ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Prospects of graph regularized super resolution in pan-sharpening

Bachelor Thesis

Mathis Erler

Friday 5th December, 2025

Advisors: Becker Alexander, Prof. Konrad Schindler
Institute of Geodesy and Photogrammetry, ETH Zürich

Abstract

Pansharpenig is an image fusion technique that fuses a high resolution black and white image a with low resolution multispectral image with the goal to increase the resolution of the multispectral image. This thesis implements a graph regularized optimization approach. The affinity graph is based on deep features and thereby learnable. A model without the graph is trained and two with the graph, with different optimization problems. The optimization based method with a spectral and a radiometric constraint performed best, while outperforming the benchmarks on some metrics. Visual inspection show more natural images, especially at large scaling factors. The optimization with only a spectral constraint is not suited to the pansharpening task. As the approach is end to end trainable, it can be integrated into a further model to optimize for a specific use case.

Contents

Contents ii

1 Introduction 1
- 1.1 Remote sensing . . . 1
- 1.2 Pan sharpening . . . 2
- 1.3 Machine Learning . . . 3

2 Background 6
- 2.1 Graph regularized super resolution . . . 6
- 2.2 Component Substitution Methods . . . 7
- 2.3 ARSIS . . . 8
- 2.4 Deep learning pansharpening approaches . . . 9

3 Data 12
- 3.1 Data source . . . 12
- 3.2 Prepocessing . . . 12
- 3.3 Issues with synthetic data . . . 13

4 Methods 14
- 4.1 Raw ResNet50 . . . 14
- 4.2 Optimized . . . 15
- 4.3 Extra Condition . . . 16
- 4.4 Details of the experimental setup . . . 16
- 4.5 Quantitative Image analysis . . . 17

5 Results 19
- 5.1 Learned Models . . . 19
- 5.2 Baselines . . . 24
- 5.3 Comparison . . . 28

6 Discussion 30

6.1 Computational cost and possible applications 30
6.2 Future Work 30
6.3 Conclusion and outlook 31

7 Acknowledgement **33**

Bibliography **34**

List of Figures **37**

A Appendix **41**
A.1 Evaluation of terminated runs 41

Introduction

1.1 Remote sensing

Since the development of the telescope in the 17th century remote sensing came a long way. With the introduction of photographic techniques, it became a quantitative science. Developments in aviation enables to mount cameras to kites and balloons, with a some experiments on pigeon mounted devices[1]. Airplanes lead to widespread adoption during the first world war. While early motivation was focused on military application, aerial imagery proofed early on to be a useful tool to tackle civil problems. As part of the New Deal, the US Agricultural Adjustment Administration conducted large scale aerial photography campaigns on US farmland to monitor the implementation of land conservation measures [2]. The development of radar in the second world war enabled another tool for remote sensing. With the launch of the first satellite sputnik a new area for remote sensing dawned. While the upfront cost of satellite launches is high, they enable the coverage of vast areas. In 1960 the Us launched TIROS, their first satellite that looked down to earth with a scientific purpose as its main intend. TIROS was a meteorological instrument on a sun synchronous orbit. ERTS(Landsat 1) followed in 1972, with a focus on the landmass of earth. Since then, numerous constellations were launched. Modern constellations like Sentinel 2 deliver global coverage at 15 m resolution at a repeat time of 6 days free to use for everybody under a creative commons licence. In addition to these science focused missions from government backed consortium, numerous private constellations are deployed. At the expense of global coverage and a consistent acquisition scheme, they allow for higher resolutions and shorter repeat times. The skysat constellation allows for over 5 daily revisits at a resolution of 50 cm [3]. An even higher resolution of 32 cm can be achieved using Maxar's Worldview constellation [4]. The introduction of these commercial constellations enabled the rise of industrial application. From high precision farming, resource monitoring, business intelligence and insurance. For governments, remote sensing data can provide an easy way to monitor its land use, evaluate nature

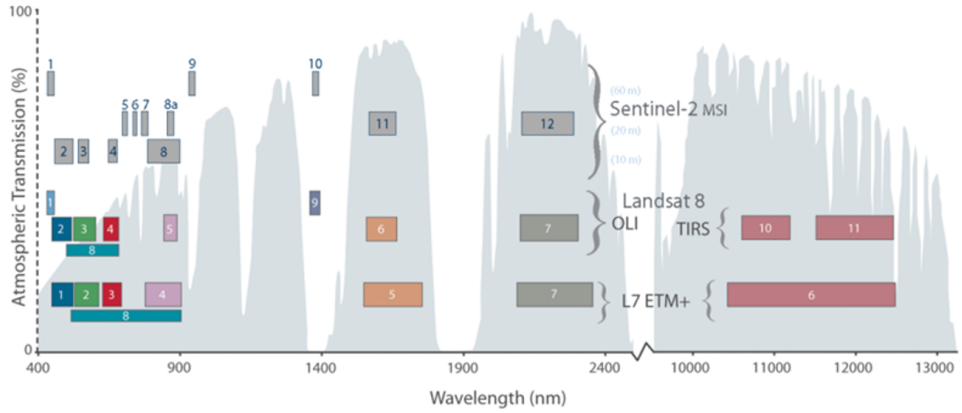

Figure 1.1: Overview of spectral bands of different space born sensors. Plot sourced from (https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-2-msi/overview).

conservation control measures. In case of disaster, satellites can provide quick information for relief coordination [5]. In recent years, the rise of cheap drones enabled cheap aerial imagery on small scales. Optical imaging systems are only one side of remote sensing. Radar enables access to a whole other suite of information while being less dependent on weather and the natural sunlight.

1.2 Pan sharpening

Under the radar of most, pan sharpened images found their way into the daily life of most people. From google earth to satellite images on the news, there is always a pansharpening step in the reprocessing. Most optical imaging systems deployed on satellites measure a range of frequency bands. As the shutter speed of space born sensors is very limited due to their high velocity, spacial and spectral resolution compete for the photons that reach the imaging sensor. Most sensors try to solve this trade-of by having multiple narrow spectral bands at a low spacial resolution and a wide spectral band (pan) at a high spatial resolution. Figure 1.1 gives an overview over the spectral bands of some space born systems. Pan sharpening is then used to transfer the details from the pan band to the multi spectral image. As the human eye is more sensitive to intensity fluctuations than to spectral fluctuations, the fused image is very pleasing to the eye. Since the deployment of color sensors, methods for pansharpening were developed. From simple component replacement methods to advanced histogram matching algorithms to deep learning approaches, the hunt for sharper images continues.

Notation and mathematical problem statement lets introduce the notation that will be used in the following chapters. Bold capital letters $\mathbf{A}$ represent matrices or higher order tensors. Lowercase bold letters $\mathbf{a}$ denote their one dimension flattened counterparts. The source $\mathbf{S}$ and the guide $\mathbf{G}$ are the input to the models. Each model predicts $\mathbf{Y}$, at the same resolution as the guide. $\mathbf{Y}$ is compared to the ground truth $\mathbf{Y_{gt}}$ to quantify the performance of a model. The source is the low resolution colour image with multiple channels, the guide is a single channel black and white image. The scaling factor k denotes the ratio between the target resolution H, W and the source resolution h, w. The goal of the pansharpening task is to find some function Φ that optimizes a given error function $\arg\operatorname{opt}_{\Phi} \|\Phi(\mathbf{S}, \mathbf{G}), \mathbf{Y_{gt}}\|$.

1.3 Machine Learning

With the rise of AI chatbots, machine learning found its way to the daily life of most people. It took the public focus away from other machine learning fields. Long before transformers, a large range of ml architectures were developed. Convolutional neural networks (CNNs) [6], offer an end to end trainable solution to computer vision tasks. The introduction of residual connection [7] between layers enabled deeper networks, without degeneration like its predecessors. The spotlight of computer vision did not last long on ResNet, as GANs [8] and, in the past few years, diffusion models [9] showed incredible performance in image generation.

As the architecture used in our models has a ResNet as its backbone, we want to elaborate more on the working principle of CNNs and the architecture of ResNet itself.

The concept of **convolution** was first introduced by Euler in the 1760s. While not getting much traction in the beginning,from the 1890 on it showed to be a useful tool in a wide range of areas. The discretized formula that can be applied on images, is given under eq: 1.1.

$$G[m,n] = \sum_i \sum_j F[m+i, n+j] \cdot K[i,j] \tag{1.1}$$

A discrete function F is convoluted by a filter K. One addition to the given formula is stride, where F is a function of a multiple of m, n. Thereby, the size of the output is reduced. Convolution comes in handy in all kind of signal processing tasks. The convolution theorem states that convolution is equivalent to multiplication in the Fourier space. This has important implications. The computational complexity of the naive convolution operation is $O(n^2)$. By applying the Fast Fourier transformation first the complexity can

be reduced to $O(n\ log(n))$. This makes it useful even for large datasets, such as images. By choosing the filter wisely, complex features can be extracted out of an image. The filter in fig: 1.2 is used for edge detection. By taking a part of an image as the filter and convoluting it by the image, the position of the patch in the image can be found. In a similar manner a filter can be used for object detection. How to choose these filters correctly is non-trivial, therefore it would be nice to learn them from data.

Incoming **ConvolutionalNeuronalNetwork** [6]. If we combine a convolutional layer with an activation function (fig:1.2), we get a layer of a convolutional neuronal network. As all the steps mentioned are differentiable, the gradient can be propagated through the network and learn the filter that extracts the feature we want from data. By using the extracted features as input of a conventional neuronal network, CNNs can be used on classification tasks. Invariant behaviour to rotations, translation and similar transformations can be achieved by applying pooling layers after the convolution. These pooling layers take the minimum, maximum or mean over a certain area of the input. The CNN architecture introduced so far has one fatal flaw. Due to vanishing and exploding gradients gradients [11] it is not easy to map the identity form one layer to the next. Therefore the performance of a model can can decrease with added layers.

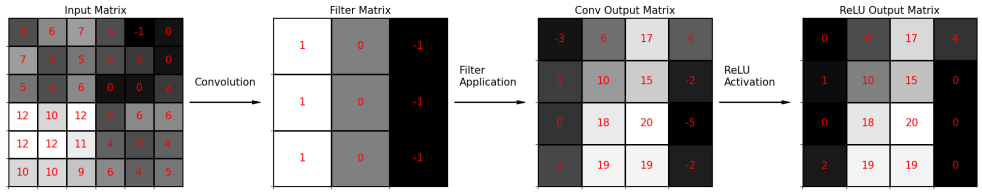

Figure 1.2: Scheme of a convolution layer with a 3x3 filter, zero padding and a stride of one. After the convolution a relu activation is applied.

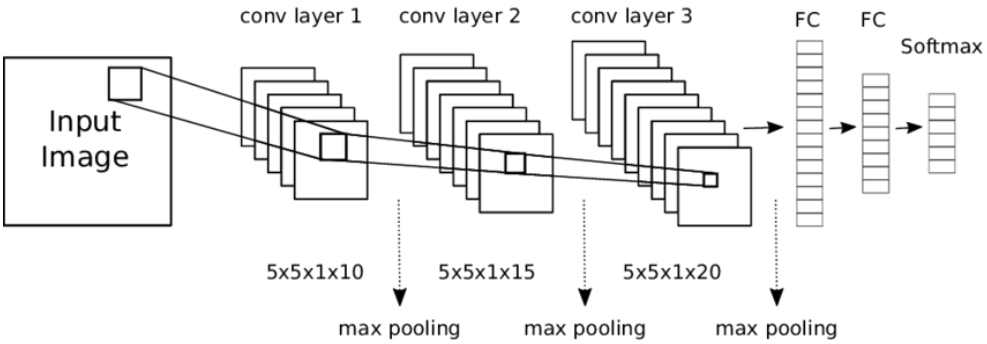

Figure 1.3: Architecture of a basic convolutional neuronal network. Graphic from [10].

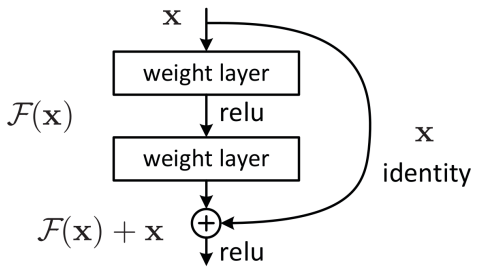

Figure 1.4: Single layer with a residual connection from a ResNet [7]. The weight layers in this case are a convolution layer form fig:1.2

This problem sets the stage for **ResNet** [7]. ResNets add a residual path to the convolutional layer (fig:1.4). The identity function has a single weight that can be trained instead of all the filter weights in a conventional CNN. This makes it easier to skip a layer if no useful progress for the loss minimization can be achieved with it. The paper shows that a network with 1202 layers has a similar training error as a network with 110 layers. The test error of the large network is higher. This is probably due to overfitting.

Background

Pansharpening is a special case of guided super resolution. In this chapter we want to introduce some established methods for pansharpening as well as the guided super resolution methods this work is based on.

2.1 Graph regularized super resolution

As this thesis is based on the paper Learning Graph Regularization for Guided Super-Resolution [12], it makes sense to introduce it here briefly. The super resolution case in the paper is depth map upscaling. A 3 channel RGB image $\mathbf{G}$ is used to scale up a 1 channel depth map $\mathbf{S}$. The output of the model $\mathbf{Y}$ is generated by solving the following quadratic system:

$$\arg\min_{\mathbf{y}} \quad \|\mathbf{Dy} - \mathbf{s}|_2^2 + \lambda \mathbf{y^T L y} \tag{2.1}$$

The matrix $\mathbf{D}$ samples the predicted depth map down to the resolution of the source, L is the Laplacian of the affinity graph. The affinity graph is computed on deep features extracted from the guide and the upsampled source. The Laplacian is constructed as $\mathbf{L} = \mathbf{D} - \mathbf{A}$. In the affinity matrix $\mathbf{A}$ each element contains the negative exponential affinity of deep features of pixel i and pixel j,

$$\mathbf{A_{ij}} = e^{-\frac{\|\mathbf{f_i} - \mathbf{f_j}\|_2^2}{\mu}} \tag{2.2}$$

where μ is a learnable parameter. To obtain the degree matrix $\mathbf{B}$ we sum over all connections of pixel i: $\mathbf{U_{ii}} = \sum_{\mathbf{j}} \mathbf{A_{ij}}$ In theory the affinity between all pixels will be non-zero. To enable the use of sparse matrices, they only include the four neighbourhood in the $\mathbf{A}$. All connections on a longer range are encoded in the deep feature extractor. The optimization problem of equation 2.1 is equivalent to solving the following linear equation:

$$(\lambda \mathbf{L}(\theta) + \mathbf{D}^T\mathbf{D})\mathbf{y}^* = \mathbf{D}^T\mathbf{s} \tag{2.3}$$

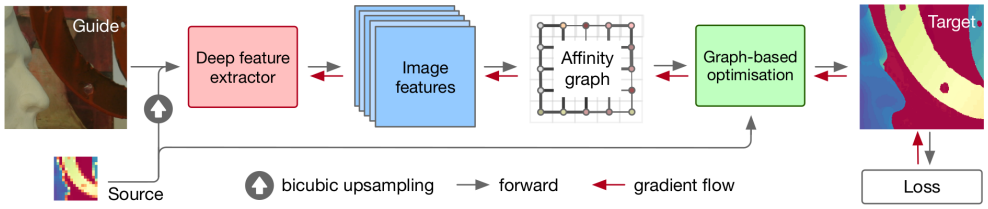

Figure 2.1: Architecture from the graph regularized super resolution model from [12]. The output depends only indirectly over the affinity graph of the deep features of the guide and the upscaled source.

During training, the weights of the feature extractor and μ are learned. Figure 2.1 shows how the gradient can be propagated through the network until the feature extractor is reached. To propagate the gradient thought the affinity graph, we can use the implicit function theorem[13]:

$$\frac{\partial l}{\partial \mathbf{L}} = -\lambda \frac{\partial l}{\partial \mathbf{D}^T \mathbf{s}} \mathbf{y}^{*T}, \quad (\lambda \mathbf{L}(\theta) + \mathbf{D}^T \mathbf{D}) \frac{\partial l}{\partial \mathbf{D}^T \mathbf{s}} = \frac{\partial l}{\partial \mathbf{y}^*} \tag{2.4}$$

By solving this linear system we get the gradients regarding the elements of the Laplacian, the rest of the back propagation can be left to the autograd functionality of pytorch.

2.2 Component Substitution Methods

The first pansharpening methods were based on component substitution. In a first step, the low resolution source is upsampled to the resolution of the guide. Next the multispectral colour space is transformed in a way so that one channel contains the intensity information. This channel is then replaced by the pan guide and the colourspace is transformed back. One popular such colourspace for RGB images is HSV[14]. In the HSV colourspace the first channels contains information on the colour, the second channel on the saturation and the third channel on the intensity. Let's look at an example of the algorithm on a flattened image. The image $\mathbf{I}$ has the dimensions n-channels by HxW.

$$\mathbf{S_{up}} = \mathbf{US} \rightarrow \mathbf{S(hsv)} = \mathbf{TS_{up}(rgb)} \rightarrow \mathbf{S(hsv)[3]} = \mathbf{G} \rightarrow \mathbf{I} = \mathbf{T^{-1}S(hsv)} \tag{2.5}$$

In the formula $\mathbf{U}$ is an upsampler, $\mathbf{T}$ is the matrix that performs the transformation into the new colourspace. Transformations like this only work on a specific number of channels. One approach to tackle this problem is to apply a transformation to the principal components[15]. The assumption is made that the first principal component is equal to the pan channel. The algorithm from equation 2.5 can be applied with the PCA transformation to get a fused image.

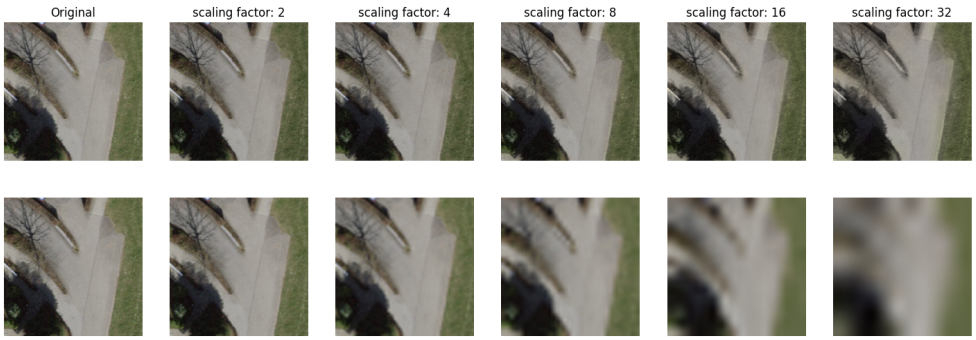

Figure 2.2: Example of a pan sharpened image using component substitution and the HSV colour space. The data used to compute is synthetic. In a first step, the image was downsampled using an inter area interpolation, then upsampled using a bicubic interpolation. To obtain the greyscale image, the colour channels were averaged. The top row are the pansharpend images, the bottom row the source images after bicubic upscaling

Depending on the task, component substitution methods work well for tasks like cartography. On visual inspection, the differences between a pansharpened image and a high resolution original can be hard to spot. Figure 2.2 shows pansharpend images at different scaling factors. Up to a scaling factor of 8 the differences are barely visible, at higher scaling factors the colour of the grass stripe fades completely. This shows some of the limitations of such methods. The sharpening only works, as long as the colour channels and the pan channel are correlated [16]. When working with real data, another problem emerges. The pan band contains frequencies that are not included in any colour band. In figure 1.1 we can get some insights into this problem. The pan band of landsat 7 has a large frequency range around 600 nm not included in neither red nor green. In addition to that, we see that the pan band of landsat 8 has a pan band with a very different frequency range. Both landsats have additional bands in the infrared. Models that are verified on data from one sensor can not be applied on a different sensor. Advantages of the approach are the computational simplicity and good performance for visual inspection.

2.3 ARSIS

ARSIS, from the french "Amélioration de la Résolution Spatiale par Injection de Structures" stands for a set of methods that try to inject high frequency features from the pan image into the colour channels[17]. The established methods rely on wavelet decomposition. Similar to the Fourier transformation, wavelet decomposition can be interpreted as a decomposition into orthogonal basis vectors. From this new representation, the original image can be reconstructed. In difference to the Fourier components, the wavelets have a stronger scale dependence. Each step of the wavelet decomposition

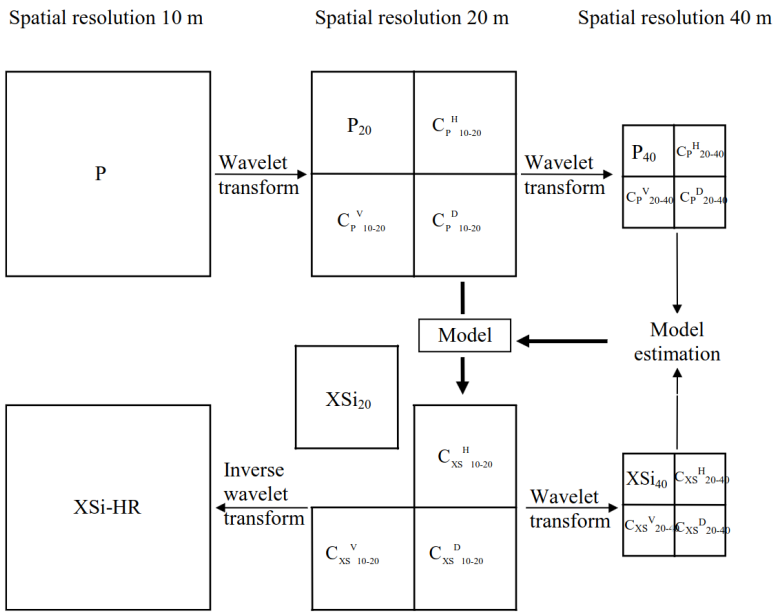

Figure 2.3: Architecture of the wavelet decomposition concept for feature injection. Wavelet decomposition is applied on the pan and the multispectral image. The largest scale wavelet is injected into the reconstruction of each colour channel. A, on a smaller scale wavelet dependant, transformation can be applied on the wavelet before injection. The figure is from [17].

can be understood as the application of a low pass filter. Figure 2.3 shows a possible architecture. The choice of the model is a topic of ongoing research, but even a simple identity results in better results than IHS and PCA [17].

2.4 Deep learning pansharpening approaches

The use of deep learning for the pansharpening task is not trivial. Even dough the amount of available pan and low resolution multispectral data is enormous, ground truth from the same sensor is newer available. Combining data from different sensors would be one way to overcome this issue, but the task of overlaying the images is non-trivial, as subpixel alignment is needed. Another issue is that the colour bands are not necessarily equal. Trials to train networks on reduced resolution show poor performance on the effective resolution of a sensor.

The current leader on the pansharpening benchmark on papers with code is Lambda-PNN [18]. It takes the lead on all WorldView and GeoEye datasets. It uses an unsupervised approach, eliminating the problem of lacking training data. By introducing a new loss function called JESSE(Joint Enhancement of

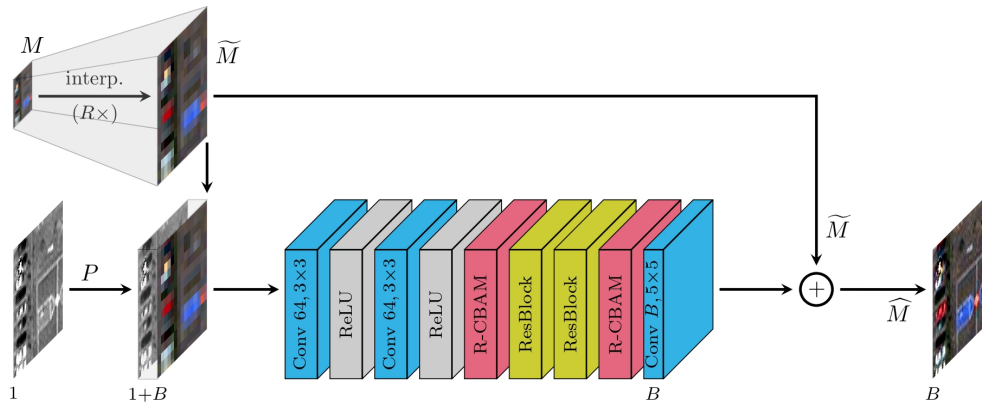

Figure 2.4: Architecture of the Lambda-PNN model. The figure is from the Lambda-PNN paper [18]

Spectral and Spatial fidElity),

$$\begin{aligned} \mathcal{L}_{JESSE} &= \mathcal{L}_\lambda + \beta \mathcal{L}_S \\ &= D_\lambda^{(K)}(\hat{M}_{\downarrow a}, M) + \gamma \text{ERGAS}(\hat{M}_{\downarrow a}, M) + \\ &\quad + \beta \left\langle (1 - \rho^\sigma)\, u\, (\rho^{\max} - \rho^\sigma) \right\rangle \end{aligned} \tag{2.6}$$

they manage multiple things at once. The $D_\lambda^{(K)}(\hat{M}_{\downarrow a}, M)$ term forces spectral similarity between the downsampled target to the source, ERGAS term forces radiometric similarity of the downsampled target to the source. Only the last term introduces the guide through the correlation $\rho^\sigma = Corr(P, \hat{M})$. The proposed architecture (fig:2.4) not only performs the pansharpening, but also band alignment during inference.

Another well performing model is "Pansharpening via Detail Injection Based Convolutional Neural Networks" [19]. It introduces multiple ways to inject details into the low resolution colour channels. Architecture a). from figure 2.5 uses a CNN to extract details from the lr-colour images and the hr pan image, while b) manages to extract features for each colour channel out of only the pan image. The decoupling from the detail generation and the colour channels makes it easier to use pretrained models on new sensors, as only the last fusion step has to be retrained.

A large amount of other models were developed in the past few years, to mention all of them here seems a lot. To get an overview, [20] can give some insights into the most recent developments.

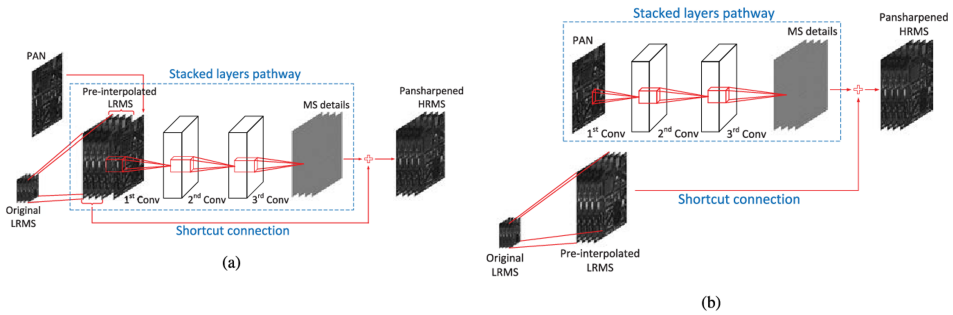

Figure 2.5: Architectures of (a) DiCNN1 and (b) DiCNN2, two feature injection architectures developed in [19]. While a) learns details from the lr-colour images and the hr pan image, b) learns the details only from the pan image

Data

3.1 Data source

Supervised machine learning approaches for pan-sharpening have the problem of lacking ground truth. To resolve this issue, either images from different sensors can be stitched together or synthetic data can be used. In this project the latter is done. All the image data is sourced from the Swissimage 10cm collection [21]. The images in the collection have a ground resolution of 10 *cm* or 25 *cm*, contain a red, blue, green colour channel and have a resolution of radiometric resolution of 8 bit (downsampled from 16 bit). Images with a ground resolution of 25 *cm* are upsampled to give a homogeneous product. The data was acquired after 2020. The images underwent further preprocessing. In a first step single image swats are ortorectified using a projection on a digital elevation model (DEM). In a next step, the image swats are stitched together using overlaps between the single acquisitions and the colours are corrected to give a homogeneous mosaic. In a last step severe errors are corrected manually. This can be in cases where the projection on the DEM fails [22].

3.2 Prepocessing

The whole Swissimage 10cm collection consists of around 42700 tiles. As the compute resources to handle such a large dataset were not available a subset was used. In total 200 tiles were selected randomly (fig: 3.1) with a train-val-test split of 50-25-25.For the random selection, algorithms from sklearn[1] were used. The tiles in each split can be found in the appendix (A). To overcome limitations while loading the images, the tiles were further divided into 1000 by 1000 pixel sub-tiles and saved into a .npy array for each split. This allows for fast access during runtime. All further preprocessing is done in the data loader. Here, from each 1000 by 1000 pixel tile, a 256 by 256

[1] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection

pixel tile is selected. To increase the amount of training data, random flips and rotations of up to 15° are applied.

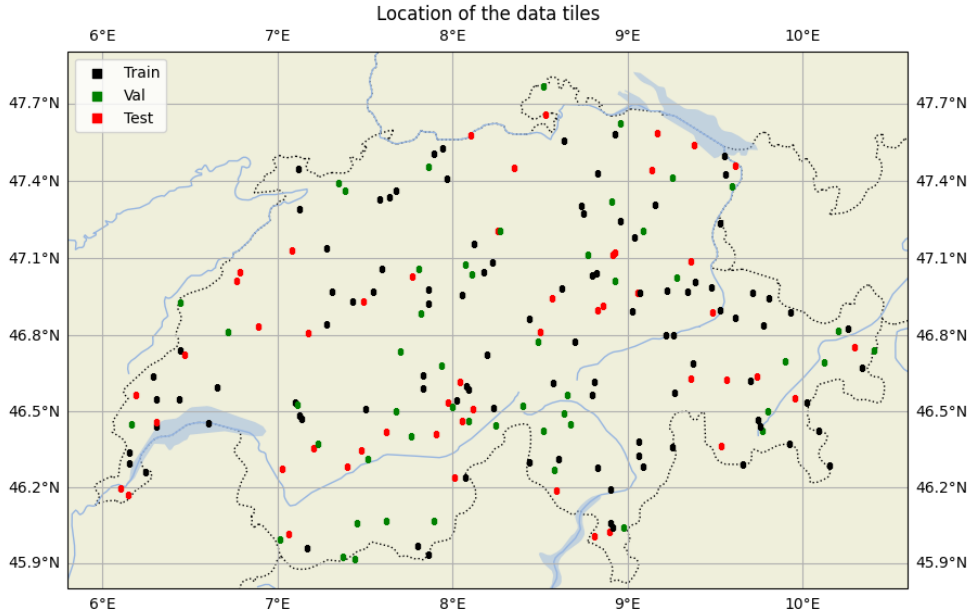

Figure 3.1: Locations of the image tiles. Each tile has a size of 1 km^2. The colours indicate to which split each tile belongs.

3.3 Issues with synthetic data

Using synthetic data has drawbacks. Some of the issues in a real camera are not easy to accurately simulate and can be device specific. In operational systems, the pan image is collected from a different sensor than the pan image. This leads to problems with the alignment of the colour channels and the pan channel [23]. These are completely disregarded in this work. Problems relating to the physical properties of the imaging system [24] that lead to bleeding from one pixel to its neighbours are neglected as well. The images in the used dataset are all orthorectified, which eliminates the problem of different viewing geometries. While this problem was constant in the traditional remote sensing satellites, current on order commercial satellites can encounter geometries rarely seen at train time.

Methods

In the following experiments, three setups are compared. First we look at the performance of a ResNet to pansharpen images, then we introduce a graph regularizer and use a ResNet to predict the weights of the graph. The resulting image is the solution to the optimization problem that the down sampled solution should match the initial low resolution image and show the structure from the learned graph. In a last step, we change the optimization problem to enforce that the average of the colour channels matches the black and white guide. The proposed methods build on the Learning Graph Regularisation for Guided Super-Resolution [12] paper.

4.1 Raw ResNet50

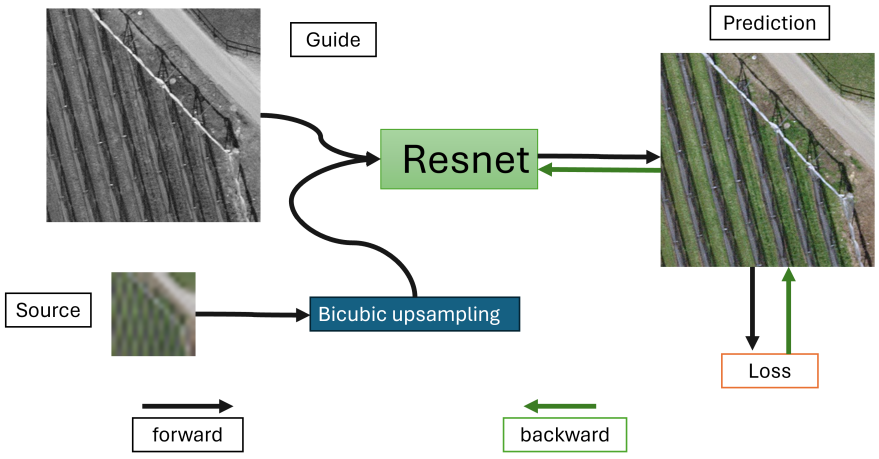

Figure 4.1: Architecture of the naive approach. With bicubic upsampling, the source gets to the same resolution as the guide. The ResNet then learns how to perform the pansharpenig during training.

As a baseline for the further experiments we look at the pansharpen capability of ResNet50. We perform bicubic upsampling on the source, concat it together with the guide and fed it into the ResNet. The network does its magic and returns a colour image with the same size as the guide.

$$\mathcal{L} = \|Y - Y_{gt}\|_1 \tag{4.1}$$

From the loss in equation 4.1 the gradient can be computed and propagated back through the network. While the most used approach for this task is to use the ResNet only to compute the residuals of the upscale image and the target, we implemented the naive version here to evaluate its performance.

4.2 Optimized

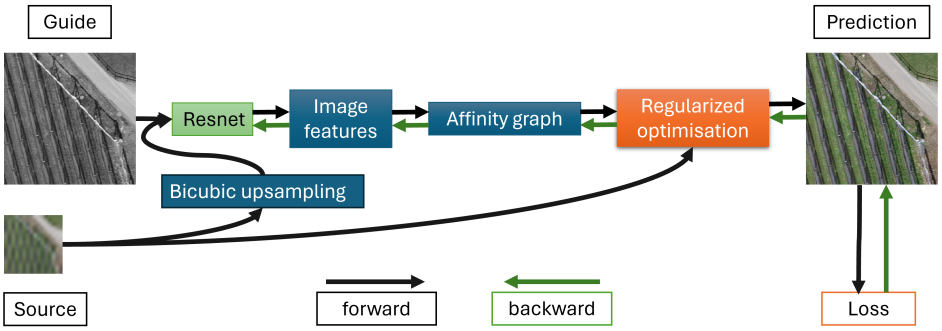

Figure 4.2: Architecture of the optimisation based approach. All steps are differentiable and therefore the network is end to end trainable

The main point of this thesis is to evaluate the usefulness of graph regularization on the pansharpening task. As a second architecture, we therefore adapt the optimization based approach introduced in [12]. The extension to three channels can be integrated into the existing workflow by concating the colour channels together. The length of $\mathbf{s}, \mathbf{g}, \mathbf{y}$ triples. The matrices $\mathbf{D}, \mathbf{L}$ become block diagonal with the former matrices as elements on the diagonal.

$$\mathbf{y} = \begin{pmatrix} y_r \\ y_g \\ y_b \end{pmatrix} \quad D = \begin{pmatrix} D' & 0 & 0 \\ 0 & D' & 0 \\ 0 & 0 & D' \end{pmatrix} \quad L = \begin{pmatrix} L' & 0 & 0 \\ 0 & L' & 0 \\ 0 & 0 & L' \end{pmatrix} \tag{4.2}$$

The computation of the Laplacian $\mathbf{L}'$ does not change, as it represents the affinity from deep features. While the adaption is simple to implement, it comes at a computational cost. This cost comes mostly into play as increased ram requirements. As the linear solver is GPU accelerated, addition of more Ram is not easily possible. Therefore, the maximal crop size that can be fed to the model is limited. Contrary to intuition, smaller scaling coefficients

require a smaller memory footprint. While the size of $\mathbf{D}$, $H \cdot W$ by $h \cdot w$, is larger for small scaling factors, the sparsity of $\mathbf{D^T D}$ is lower.

4.3 Extra Condition

The colour of a pixel has little to do with the distance to the imaging sensor. Contrary to the paper from which we borrow the method we use, we want to predict colour and not depth. In case of colour images, the intensity of the colour pixels is highly correlated to the pan image. Therefore, we can introduce an extra condition that the average of the colour channels should be equal to the pan channel. This can be done by extending equation 2.1 by an extra term.

$$\arg\min_{\mathbf{y}} \quad \|\mathbf{Dy} - \mathbf{s}\|_2^2 + \lambda \mathbf{y^T L y} + \|\mathbf{Cy} - \mathbf{g}\|_2^2 \tag{4.3}$$

The matrix C is analogue to D. While D is a spacial downsampler, C is a spectral downsampler. As the pan channel is the average of the colour channels, C is simply concated identity matrices.

$$\mathbf{C} = \frac{1}{3}\begin{pmatrix} ID & ID & ID \end{pmatrix} \tag{4.4}$$

By adding the extra condition the computation changes in some minor ways.The optimization problem of the forward pass (eq:2.1) becomes:

$$(\lambda \mathbf{L}(\theta) + \mathbf{D}^T\mathbf{D} + \mathbf{C}^T\mathbf{C})\mathbf{y}^* = \mathbf{D}^T\mathbf{s} + \mathbf{C}^T\mathbf{g} \tag{4.5}$$

In the back propagation we have to include the extra condition by changing the partial differential in equation 2.4:

$$\begin{aligned} \frac{\partial l}{\partial \mathbf{L}} &= -\lambda \frac{\partial l}{\partial(\mathbf{D}^T\mathbf{s} + \mathbf{C^T g})}\mathbf{y}^{*T} \\ (\lambda \mathbf{L}(\theta) + \mathbf{D}^T\mathbf{D} + \mathbf{C^T C})\frac{\partial l}{\partial \mathbf{D}^T\mathbf{s} + \mathbf{C}^T\mathbf{g}} &= \frac{\partial l}{\partial \mathbf{y}^*} \end{aligned} \tag{4.6}$$

All the additions do not change the underlying architecture of the model. On the surface, the computational complexity does not increase, as $\mathbf{C}$ is very sparse. The matrix $(\lambda \mathbf{L}(\theta) + \mathbf{D}^T\mathbf{D} + \mathbf{C^T C})$ thought has now an increased number of non-zero elements, which further limits the maximal crop size and the maximal batch size possible to run on a given GPU.

4.4 Details of the experimental setup

The code used to run all the experiments can be found on the Github [25]. To run the experiments, the Euler Cluster was used. While Nvidia RTX 2080ti

GPUs were used for all experiments, the CPU changed due to the nature of the queuing system. For all experiments, the dataset was stored in memory to ensure the execution time is GPU limited. The code is implemented in python 3.11[1] using pytorch 11.8[2] as the machine learning backbone and cupy 13.1[3] to enable GPU accelerated computations for the graph layer. Multiple other packets provide extra functionalities. The model without the graph layer was trained for 500 epochs and a crop size of 256 by 256 pixels. The models with the graph layer were trained for various iterations due to problems with the compute cluster and a crop size of 64 by 64 pixels. These adjustments were made to enable computation on the aforementioned GPUs in a reasonable timeframe. All experiments were conducted with the default values for the learning rate and the learning rate decay rate with an Adam optimizer. Training runs with 500 iterations could finish in 5 days. Large batch sizes tend to speed the training up, but are not feasible in some cases due to memory constraints. For the same reason, the evaluation was only performed on sub patches for the models with the graph layer. This was implemented by setting the crop-deterministic flag to False during evaluation. As the process is only pseudo random, the result is still repeatable. Due to some skill issues, the training of the models was cancelled prematurely and due to timing issues the experiments could not be repeated. In appendix A.1 is a small discussion on the magnitude of the thereby induced error. Due to this error the data for the model with the extra condition were trained for more than 200 epochs, the model without the extra condition for up to 200 epochs.

4.5 Quantitative Image analysis

The evaluation of pan sharpening results is non-trivial. The examination by the bare eye can be off, as the human eye can distinguish intensity changes better than spectral changes. The appropriate metric depends on the use case. Additional to the MAE and MSE, the PSNR, ERGAS and SAM were measured.

The mean average error is the most generic error function. It gives an average deviation from the predicted image to the ground truth on a per subpixel level. Its ideal value is positive and in the ideal case 0. The sum is over all subpixels.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} \|y_i - \hat{y}_i\| \tag{4.7}$$

[1] https://www.python.org/downloads/release/python-3110/
[2] https://pytorch.org/get-started/locally/
[3] https://docs.cupy.dev/en/stable/install.html

The mean squared error is similar to the mae, but penalizes outlayers more. The ideal value is 0.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n} \|y_i - \hat{y}_i\|^2 \tag{4.8}$$

PSNR is an established metric for various machine learning tasks. It denotes the Peak Signal to Noise Ration of a data pair. In case of images, it comes down to:

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \tag{4.9}$$

The max intensity is a reference to the original image.

The ERGAS [26], from the French "erreur relative globale adimensionnelle de synthèse", generalizes the MSE by computing it per band and dividing it by the mean of the band and introduces a scaling dependence. This reduces problems of calibration as the intensity of the colour bands are normalized. The scaling dependence allows for comparison between a large range of image sizes.

$$\text{ERGAS} = 100\frac{1}{k}\sqrt{\frac{1}{N}\sum_{j=1}^{N}\left(\frac{\text{RMSE}(j)}{\mu_j}\right)^2} \tag{4.10}$$

Where k = scaling factor, N = number of bands, ν = average value of the respective band. The optimal value is 0.

The spectral angle mapper SAM is a measure of similarity between two spectra. By summing over all pixels, we can get the spectral similarity of two images. This index has the advantage that it isolates the spectral similarity from the intensity similarity.

$$\text{SAM} = \frac{1}{N}\sum_{i}^{N} arcos\left(\frac{y_i \cdot \hat{y}_i}{\|y_i\|\|\hat{y}_i\|}\right) \tag{4.11}$$

The optimal value of the SAM is 0, which would mean an identical spectrum.

In addition to these standard errors, we implement an MSE 4.8 error that compares the down sampled prediction.

$$\text{MSE}_c = \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{s} - \mathbf{D}\hat{\mathbf{y}}\|^2, \quad \text{MSE}_p = \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{g} - \mathbf{C}\hat{\mathbf{y}}\|^2 \tag{4.12}$$

Especially in case of the graph regularized architecture, these errors are interesting, as they evaluate how well constraints are satisfied.

Results

In this chapter we want to present the results of the models we trained sections(4.1, 4.2, 4.3), as well as some baselines. Some pictures give visual clues of the performance of different models. Figure 5.7 gives an overview of all metrics.

5.1 Learned Models

For scaling factors smaller than 8 the model with the graph and the extra condition performs the best. For larger scaling factors, the model without the graphs performs better. A fascinating finding is that the model with the extra condition performs really bad on the MSE_c metric, even dough the condition should enforce it. An explanation for this behaviour is the large value of lambda. This gives the affinity graph a large weight compared to the other conditions. The model without the extra condition menages to score perfect results on the MSE_c metric. Looking at the low values of lambda, the assumption from before seems to be confirmed. The model without the extra condition fails on the pansharpening task. Even dough the metrics do not show a performance that seems too bad, visual inspection shows that the model clusters the image. The affinity graph enforces that pixels with similar deep features look similar. This assumption seems to not hold for the pansharpening case.

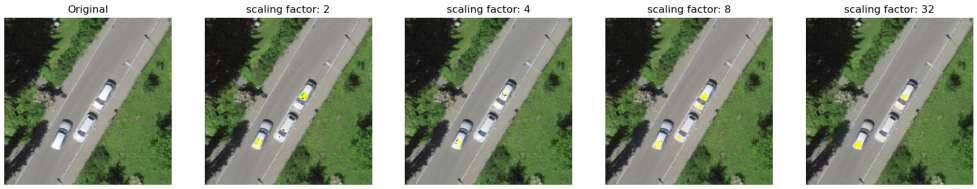

Figure 5.1: The model without the graph mostly performs well. On the roofs of the cars it fails to make a useful prediction. It seems like the error could originate from saturating a colour channel. As the results are cut off at 8 bit, the effect occurs

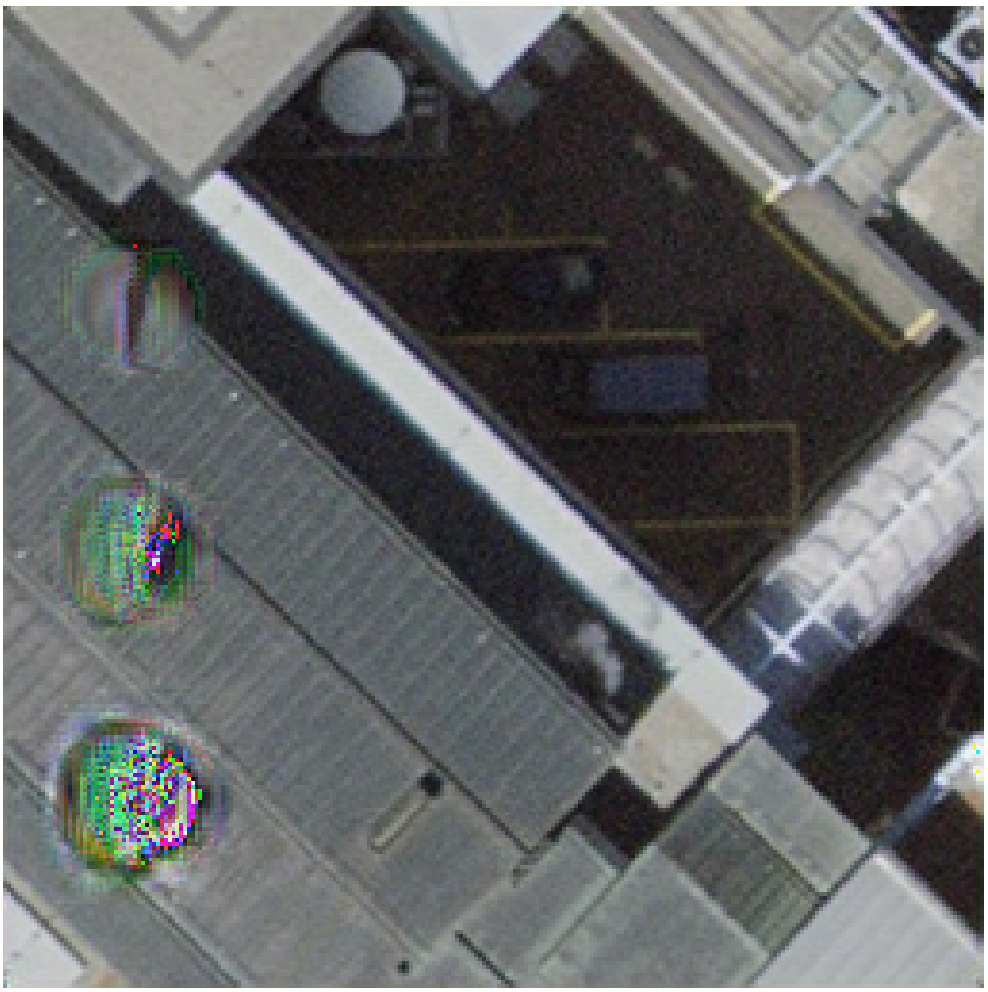

Figure 5.2: The model that only implements a ResNet shows mostly good performance. Nonetheless, there are some examples like this where the model fails completely in parts of the image.

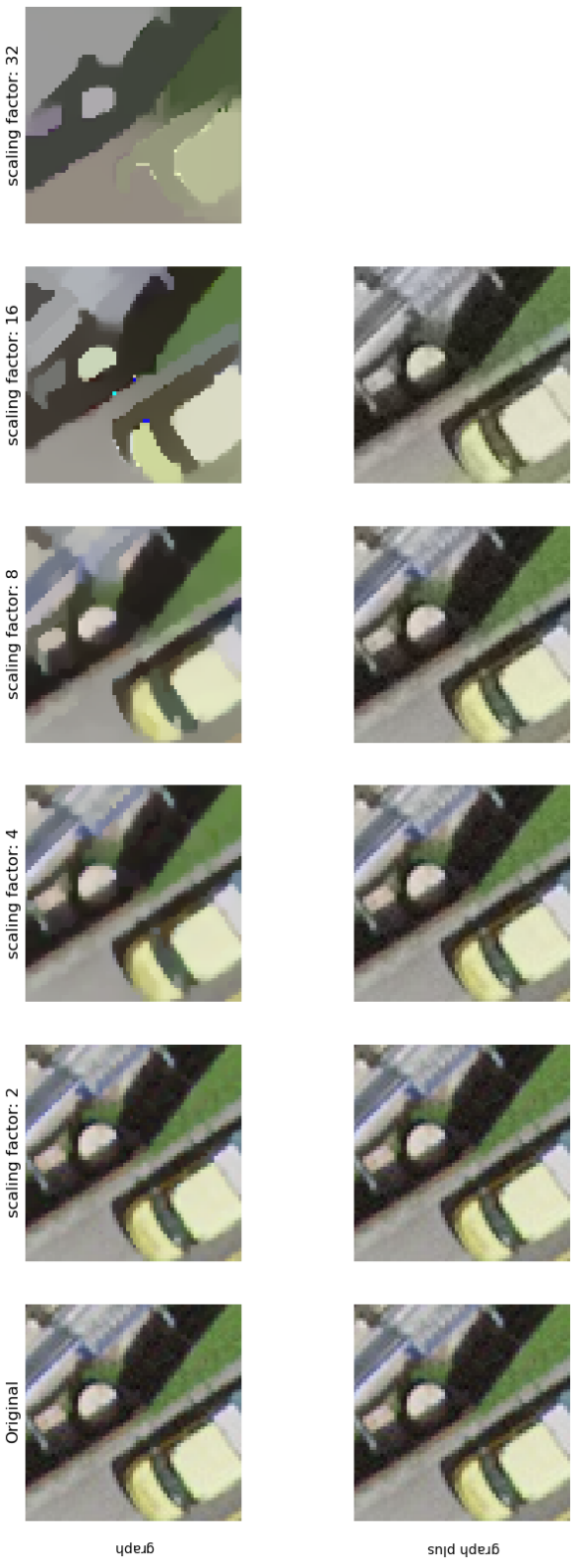

Figure 5.3: Results of the inference with different models. The model without the graph has a strong clustering effect. At a scaling factor of 16 it introduces regions with strong colours that are not in the initial image.

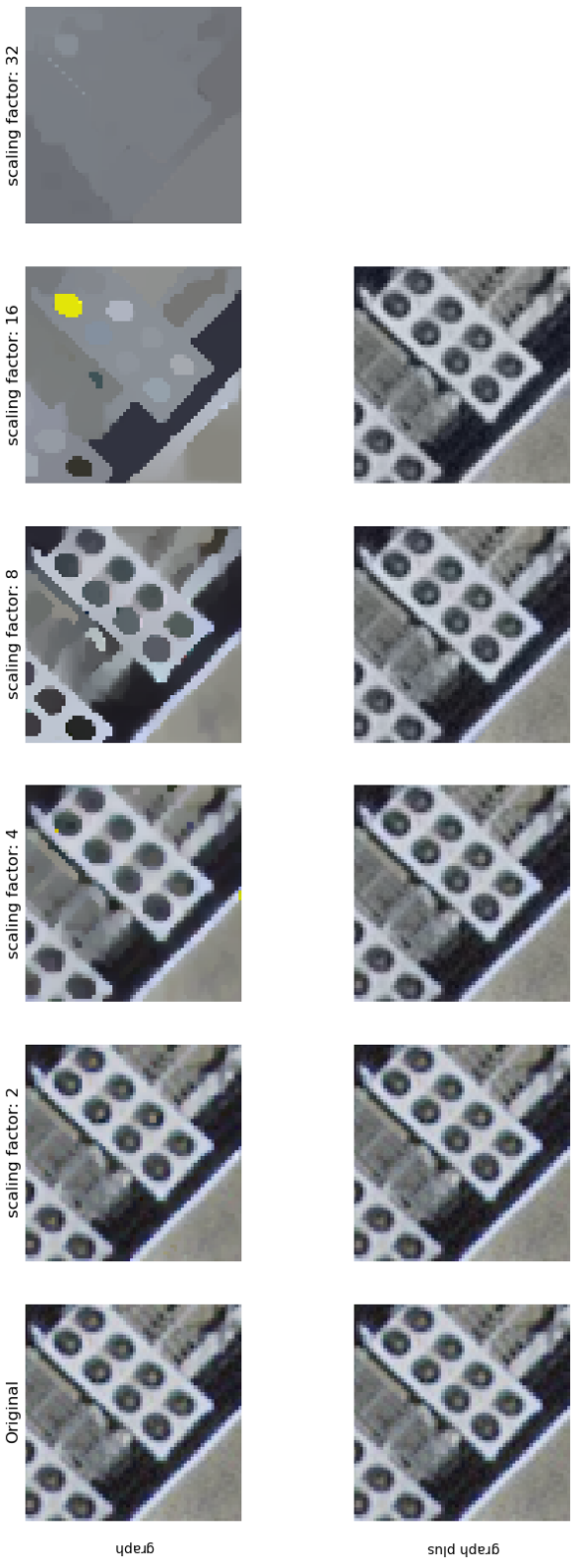

Figure 5.4: Results of the inference with different models. The model without the graph has a strong clustering effect. At large scaling factors, it introduces a yellow region. Against intuition, the downsampled image still matches the source. The model with the extra condition can not be distinguished by eye from the original.

MAE	MSE	PSNR	MSE_c	MSE_p	ERGAS	SAM	scaling	batch_size	model
0.009244	0.000250	44.064356	1.255467e-04	1.959143e-04	2.031693	0.017685	2	8	w/o-graph
0.010863	0.000879	42.565670	4.679225e-04	6.015356e-04	1.426446	0.025789	4	8	w/o-graph
0.012471	0.000781	**40.423700**	2.827437e-04	4.140108e-04	0.684820	0.032954	8	8	w/o-graph
0.014210	0.000943	**38.990962**	1.236822e-04	4.033053e-04	0.203683	**0.039012**	32	8	w/o-graph
0.007542	0.000170	43.033753	**1.767087e − 07**	1.477897e-04	1.745877	0.014046	2	4	graph
0.015447	0.000600	35.680600	**2.054780e − 07**	5.265634e-04	1.734233	0.027732	4	4	graph
0.032395	0.002259	28.126778	**1.037330e − 07**	2.089862e-03	0.865855	0.040019	16	4	graph
0.047346	0.004672	24.700745	**8.406816e − 06**	4.435297e-03	0.626980	0.045574	32	4	graph
0.002116	**0.000009**	**51.834402**	7.189170e-04	**6.835587e − 08**	**0.468988**	**0.008864**	2	4	graph-plus
0.005195	**0.000051**	**44.327204**	1.722384e-03	**1.854208e − 07**	**0.585469**	**0.021869**	4	4	graph-plus
0.007506	**0.000103**	**41.153794**	3.055342e-03	**1.032421e − 06**	**0.416642**	**0.031490**	8	4	graph-plus
0.009062	**0.000155**	39.075785	4.476286e-03	**2.304268e − 06**	**0.249346**	0.037506	16	4	graph-plus

Table 5.1: Performance of the different models on the test split of the dataset at different scaling factors.

5.2 Baselines

Three baselines are used to evaluate the performance of the models. The simplest baseline is just bicubic upsampling. As a second baseline, we use a transformation to the HSV colour space from section 2.2. As a third baseline, the optimization problem from section 4.3 is implemented without the Lagrangian. This is implemented by first performing bicubic upsampling and then linearly scaling the average of each pixel so that it matches the pan image

$$\mathbf{S} \xRightarrow{\text{bicubic upsampling}} \mathbf{S_{up}} \xRightarrow{\text{pixelvise scaling}} \mathbf{Y} \tag{5.1}$$

We call this method averaged, as both the spectral as the radiometric averages match the source and the guide.

The newly introduced benchmark outperforms the others on all metrics on all scaling factors. The reason it performers so well lies in the synthetic data. On real world data this benchmark would not work as well. In addition, this leads to differences in perceived colour that are not well captured in the presented metrics. An interesting case is the SAM metric. All proposed methods can not improve the accuracy of bicubic upscaling. While this behaviour is expected for the newly introduced method, the reason why do HSI method does not make a difference is not obvious on first sight. This new pansharpening method fulfils the both conditions from in equation 4.3. The Lagrangian in the equation should help to choose a good solution to this ill posed problem.

The results of upscaling with these benchmarks are presented in table 5.2, sample images can be seen in figure 5.6. A visual inspection of this image is difficult, as all models perform well.

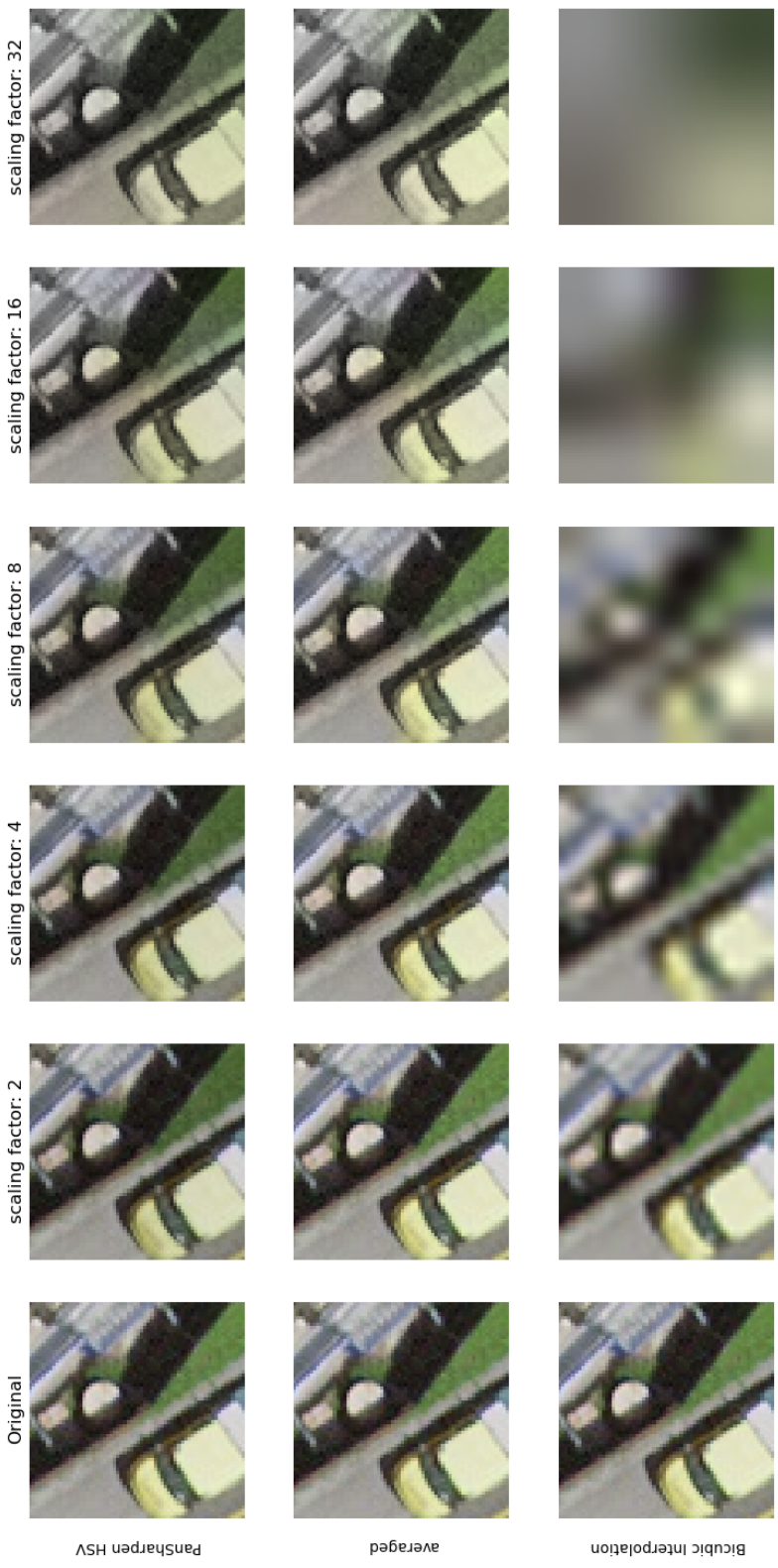

Figure 5.5: Result of the benchmarks on an image. From a scaling factor of 8 upwards, the colours are not accurate any more. The colours of the HSV method look better. The roof of the car as an example is over saturated in the averaged method.

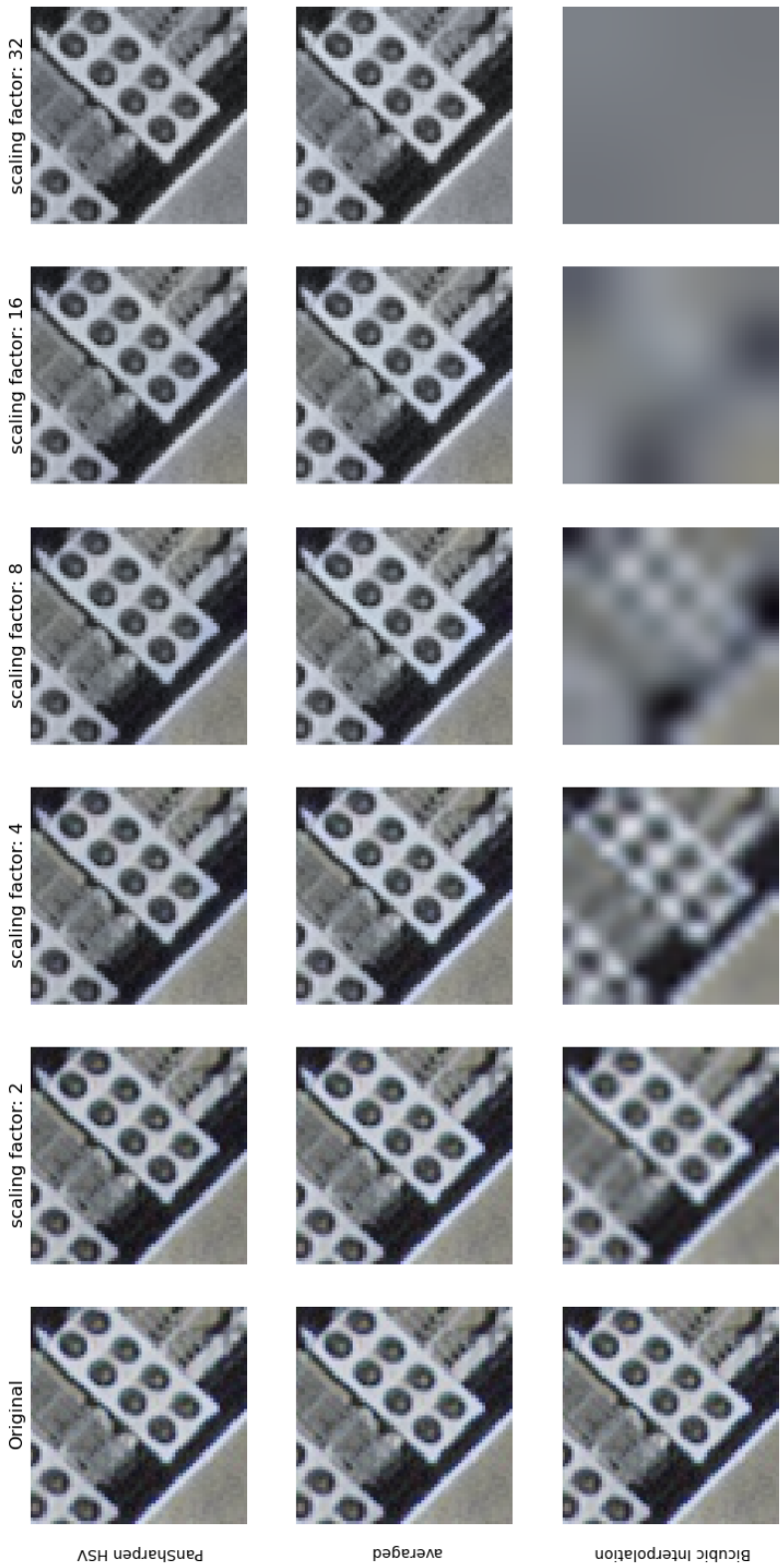

Figure 5.6: Result of the benchmarks on an image. On images with little colour like this, both conventional methods perform well.

MAE	MSE	PSNR	MSE_c	MSE_p	ERGAS	SAM	scaling	model
0.000000	0.000000	inf	0.000000	0.000000e+00	0.000000	0.000099	1	bicubic
0.018117	0.000813	40.313703	0.000052	8.075353e-04	3.810171	0.010859	2	bicubic
0.030536	0.001925	33.192776	0.000089	1.859785e-03	3.198166	0.025344	4	bicubic
0.042207	0.003413	29.470503	0.000107	3.275435e-03	2.185019	0.034245	8	bicubic
0.052460	0.005131	27.275885	0.000123	4.923734e-03	1.343403	0.039393	16	bicubic
0.063485	0.007298	25.560318	0.000149	7.006936e-03	0.800678	0.044235	32	bicubic
0.046069	0.002740	30.688425	0.002740	2.634568e-03	14.817418	0.000101	1	HSV
0.045982	0.002768	30.656207	0.002723	2.643779e-03	7.471552	0.010859	2	HSV
0.045785	0.002807	30.616595	0.002667	2.620215e-03	3.774836	0.025344	4	HSV
0.045725	0.003057	30.557418	0.002605	2.726710e-03	1.904969	0.034245	8	HSV
0.045651	0.002954	30.534804	0.002530	2.584024e-03	0.956791	0.039393	16	HSV
0.045580	0.003167	30.524782	0.002447	2.689445e-03	0.479106	0.044235	32	HSV
NaN	NaN	NaN	NaN	NaN	NaN	0.000106	1	averaged
0.003181	0.000354	54.862228	0.000086	1.591612e-15	0.757387	0.010859	2	averaged
0.007150	0.033435	46.641419	0.002089	1.616181e-15	0.857276	0.025344	4	averaged
0.009707	0.000698	43.569338	0.000013	1.596788e-15	0.521977	0.034244	8	averaged
0.011318	0.000501	41.964550	0.000005	1.597291e-15	0.302996	0.039392	16	averaged
0.012848	0.000500	40.818631	0.000005	1.598467e-15	0.169011	0.044234	32	averaged

Table 5.2: Performance of the benchmarks on the test split of the dataset at different scaling factors. The method average denotes the newly introduced pansharpening method. There is an interesting anomaly in newly introduced method at a scaling factor of 4. The MSE and MSE_c are unexpected high, while all other errors are as expected. On all other scaling factors, the new upscaling method performs best on all metrics

5.3 Comparison

The result of the benchmarks are very good. On images with a lot of colour, our model with the graph and the extra condition can separate regions of different colour better, which leads to a more natural image. On images with little colour the images are nearly indistinguishable regardless of the sharpening method excluding the graph without the extra condition. The effect becomes larger for large scaling factors.

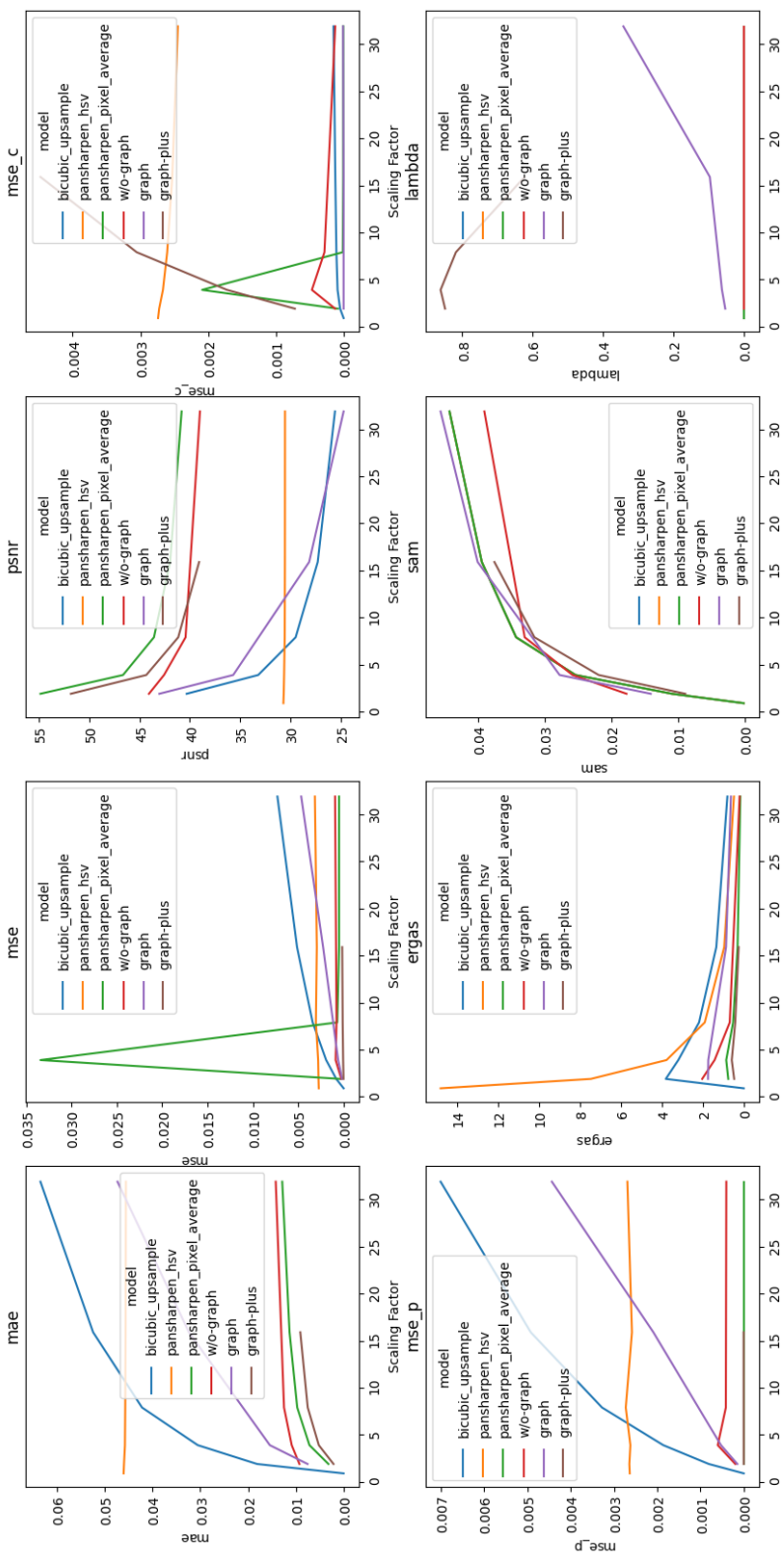

Figure 5.7: Performance of the models and benchmarks on different metrics. The last plot shows the lambda parameter. The lambda parameter is the weight of the graph regularizer. The averaged method shows an inconsistency at a scaling factor of 4. Probably the cause of the issue is a numerical problem in the computation.

Discussion

6.1 Computational cost and possible applications

The computational cost of the proposed methods is high. While an inference time of around 25ms per patch seems fast, the amount of data in the remote sensing case is very large. Using a GPU with more RAM could enable some use cases in a commercial setting where the observed area is relatively small, but the extra precision is needed for the application.

6.2 Future Work

Add Parameter In the results we saw that the performance of the graph with the extra condition is good in all cases except on MSE_C. Currently, the architecture has equal weights on both condition. It is not clear why that should lead to an optimal solution. Adding a learnable parameter could improve the overall errors, especially MSE_C, because it could give the condition that lies behind it more weight.

Equal results The number of epochs was not consistent over all experiments. While for the graph with the extra condition a model with more than 200 epochs is available, the performance of the graph alone could improve significantly with longer training.

Train on cities The dataset was sampled randomly over Switzerland. Most use cases that would benefit from a pansharpening method like this have a limited variety of locations. To sample only from settled areas or agricultural fields could improve the model for such models.

Adapt to real data At the current state, the project is not adaptable to real data. The extra condition of the graph is only works so well because we generated the pan images with exactly that condition. To apply the methods

on real data, the condition has to be relaxed.

$$\mathbf{C} = \begin{pmatrix} \lambda_1 ID & \lambda_2 ID & \lambda_3 ID \end{pmatrix} \tag{6.1}$$

Adapt to use case For training a l1 loss was used. While this provides an established method to train CNNs, the l1 does not know the use case on which we want to shine on. For example on the SAM metric we see little improvement. The change to a loss that factors in SAM directly could yield improvements for use cases in which the spectral quality is more important than the radiometric resolution.

Use for clustering While the graph without the extra condition performed badly on the pansharpening task, it clustered the input images. Maybe there is an usecase for the model as a preprocessing step of a clustering algorithm.

Improve efficiency In the current implementation, the code runs too slow for large scale use. Two things have to be adapted to change this. For one, the crop size has to be increased. This can be done by choosing a better implementation of sparse matrices or using GPUs with more memory. These do not have to be new, as analysis of the GPU utilization shows that the limiting factor is not compute power. In the current implementation with the extra condition, the training is limited by the convergence of the linear solver in the backwards pass. Better convergence should be achievable by tuning the parameters of the solver.

LORA One way to improve both the computational cost as well as the adaptability is LOw Rank Adoption[27]. It decomposes the weight matrix into two low rank matrices. Only one of them has to be fine-tuned for a new sensor, resulting in fewer parameters to be learned and a lower memory footprint during training.

6.3 Conclusion and outlook

This thesis adapts the graph regularized super resolution method to color images. We can show that the method outperforms the spectral benchmarks. We showed that the graph alone with the condition that the down sampled image has to match the source does not perform well in the pansharpening task.

Machine learning While performing well on visual inspection and the spectral benchmarks, the proposed method does not scale. This leads to the issue that niche applications that would benefit from the increased sharpness probably feed the image into another machine learning algorithm. The

proposed method has no information on how the ideal output for the next processing step looks like. An end to end trainable method will probably outperform one with a separate pansharpening step. As the proposed model is end to end trainable, it can be integrated into the training of the overall system.

Acknowledgement

I sincerely thank Alexander Becker for the supervision of the project. In addition to that, I would like to thank all the people in the IGP student lab that helped with random bug fixes and explanations on how to use the Euler cluster.

Bibliography

[1] *Des pigeons photographes ?* Oct. 5, 2024. [Online]. Available: https://web.archive.org/web/20110706223037/http://www.cameramuseum.ch/upload/press/pigeons%20catalogue.pdf.

[2] H. TUBIS, "Aerial photography maps our farmlands," *Photogrammetric Engineering and Remote Sensing*, pp. 21–23, Jun. 1937. [Online]. Available: https://www.asprs.org/wp-content/uploads/pers/1937journal/jun/1937_jun_21-23.pdf.

[3] *Planet.com*, https://www.planet.com/products/hi-res-monitoring/, Accessed: 10.05.2024, 2024.

[4] *Maxar.com*, https://www.maxar.com/maxar-intelligence/constellation, Accessed: 10.05.2024, 2024.

[5] *Emergency.copernicus.eu*, https://emergency.copernicus.eu/mapping/#zoom=2&lat=13.56036&lon=33.82273&layers=00B0T, Accessed: 10.05.2024, 2024.

[6] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. DOI: 10.1109/5.726791.

[7] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. arXiv: 1512.03385 [cs.CV].

[8] I. J. Goodfellow *et al.*, *Generative adversarial networks*, 2014. arXiv: 1406.2661 [stat.ML].

[9] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, *Deep unsupervised learning using nonequilibrium thermodynamics*, 2015. arXiv: 1503.03585 [cs.LG].

[10] T. Hinz, P. Barros, and S. Wermter, "The effects of regularization on learning facial expressions with convolutional neural networks," Sep. 2016, pp. 80–87, ISBN: 978-3-319-44780-3. DOI: 10.1007/978-3-319-44781-0_10.

[11] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Y. W. Teh and M. Titterington, Eds., ser. Proceedings of Machine Learning Research, vol. 9, Chia Laguna Resort, Sardinia, Italy: PMLR, May 2010, pp. 249–256. [Online]. Available: https://proceedings.mlr.press/v9/glorot10a.html.

[12] R. de Lutio, A. Becker, S. D'Aronco, S. Russo, J. D. Wegner, and K. Schindler, "Learning graph regularisation for guided super-resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[13] J. T. Barron and B. Poole, *The fast bilateral solver*, 2016. arXiv: 1511.03296 [id=cs.CV].

[14] A. Smith, "Color gamut transform pairs," vol. 12, Aug. 1978, pp. 12–19. DOI: 10.1145/800248.807361.

[15] P. Chavez, S. C. Sides, J. A. Anderson, *et al.*, "Comparison of three different methods to merge multiresolution and multispectral data-landsat tm and spot panchromatic," *Photogrammetric Engineering and remote sensing*, vol. 57, no. 3, pp. 295–303, 1991.

[16] C. Thomas, T. Ranchin, L. Wald, and J. Chanussot, "Synthesis of multispectral images to high spatial resolution: A critical review of fusion methods based on remote sensing physics," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 46, no. 5, pp. 1301–1312, 2008. DOI: 10.1109/TGRS.2007.912448.

[17] T. Ranchin and L. Wald, "Fusion of high spatial and spectral resolution images: the ARSIS concept and its implementation," *Photogrammetric engineering and remote sensing*, vol. 66, no. 1, pp. 49–61, 2000. [Online]. Available: https://hal.science/hal-00356168.

[18] M. Ciotola, G. Poggi, and G. Scarpa, "Unsupervised deep learning-based pansharpening with jointly enhanced spectral and spatial fidelity," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1–17, 2023. DOI: 10.1109/TGRS.2023.3299356.

[19] L. He *et al.*, "Pansharpening via detail injection based convolutional neural networks," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, pp. 1188–1204, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:64494656.

[20] *Recent pansharpening methods overview*, https://paperswithcode.com/task/pansharpening, Accessed: 22.05.2024, 2024.

[21] "Swissimage 10 cm," in Federal Office of Topography swisstopo, 2024. [Online]. Available: https://www.swisstopo.admin.ch/de/orthobilder-swissimage-10-cm#Weiterf%C3%BChrende-Informationen.

[22] *Detaillierte produktinformation swissimage*, Federal Office of Topography swisstopo, 2024. [Online]. Available: https://backend.swisstopo.admin.ch/fileservice/sdweb-docs-prod-swisstopoch-files/files/2023/11/14/a84642dc-5feb-48e5-af6b-55df4ae7a10b.pdf.

[23] H.-H. Kim and M. Kim, "Deep spectral blending network for color bleeding reduction in pan-sharpening images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 62, pp. 1–14, 2024. DOI: 10.1109/TGRS.2024.3378158.

[24] K. Trantham and T. J. Reece, "Demonstration of the Airy disk using photography and simple light sources," *American Journal of Physics*, vol. 83, no. 11, pp. 928–934, Nov. 2015, ISSN: 0002-9505. DOI: 10.1119/1.4927525. eprint: https://pubs.aip.org/aapt/ajp/article-pdf/83/11/928/13111095/928_1_online.pdf. [Online]. Available: https://doi.org/10.1119/1.4927525.

[25] *Code repository on github*, https://github.com/Mdmdma/graph-super-resolution-pan, Accessed: 28.05.2024, 2024.

[26] L. Wald, *Data Fusion. Definitions and Architectures - Fusion of Images of Different Spatial Resolutions*. Jan. 2002, p. 161.

[27] E. J. Hu *et al.*, *Lora: Low-rank adaptation of large language models*, 2021. arXiv: 2106.09685 [cs.CL].

List of Figures

1.1 Overview of spectral bands of different space born sensors. Plot sourced from (https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-2-msi/overview). 2
1.2 Scheme of a convolution layer with a 3x3 filter, zero padding and a stride of one. After the convolution a relu activation is applied. 4
1.3 Architecture of a basic convolutional neuronal network. Graphic from [10]. 4
1.4 Single layer with a residual connection from a ResNet [7]. The weight layers in this case are a convolution layer form fig:1.2 . . . 5

2.1 Architecture from the graph regularized super resolution model from [12]. The output depends only indirectly over the affinity graph of the deep features of the guide and the upscaled source. 7
2.2 Example of a pan sharpened image using component substitution and the HSV colour space. The data used to compute is synthetic. In a first step, the image was downsampled using an inter area interpolation, then upsampled using a bicubic interpolation. To obtain the greyscale image, the colour channels were averaged. The top row are the pansharpend images, the bottom row the source images after bicubic upscaling 8
2.3 Architecture of the wavelet decomposition concept for feature injection. Wavelet decomposition is applied on the pan and the multispectral image. The largest scale wavelet is injected into the reconstruction of each colour channel. A, on a smaller scale wavelet dependant, transformation can be applied on the wavelet before injection. The figure is from [17]. 9
2.4 Architecture of the Lambda-PNN model. The figure is from the Lambda-PNN paper [18] 10

2.5 Architectures of (a) DiCNN1 and (b) DiCNN2, two feature injection architectures developed in [19]. While a) learns details from the lr-colour images and the hr pan image, b) learns the details only from the pan image 11

3.1 Locations of the image tiles. Each tile has a size of 1 km^2. The colours indicate to which split each tile belongs. 13

4.1 Architecture of the naive approach. With bicubic upsampling, the source gets to the same resolution as the guide. The ResNet then learns how to perform the pansharpenig during training. 14
4.2 Architecture of the optimisation based approach. All steps are differentiable and therefore the network is end to end trainable . 15

5.1 The model without the graph mostly performs well. On the roofs of the cars it fails to make a useful prediction. It seems like the error could originate from saturating a colour channel. As the results are cut off at 8 bit, the effect occurs 19
5.2 The model that only implements a ResNet shows mostly good performance. Nonetheless, there are some examples like this where the model fails completely in parts of the image. 20
5.3 Results of the inference with different models. The model without the graph has a strong clustering effect. At a scaling factor of 16 it introduces regions with strong colours that are not in the initial image. 21
5.4 Results of the inference with different models. The model without the graph has a strong clustering effect. At large scaling factors, it introduces a yellow region. Against intuition, the downsampled image still matches the source. The model with the extra condition can not be distinguished by eye from the original. 22
5.5 Result of the benchmarks on an image. From a scaling factor of 8 upwards, the colours are not accurate any more. The colours of the HSV method look better. The roof of the car as an example is over saturated in the averaged method. 25
5.6 Result of the benchmarks on an image. On images with little colour like this, both conventional methods perform well. 26
5.7 Performance of the models and benchmarks on different metrics. The last plot shows the lambda parameter. The lambda parameter is the weight of the graph regularizer. The averaged method shows an inconsistency at a scaling factor of 4. Probably the cause of the issue is a numerical problem in the computation. 29

A.1 Curve of the l1 optimization loss used to train the models. All models but the one of run 147 seem to have converged to logarithmic improvement. 41

A.2 Curve of the PSNR during training on a logarithmic scale. All models seem to have at least linear behaviour. some models that reached 200 training epochs seem to still have linear improvements, so further training could lead to better results. 42

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every written paper or thesis authored during the course of studies. In consultation with the supervisor, one of the following three options must be selected:

○ I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used no generative artificial intelligence technologies[1].

○ I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used and cited generative artificial intelligence technologies[2].

◉ I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used generative artificial intelligence technologies[3]. In consultation with the supervisor, I did not cite them.

Title of paper or thesis:

Prospects of graph regularized super resolution in pan-sharpening

Authored by:
If the work was compiled in a group, the names of all authors are required.

Last name(s):	**First name(s):**
Erler	Mathis

With my signature I confirm the following:
- I have adhered to the rules set out in the Citation Guide.
- I have documented all methods, data and processes truthfully and fully.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for originality.

Place, date	**Signature(s)**
Zürich 18.06.2024	Mathis Erler

If the work was compiled in a group, the names of all authors are required. Through their signatures they vouch jointly for the entire content of the written work.

[1] E.g. ChatGPT, DALL E 2, Google Bard
[2] E.g. ChatGPT, DALL E 2, Google Bard
[3] E.g. ChatGPT, DALL E 2, Google Bard

Appendix

A.1 Evaluation of terminated runs

Due to an error, some of the training runs were terminated early. In this part we want to evaluate how usefully the obtained data still can be. Both figures show that there are still improvements expected, but as the plots are logarithmic, the amount of expected improvement is not very high. While it is far from ideal to use these models in the analysis, the conclusions we make should still be representative.

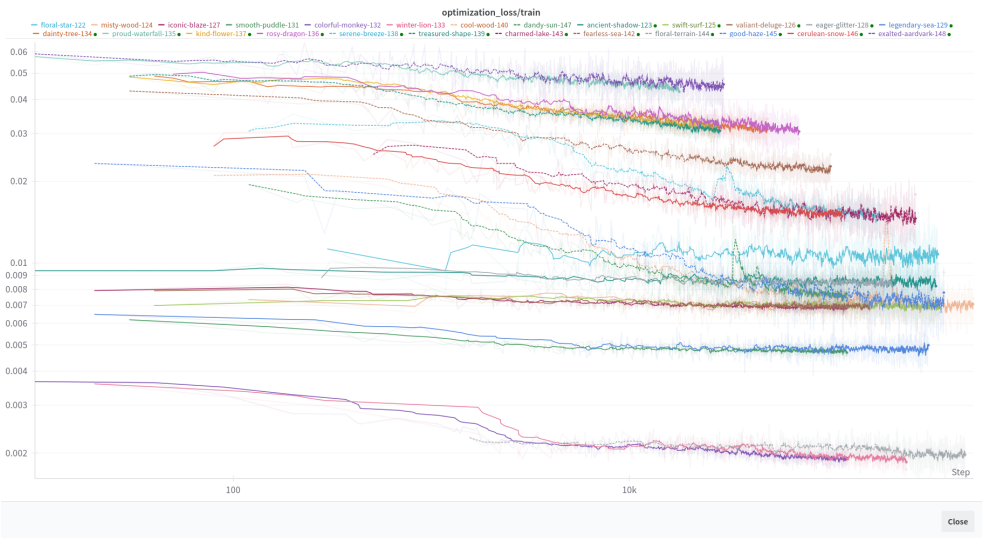

Figure A.1: Curve of the l1 optimization loss used to train the models. All models but the one of run 147 seem to have converged to logarithmic improvement.

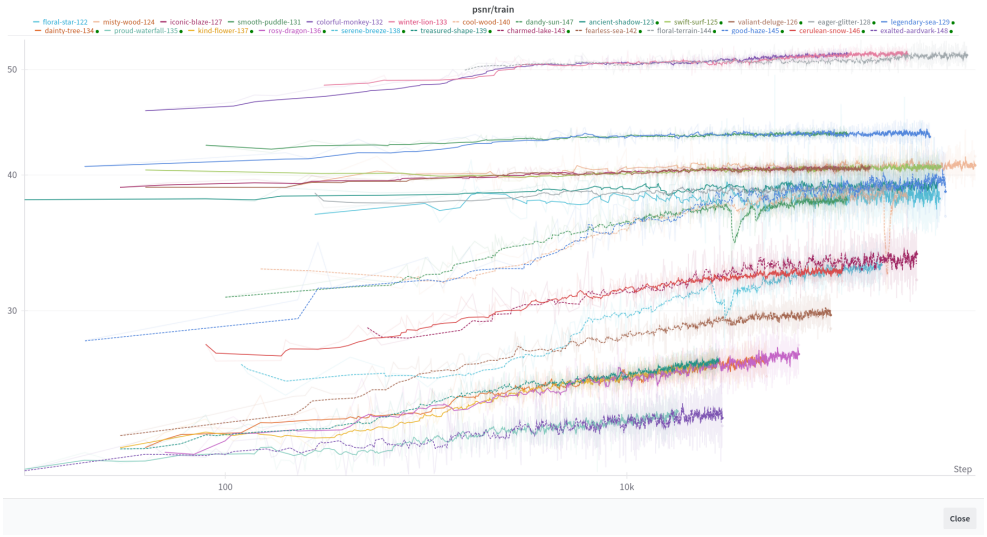

Figure A.2: Curve of the PSNR during training on a logarithmic scale. All models seem to have at least linear behaviour. some models that reached 200 training epochs seem to still have linear improvements, so further training could lead to better results.