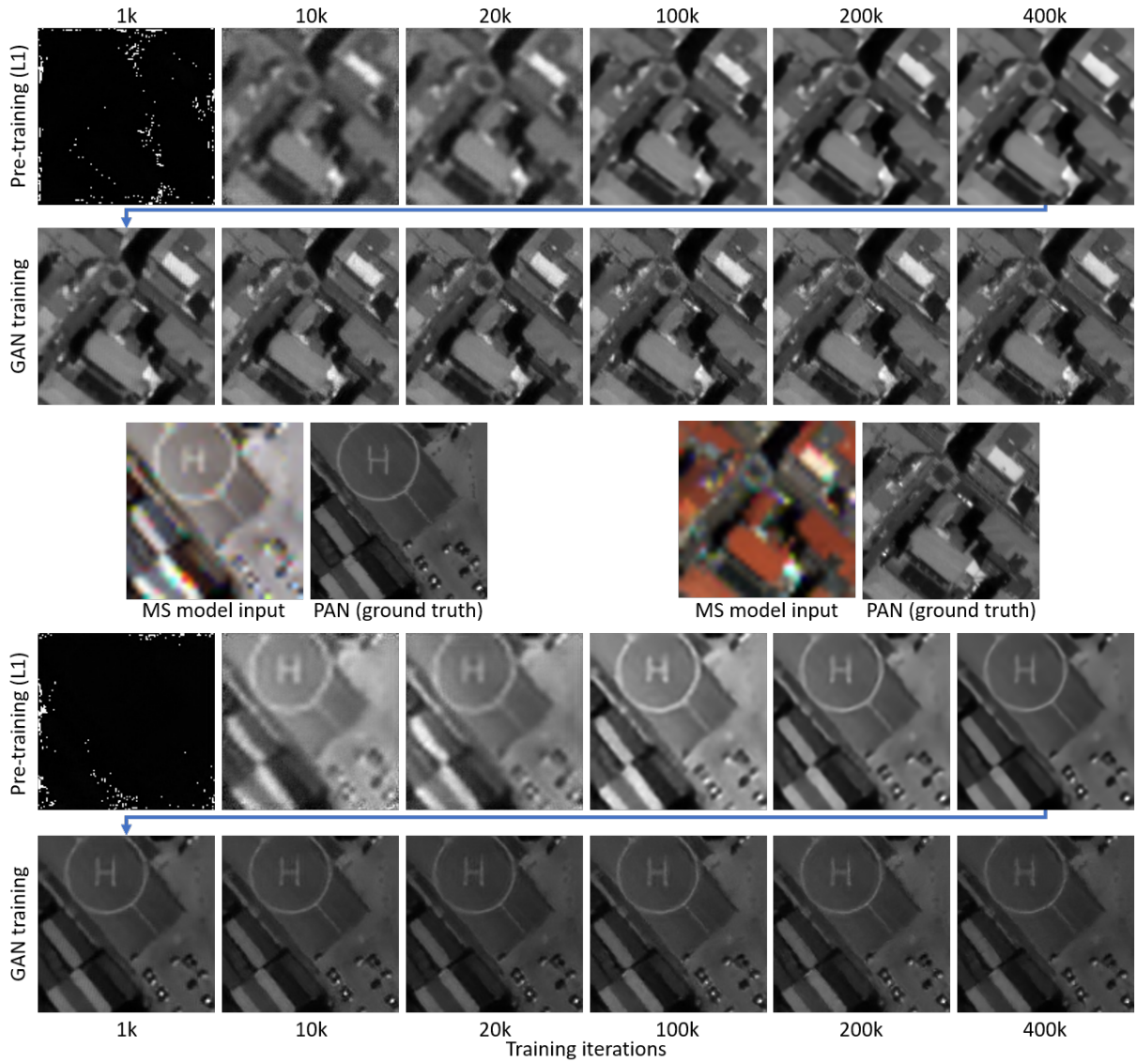


Figure 5.6: Image patches at different stages of pretraining and GAN training. The training set image patches were extracted during training of the m6 model in Experiment 1. Top: La Spezia 2018-07-06. Bottom: Toulon 2014-04-06 *Satellite image © 2021 Maxar Technologies*

5.3.1. Computing NIQE, Ma et al., and Perceptual Index (PI) metrics

We wanted to evaluate performance on the perception-distortion plane (see Figure 1.2), using the same metrics as in the original ESRGAN paper (X. Wang, Yu, Wu, et al., 2018) and the PIRM Challenge (Shoeiby et al., 2019). These metrics were NIQE, Ma et al. and PI (see Section 4.2). Official releases of the algorithms are MATLAB implementations and to our knowledge there are no working and validated Python implementations. [3]

This left two alternatives: (1) Evaluate performance in a separate MATLAB environment,

[3]The scikit-video Python package includes a NIQE function, but during testing this algorithm failed to reproduce the same results as the official implementation.

or (2) call on the MATLAB functions from our Python environment with the MATLAB Engine API for Python. As the first option would be unnecessarily cumbersome, we opted for the second option. We implemented calls to the official MATLAB NIQE and Ma et al. libraries from inside our TensorFlow Keras ESRGAN model and computed PI in TensorFlow. Consequently we were able to evaluate performance on these metrics just as if they were native TensorFlow metrics like MSE and PSNR. To our knowledge this is a first, at least among publicly available source code and research.

5.4. Adapting ESRGAN to the MS-to-PAN task

The unofficial TensorFlow 2 implementation of ESRGAN by K.-Y. Huang, 2020 was selected as the starting point for this project.[4] We have since made significant changes to the code base. Some changes are unrelated to the specifics of the MS-to-PAN task, such as our implementation of ESRGAN as a subclassed TensorFlow Keras Model (`tf.keras.Model`), allowing for the use of nifty and time-saving features already implemented by the Keras team. Other changes are very specific to the MS-to-PAN task, such as designing a data pipeline that enables efficient training on GeoTIFF satellite images. Since the data pipeline is not directly related to ESRGAN, it can be used to train other models as well, we have dedicated a separate chapter, Chapter 6, on how we designed and implemented our data pipeline.

Changes to the ESRGAN network architecture and training configuration are relatively few, and we will cover them in the next few sections. For a complete overview of all/most configuration hyperparameters, not just the ones that differ from the original ESRGAN , see Table 5.4.

5.4.1. Changes to the network architecture

There are really only three changes to the ESRGAN network architecture (also visible in Figure 4.5 and Figure 4.8):

1. No degradation algorithm is needed to produce LR images from HR images. MS images are used directly as input to the generator.
2. The number of channels in the input layer is configurable and depends on the number of MS bands that will be used. The original implementation used 3 (RGB).
3. The number of output channels is fixed to one, since the PAN image consists of only one band. The original implementation used 3 (RGB).

[4] PyTorch is used in the official implementation (X. Wang, 2019). The author was and is more proficient in the use of TensorFlow than PyTorch, and this is the rationale behind the choice of an unofficial implementation as a starting point.

Training settings	Value	
	Pretraining	**GAN-training**
Optimizer:		
Training iterations	400k	400k
Type of optimizer	Adam	
Generator learning rate	$5 * 10^{-5}$	$2 * 10^{-5}$
Discriminator learning rate		$2 * 10^{-5}$
Generator β_1	0.9	0.9
Generator β_2	0.999	0.999
Generator ϵ		0.9
Discriminator β_1		0.9
Discriminator β_2		0.999
Generator ϵ		10^{-7}
Loss coefficients:		
L_1	1	0.01
L_{percep}	0	1
L_{GAN}	0	0.005

Model settings	Value
Number of RRDB blocks	16
Number of filters	64
Input: Number of channels	8, 6, 4 or 3
Input: Height, Width	None, None
Output: Number of channels	1
Output: Height, Width	None, None

Data settings	Value	
	Training set	**Validation set**
Number of images	22	19
MS patch size	32x32	128x128
PAN patch size	128x128	512x512
Expected patch density	2.0 patches/m^2	
Number of patches	129 221	8 113
Batch size	16	8
Cloud/Sea classifier:		
Probability threshold	0,95	
Keep rate	0,10	

Table 5.4: Configuration and hyperparameter settings of the MS-to-PAN version of ESRGAN used in the experiments

In addition it is worth mentioning that we only employ the shallower 16 Residual-in-Residual Dense Blocks (RRDB) version of ESRGAN. The authors of ESRGAN trained both a 16 and a 23 block version. We made the choice in order to decrease training time and because we observed only negligible differences in performance during initial testing of the model.

5.4.2. Changes to the training configurations

Unsurprisingly, the training configurations presented in the ESRGAN paper (X. Wang, Yu, Wu, et al., 2018) did not work out-of-the-box on our MS-to-PAN task. After all, the authors trained ESRGAN on images of people, buildings, bananas and dogs; content very different from multispectral satellite images. Pre-training worked, although not optimally, but during GAN training the loss frequently diverged. The hyperparameters of the Adam optimizer as well as the loss function (4.5) were hot candidates in the search for what needed to be changed. A set of minor experiments were run on a limited set of data, with TensorBoard used as a tool to investigate performance across metrics. After a decent amount of root-cause-analysis, we ultimately found it necessary to only change the configuration of the Adam optimizer. The following changes were made:

1. We removed the learning rate decay scheduler (halving of the learning rate at fixed intervals). Using learning rate decay would probably work and provide regularization benefits,

but it complicated the search for feasible learning rates. In addition, learning rate decay also adds some complexity to the interpretation of loss curves.

2. Pretraining generator learning rate changed from $2 \cdot 10^{-4}$ to $5 \cdot 10^{-5}$
3. GAN training generator learning rate changed from $1 \cdot 10^{-4}$ to $2 \cdot 10^{-5}$
4. GAN training discriminator learning rate changed from $1 \cdot 10^{-4}$ to $2 \cdot 10^{-5}$

Ideally these changes should be made after a set of rigorous hyperparameter search experiments. This was not done due to time constraints. As a consequence the hyperparameters are probably still sub-optimal.

Chapter 6

Data pipeline

Implementing an efficient data pipeline was essential in order to maximize the utilization of the graphics processing unit (GPU), in our case an NVIDIA GeForce RTX 2080 Ti. Training ESRGAN is computationally expensive, even compared to many computer vision problems. As depicted in Figure 4.5 it first requires forward passes through three different deep neural networks: the generator, the discriminator and the separate VGG19 network used to compute perceptual loss. It then requires backpropagation and optimization of two networks: the generator and the discriminator.

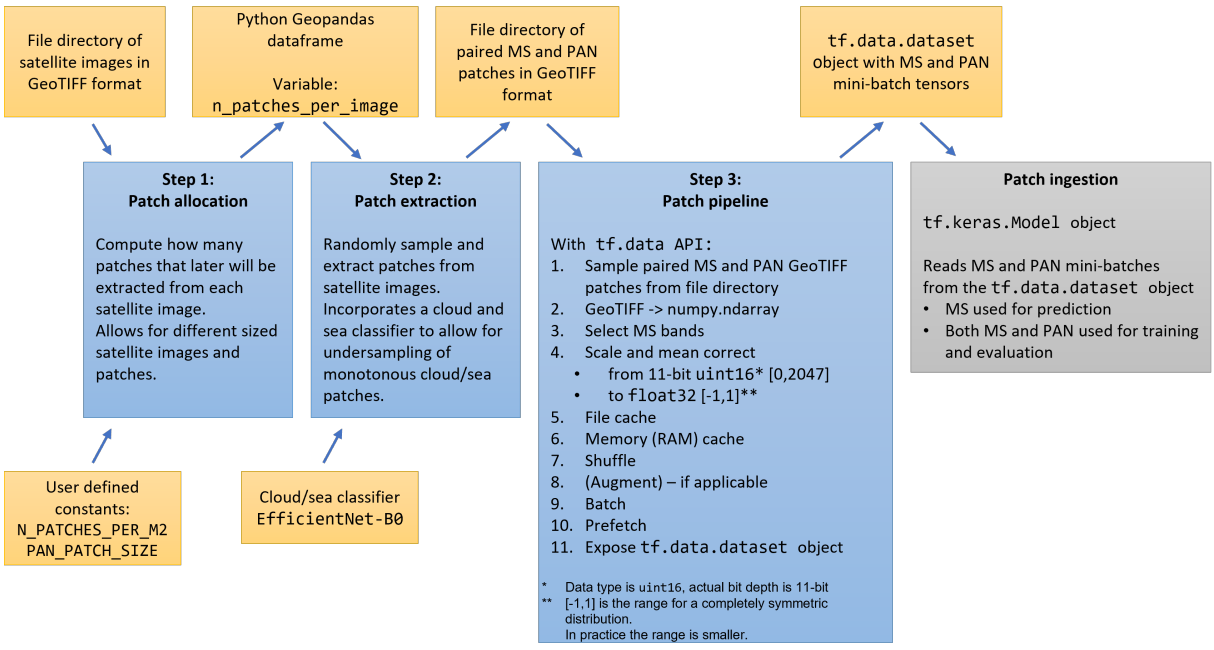


Figure 6.1: Overview of the data pipeline

The data pipeline was developed and improved iteratively over the course of the project. A summary of the final pipeline is seen in Figure 6.1. It describes the steps involved in processing large satellite images, stored as georeferenced TIFF files (GeoTIFFs) on disk, into smaller fixed-size tensors stored in GPU memory, ready to be consumed by the ESRGAN model.

The central component of the pipeline is the tf.data API, TensorFlow's highly optimized module for data pipelines. With tf.data it would be possible to implement an end-to-end pipeline combining steps 1 through 3 in Figure 6.1. However, despite the apparent elegance of such a pipeline, it would require reading large GeoTIFF images from disk during training. We found this to be a bottleneck, and instead chose a step-wise approach where allocation and extraction of patches was separated from the other processes described in step 3.

6.1. Image patches and the fully-convolutional neural network

Despite the apparent complexity of the ESRGAN generator network architecture (see Figure 4.8) it is at its core a series of convolutional operations and is considered a so-called *fully convolutional network* (Long et al., 2015). This means that the generator is input size invariant (height and width), a very attractive property for a SISR algorithm. For instance, the same 4x ESRGAN generator can accept both a $32 \times 32 \times 3$ and a $128 \times 128 \times 3$ input image. Yet, a fully convolutional network is *not* invariant to the number of channels (3 in the preceding example). The number of trainable parameters in the first convolutional layer will change, depending on the number of channels of the input. Consequently, the network architecture is different: the same generator cannot accept a $32 \times 32 \times 3$ and a $32 \times 32 \times 4$ input image.

The discriminator (see Figure 4.7) is *not* fully convolutional. It contains dense layers at the end, and the number of trainable parameters in these layers are dependent on the size of the input.

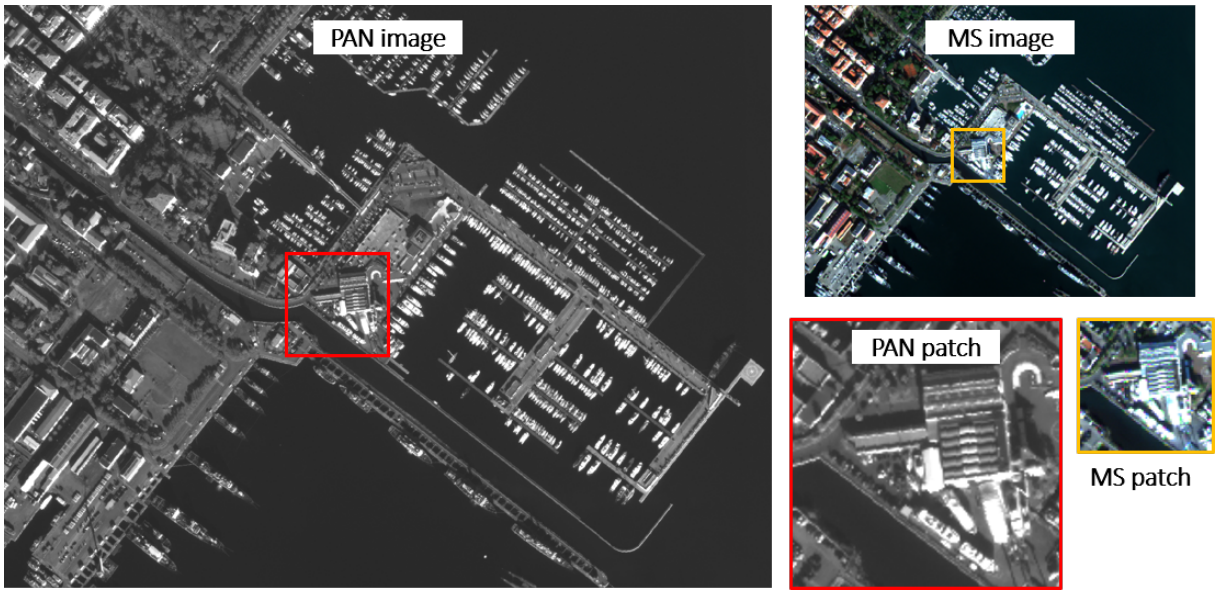


Figure 6.2: Extraction of paired MS and PAN image patches from larger satellite images. *Satellite image © 2021 Maxar Technologies*

The implications of the preceding properties is that fixed and equal input sizes are required during training. However, during inference, the discriminator is not used. Thus, the model may accept any input size (height and width)[1], when performing inference, but the number of channels must be fixed at all times.

In practice, SISR models like ESRGAN are usually trained on smaller fixed-size patches extracted from larger variable-sized images. ESRGAN is trained on 32×32 LR and 128×128 HR patches. As seen in Table 6.1 we use the same sizes in our setup. Where we depart from common configurations is that we have used a four times higher input size, 512×512, during validation and testing. This is a result of our active use of the Ma, NIQE and PI metrics during hyperparameter tuning and model selection. These metrics require larger patch sizes, more content, in order to give meaningful results. A small study was conducted to investigate the impact of patch size on these metrics and select an appropriate patch size for the validation and test sets.[2]

	Multispectral (MS)	Panchromatic (PAN)
Training	$32 \times 32 \times C$	$128 \times 128 \times 1$
Validation	$128 \times 128 \times C$	$512 \times 512 \times 1$
Test	$128 \times 128 \times C$	$512 \times 512 \times 1$

Table 6.1: Different patch sizes for different partitions. C varies across experiments.

6.2. Step 1: Patch allocation

The first step is relatively straight-forward and would not be necessary were it not for the difference in satellite image size. We wanted to randomly sample smaller patches from several satellite images, with an equal sampling density across all images. To allow for both different image and patch sizes we introduce (6.1) to compute the number of patches, n, that should be sampled from an individual satellite image.

$$n = C \cdot \frac{H_{PAN} \cdot W_{PAN}}{h_{PAN} \cdot w_{PAN}} \tag{6.1}$$

where H_{PAN} and W_{PAN} are the pixel dimensions of the satellite image, h_{PAN} and w_{PAN} are the patch pixel dimensions, and C is a hyperparameter that defines the sampling density. We used $C = 2.0$ as a baseline in experiment E1 and E2, and $C = 10.0$ for the models that use

[1] Memory requirements impose certain limitations, which depend on whether a GPU is used for inference and the specifics of that GPU. See Siu et al., 2018 for a review of the topic.

[2] The patch size study is available as a Jupyter Notebook in the project's public GitHub repository, https://github.com/onordberg/multispectral-super-resolution

oversampling as a regularization technique in E2 and E3 (see Figure 5.1). The effect of changing C and the size of the patches can be observed in the Figure 6.4 patch density heat map.

6.3. Step 2: Patch extraction

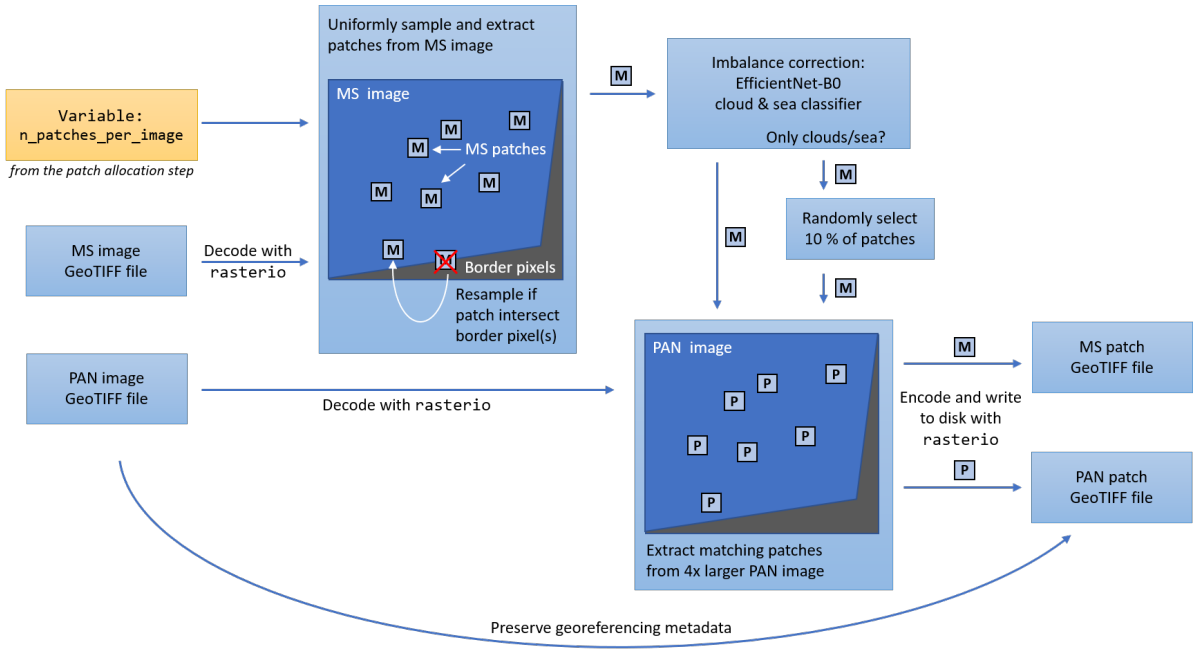


Figure 6.3: Overview of patch extraction process

The second step of the data pipeline centers around the random extraction of paired low-resolution MS and high-resolution PAN patches from the satellite images. The process is summarized in Figure 6.3. Rasterio, a Python library well-suited to read and write georeferenced raster data, is used to read the MS and PAN GeoTIFF files stored on disk. A suite of custom Python functions then uniformly sample matched pairs of MS and PAN patches.

6.3.1. Border pixels

Some GeoTIFF files include NoData pixels encoded as 0. In Figure 6.3 these pixels are referred to as *border pixels* and occur because raster files must be rectangular, no matter what shape the underlying data has organically. Whenever a randomly sampled patch intersects with at least one border pixel, the patch is dropped and replaced. While it would be possible to proceed with patches including border pixels, such an artificial boundary could complicate both training and evaluation. Consequently, since handling border pixels is not central to the topic under research it was decided to avoid them.

6.3.2. Cloud and sea classifier

A major component of step 2 is the use of a cloud and sea classifier to significantly undersample patches that only consist of clouds and/or sea surfaces. We touched upon the rationale behind this decision when we introduced Toulon and La Spezia in Section 5.1.1. To reiterate, the dominance of monotonous surfaces is problematic. Early testing indicated that it could lead to mode collapse during GAN training, a condition where the generator only produces one type of output, in our case sea surfaces. We hypothesize that this was caused by the dominance of sea surface patches in the training data.

One straight-forward and widely used strategy to mitigate the effects of such data imbalance is to undersample the dominant class, or equivalently to oversample the non-dominant class. We decided to undersample cloud and sea patches. However, in order to undersample, we first needed to detect whether a patch was a cloud or a sea patch.

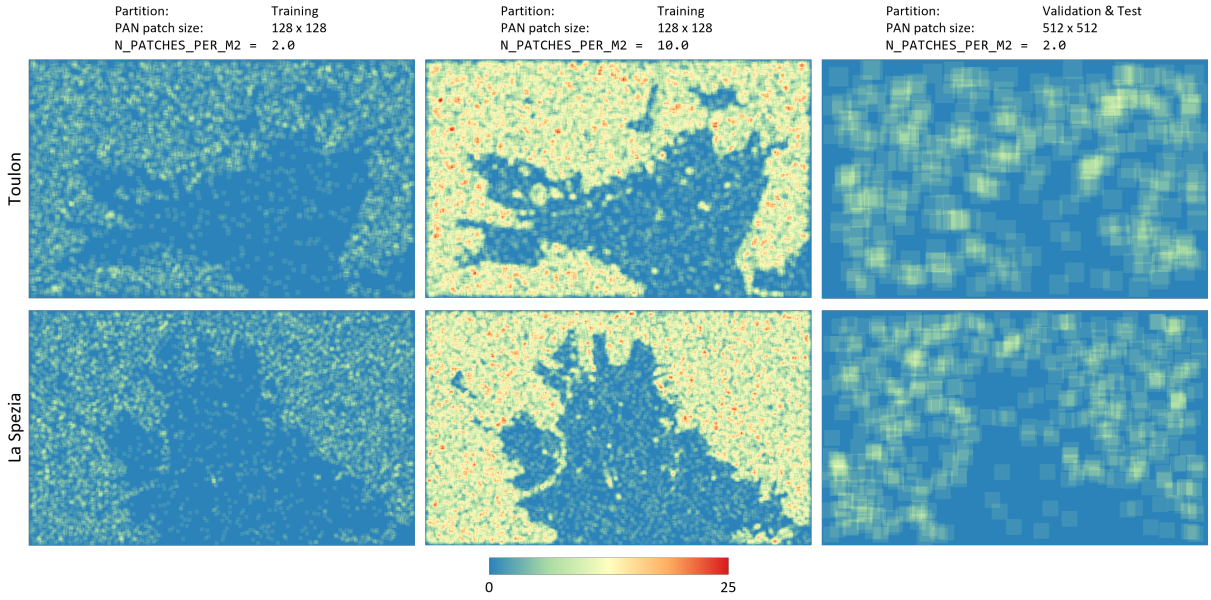


Figure 6.4: Density maps of sampled patches with different sampling densities and different patch sizes. A cloud and sea classifier has been utilized to undersample patches that with a high degree of confidence only contain clouds and/or sea surfaces.

We did this through training our own cloud and sea classifier. 2500 patches of varying sizes was sampled from the complete dataset. Every patch was then manually labelled as either 1: "Completely covered by either sea and/or clouds", or 0: "The inverse". 750 of the patches were held out in a validation set, while the remaining 1750 patches were used to train a convolutional neural network classifier, EfficientNetB0, from scratch. (Tan & Le, 2020). Data augmentation techniques like random rotates, flips, translation and contrast adjustments were implemented to artificially increase the diversity of the training set. The chosen classifier achieved accuracy

scores of above 0.95 on the validation set. Even after accounting for some over-fitting of hyperparameters to the validation set, we consider this well above the requirements for our use case.[3]

Returning to Figure 6.3, every sampled patch is fed through the cloud and sea classifier. Since we do not want to discard valuable non-sea/cloud patches we use a conservative prediction cutoff of 0.95 when deciding whether to classify a patch as cloud and/or sea. 90 % of patches classified as either cloud and/or sea is then discarded without replacement.

At the end of step 2, patches are saved to disk as paired MS and PAN GeoTIFF files. Georeferencing metadata is preserved through the process, enabling the possibility to display patches on top of satellite images and web maps at a later stage.

6.4. Step 3: Patch pipeline

The third and final step of the data pipeline reads paired MS and PAN GeoTIFF patches from disk and exposes mini-batches of these as tensors ready to be consumed by a TensorFlow model. A sequential overview of the operations involved is depicted in Figure 6.1. Some operations are strictly necessary, for instance GeoTIFFs need to be decoded and it is imperative to shuffle data during training and batch several samples together in a mini-batch. Other operations, like caching and prefetching, are included to optimize performance and maximize the utilization of the GPU. This is conceptualized in Figure 6.5. The combined effects of all implemented optimizations are substantial: we observed improvements on the order of 10x compared to a naive, sequential approach. Parallelizing non-TensorFlow functions during GeoTIFF decoding contributed most to these improvements.

Our patch pipeline, `GeotiffDataset`, is implemented as a Python class, and can be considered an end-to-end data pipeline in itself. This allows us to call on a relatively complex pipeline through a few lines of code:

```
ds_train = GeotiffDataset(tiles_path=TILES_PATH, batch_size=16,
                          ms_tile_shape=(32, 32, 4), pan_tile_shape=(128, 128, 1),
                          sensor='WV02', band_selection=(1, 2, 4, 6),
                          mean_correction=MEAN, cache_memory=True,
                          cache_file=CACHE_PATH, repeat=True,
                          shuffle=True, shuffle_buffer_size=SHUFFLE_BUFFER_SIZE).get_dataset()
```

The above `ds_train` object is now a `tf.data.Dataset` instance that can be directly consumed

[3] The cloud and sea classifier is documented in a Jupyter Notebook in the project's public GitHub repository, https://github.com/onordberg/multispectral-super-resolution

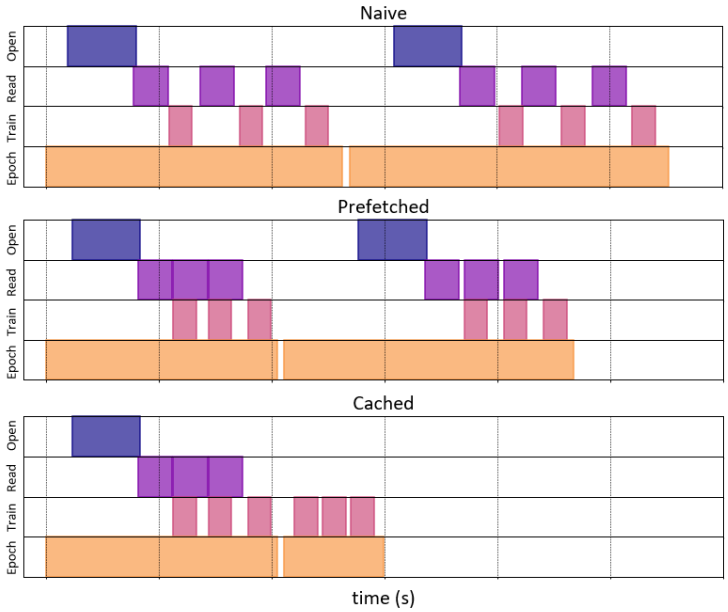


Figure 6.5: Effect of different pipeline optimization techniques (TensorFlow, 2021). Modified figure based on work created and shared by Google and used according to terms described in the Creative Commons 4.0 Attribution License.

by a TensorFlow model.

6.4.1. Normalization

The dynamic range of both WorldView-2 and GeoEye-1 images is 11 bits per pixel , meaning that the range of every pixel in the image is $[0, 2047]$ (Maxar, 2019a, 2019c). 11-bit unsigned integers is a non-standard data type so the pixels in the actual GeoTIFF images are encoded as 16-bit unsigned integers. We normalized the image patches with (6.2), a variant of min-max scaling with zero-centering, normalizing values to within a hard $[-1, 1]$ range and subtracting the mean.[4]

$$\mathbf{Z} = \frac{\mathbf{X} - \mu}{\max\left(|0 - \mu|, |2047 - \mu|\right)} \tag{6.2}$$

where $\mathbf{X}$ is a single MS or PAN image patch and μ is the empirical mean of all image patches in the training set. The empirical mean of our training set varied a bit depending on the experiment, but generally hovered around $\mu = 340$, producing a normalized range $\approx [-0.199, 1.0]$.

[4]The output of the ESRGAN generator model (Figure 4.8) is actually unbounded: there is no final activation layer. In retrospect a min-max approach to scaling is therefore not necessary. $\mathbf{Z} = \frac{\mathbf{X} - \mu}{\sigma}$ would suffice. We did at one point consider modifying the generator by adding a tanh activation layer at the end. This would require pixels in the PAN image to be bounded within a $[-1, 1]$ range, hence the choice of normalizing with (6.2).

Chapter 7

Results

In this chapter we will present and discuss the results from the baseline experiment (E1), the regularization experiment (E2) and from the final run of the best model on the test set (E3).

7.1. E1. The baseline experiment

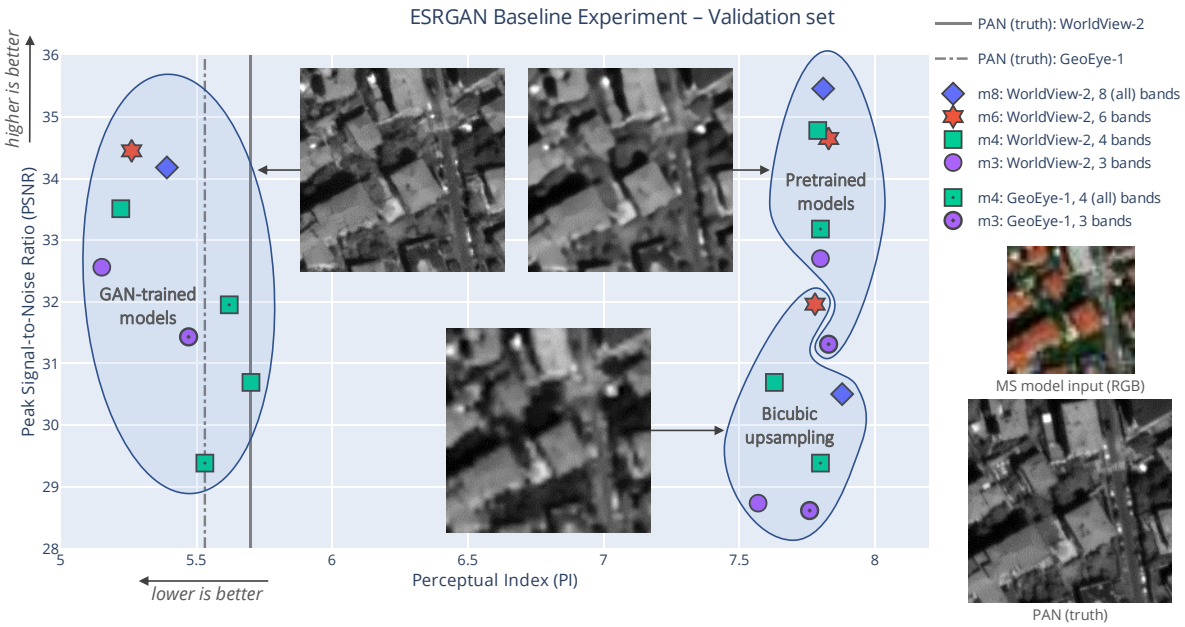


Figure 7.1: The baseline experiment results summarized on the perception-distortion plane. The different models (m3, m4, m6 and m8) have been evaluated on the validation set. The points on the plane represent the mean PSNR (higher is better) and Perceptual Index (lower is better) scores. *Satellite image © 2021 Maxar Technologies*

The baseline experiment gave somewhat predictable, yet interesting results. Firstly, we observe in Figure 7.1 how 400k iterations of GAN-training have increased the perceptual quality substantially. There is a notable jump in Perceptual Index from the pretrained to the GAN-trained versions of the models. Secondly, we observe that the increase in perceptual quality has come at the cost of more distortion, i.e, a lower PSNR score. The perception-distortion trade-off

is in line with previous research.

The third observation is more specific to our MS-to-PAN satellite imagery task. We notice a relationship between the number of MS bands and the level of distortion. More MS bands generally result in less distortion (higher PSNR scores). A possible interpretation is that the model is able to utilize and exploit information in the additional bands. In Figure 2.7 we noticed how the spectral range of the PAN band approximately overlapped with six of the WorldView-2 MS bands (all eight except the low wavelength Coastal band and the high wavelength NIR2 band). Model m6 uses the same six bands as its input, and in Figure 7.1 we notice that m6 perform well compared to the eight band m8. Evidently, there is no indication that using MS bands outside the spectral range of the PAN band increases the performance of the ESRGAN model. This fact may also be interpreted as a validation of the model; it adheres to the laws of physics.

Finally, a fourth observation relates to the performance of models m3 and m4 on the GeoEye-1 validation set. We notice that the performance lags behind the that of their WorldView-2 counter-parts. This was expected. The models are trained on WorldView-2 images, and GeoEye-1 validation images are not drawn from the same distribution. A decrease in performance is therefore to be expected and the remaining question is whether the performance is *good enough*.

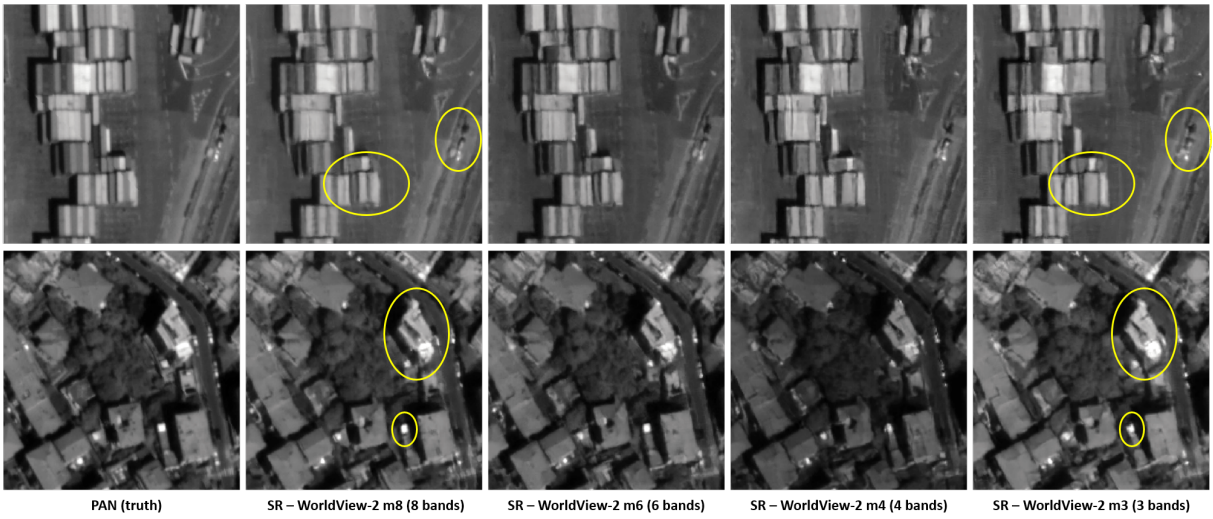


Figure 7.2: Comparison of baseline models with different number of MS bands. The image patches are from the WorldView-2 validation set. Yellow circles highlight some areas with a noticeable difference in performance. *Satellite image © 2021 Maxar Technologies*

Are the differences between the different models noticeable by the human eye? The difference between a pretrained and a GAN-trained model is apparent (see image patches in Figure 7.1 and randomly drawn image patches in Appendix A and Appendix B). Pretrained models produce blurry, smooth and somewhat conservative estimates, while GAN-trained models produce images

with sharper edges and more details, some correct and some erroneous. We call the latter SISR *artefacts*.

The difference between the different GAN-trained models are not that obvious. In Figure 7.2 we compare two patches across the m8, m6, m4 and m3 models. In the topmost comparison, the one containing shipping containers, we notice for instance that some shipping containers are combined into one in the m3 model. The m8 and m6 models, on the other hand, are able to separate the containers, and when comparing with the PAN ground truth image, we see that this is indeed closer to the truth. A similar effect is noticeable with the cars. The m3 model produces a noisy blob where the m8 model outputs objects that more closely resemble cars. Meanwhile, the comparison at the bottom of Figure 7.2 displays similar differences. The houses lack and/or misrepresent details in m3, compared to m8.

7.1.1. GeoEye-1 performance

In Figure 7.1, the model performances are represented only by their mean PSNR and PI, computed from a sample of approximately 4000 image patches per sensor in the validation set. When investigating performance on images from the GeoEye-1 sensor it is of interest to peek behind these statistics and plot individual image patches on the perception-distortion plane.

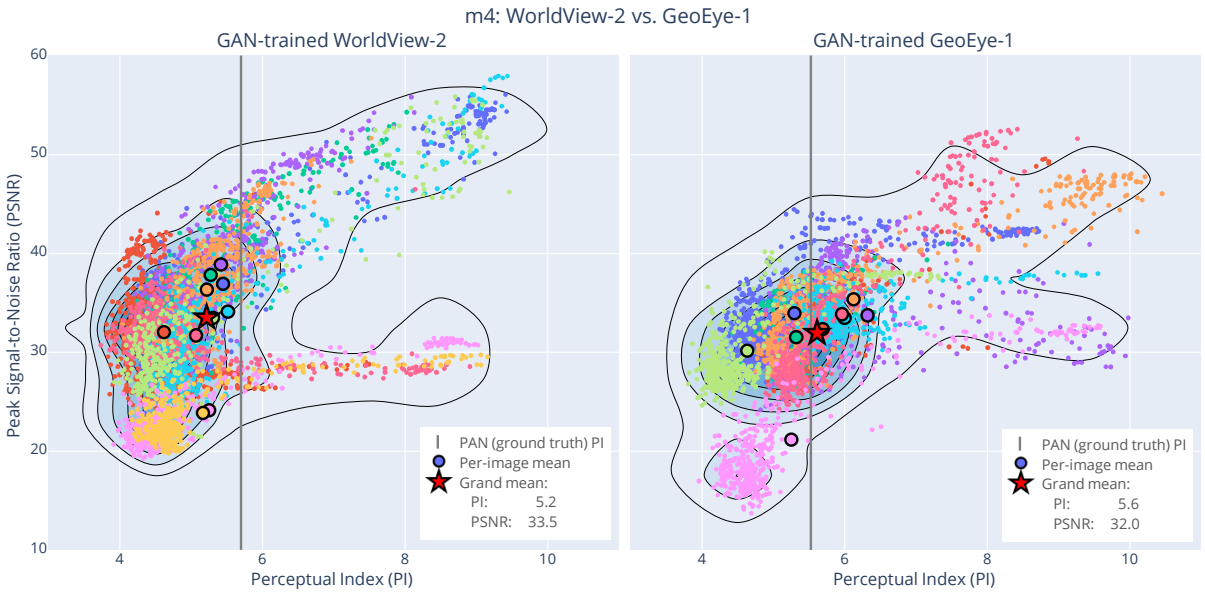


Figure 7.3: Scatter plot of individual image patches from the m4 model, comparing WorldView-2 (left) with GeoEye-1 image patches (right). Each point represents a 512x512 patch that is a part of a larger satellite image. The points are colored by satellite image and the colors are not comparable across satellites, e.g., the pink WorldView-2 is not the same as the pink GeoEye-1 image.

Not surprisingly, Figure 7.3 reveals a distribution with some complexities. Comparing the

left plot (WorldView-2) with the right plot (GeoEye-1), we notice that the distributions are not entirely different. This is promising for our attempt to generalize the models to work well on GeoEye-1 images (research question R2). What we would *not* want to see in this plot is entirely different distributions, for instance a GeoEye-1 distribution where points are more clustered together with other points from the same satellite image. This seems to be the case for only one of the satellite images in the GeoEye-1 plot, the image represented with pink-colored points.[1]

Remember that we want our images to be located in the upper left corner of the perception-distortion plane. In both plots we notice that as PSNR increases there is a tendency of PI to also increase, establishing an upper left boundary on the plane that no outliers seem to cross. In general, we may say that the the scatter plot reveal non-linear relationships within and between satellite images. We have not investigated this in detail, but can note that based on manual inspection the upper right region/tail is dominated by sea and cloud patches. It is easy for the models to achieve high PSNR scores on very monotonous patches, but these almost all-black images are not very natural-looking and thus receives low PI scores.

7.1.2. Are we overfitting?

When dealing with an unregularized neural network like ESRGAN it is natural to be concerned about overfitting of the model to the training data. Indeed, we observe a gap between training loss and validation loss. This is to be expected from an unregularized setup. Still, validation PSNR curves in the upper left corner of Figure 7.4 generally appear to stabilize and converge. This behaviour contradicts classical bias-variance trade-off theory, but is more in line with recent research into how high-capacity models, like deep neural networks, behave (Belkin et al., 2019). Still, in Section 7.2, we will see whether regularization helps to further stabilize and increase performance.

Expanding the focus beyond PSNR curves from pretraining, in Figure 7.4 we see PSNR and NIQE curves from both pretraining and GAN-training. In this instance, NIQE represents the perception axis of the perception-distortion plane.[2] From these curves we make a few observations. First, we note the immediate drop of all PSNR and NIQE curves from the last iteration of pretraining to the first completed iteration of GAN-training.

The second observation is the rising NIQE curves during GAN-training. A lower NIQE score is supposedly better, and a rising curve may be a sign of overfitting. Still, the concern is somewhat

[1] Image ID: `GE01_Toulon 2019_10_07_011651194010_0`. See Appendix C for more details.

[2] The reason for not reporting Perceptual Index (PI) as in previous plots is because PI is a function of the Ma et al. metric, and the computation of this metric at several iteration steps is very time consuming (several weeks on our hardware setup).

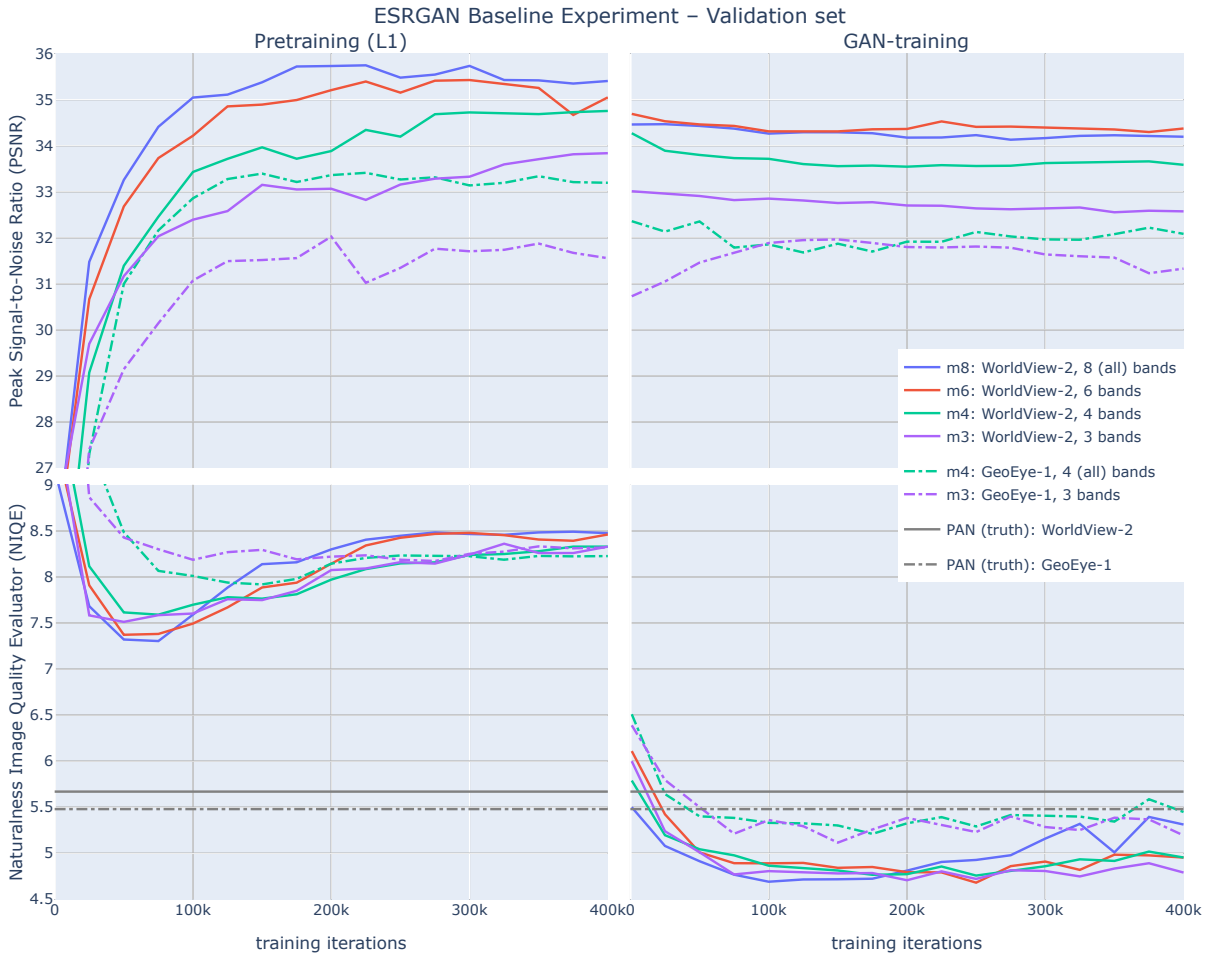


Figure 7.4: Learning curves: PSNR and NIQE plotted against training iterations in E1. Evaluated against WorldView-2 and GeoEye-1 validation sets. All image patches in the validation set have been evaluated every 25k iterations.

mitigated by the NIQE scores of the actual, true PAN images (gray, horizontal lines). These are generally higher than their SISR counterparts (especially true for WorldView-2). In theory, on the assumption that NIQE is a good measurement of the perceptual quality of satellite images, NIQE should be lower on ground truth images than on SISR images. NIQE's usefulness as a metric for satellite image SISR is therefore weakened by these findings. A reasonable explanation is that NIQE is trained on a set of natural, pristine images – not on satellite images (see Section 4.2.2). The learning curves indicate that NIQE is a reasonable choice for a satellite image perceptual metric, but only up to a certain point. The metric captures the transition from pretraining on L1 loss to GAN training very well, but when the network starts fine-tuning features in the image that is specific to satellite images, NIQE scores increase. This belief has been validated by manual visual comparisons of image patches from training iterations 150k and 400k. Image patches from the 400k-th iteration have sharper edges and seem to capture more details than those from the 150k-th. Still, we suggest further validation of perceptual performance in the

later stages of GAN training.

The third observation is something that should be noted, but is not explored further in this thesis. The NIQE curve of the m8 model seems to behave erratically towards the end of GAN-training. Combined with the approximately equal PSNR performance of m6 and m8, and the fact that m6 is a simpler model, this suggests that m6 is preferred over m8 for WorldView-2 models, i.e., models trained and tested on images from the same sensor (R1).

7.2. E2. The regularization experiment

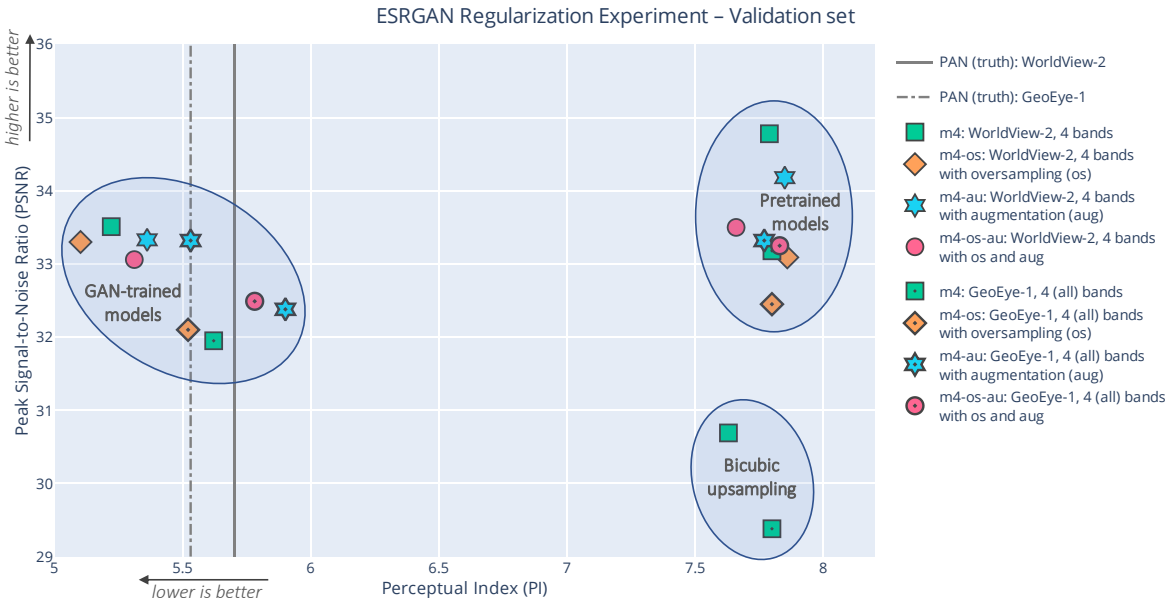


Figure 7.5: The regularization experiment results summarized on the perception-distortion plane. The different models, the baseline m4 from E1 and its regularized versions, have been evaluated on the WorldView-2 and GeoEye-1 validation sets. The points on the plane represent the mean PSNR (higher is better) and Perceptual Index (lower is better) scores.

Results from the regularization experiment is somewhat mixed, and point in a couple of different directions. We notice from Figure 7.5 and Figure 7.4 that overall performance on the WorldView-2 validation set is actually negatively affected by regularization. By contrast, performance on the GeoEye-1 validation set is generally improved. What could explain this difference?

One hypothesis is that the m4 model memorizes specific features in the WorldView-2 training set, for instance a special looking building or ship, in a way that transfers to the WorldView-2 validation set. This, despite the variation in lighting conditions, angles and other ground conditions from image to image (see Section 5.1.3). In certain cases memorization may be a useful feature, but it could also lead to a model that is hesitant to suggest new features, opting

for what it has seen before instead. It could for instance be hesitant to suggest a new roof on a building despite evidence in the MS image that the roof has been replaced. Our regularization techniques, especially the flips and rotations, are probably limiting the network's ability to memorize specific features. Memorization of such specific features is thus likely less rewarded, replaced with an incentive to learn more generalizable features, for instance the features of *typical* roofs.

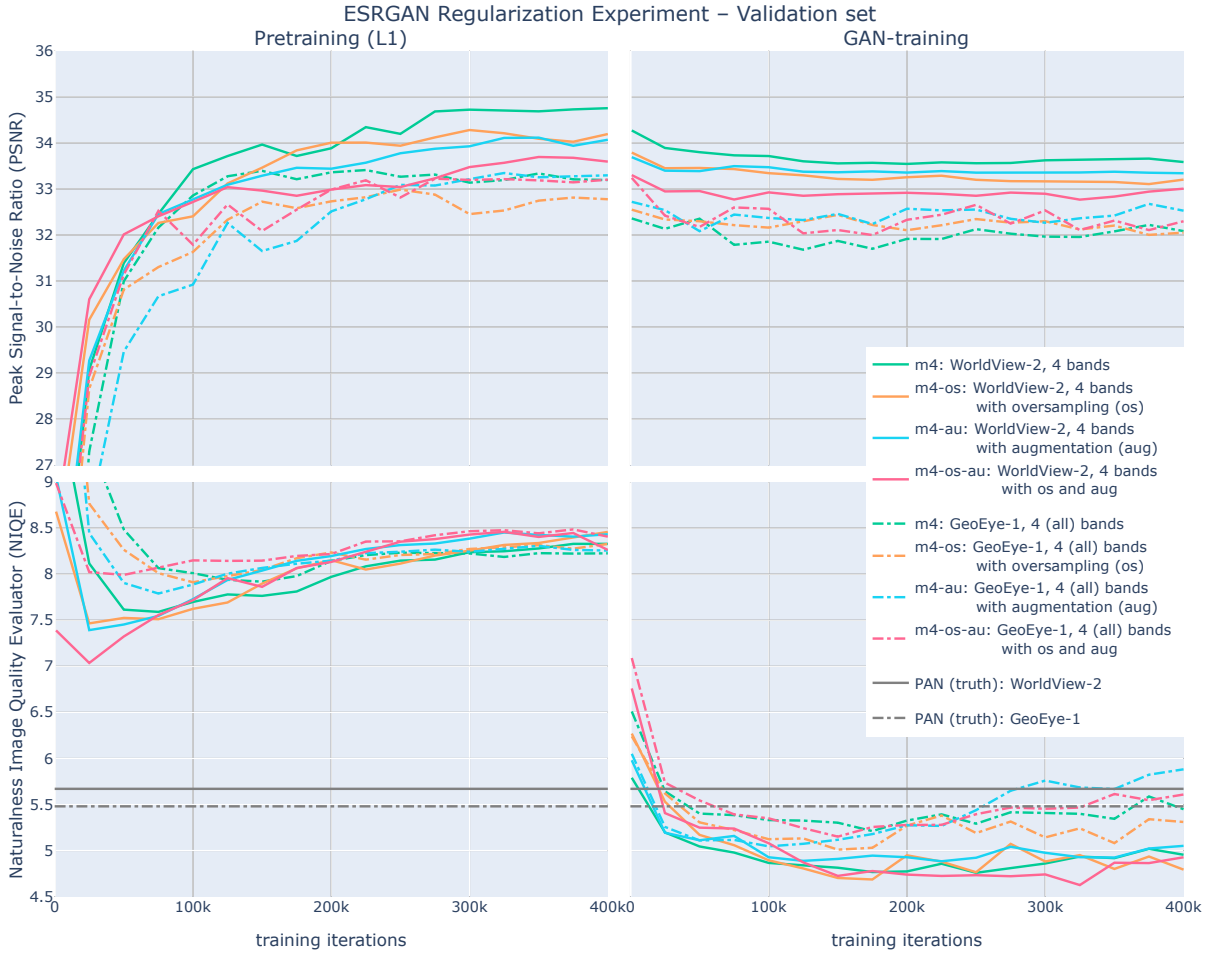


Figure 7.6: Learning curves: PSNR and NIQE plotted against training iterations in E2. Evaluated against WorldView-2 and GeoEye-1 validation sets. All image patches in the validation set have been evaluated every 25k iterations.

Under the same hypothesis, m4 is not able to utilize the memorization as well on images taken by GeoEye-1. It could be that the network tries to use memorized features, is somewhat unsuccessful and consequently receives a lower PSNR score. Regularization helps, simply because the model is relying less on memorization, and more on generalized representations.

7.2.1. The best models

It is difficult to select a single best-performing model based on results from the regularization experiment (E2). As mentioned, there are indications that for WorldView-2 images, an unregularized model is best, and for GeoEye-1 images some form of regularization helps. Still, the margins are small, learning curves fluctuate (see Figure 7.6) and the ranking of models at 400k iterations of GAN-training may be a result of random chance. If we for a moment assume that all three regularization models (m4-os, m4-aug, m4-os-aug) perform equally well on the GeoEye-1 validation set, then established theory suggests that the most heavily regularized model should be selected. With the types of regularization we have introduced, it is hard to imagine that they will make the model less generalizable. By this argument, the model with both oversampling and data augmentation, m4-os-aug, is selected as "the best" model for GeoEye-1 images.

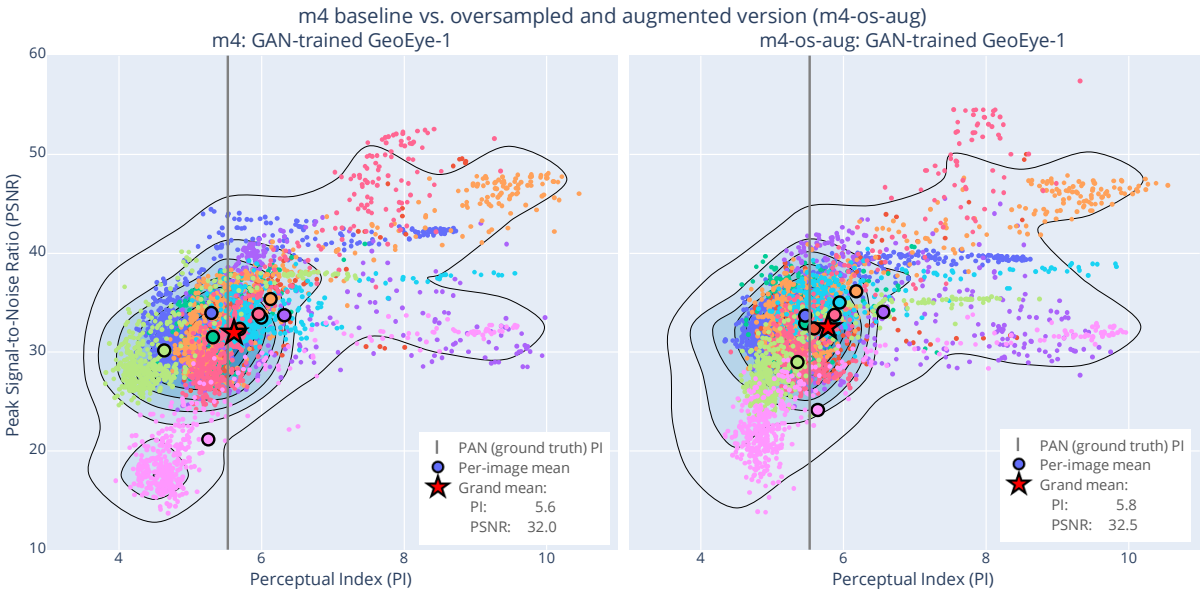


Figure 7.7: Scatter plot of individual image patches, comparing the baseline m4 model from E1 with the oversampled and augmented version (m4-os-aug) from E2. Each point represents a 512x512 patch that is a part of a larger satellite image. The points are colored by satellite image.

Comparing the baseline m4 model with the regularized m4-os-aug model in Figure 7.7, we observe a few indications of a better performing model to the right. In addition to the already mentioned increase in PSNR, we note less clustering between images, i.e. the colors are less separated, indicating a more generalizable model. Consequently, the performance is less dependent on which image the patch is extracted from.

7.3. E3. The final evaluation – Test set

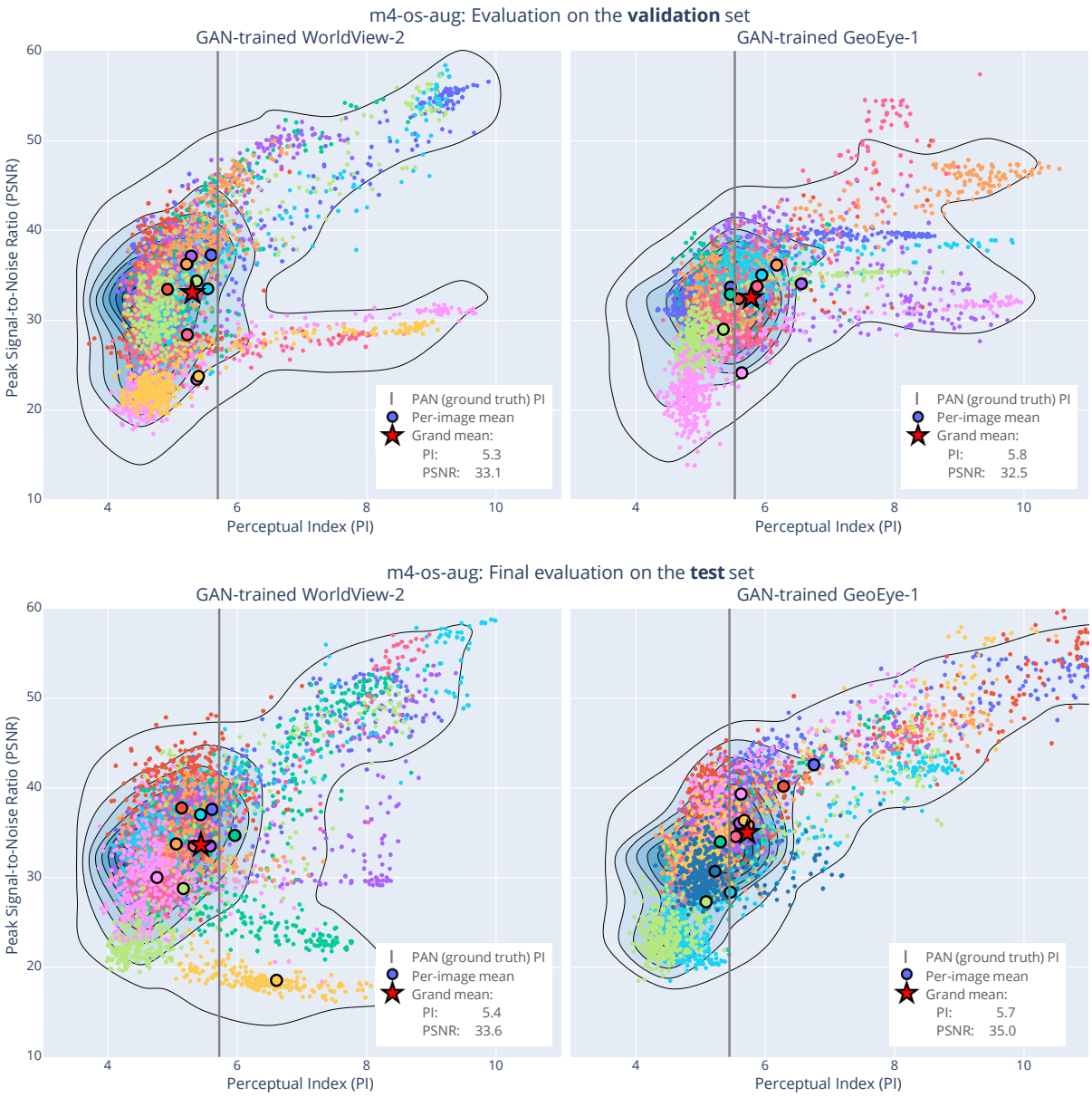


Figure 7.8: Scatter plot of individual image patches, comparing the oversampled and augmented version of m4 (m4-os-aug) on both validation and test set. Each point represents a 512x512 patch that is a part of a larger satellite image. The points are colored by satellite image.

Overall, results from the final evaluation on the test set indicates that our selected *best* model from E2, m4-os-aug, is *not* overfitting on the validation set. As a matter of fact, performance is generally better on the test set than on our validation set. This is particularly true in the case of GeoEye-1 where we in Figure 7.8 notice better PSNR scores, similar PI scores and what looks to be less clustering by individual satellite image (colors are less clustered). In addition we see a more well defined perception-distortion boundary (see Figure 1.2) along the upper left portion of

the plane. This indicates that image patches have been pulled more towards an optimal trade-off between perceptual quality and distortion than in the other plots.

A better performance on the test set than the validation set is somewhat uncommon. One generally assumes that the model selected in the model selection phase, to a certain degree overfits on the validation set. In our case, overfitting might have been mitigated by us selecting the most regularized version of the model.

Additionally, we increase generalizability during final training because we also train on the validation set (see Figure 3.1). In our case we increase the total number of training satellite images from 22 to 32 (see Table 5.2), corresponding to an increase in the total number of extracted image patches from ca. 645k to ca. 935k (31% increase). At last, we should not underestimate random effects. We evaluate on a test set of 9315 image patches, but they are extracted from a set of only 11 GeoEye-1 and 10 WorldView-2 satellite images. These 21 images could, by chance, fit our model well and be well-suited for the MS-to-PAN SISR task. In other words, the *increase* in performance when evaluated on the test set could also *partly* be explained by luck.

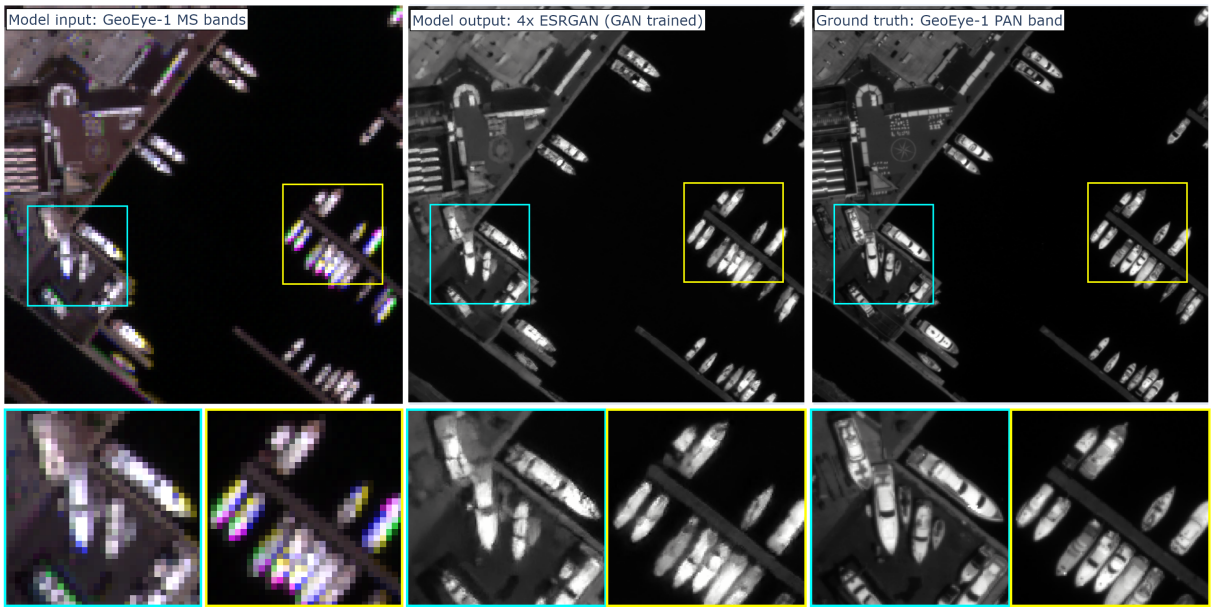


Figure 7.9: GeoEye-1 test set: Comparison between an MS image patch, an ESRGAN estimated PAN patch and an PAN ground truth patch. La Spezia 2013-07-15, *Satellite image © 2021 Maxar Technologies*

Finally, in Figure 7.9 we see a phenomenon indicative of a general pattern in our results. Here, two interesting areas are highlighted. In the yellow rectangle we notice that boat structures and textures are well estimated. What is lacking are details that in many ways are impossible for the model to estimate. Meanwhile, in the blue rectangle we notice severe artefacts in at least one of the boats located on land. A boat on land is a rare event in the training set compared to a boat on the water, and we generally notice that ESRGAN captures the structure of frequent textures

very well, but fails to reproduce these types of rarer textures. SISR estimates of forested areas are for instance almost indistinguishable from its ground truth counterpart.

Chapter 8

Conclusion

A deep learning-based SISR model like ESRGAN can be successfully applied to the 4x SISR MS-to-PAN task, without much modification to either the network architecture nor the training configuration. On images from the WorldView-2 satellite we have established that it is possible to reconstruct the PAN band to a degree of similarity and image quality that far outperforms the most commonly used deterministic upsampling method: bicubic upsampling.[1] Even more interestingly, we have shown that it is possible to train the model on images from one satellite (WorldView-2), and directly apply the model on images from a different satellite (GeoEye-1). In both cases the super-resoluted PAN images were evaluated on unseen test data and performance was measured with both a distortion-type full-reference metric, PSNR, as well as a perception-type no-reference metric, Perceptual Index. When using the same number of MS bands as input (RGB+NIR), performance on the unseen GeoEye-1 test imagery were similar to the performance on the unseen WorldView-2 test imagery. These results were achieved with a version of ESRGAN implementing data augmentation (flips and 90 degree rotations) on image patches randomly sampled with a high degree of overlap from the underlying satellite images.

The ability of applying a SISR model trained on images taken by one satellite, to images taken by another satellite is significant. It opens the possibility of enhancing images taken by smaller satellites not capable of capturing a high-resolution PAN band. These cheaper nanosatellites are plentiful and their significance within the domain of satellite imagery is only increasing.

8.1. Ideas for future research

To our knowledge there is no publicly available research directly related to the cross-sensor MS-to-PAN SISR topic. We are hopeful that this will change and would therefore like to conclude with pointing to some interesting directions for future research.

[1] See Appendix A and B for examples of bicubic upsampling.

8.1.1. Apply SISR to satellite images without a PAN band

A natural extension of the research done in this thesis would be to apply the same, or a similar SISR model to satellite imagery with no PAN band. Imagery from Planet's PlanetScope nanosatellite constellation is a possible candidate. This imagery has four multispectral band (RGB+NIR) with a spatial resolution (GSD) of around 4 meters, compared to the approximately 2 meter resolution of WorldView-2 and GeoEye-1 (Planet, 2021). If WorldView-2, or imagery from similar satellites are to be used for training, one would probably need to downsample the imagery to a resolution more similar to PlanetScope's resolution. Another approach could be to use training imagery from a satellite with an MS resolution closer to PlanetScope's 4 meter resolution. Satellites with this specifications are few and far between. However, historical imagery from the decommissioned IKONOS satellite could fit the bill.(DigitalGlobe, 2013)

A major challenge when evaluating performance on imagery without a PAN band is that full-reference image quality metrics cannot be used. There is no ground truth, no $\mathbf{X}_{HR}$, available to compare the super-resoluted PAN image with. We have seen that relying solely on no-reference algorithmic image quality metrics developed for non-satellite imagery is difficult. One would therefore need to explore and/or develop alternative performance metrics, alternatively rely solely on human evaluation.

8.1.2. Develop alternative performance metrics

There is a need for image quality metrics that are more tailored towards measuring satellite imagery quality in general, and the super-resolution of satellite imagery specifically. Perceptual quality metrics like NIQE, Ma et al. and Perceptual Index have been developed with non-satellite imagery in mind. NIQE, for instance, has been trained on a set of pristine, natural images. One could retrain NIQE on pristine satellite images and presumably get a no-reference metric more suited to the satellite imagery SISR task.

A quite different approach would be to research and develop what we in Figure 4.1 dub *Computer Vision IQA* metrics. These differ from perceptual metrics in that they would measure the impact the quality of a satellite imagery has on computer vision tasks, like object detection and image segmentation. One could for instance establish a labelled benchmark dataset, possibly building off one of the labelled datasets from the SpaceNet challenges. This labelled dataset could then be used to evaluate whether for instance the performance of an object detection model is improved by super-resolution. Rabbi et al., 2020 used this approach to evaluate their ability to use ESRGAN to enhance the ability to detect small objects, but their labelled datasets were not appropriate to use for the MS-to-PAN task.

8.1.3. Generalize beyond two towns and a temporal dataset

Our research has been focused on a dataset with multiple overlapping images of two small areas of the Earth's surface. In Section 5.1.3 we discussed how we likely have temporal correlations within our dataset and how this could motivate the model into memorizing specific areas of the dataset, instead of learning generalizable features. While this may be viewed as a feature rather than a bug, memorization's effects on performance should be further researched. One possible way to study this is to spatially divide up the dataset so that a temporally and a non-temporally trained model can be compared on as equal terms as possible. For instance, one model could be allowed to train on areas (but not images!) included in the test set, and the other not.

8.1.4. Train on less processed images

We have trained on images that have been through several post-processing steps after their capture by the arrays of CCD sensors onboard the satellite. Specifically, our dataset only contains Level 2A imagery. In Section 2.3 we introduced the NASA processing levels, and from these we read that Level 2A images have been geometrically corrected. Geometric corrections will cause image resampling: original pixels are mapped to a new grid of pixels. No matter which resampling method used, this will lead to some level of degradation and could cause artefacts, for instance jaggedness. Consequently, introducing SISR at an earlier stage in the post-processing pipeline should be further researched.

Bibliography

Agustsson, E., & Timofte, R. (2017). NTIRE 2017 challenge on single image super-resolution: Dataset and study. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. Retrieved August 28, 2021, from https://openaccess.thecvf.com/content_cvpr_2017_workshops/w12/html/Agustsson_NTIRE_2017_Challenge_CVPR_2017_paper.html

Athar, S., & Wang, Z. (2019). A comprehensive performance evaluation of image quality assessment algorithms [Conference Name: IEEE Access]. *IEEE Access*, *7*, 140030–140070. https://doi.org/10.1109/ACCESS.2019.2943319

Baghdadi, N., & Zribi, M. (2016). *Optical remote sensing of land surface: Techniques and methods*. Elsevier.

Beck, A. (2016, December 27). *Sunsynchronous orbit*. Retrieved October 30, 2021, from https://commons.wikimedia.org/wiki/File:Sunsynchronous_orbit_en.jpg

Belkin, M., Hsu, D., Ma, S., & Mandal, S. (2019). Reconciling modern machine learning practice and the bias-variance trade-off. *arXiv:1812.11118 [cs, stat]*, arxiv 1812.11118. Retrieved July 11, 2021, from http://arxiv.org/abs/1812.11118

Bi, X., Tang, X., Yuan, Y., Zhang, Y., & Qu, A. (2021). Tensors in statistics [_eprint: https://doi.org/10.1146/an statistics-042720-020816]. *Annual Review of Statistics and Its Application*, *8*(1), 345–368. https://doi.org/10.1146/annurev-statistics-042720-020816

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Blau, Y., Mechrez, R., Timofte, R., Michaeli, T., & Zelnik-Manor, L. (2019). The 2018 PIRM challenge on perceptual image super-resolution. *arXiv:1809.07517 [cs]*, arxiv 1809.07517. Retrieved October 12, 2020, from http://arxiv.org/abs/1809.07517

Blau, Y., & Michaeli, T. (2018). The perception-distortion tradeoff. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Retrieved July 13, 2021, from https://openaccess.thecvf.com/content_cvpr_2018/html/Blau_The_Perception-Distortion_Tradeoff_CVPR_2018_paper.html

Brandir, & XZise. (2018, February 1). *Heliosynchronous orbit*. Retrieved October 30, 2021, from https://commons.wikimedia.org/wiki/File:Heliosynchronous_orbit.svg

Buda, M., Maki, A., & Mazurowski, M. A. (2018). A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, *106*, 249–259. https://doi.org/10.1016/j.neunet.2018.07.011

Courtrai, L., Pham, M.-T., & Lefèvre, S. (2020). Small object detection in remote sensing images based on super-resolution with auxiliary generative adversarial networks [Number: 19 Publisher: Multidisciplinary Digital Publishing Institute]. *Remote Sensing*, *12*(19), 3152. https://doi.org/10.3390/rs12193152

Deng, J., Dong, W., Socher, R., Li, L., Kai Li, & Li Fei-Fei. (2009, June). ImageNet: A large-scale hierarchical image database [ISSN: 1063-6919], In *2009 IEEE conference on computer vision and pattern recognition*. 2009 IEEE Conference on Computer Vision and Pattern Recognition. ISSN: 1063-6919. https://doi.org/10.1109/CVPR.2009.5206848

Deng, L. (2014). Deep learning: Methods and applications. *Foundations and Trends® in Signal Processing*, *7*(3), 197–387. https://doi.org/10.1561/2000000039

DigitalGlobe. (2009, June 4). WorldView-2 overview. Retrieved October 3, 2021, from https://content.satimagingcorp.com.s3.amazonaws.com/static/satellite-sensor-specification/WorldView-2-PDF-Download.pdf

DigitalGlobe. (2013, June). IKONOS data sheet. https://dg-cms-uploads-production.s3.amazonaws.com/uploads/document/file/96/DG_IKONOS_DS.pdf

DigitalGlobe. (2014, October 22). Spectral response for DigitalGlobe earth imaging instruments. Retrieved January 18, 2021, from https://dg-cms-uploads-production.s3.amazonaws.com/uploads/document/file/105/DigitalGlobe_Spectral_Response_1.pdf

Dong, C., Loy, C. C., He, K., & Tang, X. (2016). Image super-resolution using deep convolutional networks [Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence]. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *38*(2), 295–307. https://doi.org/10.1109/TPAMI.2015.2439281

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, *12*(7).

ESA. (2020, March 30). *Types of orbits*. Retrieved October 3, 2021, from https://www.esa.int/Enabling_Support/Space_Transportation/Types_of_orbits

ESA, E. O. P. (2021a). *GeoEye-1 - eoPortal directory - satellite missions*. Retrieved October 11, 2021, from https://directory.eoportal.org/web/eoportal/satellite-missions/g/geoeye-1

ESA, E. O. P. (2021b). *WorldView-2 - eoPortal directory - satellite missions*. Retrieved October 11, 2021, from https://earth.esa.int/web/eoportal/satellite-missions/v-w-x-y-z/worldview-2

Foster, D. (2019, June 28). *Generative deep learning: Teaching machines to paint, write, compose, and play* [Google-Books-ID: RKegDwAAQBAJ]. "O'Reilly Media, Inc.".

Gleason, M. (2020, October 21). *HD satellite imagery and machine learning: More accurately detect and...* [Maxar blog]. Retrieved July 20, 2021, from https://blog.maxar.com/earth-intelligence/2020/hd-satellite-imagery-and-machine-learning-more-accurately-detect-and-locate-features-of-interest-with-greater-consistency

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks, In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, JMLR Workshop; Conference Proceedings.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial networks. *arXiv:1406.2661 [cs, stat]*, arxiv 1406.2661. Retrieved February 18, 2021, from http://arxiv.org/abs/1406.2661

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. http://www.deeplearningbook.org

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. Proceedings of the IEEE International Conference on Computer Vision. Retrieved August 28, 2021, from https://openaccess.thecvf.com/content_iccv_2015/html/He_Delving_Deep_into_ICCV_2015_paper.html

He, K., Zhang, X., Ren, S., & Sun, J. (2016a). Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Retrieved April 12, 2021, from https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html

He, K., Zhang, X., Ren, S., & Sun, J. (2016b). Identity mappings in deep residual networks (B. Leibe, J. Matas, N. Sebe, & M. Welling, Eds.). In B. Leibe, J. Matas, N. Sebe, & M. Welling (Eds.), *Computer vision – ECCV 2016*, Cham, Springer International Publishing. https://doi.org/10.1007/978-3-319-46493-0_38

Hinton, G., Srivastava, N., & Swersky, K. (2012). *Neural networks for machine learning: Lecture 6e rmsprop: Divide the gradient by a running average of its recent magnitude.*

Horé, A., & Ziou, D. (2010, August). Image quality metrics: PSNR vs. SSIM [ISSN: 1051-4651], In *2010 20th international conference on pattern recognition*. 2010 20th International Conference on Pattern Recognition. ISSN: 1051-4651. https://doi.org/10.1109/ICPR.2010.579

Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. Proceedings of the IEEE Conference on Computer Vision and Pattern

Recognition. Retrieved March 9, 2021, from https://openaccess.thecvf.com/content_cvpr_2017/html/Huang_Densely_Connected_Convolutional_CVPR_2017_paper.html

Huang, K.-Y. (2020, June 4). *peteryuX/esrgan-tf2* (Version 90e8eeced35a677aa80851f185e1227a71fad1ac) [original-date: 2019-12-31T16:49:49Z]. Retrieved March 2, 2021, from https://github.com/peteryuX/esrgan-tf2

Ioffe, S., & Szegedy, C. (2015, June 1). Batch normalization: Accelerating deep network training by reducing internal covariate shift [ISSN: 1938-7228], In *International conference on machine learning*. International Conference on Machine Learning, PMLR. ISSN: 1938-7228. Retrieved April 12, 2021, from http://proceedings.mlr.press/v37/ioffe15.html

Johnson, J., Alahi, A., & Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution (B. Leibe, J. Matas, N. Sebe, & M. Welling, Eds.). In B. Leibe, J. Matas, N. Sebe, & M. Welling (Eds.), *Computer vision – ECCV 2016*, Cham, Springer International Publishing. https://doi.org/10.1007/978-3-319-46475-6_43

Jolicoeur-Martineau, A. (2018). The relativistic discriminator: A key element missing from standard GAN. *arXiv:1807.00734 [cs, stat]*, arxiv 1807.00734. Retrieved February 18, 2021, from http://arxiv.org/abs/1807.00734

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kolda, T. G., & Bader, B. W. (2009). Tensor decompositions and applications [Publisher: Society for Industrial and Applied Mathematics]. *SIAM Review*, *51*(3), 455–500. https://doi.org/10.1137/07070111X

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*, 1097–1105.

Lanaras, C., Bioucas-Dias, J., Baltsavias, E., & Schindler, K. (2017). Super-resolution of multispectral multiresolution images from a single sensor. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. Retrieved July 13, 2021, from https://openaccess.thecvf.com/content_cvpr_2017_workshops/w18/html/Lanaras_Super-Resolution_of_Multispectral_CVPR_2017_paper.html

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition [Conference Name: Proceedings of the IEEE]. *Proceedings of the IEEE*, *86*(11), 2278–2324. https://doi.org/10.1109/5.726791

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition [Publisher: MIT Press]. *Neural computation*, *1*(4), 541–551.

Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., & Shi, W. (2017). Photo-realistic single image super-resolution using a generative adversarial network. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Retrieved October 12, 2020, from https://openaccess.thecvf.com/content_cvpr_2017/html/Ledig_Photo-Realistic_Single_Image_CVPR_2017_paper.html

Lexico. (2021). *TEMPORAL | definition of TEMPORAL by oxford dictionary on lexico.com also meaning of TEMPORAL* [Lexico dictionaries | english]. Retrieved October 11, 2021, from https://www.lexico.com/definition/temporal

Lim, B., Son, S., Kim, H., Nah, S., & Mu Lee, K. (2017). Enhanced deep residual networks for single image super-resolution. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. Retrieved April 13, 2021, from https://openaccess.thecvf.com/content_cvpr_2017_workshops/w12/html/Lim_Enhanced_Deep_Residual_CVPR_2017_paper.html

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. *arXiv:1411.4038 [cs]*, arxiv 1411.4038. Retrieved July 6, 2021, from http://arxiv.org/abs/1411.4038

Ma, C., Yang, C.-Y., Yang, X., & Yang, M.-H. (2017). Learning a no-reference quality metric for single-image super-resolution. *Computer Vision and Image Understanding*, *158*, 1–16. https://doi.org/10.1016/j.cviu.2016.12.009

Ma, C., Rao, Y., Cheng, Y., Chen, C., Lu, J., & Zhou, J. (2020). Structure-preserving super resolution with gradient guidance. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Retrieved July 20, 2021, from https://openaccess.thecvf.com/content_CVPR_2020/html/Ma_Structure-Preserving_Super_Resolution_With_Gradient_Guidance_CVPR_2020_paper.html

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models, In *In ICML workshop on deep learning for audio, speech and language processing*.

Maxar. (2019a, August 26). GeoEye-1 data sheet. Maxar. Retrieved January 18, 2021, from https://resources.maxar.com/data-sheets/geoeye-1

Maxar. (2019b). Maxar technologies reports failure of its WorldView-4 imaging satellite. Retrieved July 13, 2021, from https://www.prnewswire.com/news-releases/maxar-technologies-reports-failure-of-its-worldview-4-imaging-satellite-300773567.html

Maxar. (2019c, August 29). WorldView-2 data sheet. Maxar. Retrieved January 18, 2021, from https://resources.maxar.com/data-sheets/worldview-2

Maxar. (2020, July 23). View-ready imagery. Retrieved October 12, 2021, from https://resources.maxar.com/satellite-access/view-ready-imagery-data-sheet

Mika, A. M. (1997). Three decades of landsat instruments [Publisher: [Falls Church, Va.] American Society of Photogrammetry.]. *Photogrammetric Engineering and Remote Sensing*, *63*(7), 839–852.

Mitchell, T. M. (1997). *Machine learning*. New York, McGraw-Hill.

Mittal, A., Soundararajan, R., & Bovik, A. C. (2013). Making a "completely blind" image quality analyzer [Conference Name: IEEE Signal Processing Letters]. *IEEE Signal Processing Letters*, *20*(3), 209–212. https://doi.org/10.1109/LSP.2012.2227726

Müller, M. U., Ekhtiari, N., Almeida, R. M., & Rieke, C. (2020). Super-resolution of multispectral satellite images using convolutional neural networks. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, *V-1-2020*arxiv 2002.00580, 33–40. https://doi.org/10.5194/isprs-annals-V-1-2020-33-2020

NASA. (2018). *Landsat: Benefiting society for fifty years*. Retrieved October 3, 2021, from https://landsat.gsfc.nasa.gov/how-landsat-helps/case-studies

NASA. (2021a, July 13). *Data processing levels | earthdata*. Retrieved October 12, 2021, from https://earthdata.nasa.gov/collaborate/open-data-services-and-software/data-information-policy/data-levels/

NASA. (2021b, September 28). *NASA and USGS launch landsat 9* [Publisher: NASA Earth Observatory]. Retrieved October 3, 2021, from https://earthobservatory.nasa.gov/images/148888/nasa-and-usgs-launch-landsat-9

Ng, A. (2018, September). *Machine learning yearning* (Draft Version 2018-09). Unpublished. Retrieved January 25, 2021, from https://www.deeplearning.ai/machine-learning-yearning/

Planet, L. (2021, February). Planet imagery product specifications. Retrieved July 13, 2021, from https://www.planet.com/products/satellite-imagery/files/1610.06_Spec%20Sheet_Combined_Imagery_Product_Letter_ENGv1.pdf

Pouliot, D., Latifovic, R., Pasher, J., & Duffe, J. (2018). Landsat super-resolution enhancement using convolution neural networks and sentinel-2 for training [Number: 3 Pub-

lisher: Multidisciplinary Digital Publishing Institute]. *Remote Sensing*, *10*(3), 394. https://doi.org/10.3390/rs10030394

Rabbi, J., Ray, N., Schubert, M., Chowdhury, S., & Chao, D. (2020). Small-object detection in remote sensing images with end-to-end edge-enhanced GAN and object detector network [Number: 9 Publisher: Multidisciplinary Digital Publishing Institute]. *Remote Sensing*, *12*(9), 1432. https://doi.org/10.3390/rs12091432

Radford, A., Metz, L., & Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv:1511.06434 [cs]*, arxiv 1511.06434. Retrieved March 2, 2021, from http://arxiv.org/abs/1511.06434

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors [Number: 6088 Publisher: Nature Publishing Group]. *Nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, *61* arxiv 1404.7828, 85–117. https://doi.org/10.1016/j.neunet.2014.09.003

Shermeyer, J., & Van Etten, A. (2019). The effects of super-resolution on object detection performance in satellite imagery. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. Retrieved April 19, 2021, from https://openaccess.thecvf.com/content_CVPRW_2019/html/EarthVision/Shermeyer_The_Effects_of_Super-Resolution_on_Object_Detection_Performance_in_Satellite_CVPRW_2019_paper.html

Shoeiby, M., Robles-Kelly, A., Wei, R., & Timofte, R. (2019). PIRM2018 challenge on spectral image super-resolution: Dataset and study. *arXiv:1904.00540 [cs]*, arxiv 1904.00540. Retrieved October 12, 2020, from http://arxiv.org/abs/1904.00540

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556 [cs]*, arxiv 1409.1556. Retrieved April 12, 2021, from http://arxiv.org/abs/1409.1556

Siu, K., Stuart, D. M., Mahmoud, M., & Moshovos, A. (2018, September). Memory requirements for convolutional neural network hardware accelerators, In *2018 IEEE international symposium on workload characterization (IISWC)*. 2018 IEEE International Symposium on Workload Characterization (IISWC). https://doi.org/10.1109/IISWC.2018.8573527

Soh, J. W., Park, G. Y., Jo, J., & Cho, N. I. (2019). Natural and realistic single image super-resolution with explicit natural manifold discrimination. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Retrieved July 20, 2021, from https://openaccess.thecvf.com/content_CVPR_2019/html/Soh_Natural_and_

Realistic _ Single _ Image _ Super - Resolution _ With _ Explicit _ Natural _ Manifold _ CVPR _ 2019 _ paper.html

Stathaki, T. (2011, August 29). *Image fusion: Algorithms and applications* [Google-Books-ID: VmvY4MTMFTwC]. Elsevier.

Steele, A. (2018, June 20). *Satellite data processing & multispectral data analysis.*

Szandała, T. (2021). Review and comparison of commonly used activation functions for deep neural networks (A. K. Bhoi, P. K. Mallick, C.-M. Liu, & V. E. Balas, Eds.). In A. K. Bhoi, P. K. Mallick, C.-M. Liu, & V. E. Balas (Eds.), *Bio-inspired neurocomputing*. Singapore, Springer. https://doi.org/10.1007/978-981-15-5495-7_11

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017, February 12). Inception-v4, inception-ResNet and the impact of residual connections on learning, In *Thirty-first AAAI conference on artificial intelligence*. Thirty-First AAAI Conference on Artificial Intelligence. Retrieved August 28, 2021, from https://www.aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14806

Tan, M., & Le, Q. V. (2020). EfficientNet: Rethinking model scaling for convolutional neural networks. *arXiv:1905.11946 [cs, stat]*, arxiv 1905.11946. Retrieved July 6, 2021, from http://arxiv.org/abs/1905.11946

TensorFlow. (2020, October 1). *Tf.keras.layers.LeakyReLU API documentation* [TensorFlow]. Retrieved March 9, 2021, from https://www.tensorflow.org/versions/r2.3/api_docs/python/tf/keras/layers/LeakyReLU

TensorFlow. (2021, March 19). *Better performance with the tf.data API | TensorFlow core* [TensorFlow]. Retrieved July 7, 2021, from https://www.tensorflow.org/guide/data_performance

Timofte, R., Agustsson, E., Van Gool, L., Yang, M.-H., & Zhang, L. (2017). NTIRE 2017 challenge on single image super-resolution: Methods and results. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. Retrieved August 28, 2021, from https://openaccess.thecvf.com/content_cvpr_2017_workshops/w12/html/Timofte_NTIRE_2017_Challenge_CVPR_2017_paper.html

Updike, T., & Comp, C. (2010). Radiometric use of WorldView-2 imagery, 17.

USGS. (1976, July 5). Landsat 2 image EMP217r18_2m19760705. Retrieved October 24, 2021, from https://earthexplorer.usgs.gov/scene/metadata/full/5e83a358acceddc0/EMP217R18_2M19760705/

Vasu, S., Thekke Madam, N., & Rajagopalan, A. N. (2018). Analyzing perception-distortion tradeoff using enhanced perceptual super-resolution network. Proceedings of the Euro-

pean Conference on Computer Vision (ECCV) Workshops. Retrieved April 13, 2021, from https://openaccess.thecvf.com/content_eccv_2018_workshops/w25/html/Vasu_Analyzing_Perception-Distortion_Tradeoff_using_Enhanced_Perceptual_Super-resolution_Network_ECCVW_2018_paper.html

Wang, X. (2019, June 2). *Xinntao/ESRGAN* [original-date: 2018-08-31T08:18:41Z]. Retrieved July 8, 2021, from https://github.com/xinntao/ESRGAN

Wang, X., Yu, K., Dong, C., & Loy, C. C. (2018). Recovering realistic texture in image super-resolution by deep spatial feature transform. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Retrieved August 28, 2021, from https://openaccess.thecvf.com/content_cvpr_2018/html/Wang_Recovering_Realistic_Texture_CVPR_2018_paper.html

Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., Loy, C. C., Qiao, Y., & Tang, X. (2018). ESRGAN: Enhanced super-resolution generative adversarial networks [version: 2]. *arXiv:1809.00219 [cs]*, arxiv 1809.00219. Retrieved October 29, 2020, from http://arxiv.org/abs/1809.00219

Wang, Z., & Bovik, A. C. (2006, December 1). *Modern image quality assessment* [Google-Books-ID: CXhgAQAAQBAJ]. Morgan & Claypool Publishers.

Wang, Z., & Bovik, A. C. (2009). Mean squared error: Love it or leave it? a new look at signal fidelity measures [Conference Name: IEEE Signal Processing Magazine]. *IEEE Signal Processing Magazine*, *26*(1), 98–117. https://doi.org/10.1109/MSP.2008.930649

Yang, C.-Y., Ma, C., & Yang, M.-H. (2014). Single-image super-resolution: A benchmark (D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars, Eds.). In D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer vision – ECCV 2014*, Cham, Springer International Publishing. https://doi.org/10.1007/978-3-319-10593-2_25

Zhang, Y., Tian, Y., Kong, Y., Zhong, B., & Fu, Y. (2018). Residual dense network for image super-resolution. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Retrieved March 9, 2021, from https://openaccess.thecvf.com/content_cvpr_2018/html/Zhang_Residual_Dense_Network_CVPR_2018_paper.html

Zhou Wang, Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image quality assessment: From error visibility to structural similarity [Conference Name: IEEE Transactions on Image Processing]. *IEEE Transactions on Image Processing*, *13*(4), 600–612. https://doi.org/10.1109/TIP.2003.819861

Appendix A

Random patches from the GeoEye-1 test set

The images in this appendix are intended to be viewed on a high resolution monitor.

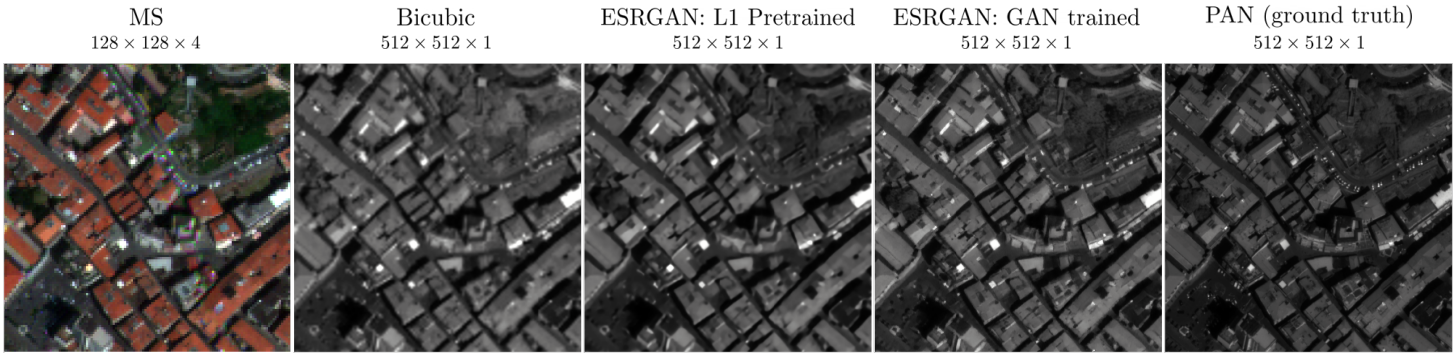


Figure A.1: La Spezia 2013-07-15, *Satellite image © 2021 Maxar Technologies*

Figure A.2: La Spezia 2019-01-03, *Satellite image © 2021 Maxar Technologies*

Figure A.3: Toulon 2016-09-21, *Satellite image © 2021 Maxar Technologies*

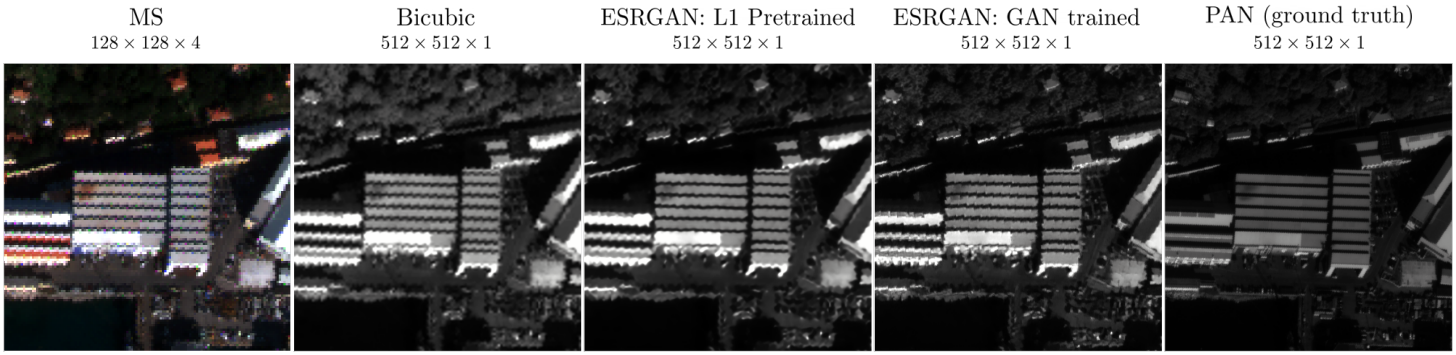


Figure A.4: La Spezia 2011-01-24, *Satellite image © 2021 Maxar Technologies*

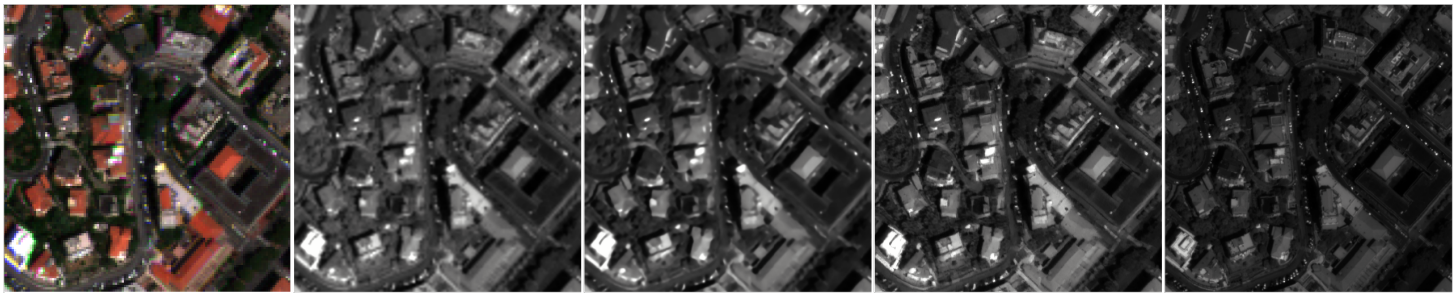


Figure A.5: La Spezia 2013-07-15, *Satellite image © 2021 Maxar Technologies*

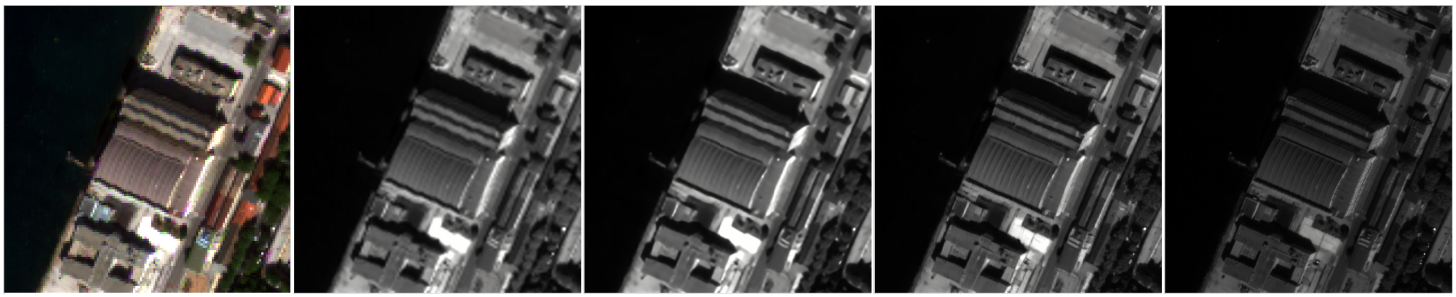


Figure A.6: Toulon 2016-09-21, *Satellite image © 2021 Maxar Technologies*

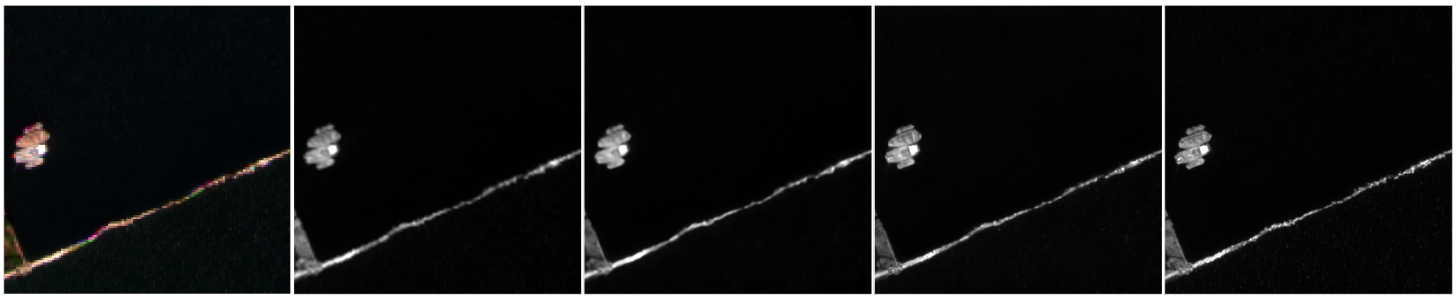


Figure A.7: La Spezia 2013-07-23, *Satellite image © 2021 Maxar Technologies*

Figure A.8: La Spezia 2013-07-18, *Satellite image © 2021 Maxar Technologies*

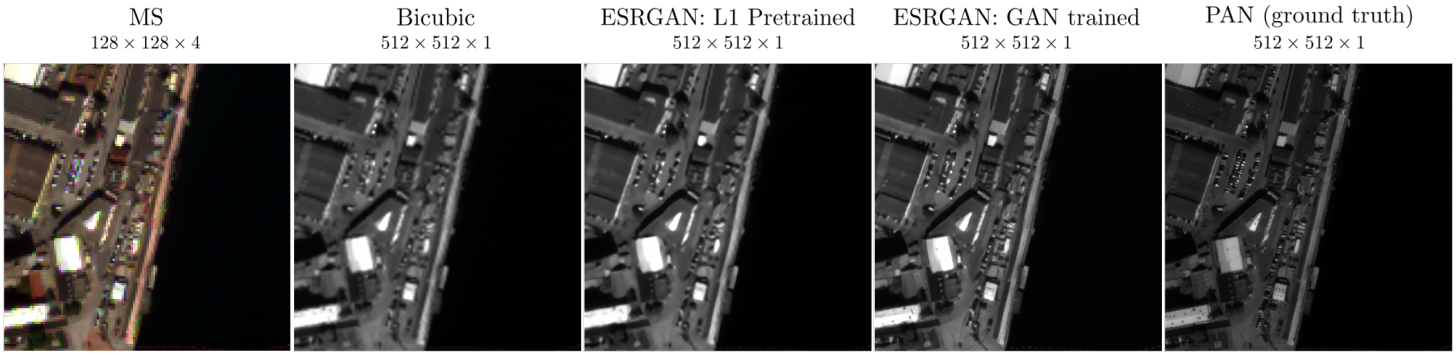


Figure A.9: Toulon 2016-09-21, *Satellite image © 2021 Maxar Technologies*

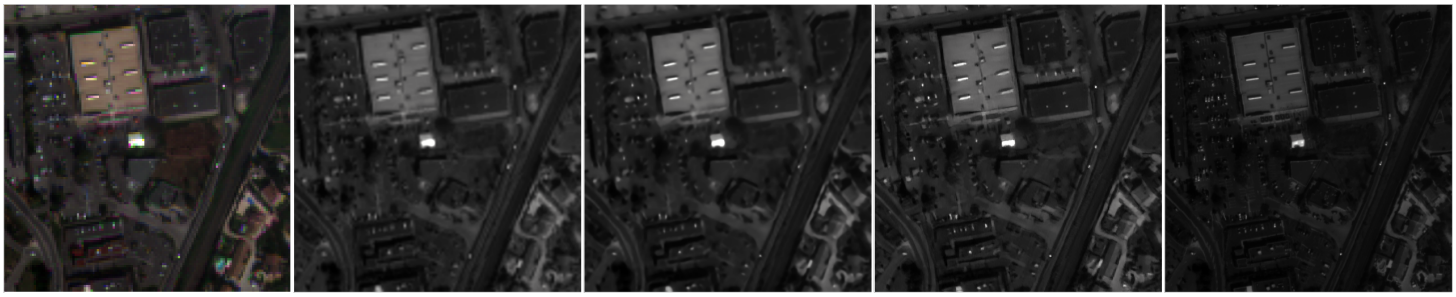


Figure A.10: Toulon 2016-03-26, *Satellite image © 2021 Maxar Technologies*

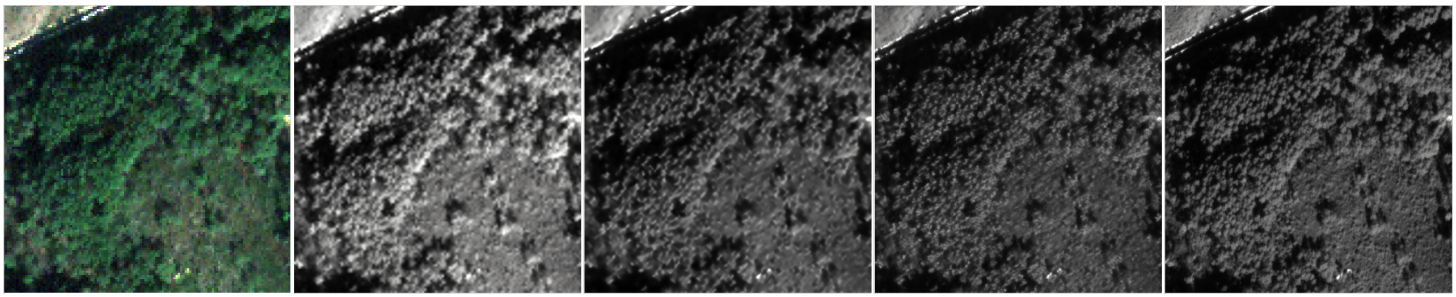


Figure A.11: La Spezia 2011-01-24, *Satellite image © 2021 Maxar Technologies*

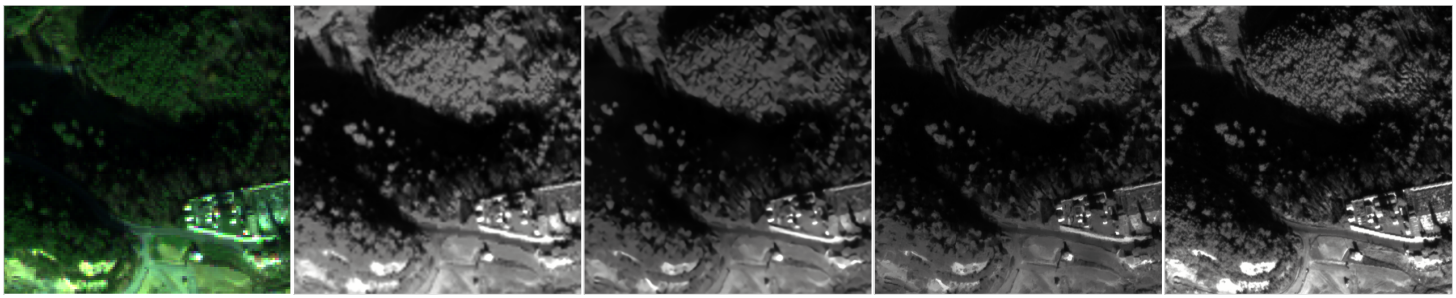


Figure A.12: La Spezia 2019-01-03, *Satellite image © 2021 Maxar Technologies*

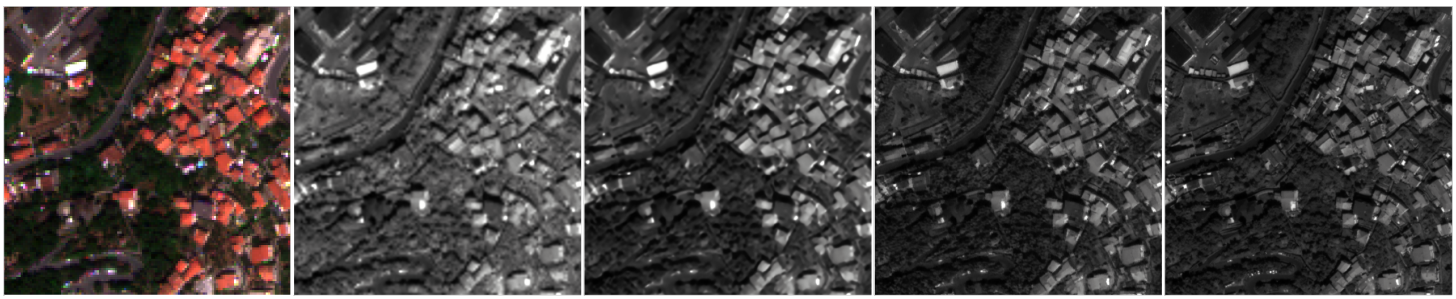


Figure A.13: La Spezia 2013-07-23, *Satellite image © 2021 Maxar Technologies*

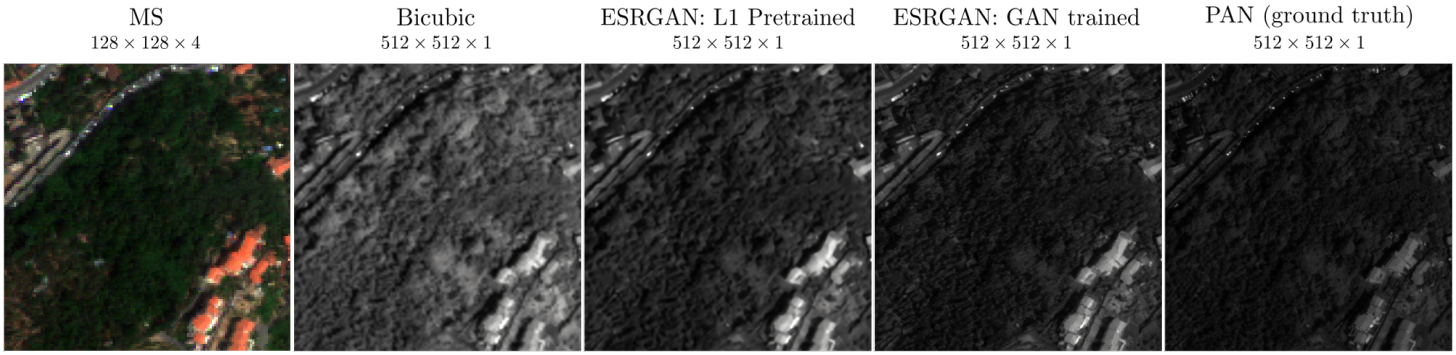


Figure A.14: La Spezia 2013-07-18, *Satellite image © 2021 Maxar Technologies*

Figure A.15: Toulon 2017-09-05, *Satellite image © 2021 Maxar Technologies*

Figure A.16: La Spezia 2013-07-23, *Satellite image © 2021 Maxar Technologies*

Figure A.17: Toulon 2016-03-26, *Satellite image © 2021 Maxar Technologies*

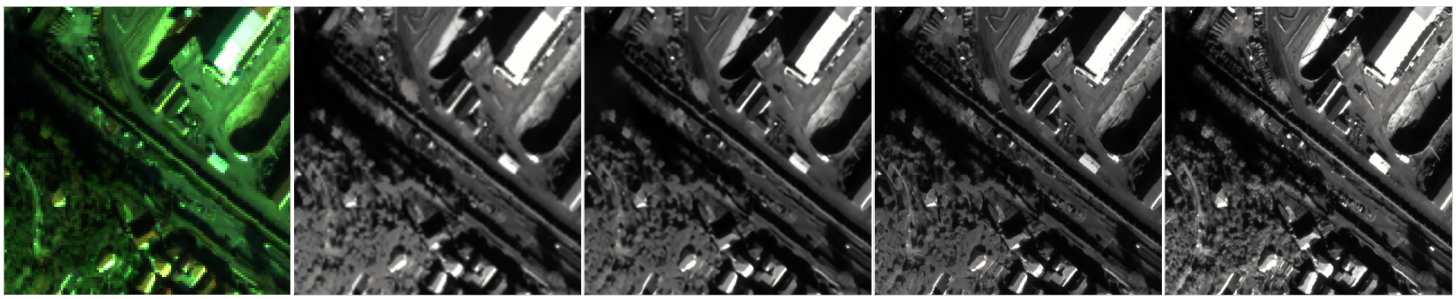


Figure A.18: La Spezia 2019-01-03, *Satellite image © 2021 Maxar Technologies*

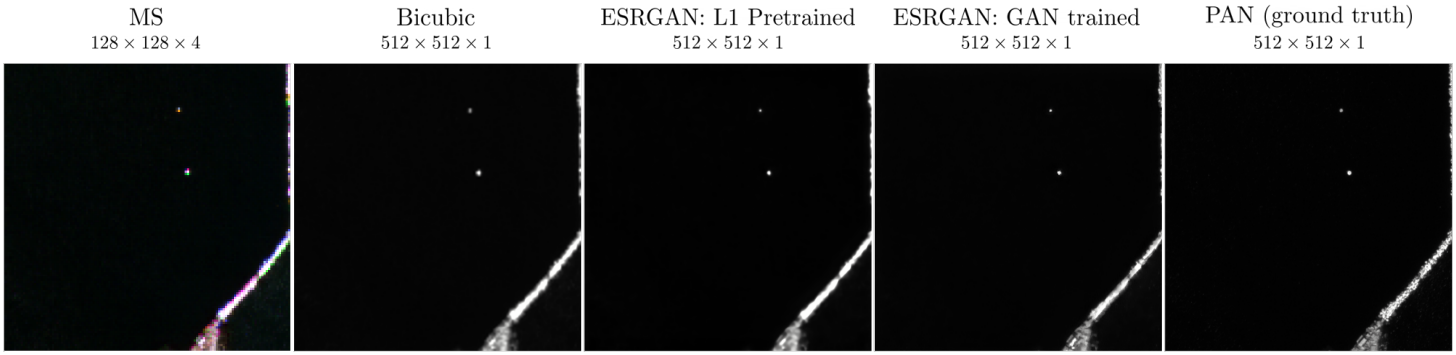


Figure A.19: La Spezia 2013-07-23, *Satellite image © 2021 Maxar Technologies*

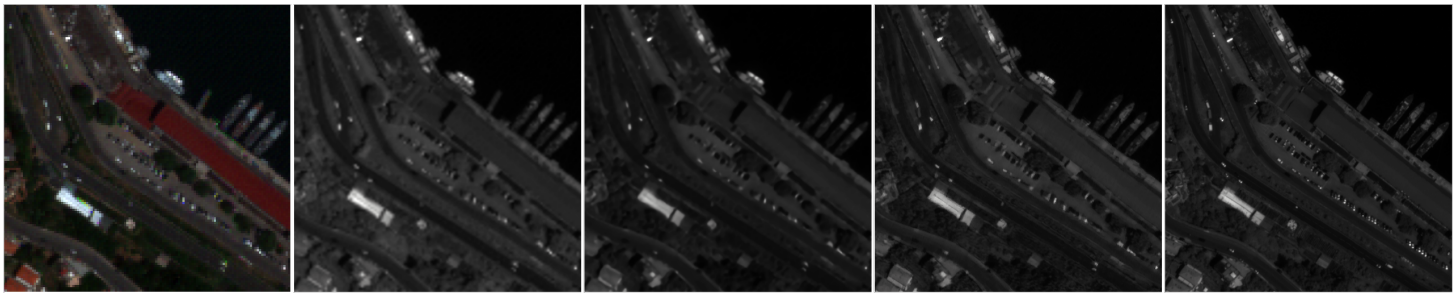


Figure A.20: La Spezia 2013-07-15, *Satellite image © 2021 Maxar Technologies*

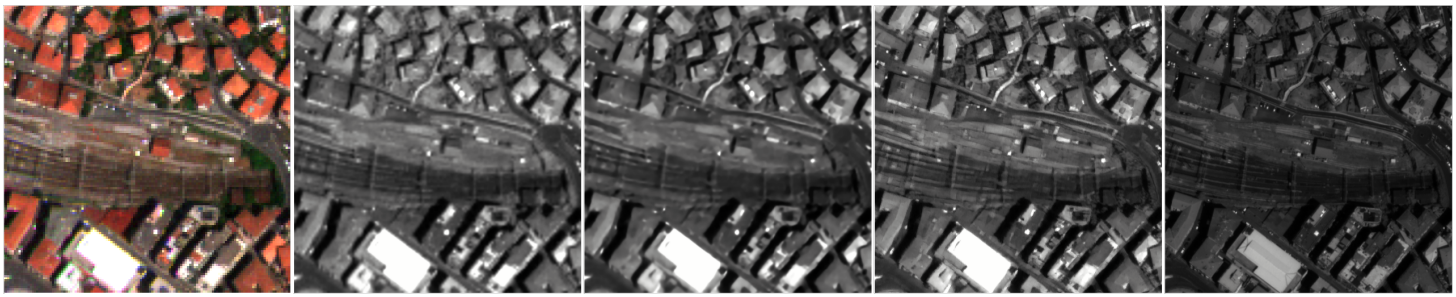


Figure A.21: La Spezia 2013-07-15, *Satellite image © 2021 Maxar Technologies*

Figure A.22: La Spezia 2013-07-23, *Satellite image © 2021 Maxar Technologies*

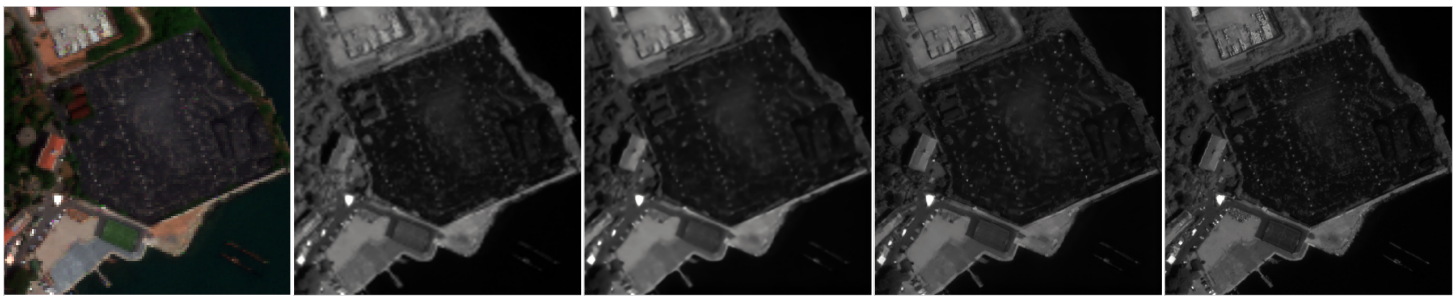


Figure A.23: La Spezia 2013-07-18, *Satellite image © 2021 Maxar Technologies*

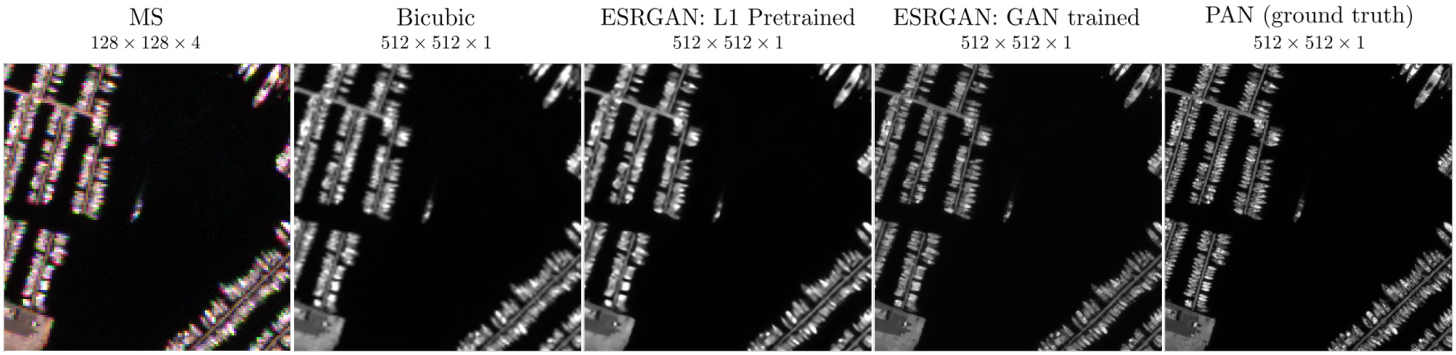


Figure A.24: Toulon 2010-06-08, *Satellite image © 2021 Maxar Technologies*

Figure A.25: Toulon 2016-09-21, *Satellite image © 2021 Maxar Technologies*

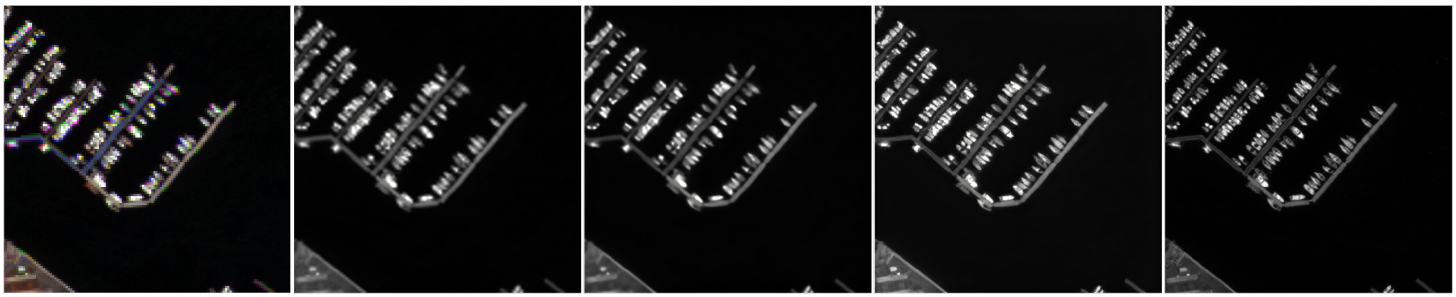


Figure A.26: La Spezia 2013-07-18, *Satellite image © 2021 Maxar Technologies*

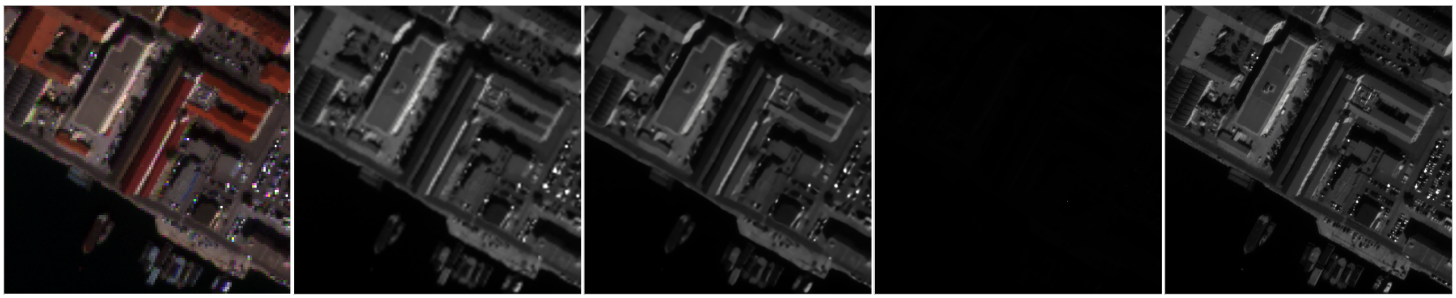


Figure A.27: Toulon 2016-09-21, *Satellite image © 2021 Maxar Technologies*

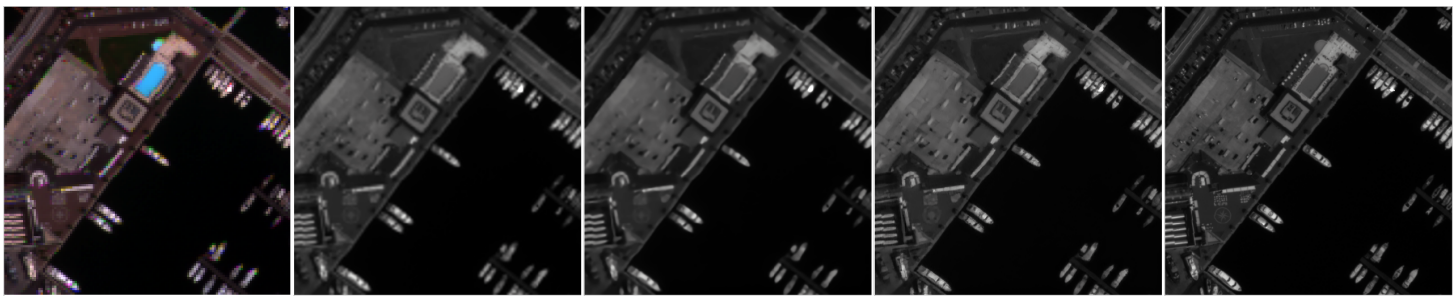


Figure A.28: La Spezia 2013-07-18, *Satellite image © 2021 Maxar Technologies*

Appendix B

Random patches from the WorldView-2 test set

The images in this appendix are intended to be viewed on a high resolution monitor.

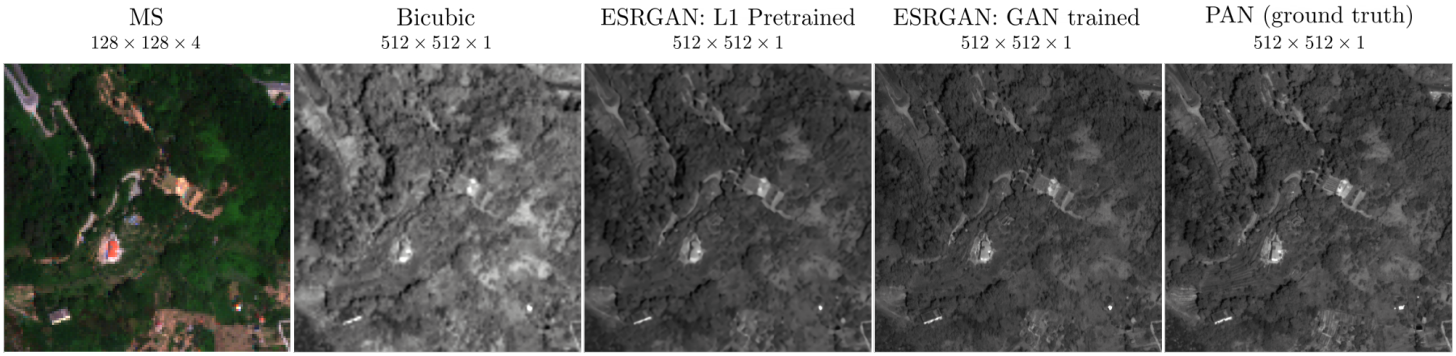


Figure B.1: La Spezia 2015-05-21, *Satellite image © 2021 Maxar Technologies*

Figure B.2: La Spezia 2017-07-31, *Satellite image © 2021 Maxar Technologies*

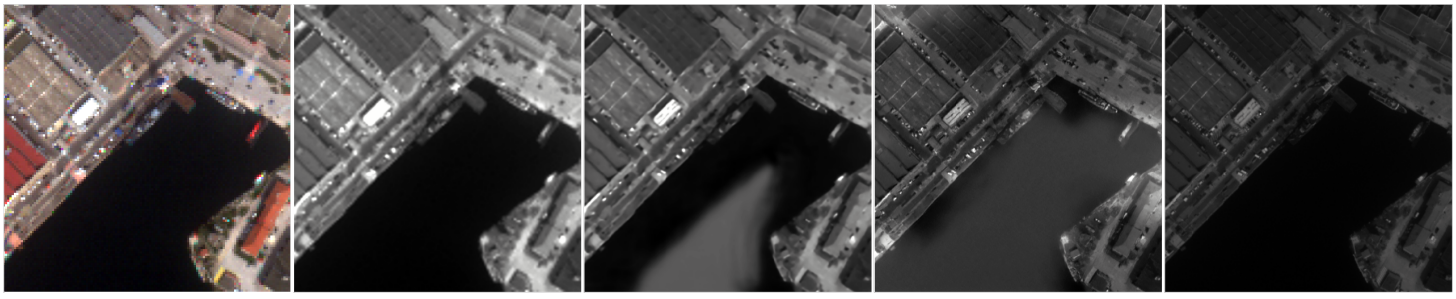


Figure B.3: Toulon 2016-03-25, *Satellite image © 2021 Maxar Technologies*

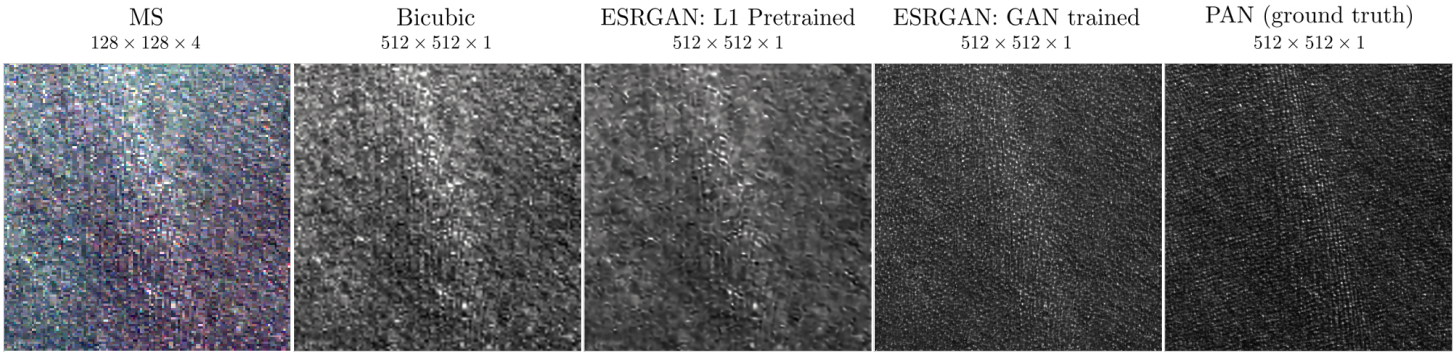


Figure B.4: La Spezia 2015-03-19, *Satellite image © 2021 Maxar Technologies*

Figure B.5: La Spezia 2017-03-10, *Satellite image © 2021 Maxar Technologies*

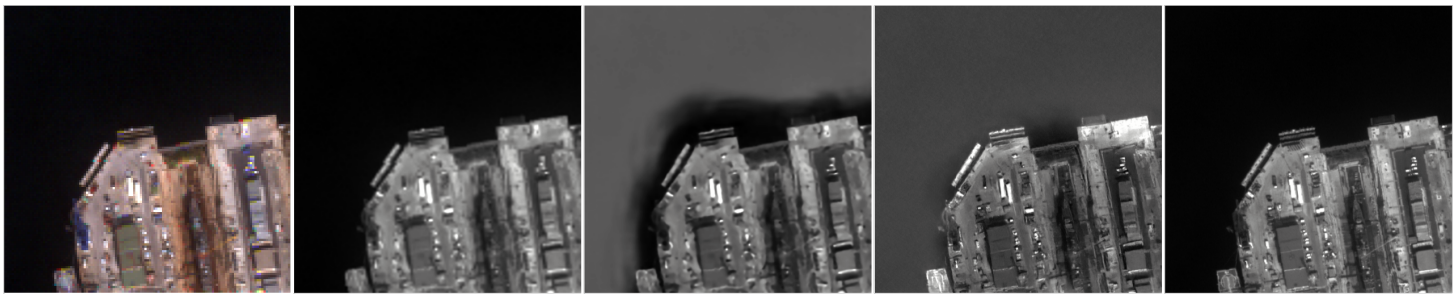


Figure B.6: Toulon 2016-03-25, *Satellite image © 2021 Maxar Technologies*

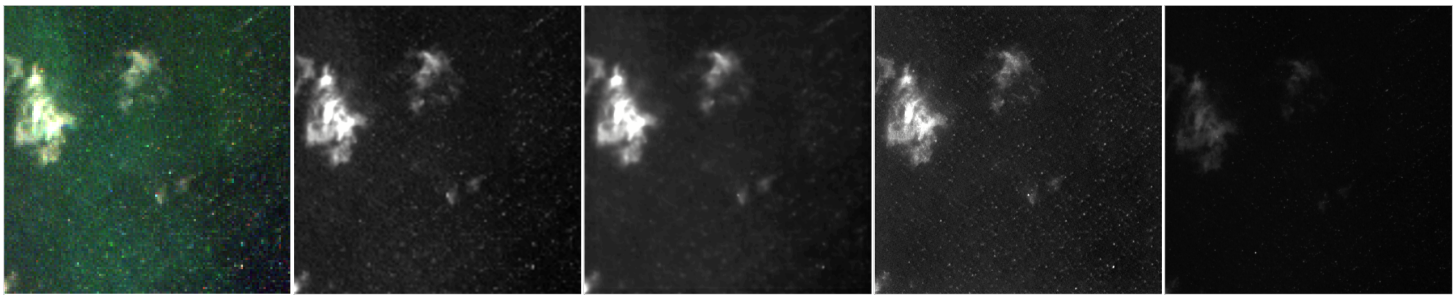


Figure B.7: La Spezia 2015-05-21, *Satellite image © 2021 Maxar Technologies*

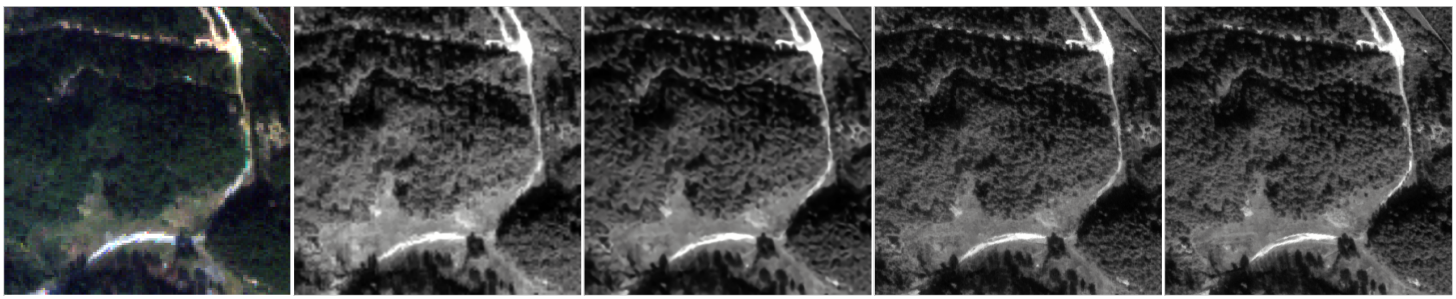


Figure B.8: La Spezia 2015-03-19, *Satellite image © 2021 Maxar Technologies*

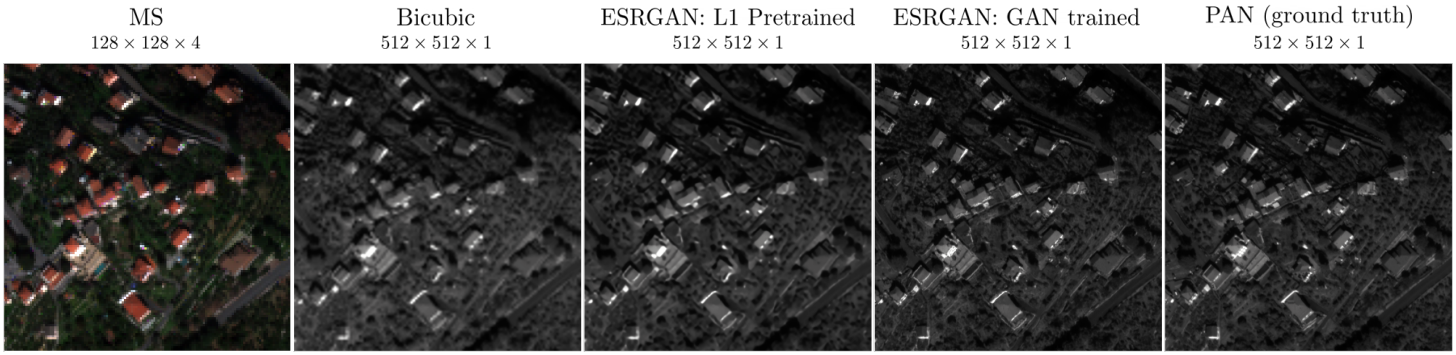


Figure B.9: La Spezia 2017-03-10, *Satellite image © 2021 Maxar Technologies*

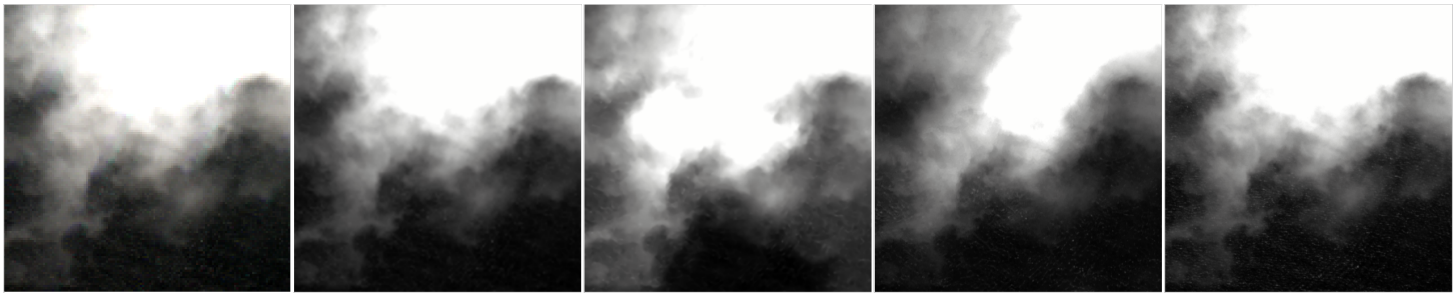


Figure B.10: Toulon 2016-03-30, *Satellite image © 2021 Maxar Technologies*

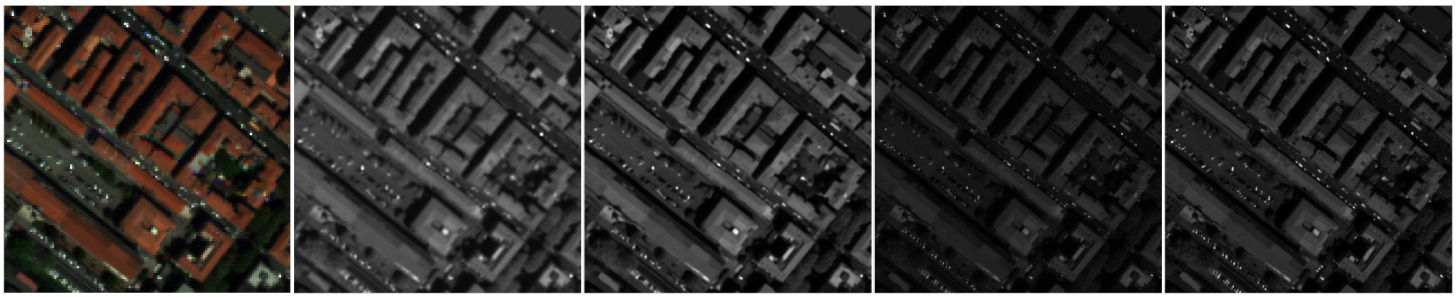


Figure B.11: La Spezia 2017-07-31, *Satellite image © 2021 Maxar Technologies*

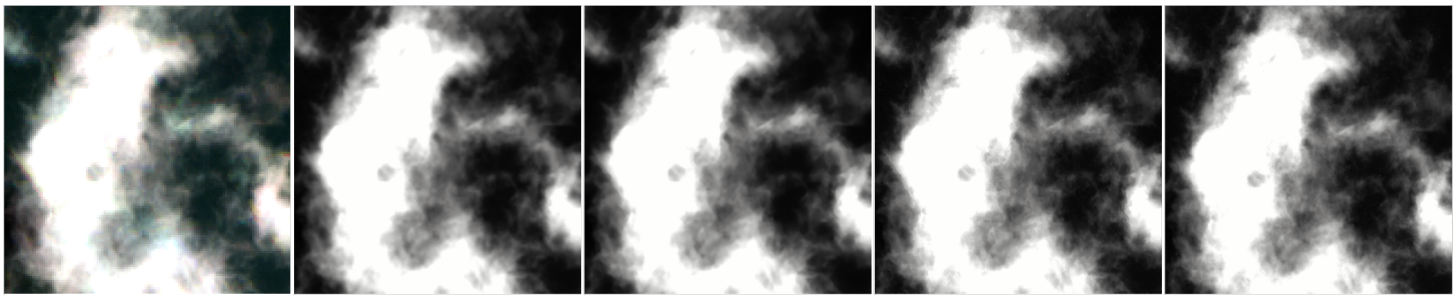


Figure B.12: La Spezia 2015-05-21, *Satellite image © 2021 Maxar Technologies*

Figure B.13: Toulon 2016-03-25, *Satellite image © 2021 Maxar Technologies*

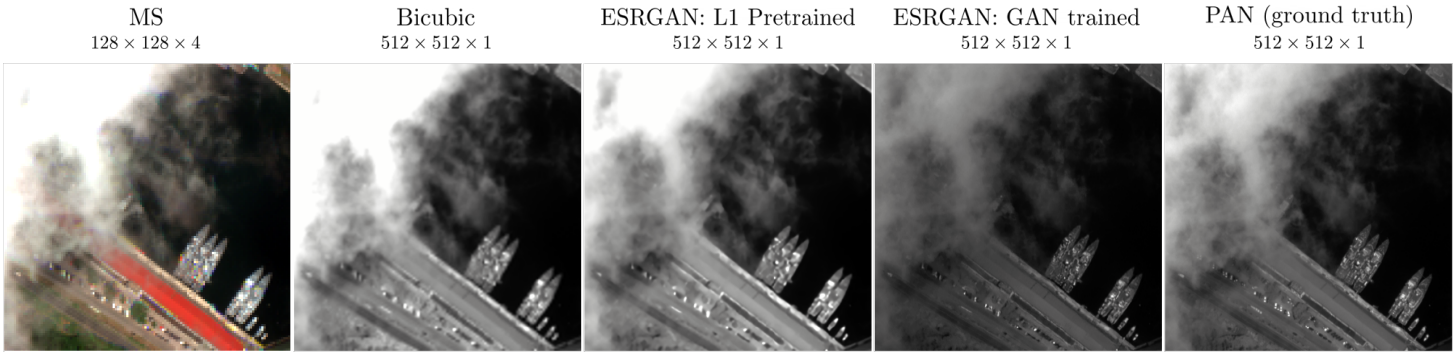


Figure B.14: La Spezia 2015-05-21, *Satellite image © 2021 Maxar Technologies*

Figure B.15: Toulon 2016-03-25, *Satellite image © 2021 Maxar Technologies*

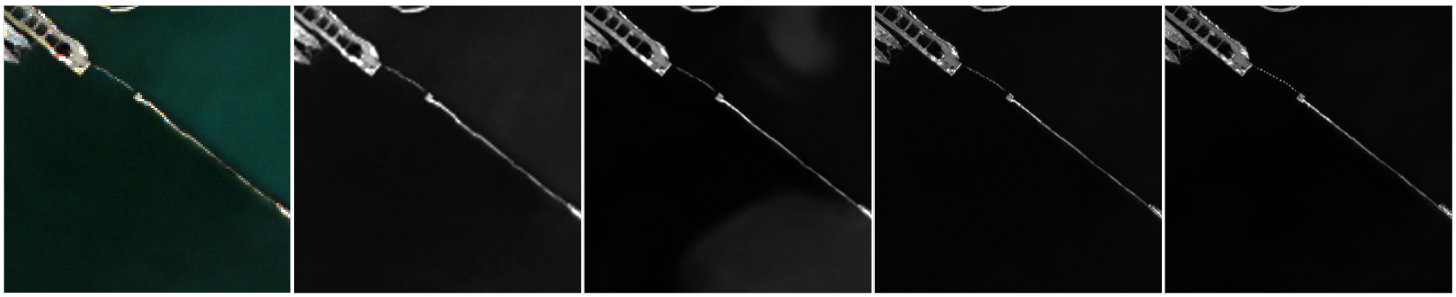


Figure B.16: La Spezia 2017-03-10, *Satellite image © 2021 Maxar Technologies*

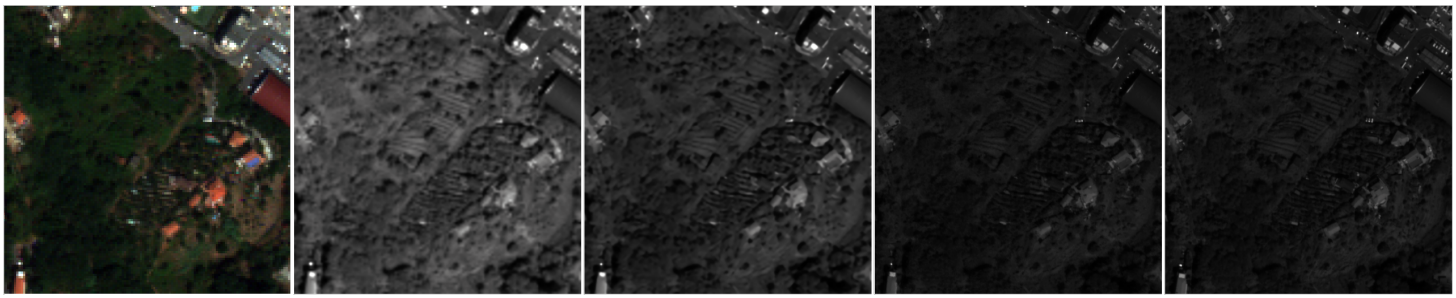


Figure B.17: La Spezia 2017-07-31, *Satellite image © 2021 Maxar Technologies*

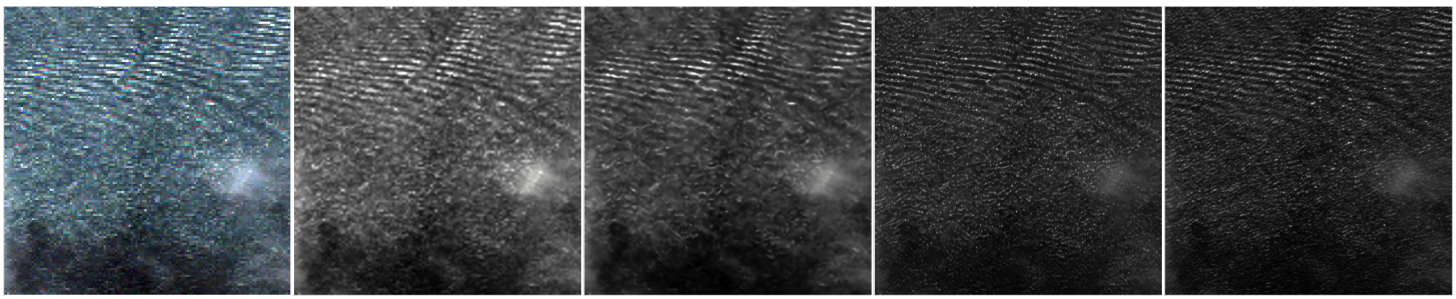


Figure B.18: Toulon 2016-03-30, *Satellite image © 2021 Maxar Technologies*

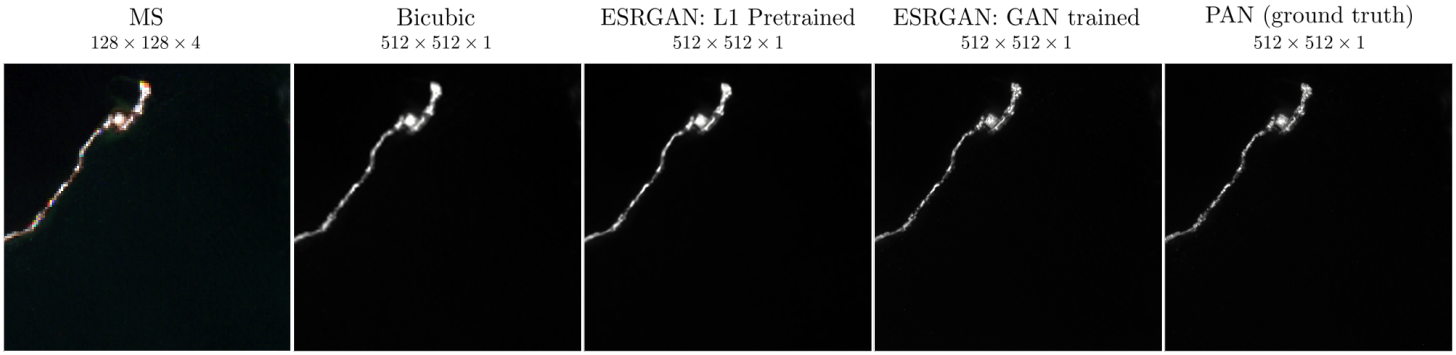


Figure B.19: La Spezia 2015-05-21, *Satellite image © 2021 Maxar Technologies*

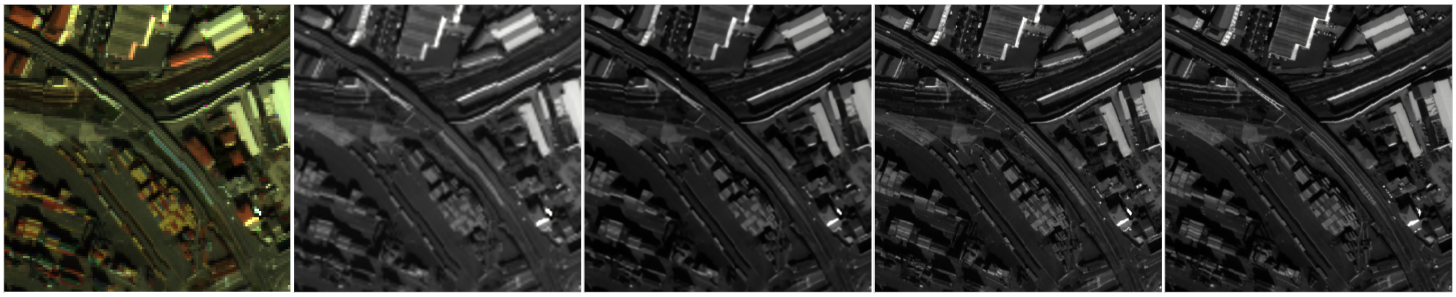


Figure B.20: La Spezia 2017-03-10, *Satellite image © 2021 Maxar Technologies*

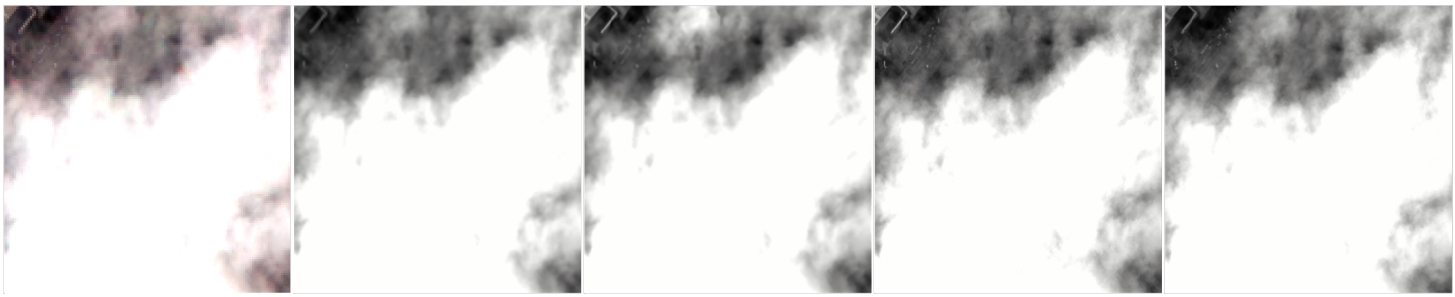


Figure B.21: La Spezia 2015-05-21, *Satellite image © 2021 Maxar Technologies*

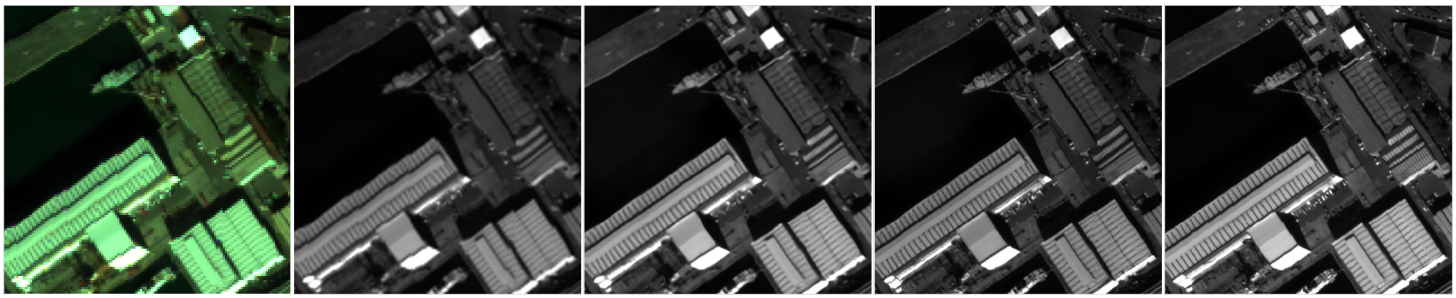


Figure B.22: La Spezia 2017-03-10, *Satellite image © 2021 Maxar Technologies*

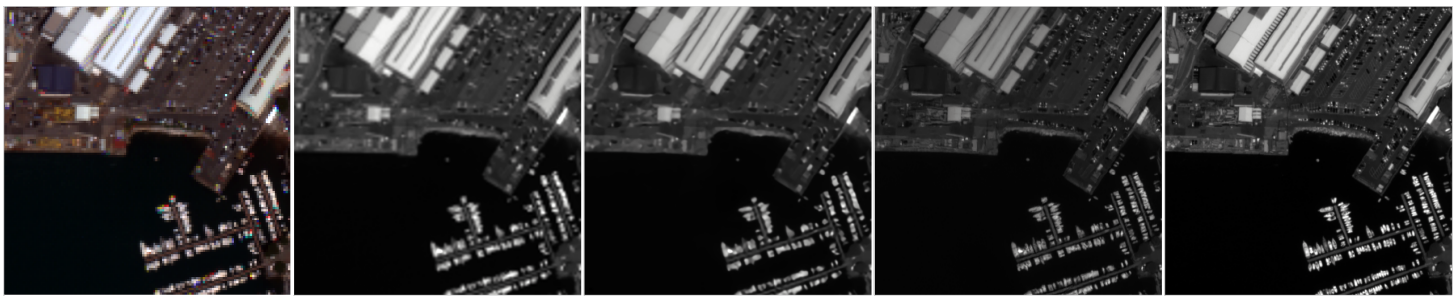


Figure B.23: La Spezia 2015-05-21, *Satellite image © 2021 Maxar Technologies*

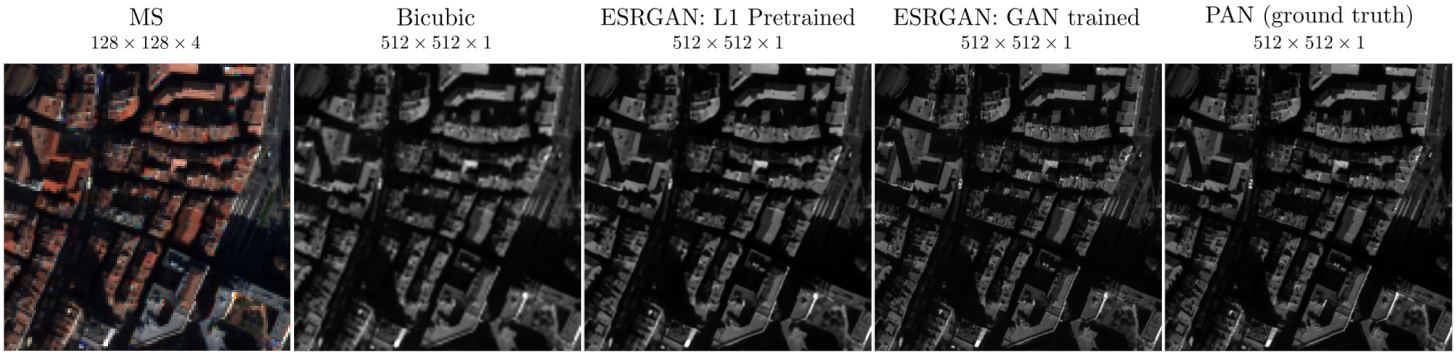


Figure B.24: Toulon 2011-12-22, *Satellite image © 2021 Maxar Technologies*

Figure B.25: Toulon 2016-03-30, *Satellite image © 2021 Maxar Technologies*

Figure B.26: La Spezia 2015-05-21, *Satellite image © 2021 Maxar Technologies*

Figure B.27: Toulon 2016-03-30, *Satellite image © 2021 Maxar Technologies*

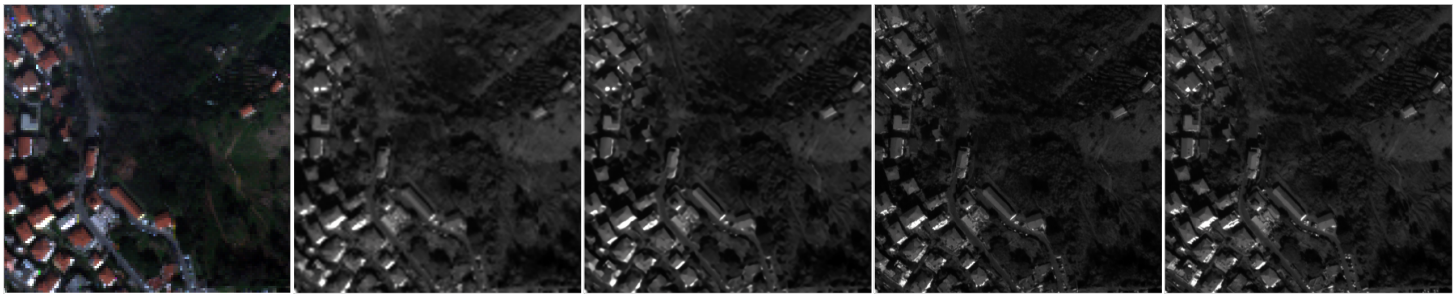


Figure B.28: La Spezia 2017-03-10, *Satellite image © 2021 Maxar Technologies*

Appendix C

Satellite image metadata

The table contains the identification numbers needed to acquire the same satellite images from Maxar as was used in the experiments. More columns of metadata, for instance exact time of capture, image size and exact spatial resolution, are available in the project's GitHub repository, https://github.com/onordberg/multispectral-super-resolution.

uid	string_uid	part	pan_catalog_id	ms_catalog_id
0	GE01_La_Spezia_2009_09_25_011651186010_0	test	2030010563489B00	2030010563489800
1	GE01_La_Spezia_2011_01_24_011651197010_0	test	2030010563690B00	2030010563690500
2	GE01_La_Spezia_2012_02_23_011651192010_0	val	2030010563727300	2030010563727000
3	GE01_La_Spezia_2012_05_07_011651189010_0	val	2030010563738800	2030010563738300
4	GE01_La_Spezia_2012_05_16_011651187010_0	val	2030010563726F00	2030010563726D00
5	GE01_La_Spezia_2013_07_07_011651184010_0	val	2030010563735C00	2030010563735A00
6	GE01_La_Spezia_2013_07_15_011651183010_0	test	2030010563737600	2030010563737400
7	GE01_La_Spezia_2013_07_18_011651195010_0	test	2030010563719700	2030010563719400
8	GE01_La_Spezia_2013_07_23_011651202010_0	test	20300105636D9400	20300105636D9200
9	GE01_La_Spezia_2017_04_16_011651188010_0	val	2030010563736D00	2030010563736900
10	GE01_La_Spezia_2019_01_03_011651196010_0	test	203001056371F100	203001056371EC00
11	GE01_Toulon 2009_03_23_011651190010_0	val	2030010563738900	2030010563738500
12	GE01_Toulon 2010_06_08_011651191010_0	test	2030010563733400	2030010563733000
13	GE01_Toulon 2013_08_20_011651198010_0	val	2030010563465E00	2030010563465800
14	GE01_Toulon 2014_10_16_011651185010_0	test	203001056372E200	203001056372DA00
15	GE01_Toulon 2016_03_26_011651193010_0	test	2030010563482700	2030010563482300
16	GE01_Toulon 2016_09_21_011651200010_0	test	203001056372F600	203001056372F300
17	GE01_Toulon 2017_09_05_011651201010_0	test	2030010563733200	2030010563732D00
18	GE01_Toulon 2018_02_13_011651199010_0	val	203001056371EB00	203001056371E400
19	GE01_Toulon 2019_10_07_011651194010_0	val	2030010563158400	2030010563158200
20	WV02_La_Spezia_2010_08_06_011650744010_0	test	20300105635BC200	20300105635BC400
21	WV02_La_Spezia_2011_05_02_011650586010_0	train	2030010563517600	2030010563517A00
22	WV02_La_Spezia_2011_05_10_011650587010_0	train	203001056350B200	203001056350B400
23	WV02_La_Spezia_2011_10_28_011650745010_0	train	2030010563350A00	2030010563350D00
24	WV02_La_Spezia_2012_01_10_011650582010_0	train	2030010563525500	2030010563525B00

uid	string_uid	part	pan_catalog_id	ms_catalog_id
25	WV02_La_Spezia_2012_05_17_011650585010_0	train	2030010563306700	2030010563306900
26	WV02_La_Spezia_2013_10_31_011650588010_0	train	203001056351E500	203001056351E700
27	WV02_La_Spezia_2013_12_08_011650589010_0	train	2030010563517300	2030010563517800
28	WV02_La_Spezia_2014_11_20_011650595010_0	train	2030010563525300	2030010563525A00
29	WV02_La_Spezia_2015_01_05_011650746010_0	val	2030010563598800	2030010563598A00
30	WV02_La_Spezia_2015_03_19_011650598010_0	val	203001056350CE00	203001056350D000
31	WV02_La_Spezia_2015_03_19_011650600010_0	test	203001056350EC00	203001056350EE00
32	WV02_La_Spezia_2015_04_24_011650748010_0	train	203001056359B800	203001056359BB00
33	WV02_La_Spezia_2015_05_21_011650592010_0	test	2030010563517D00	2030010563517F00
34	WV02_La_Spezia_2017_03_10_011650750010_0	test	2030010563573100	2030010563573400
35	WV02_La_Spezia_2017_07_31_011650593010_0	test	203001056351CF00	203001056351D100
36	WV02_La_Spezia_2018_05_18_011650597010_0	val	2030010563526100	2030010563526300
37	WV02_La_Spezia_2018_05_29_011650583010_0	val	203001056351E200	203001056351E400
38	WV02_La_Spezia_2018_07_06_011650749010_0	train	2030010563593500	2030010563593700
39	WV02_La_Spezia_2019_05_23_011650590010_0	train	20300105634FCE00	20300105634FD000
40	WV02_La_Spezia_2019_06_03_011650596010_0	val	203001056350B800	203001056350BA00
41	WV02_La_Spezia_2019_10_11_011650594010_0	train	203001056301AE00	203001056301B100
42	WV02_Toulon_2010_11_04_011651051010_0	val	20300105636D8800	20300105636D8A00
43	WV02_Toulon_2011_05_04_011651058010_0	train	203001056312A600	203001056312A800
44	WV02_Toulon_2011_12_22_011651063010_0	test	20300105636DC500	20300105636DC800
45	WV02_Toulon_2013_03_16_011651062010_0	train	2030010563428C00	2030010563429000
46	WV02_Toulon_2013_04_07_011651056010_0	train	203001056312A000	203001056312A200
47	WV02_Toulon_2013_04_10_011651050010_0	val	20300105636D7B00	20300105636D7F00
48	WV02_Toulon_2013_10_09_011651061010_0	test	20300105636DA100	20300105636DA300
49	WV02_Toulon_2014_04_06_011651052010_0	train	20300105636D8200	20300105636D8400
50	WV02_Toulon_2015_11_16_011651049010_0	val	203001056359AD00	203001056359AF00
51	WV02_Toulon_2016_03_14_011651064010_0	train	20300105636D9800	20300105636D9A00
52	WV02_Toulon_2016_03_22_011651057010_0	train	20300105636D8500	20300105636D8700
53	WV02_Toulon_2016_03_25_011651060010_0	test	20300105636D8F00	20300105636D9100
54	WV02_Toulon_2016_03_30_011651053010_0	test	20300105636D9500	20300105636D9700
55	WV02_Toulon_2016_09_28_011651048010_0	val	20300105636D2E00	20300105636D3000
56	WV02_Toulon_2017_08_19_011651059010_0	test	20300105636D9D00	20300105636D9F00
57	WV02_Toulon_2019_07_29_011650877010_0	train	20300105630ECF00	20300105630ED300
58	WV02_Toulon_2019_08_04_011650878010_0	train	20300105630ECC00	20300105630ECE00
59	WV02_Toulon_2019_09_11_011650876010_0	train	20300105630EBC00	20300105630EBF00
60	WV02_Toulon_2019_10_16_011650874010_0	val	20300105630EC000	20300105630EC200
61	WV02_Toulon_2019_12_15_011650875010_0	train	20300105630EC600	20300105630EC800