



Università degli Studi di Padova

---

DEPARTMENT OF INFORMATION ENGINEERING

*Master Thesis in ICT FOR INTERNET AND MULTIMEDIA*

**Semi-supervised deep learning techniques for  
spectrum reconstruction from RGB images**

*Supervisor*

PIETRO ZANUTTIGH  
UNIVERSITÀ DI PADOVA

*Co-supervisor*

VINCENT PARRET  
SONY EUROPE B.V.

*Master Candidate*

ADRIANO SIMONETTO

Padova, 09 September 2019

---

ACADEMIC YEAR 2018/2019



# Abstract

The task of spectrum reconstruction tries to recover the hyperspectral data of an image based only on the RGB information. Spectrum reconstruction proves to be particularly important as acquiring hyperspectral images is a very long and expensive task, specially when compared to modern RGB cameras.

As it is possible to imagine, the RGB to hyperspectral reconstruction is a far from easy problem, which can be solved with the aid of deep neural networks and a sufficient amount of training data. We therefore decided to investigate some best practices to perform the trainings even when the hyperspectral data at our disposal is limited; we also tried to relate our study as closely as possible to real case scenarios.

In this thesis we introduce some techniques reaching close to state of the art performance while requiring in input only a few hyperspectral images or even pixels. We also show the importance of exploiting the physical model in the training pipeline to constraint the output and ease out the whole process.





# Contents

ABSTRACT	ii
1 INTRODUCTION	1
2 SPECTRUM RECONSTRUCTION	5
2.1 Applications of hyperspectral images . . . . .	6
2.2 Spectrum reconstruction . . . . .	6
2.2.1 Feasibility of the task . . . . .	10
2.3 Related Work . . . . .	12
3 TRAINING WITH LIMITED SUPERVISION	15
3.1 Physical model . . . . .	16
3.1.1 The sRGB colour space . . . . .	18
3.2 Unsupervised training . . . . .	22
3.3 Semi-supervised training . . . . .	23
3.3.1 Alternated training . . . . .	24
3.3.2 Pixel loss . . . . .	24
3.4 Fine tuning . . . . .	32
4 PIPELINE AND TRAINING TECHNIQUES	35
4.1 Convolutional Neural Networks . . . . .	35
4.1.1 Residual Neural Networks . . . . .	38
4.2 Datasets . . . . .	40
5 SETUP AND EXPERIMENTAL RESULTS	45
5.1 Experimental setup . . . . .	45
5.2 Error metrics . . . . .	46
5.3 Results and discussion . . . . .	47
6 CONCLUSION	59
LIST OF FIGURES	61
LIST OF TABLES	63
REFERENCES	64



# 1

## Introduction

The worldwide diffusion of RGB cameras is linked to their ease of use, reliability and affordability. There is a wide variety of applications however, where the information provided by the three canonical channels proves to be insufficient. Applications such as remote sensing [1], medical diagnosis [2] and anomaly detection [3] for example, heavily rely on multispectral or even hyperspectral information.

Acquiring a more accurate representation of the spectrum reflected or emitted by a scene however, comes at a cost. Hyperspectral (HS) cameras are much pricier than their RGB counterpart, and depending on the technology, require longer exposure times or provide lower resolutions [4].

Given the previously described points, it is easy to understand the rising interest in the topic of spectrum reconstruction, also known as spectrum super-resolution. The objective is to start from a low-dimensional frequency wise image (usually a 3-channels RGB), and from it recover the missing information of the spectrum, producing as output an image with a higher number of channels.

The benefit of this technique are twofold: it makes it possible to get the information of hyperspectral images, while using cheaper and easy to handle hardware.

As awesome as it sounds, the advantages are offset by the fact that the problem is really ill-posed as we are trying to get from a lower to a higher dimensionality domain, making spectrum reconstruction particularly challenging.

The first machine learning related work addressing the task of spectrum reconstruction based uniquely from RGB images, was proposed by Arad et al. [5]. They introduced a method based on a dictionary linking hyperspectral and RGB domain. The work of Arad and Ben-Shahar brought to the public also the ICVL hyperspectral dataset, which is still the biggest one up to date. This was of great help for the development of deep learning approaches in the field, since just a year later we can find the work of Galliani et al. [6] that proposed a Convolutional Neural Network (CNN). They modified the structure of the Tiramisù network, previously used for segmentation, and employed it for spectrum reconstruction.

The current state of the art was set in 2018, during the NTIRE 2018 challenge [7]. There, the two ResNet structures proposed by Xiong et al. [8] reached respectively the first and second spot in the challenge. As a matter of fact, deep learning approaches have shown a clear superiority over the competition, as all the entries of the NTIRE challenge and in practice all the better performing methods employ neural networks.

A severe constraint limiting deep learning approaches however, is the unavailability of data. As previously suggested, the various flaws of hyperspectral cameras make it difficult to acquire large hyperspectral datasets, the biggest one amounting only to 256 images [7].

For this reason, we decided to investigate semi-supervised techniques and compare their performance to the current best practices. The main objective was to devise training pipelines requiring only a limited amount of ground truth while retaining competitive performances. To our knowledge, there are no closely related works in the literature, with only [9] proposing some good practices for data augmentation for hyperspectral images, but without any focus on the training side.

The main contributions shown in this thesis are the following: we introduce the physical model inside the training pipeline, constraining the outputs to a smaller manifold of possible solutions and easing out overall the training. We employ said model to develop a technique reaching close to state of the art performance while relying only on a few hyperspectral images as ground truth. On the same line we developed a training approach which only requires a small amount hyperspectral pixels in input which still provides top tier results. Finally, the same approach is used

to successfully fine tune a model on a different dataset, matching the performance obtained in the fully supervised case.

The thesis is organized as follows: in chapter 2 we will introduce the current literature on hyperspectral images and define the task of spectrum reconstruction; chapter 3 will be devoted to the description of the training pipeline used and the proposed novelties; in chapter 4 we will focus on the actual architectures employed and the datasets at our disposal, while in chapter 5 we will compare our results to the current best practices; finally in chapter 6 we will draw our conclusions and propose some lines along which our work can be further developed.



# 2

## Spectrum reconstruction

The perception of colour in the human eye is due to particular cells in the retina called *cones*. These cells come in three different types each with its specific spectral sensitivity, thus making human vision trichromatic. The human eye therefore synthesizes the incoming light into three different categories which can be roughly identified with the blue, green and red colours.

RGB cameras were built with the eye model in mind, and then adapted for specific applications. While amongst the years this technology became more and more accessible and easy to use, it still presents one main drawback that makes it unsuitable for more demanding applications: the three channel representation is a very lossy way to summarize the incoming information.

In order to extract a more complete description of the spectrum of an image we need a technology capable of capturing a higher amount of bands. No clear definition differentiating between *multispectral* and hyperspectral cameras exists; it is possible to say that we are working with multispectral technology if the bands are considerably wide and come in small amounts, while we can talk of hyperspectral images if we deal with a lot of narrow bands. To give some insights we refer to a recent survey where they draw the line between the two at 15 bands [4].

## 2.1 Applications of hyperspectral images

Multispectral and hyperspectral images have a wide variety of applications, ranging from medical diagnosis to agriculture, image segmentation and food inspection.

For example it is common practice to employ an Unmanned Aerial Vehicle (UAV) to take hyperspectral pictures of a field from above. From the spectrum of the image it is then possible to determine the location of invasive plant species, the health and the overall ripeness of the crops [10] (e.g. in Fig. 2.1).

In remote sensing instead [1], the spectral information is exploited in order to classify each pixel of the image, as can be seen in Fig. 2.2.

Another possible application comes from the medical field, where the additional information provided by hyperspectral images helps differentiating tissues and improves the diagnostic capabilities [2].

A growing interest towards hyperspectral imaging is raising also in the field of food quality and safety control. RGB images lack the possibility of monitoring bands outside the visible range, and can end up missing harmful components which would otherwise be spotted employing hyperspectral imagery [12]. As a trivial but meaningful example, we can refer to figure 2.3, where in the RGB image it is hard to discern between rice and plastic, but the distinction suddenly becomes clear when we use a hyperspectral camera.

## 2.2 Spectrum reconstruction

Along with their perks and wide range of applications, hyperspectral technologies present some definite downsides; the current cameras are pricey and either require very long exposure times (which makes it unfeasible to record hyperspectral videos) or provide a lower resolution [4]. The most common approaches can be seen in figure 2.4 with the respective performances regarding resolution of the acquired images and acquisition speed. On the same line it has to be taken into account also the price of the hardware which tends to be inversely proportional to speed and resolution.

In order to overcome these flaws, a lot of focus was put on the processing side to recover the needed information from low resolution hyperspectral images, sometimes



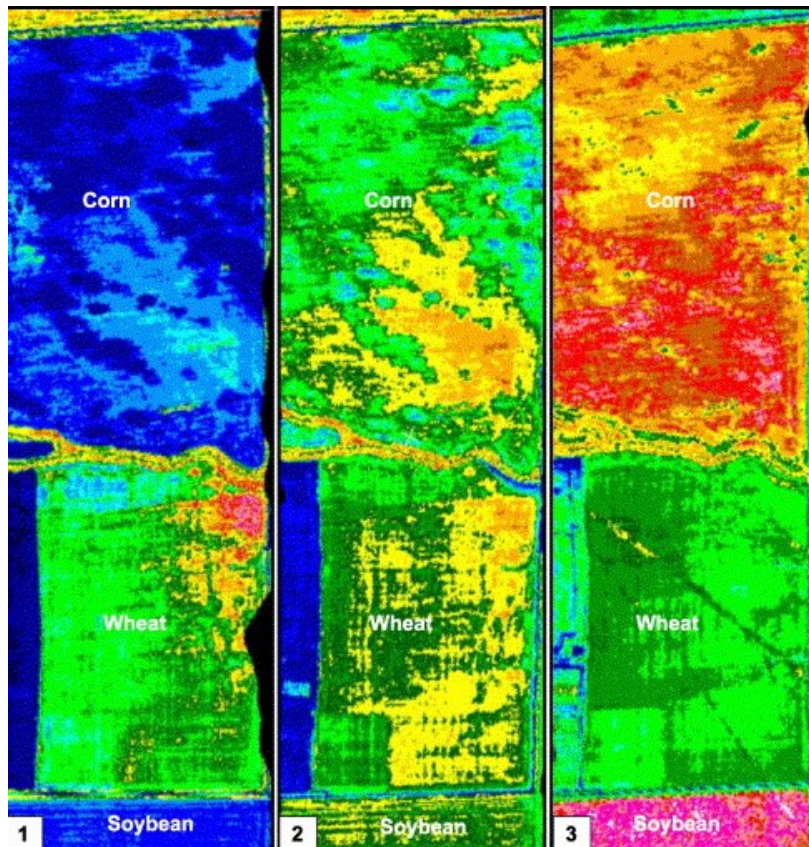


Figure 2.1: Use of hyperspectral imaging to monitor the growth of crops [11]

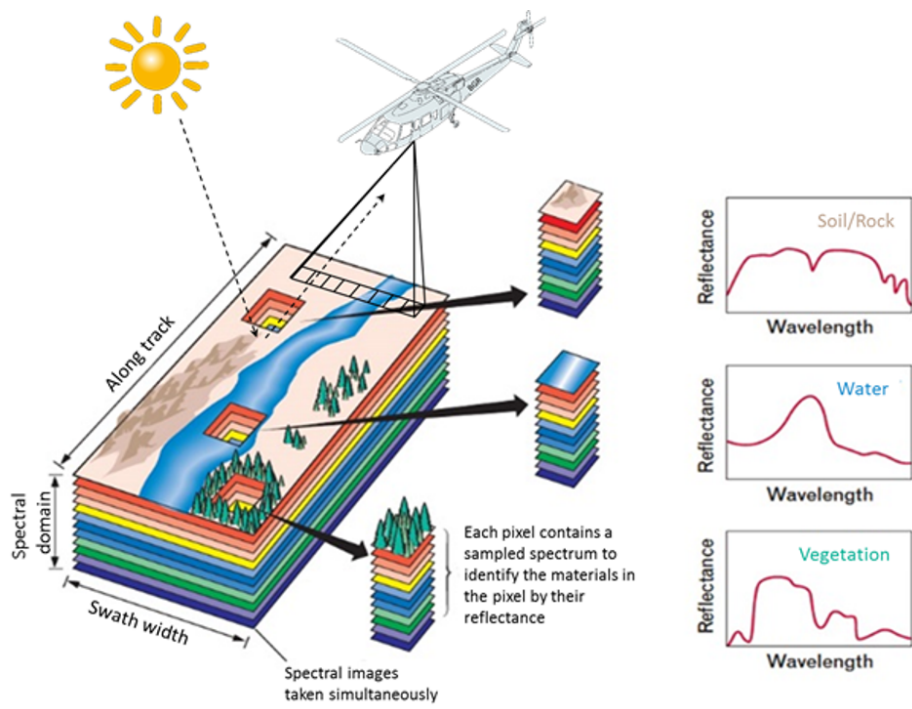


Figure 2.2: Hyperspectral remote sensing [1]

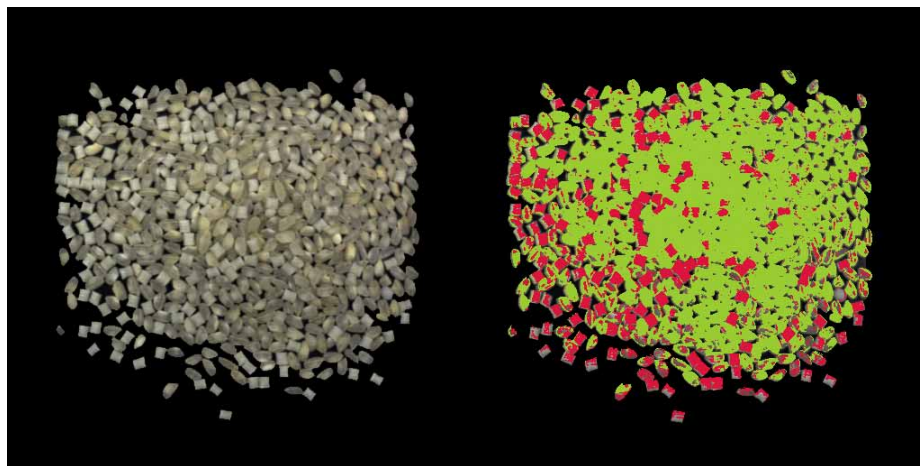


Figure 2.3: Mixture of rice and plastic [13]

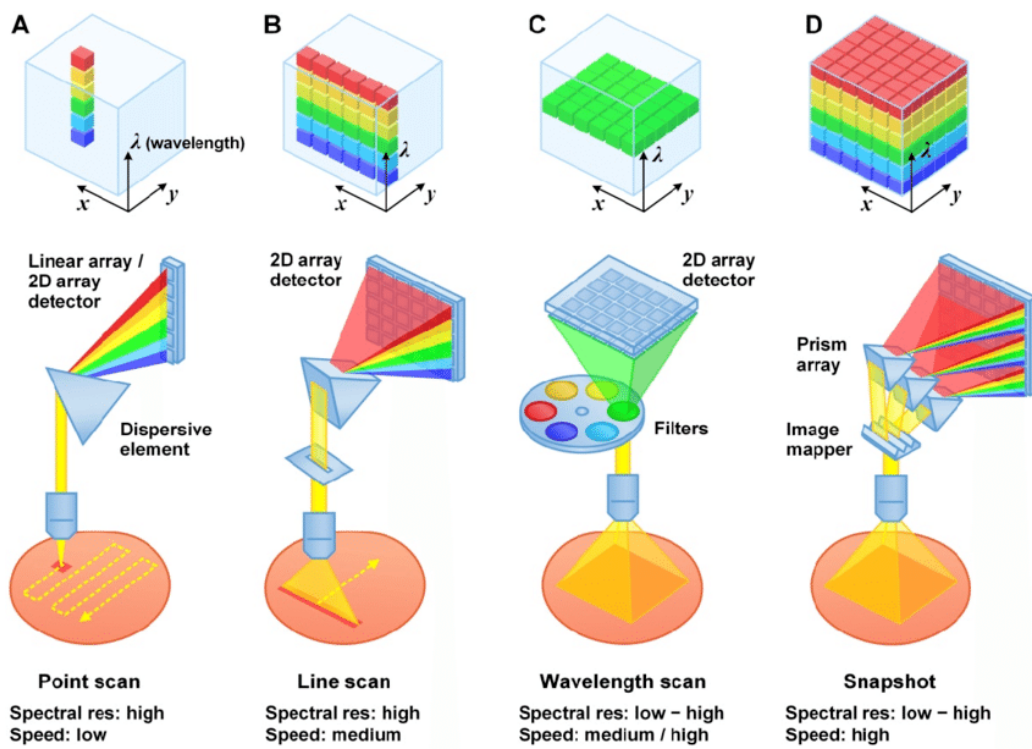


Figure 2.4: Typical hyperspectral imaging approaches [14]

coupled with a high resolution RGB counterpart.

For example, the task of hyperspectral unmixing starts from the idea that each pixel can contain information deriving from different objects and aims at recovering their spectral signature together with their abundance [15]. Hyperspectral super-resolution instead starts with a low resolution hyperspectral image and from it, and possibly a matched high-res RGB, tries to enhance the image in the spatial domain [16].

On a different note, *spectrum reconstruction* tries to limit as much as possible the use of hyperspectral hardware: at inference time the pipeline would get in input an RGB image and provide in output the hyperspectral counterpart [5]. The procedure can also be seen as image super-resolution in the spectral domain, since it is not the size of the image that changes but the number of channels (as shown in Fig. 2.5).

### 2.2.1 Feasibility of the task

One of the first concerns when talking about spectrum reconstruction is linked to very definition of the task. The problem as we are facing it is very ill-posed since there are infinite spectra corresponding to the same RGB values. This issue, known as *metamerism*, arises when lights with different spectral distributions appear the same from the point of view of the sensor system. With the problem being so ambiguous and underconstrained (the mapping is three to many), it appears far from obvious why it should actually be feasible.

However, it has been proved that the frequency of metameric spectra in *natural scenes* can be as low as  $10^{-6}$  to  $10^{-4}$  [17], which means that under the mentioned condition, metamerism can be considered marginal.

Moreover, it has been shown that the minimum amount of degrees of freedom (DoF) required for a reliable description of the visible spectrum, are not that many. In [18], a total of 8 components proved to be sufficient for a correct reconstruction while [19], claimed 23 were sufficient.

Even more interestingly, considering at the same time both the spectral and the spatial dimension, Chakrabarti and Zickler showed that the first 20 principle components account for 99% of the sample variance [20]

From this we can also grasp that the spatial information is going to play an extremely

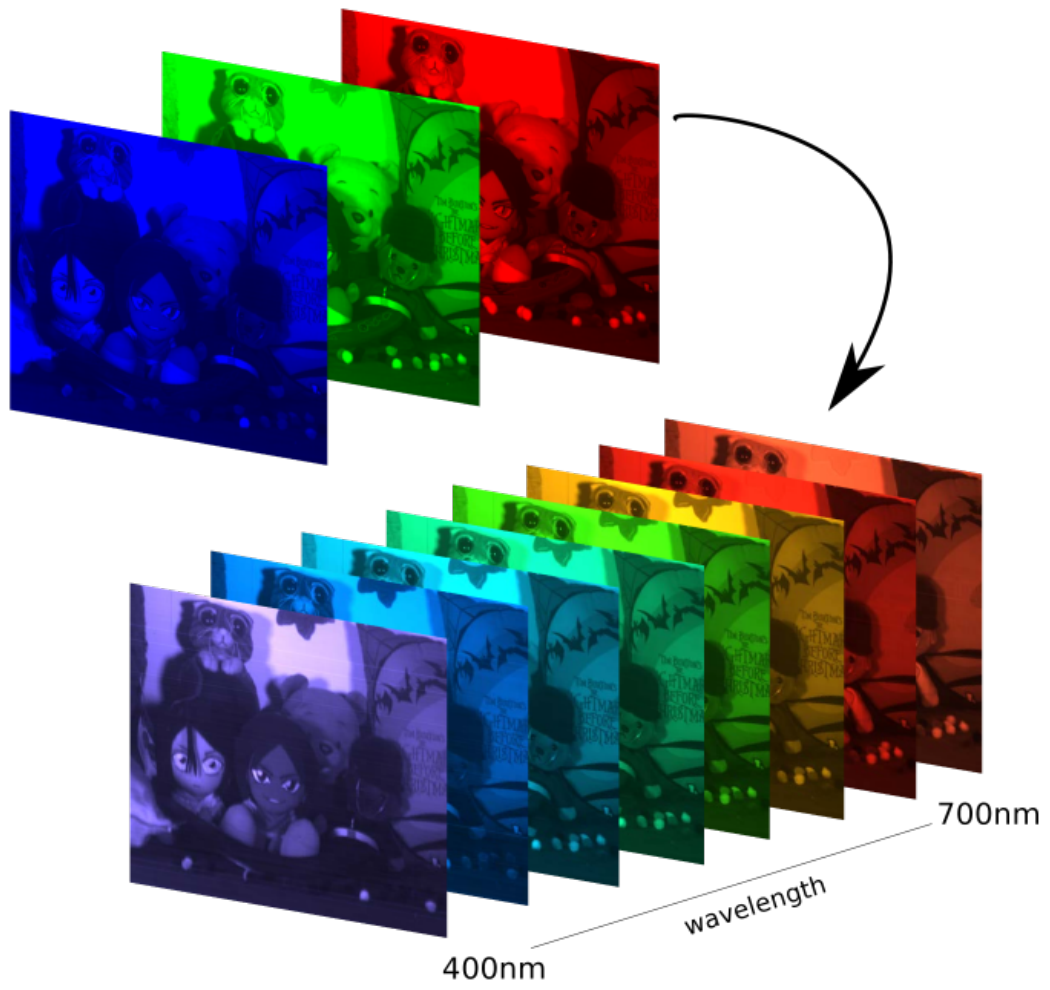


Figure 2.5: Spectrum reconstruction [6]

important role in the reconstruction. In order to exploit it in an efficient way, it becomes rather clear the need to employ convolutional neural networks, given their ability to handle contextual information. In an ideal scenario with copious amounts of hyperspectral data at our disposal, we would be able to train a network with a receptive field big enough that it would manage to recognize objects and learn to associate them to the respective spectra. This is of course still far from our possibilities (hyperspectral databases are very small), but even with very small receptive fields the reconstruction still benefits from contextual information.

In conclusion the marginality of metameric spectra and the sparsity in the spatio-spectral dimension give us good reasons to believe spectrum reconstruction to be feasible.

## 2.3 Related Work

The first work introducing the recovery of hyperspectral information from single RGB images following a machine learning approach, was “Sparse Recovery of Hyperspectral Signal from Natural RGB Images” by Arad and Ben-Shahar [5]. They exploited the sparsity in natural hyperspectral images and built a dictionary of HS values covering the image space. Those HS values were then mapped back to RGB building the corresponding dictionary in the three-dimensional domain. Finally, after this setup step, the relationship between RGB and HS values was exploited in order to perform the backward mapping. The key contribution of their work was the idea of making the RGB to HS transformation fast, performing the most complex computation offline and only later employing the trained model.

Some other deterministic approaches were employed, as for example a work of Aeschbacher et al. [21] that further improved the dictionary method we just mentioned, but neural networks ended up outperforming all other attempts.

The first time deep learning techniques were employed was in 2017 when Galliani et al. adapted the Tiramisù network, shown in figure 2.6, for spectrum reconstruction [6]. It is important to notice that the choice of the Tiramisù network was based on the nature of the problem since the structure had been employed previously for semantic segmentation. The similarity with spectrum reconstruction can be found in the fact



that both tasks have an input image whose size is the same to the output one but for the number of channels (which increases in both instances). Some modifications were of course needed as spectrum reconstruction is a regression problem while semantic segmentation a classification one but the main structure remained the same due to the analogies between the two tasks.

It was only in 2018 with the “NTIRE 2018 challenge for spectrum reconstruction” [7]

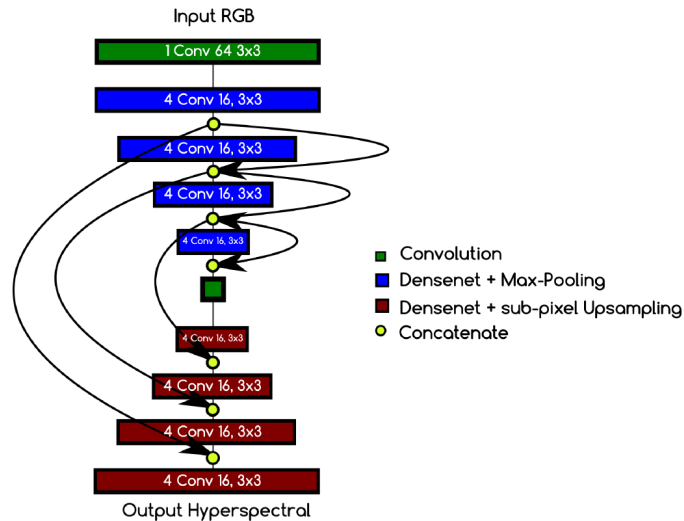


Figure 2.6: The Tiramisù network proposed by Galliani et al. [6]

that the current state of the art was set; all the participants employed deep learning approaches, adapting architectures such as ResNet [22], U-Net [23] and Generative Adversarial Networks (GANs) [24]. The winner of the challenge was the team of Shi et al. that managed to get first and second place with their HSCNN-D and HSCNN-R networks [8] (two architectures based on the residual network model). The networks are based on a structure previously employed for images superresolution [25], a task closely related to spectrum reconstruction as it is too a regression problem where the output has a higher dimensionality with respect to the input.

Finally, another CNN structure worth mentioning is the one from Can and Timofte [47], which reached sixth place in the NTIRE challenge but is the best performing one on other datasets.

Apart from the most successful techniques, it is important to mention the principal hyperspectral databases, without which deep learning approaches would turn out to

be unfeasible.

Worth mentioning is the CAVE dataset, composed of 32 indoor images and collected by Yasuma et al. [26]. From 2011 instead we have the Harvard dataset, a collection of 77 between outdoor and indoor images by Chakrabarti and Zickler [27] and in 2014 the NUS dataset which amounts to a total of 66 images by Nguyen et al. [28]. Finally, the most recent and largest dataset is the ICVL, which was acquired by Arad and Chakrabarti [5]. The dataset was initially composed of 201 images but it was then extended in the occasion of the NTIRE challenge and brought to a total of 256 pictures.



# 3

## Training with limited supervision

All the techniques that have been proposed up to now in the literature have to tackle the same problem they are trying to solve: the lack of hyperspectral data. As a matter of fact, if a common database of RGB images can go from tens of thousand to tens of millions of samples [29], the biggest hyperspectral database is composed of only 256 images [7].

Training a Deep Neural Network (DNN) with such a limited amount of data is a really hard challenge, and splitting the training images into patches becomes a harsh necessity. One valuable solution to alleviate the difficulties would be the use of data augmentation techniques to increase the number of training images, but for hyperspectral data this is still quite an open problem. None of the basic techniques (flipping, rotating, shearing...) seem to provide any advantage [7], and very little literature exists on performing data augmentation on the spectral dimension [9].

Starting from the issues just described, instead of focusing on augmenting the database we decided to investigate some training techniques which can provide sufficiently good performances even when the hyperspectral data at our disposal is limited (just some images), very scarce (only a few hyperspectral pixels) or even null (RGB images only).

In figure 3.1 we can see a brief summary of the approaches we will describe as follows.



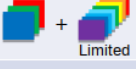



RGB images # images needed	HS images # images needed	Point-wise spectra # spectra needed	Available data	Training loss				Main message
				RGB Image- wise	HS GT Image- wise	HS Image-wise From spectra	TVM	
	+				+			Fully supervised
+				+			+	Unsupervised
+	+		 Limited	+	+			Semi-supervised (image-wise) Alternated training
+		+		+		+		Semi-supervised (point-wise) Spectra for approximated GT
+	+		 + pre-trained model	+	+			Semi-supervised (image-wise) Alternated training Fine-tuning to other dataset
+		+	 + pre-trained model	+		+		Semi-supervised (point-wise) Spectra for approximated GT Fine-tuning to other dataset

Figure 3.1: Summary of the devised approaches

### 3.1 Physical model

One of the key tools employed throughout all of the proposed techniques is the so called *physical model*. The main point is that, whenever possible, it proves to be extremely useful to use additional constraints based on the physical principles behind the problem we are addressing. For example in [30] the authors' objective is to retrieve illumination and reflectance information from RGB images; as from these two components it is possible to uniquely identify the original image (by simply computing the product), they took pictures with identical reflectance and different illumination to constraint the training. In [31] instead, they use a similar approach for deblurring videos: a deblur network is applied to the input images and the output is then blurred again according to a noise model, and compared to the original image.

Finally, a much more recent example examined the problem unsupervised image decomposition from a more general point of view [32]. The point is that tasks such as image segmentation, transparency separation (separate the image into its superimposed reflection and transmission) and image dehazing (recover from a foggy image the clean part and the noise) all have the common objective of splitting the image into

layers. They use this observation and the strong self-similarity of the images after their separation to define a multi-purpose framework for image decomposition. The model works over three losses: the first one is the *reconstruction loss* ensuring that the two images after the decomposition still lead to the starting one, the *exclusion loss* minimizes the correlation between the output images and the third loss is task specific.

In our case the physical model is constituted by the hyperspectral to RGB transformation. In practice while the RGB to hyperspectral transformation is very under-constrained, the opposite relation is univocal: given a spectrum and the back-transformation to a colour space, we can always identify a single corresponding RGB triplet. This means that it does not matter what our network gives in output at any stage, it has to hold that the transformation of our output back to RGB has to be equal to the input RGB image.

This of course does not in any way mean that the output values of the DNN are the ones corresponding to the ground truth (there is an infinite amount of possibilities providing the same RGB image), but it severely constraints the manifold of possible solutions.

What the physical model effectively does is drastically reducing the amount of ground truth required by the model for a successful training. The reason for that is that it pins down the DNN preventing him to choose some solutions that we can immediately label as wrong. Based on this, it becomes clear that the addition of the physical model can greatly help in the case the hyperspectral data at our disposal is insufficient at the reduced price of additional RGB information.

The most general form of our training pipeline can be seen in Fig. 3.2 where the DNN can be any network structure (for all of our attempts we chose the HSCNN-R network from Shi et al. [8]), the physical model consists in the transformation of the DNN's output back to RGB and finally we have some additional constraints, which are crucial for determining a good solution. The physical model by itself in fact, cannot alone lead us to the correct solution since given an RGB triplet, it would always be possible to choose a spectrum with all values put to 0 but for three giving us the desired RGB components. The constraints we mentioned can be of very different nature, from the trivial comparison to ground truth up to more sophisticated losses.

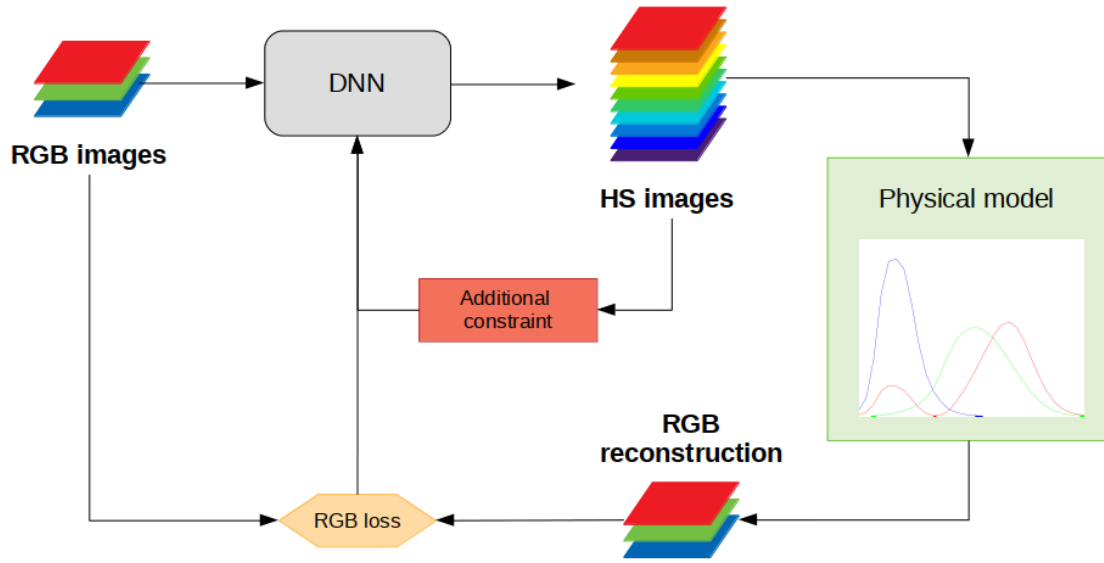


Figure 3.2: Structure of the training pipeline

### 3.1.1 The sRGB colour space

As we mentioned before, the conversion of a spectrum to RGB is univocal if we define the colour space.

Colour spaces are specific organizations of colours designed to fulfil determinate tasks, all revolving around the representations of colours with different display technologies. Just to cite some examples, we have the XYZ space which is one of the oldest, the CMYK used for printers and sRGB and Adobe RGB which are common for cameras and monitors (shown in Fig. 3.3).

The CIE 1931 XYZ colour space was created with in mind the idea of giving a quantitative link between the distributions of wavelengths in the visible spectrum and human perception. Given a spectrum  $S$ , the conversion to the XYZ colour space is performed by the three following integrals:

- $X = \int_{\lambda} S(\lambda) \bar{x}(\lambda) d\lambda$
- $Y = \int_{\lambda} S(\lambda) \bar{y}(\lambda) d\lambda$
- $Z = \int_{\lambda} S(\lambda) \bar{z}(\lambda) d\lambda$

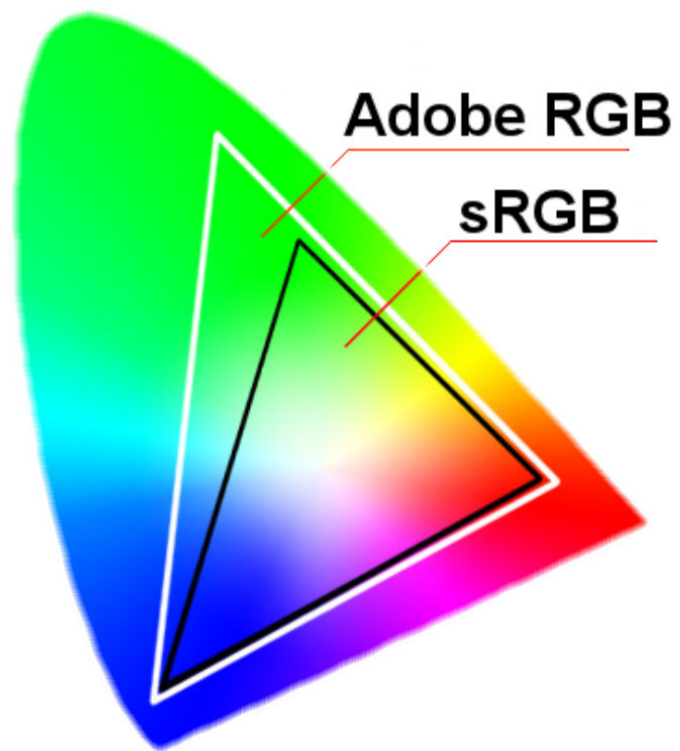


Figure 3.3: sRGB and Adobe RGB colour spaces

where  $\lambda$  is the wavelength (it goes from 400 to 700 with pace 10 in our case) and  $\bar{x}$ ,  $\bar{y}$  and  $\bar{z}$  are the XYZ colour matching functions shown in fig. 3.4. The three functions were defined trying to mimic the chromatic response of an average human.  $S$  instead is the component coming in input to the sensor, and it consists of the product between the reflectance of the object  $R$  and the illumination  $I$  due to the incident light.

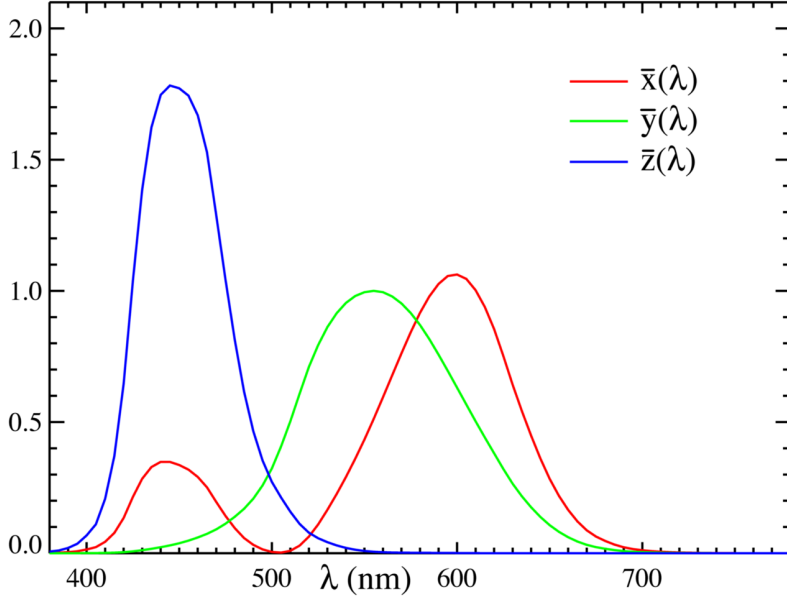


Figure 3.4: CIE 1931 colour matching functions

Starting from this we can then perform a conversion to the RGB space with a simple matrix multiplication, and finally reach the sRGB colour space applying gamma correction, which consists in the following transformation:

$$\gamma(u) = \begin{cases} 1.055u^{\frac{1}{2.4}} - 0.055 & \text{for } 1 \geq u \geq 0.0031308 \\ 12.92u & \text{for } 0.0031308 > u \geq 0 \end{cases} \quad (3.1)$$

where  $u$  stands for any of the three R,G,B values and  $\gamma(u)$  is the corresponding element in the sRGB domain .

The previous procedure synthesizes the physical model employed to get the sRGB information for all of our experiments. The physical model provides an useful loss which we are going to call *RGB loss*. Let's consider the main structure shown in fig-

ure 3.2: the output of the network is converted back to RGB via the physical model; since as we know the transformation is univocal, the reconstruction has to provide us with the input RGB image. The RGB loss is therefore based on the comparison between the input RGB image and its reconstruction after going through the DNN using the physical model.

**The physical model in a real case scenario** In a practical case we expect the user to possess a set of images taken with an RGB camera and to only have the corresponding spectra of a few of those.

Even though the sensing device is of course built with the objective of converting the incoming spectra to a colour space, let's say sRGB, the transformation will in practice be different from camera to camera due to the technology and the manufacturing process behind the sensor. This means that it would be very unwise to apply the ideal procedure in a real case, since we would end up working with RGB images belonging to different colour spaces, which would hinder the performance of the network.

In practice, for all our proposed approaches to work in a real case, the user would need the sensitivity function of the camera; this is a quite reasonable assumption as it is common for the manufacturer to provide it.

As a last remark it is worth mentioning that even with the camera sensitivity function at hand it is not possible to replicate in a perfect manner the hyperspectral to RGB transformation as we would need a rounding operation in the end since real images are only composed of discrete values. This operation however cannot be inserted in our pipeline as the model needs to be differentiable for the backpropagation algorithm to work. This deficiency however comes far from creating any real issues and has an even smaller impact the higher the number of pixels employed to represent our images.

On a different note, the proposed training pipeline is not limited to the RGB to hyperspectral case and can of course be applied to many analogous scenarios with different amounts of input (e.g. grayscale) and output channels.

## 3.2 Unsupervised training

We can begin by considering an extremely challenging scenario: when the data at our disposal is RGB only. With no hyperspectral information available, it is hard to find an useful condition leading to promising results.

We observed that the output of our network, when under the sole constraint of the physical model, was providing some extremely noisy shape of the spectra. Since the image spectrum usually shows a great amount of correlation between nearby channels, we decided to fight this unwanted behaviour by penalizing high variances.

In practice we followed the line of the Total Variation Minimization (TVM) algorithm in one dimension [33]: the original noisy output image, let's call it  $x$ , is cleaned into a smoother version  $y$  according to equation 3.2

$$y = \min_y E(x, y) + \lambda V(y) \quad (3.2)$$

where the first term is  $E(x, y) = \frac{1}{n} \sum_{i=1}^n (x_i - y_i)^2$  which measures the mean squared error between the starting signal and the clean one,  $V(y) = \sum_{i=1}^n |y_{i+1} - y_i|$  measures the amount of variation in the new signal and  $\lambda$  weights the two factors.

The main idea is to apply this minimization factor in the spectral direction together with the physical constraint (reprojection to RGB). In our particular case, the term  $E(x, y)$  proved to be unnecessary as the same effect was produced by the physical model.

In Fig. 3.5 we can see a high level description of the training pipeline. In Fig. 3.6 it is possible to see the effect of employing the TVM algorithm in the spectral direction. While using the RGB loss alone can lead to a very noisy output as in Fig.3.6a, Total Variation Minimization leads our training to a much regular output (Fig. 3.6b).

The output that we get is of course still far from optimal, but presents some very useful properties such as correlation in the spectral direction (forced by the TVM algorithm), values in the “right” range and of course knowledge of the physical model (the output has full knowledge of the input). As we are going to see some of the training techniques will present issues when trained from scratch but will instead



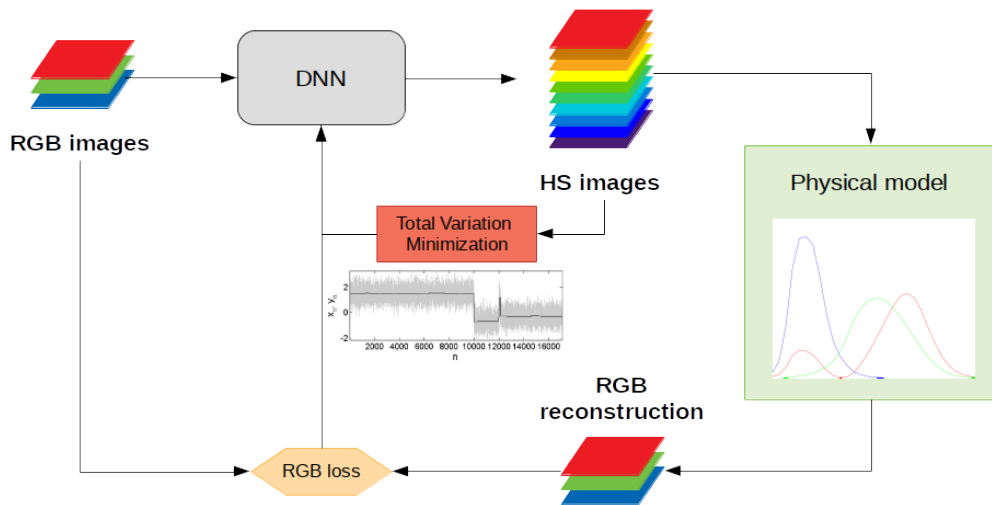


Figure 3.5: Training pipeline in an unsupervised scenario

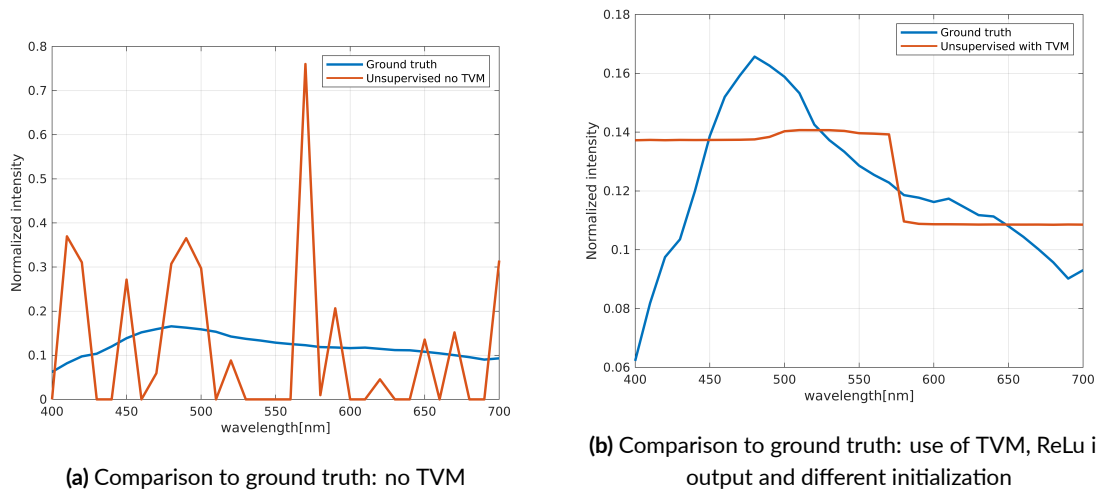
converge pretty easily when we use the pre-trained network.

As a last note, we surely believe it to be possible that some losses other than TVM (or combined with it), would have ended up giving some slightly better results, but overall such performance was definitely expected since the task is structurally very ill-posed

### 3.3 Semi-supervised training

After studying the case of unsupervised learning, we considered the event of training with a limited amount of supervision. The idea is to address some realistic scenarios where the database is mostly composed of RGB images, but there is a certain amount of hyperspectral information, be it on the form of a few images or even only some pixels.

In the first event we have the possibility of acquiring hyperspectral images but RGB data is more easily available, while in the second we are only able to acquire single HS points along with the RGB images.



**Figure 3.6:** Spectrum before and after applying TVM compared to ground truth

It is important to specify that the proposed approaches are only made possible by the physical model which gives us the opportunity of using RGB data for our trainings and at the same time constraints the solutions easing out the training.

### 3.3.1 Alternated training

In the event the hyperspectral data that we have comes in the form of images, the best option consists in alternating our training over two losses: if the patch we are considering possesses its hyperspectral counterpart we can directly evaluate the distance from the ground truth, if otherwise the patch is RGB only, we will switch to the RGB loss.

The main workflow is reported in Fig. 3.7 where we can see that the constraint now simply consists in supervision. The importance of the RGB loss in the training consists in avoiding overfitting (since typically the amount of RGB data is much larger), which is otherwise a serious issue when the dataset is too small.

### 3.3.2 Pixel loss

A much more challenging scenario, but very useful in a real case, appears if the only aid to the RGB dataset comes in the form of a few hyperspectral pixels.

Hyperspectral cameras are indeed pricey and it is reasonable to think of an event

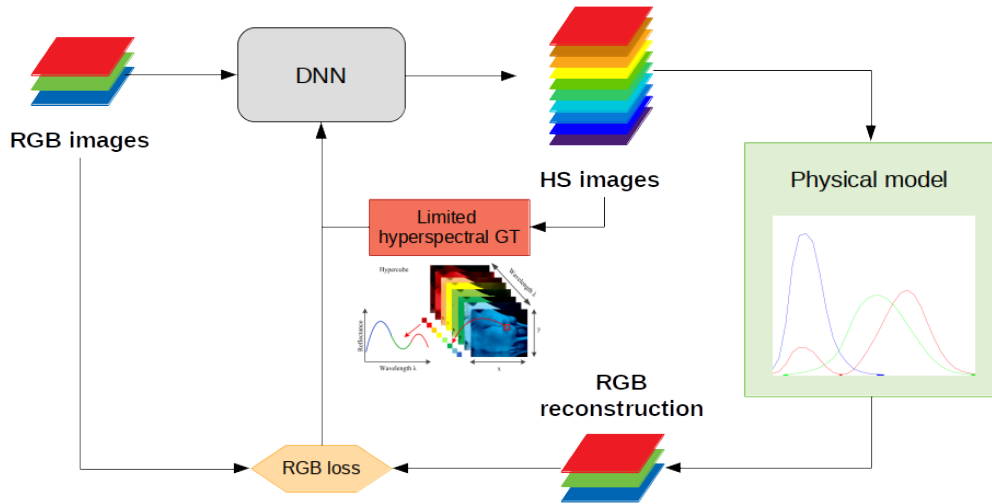


Figure 3.7: Training pipeline using a limited amount of images as supervision

where the tools at our disposal are only an RGB camera and a spectrometer. This device, while much cheaper than an hyperspectral camera, is only able to acquire the spectrum of a single pixel at a time, making it unfeasible to collect the whole image matrix. In practice, it is reasonable to think that the total amount of hyperspectral data at our disposal can vary from a few hundreds to a few thousands pixels. The same procedure can be applied to similar frameworks, as for example an event where the sensor itself is built in a way to provide an RGB image with sparse HS information.

Finally, it is worth noticing that this approach is not limited to spectrum reconstruction, but also to a closely related field on spectrum superresolution. The idea is again to reconstruct the spectral information but this time in input we have both a high resolution RGB (as in spectrum reconstruction) and a low resolution hyperspectral version [34] of the same image. The following approach can be easily applied to this case as it just requires hyperspectral pixels, without any constraint on their spatial distribution.

In figure 3.8, we can see the kind of data we are dealing with: our images are mainly RGB apart from a few pixels that possess hyperspectral ground truth.

This setup, without a doubt challenging, turns out to be even more problematic when

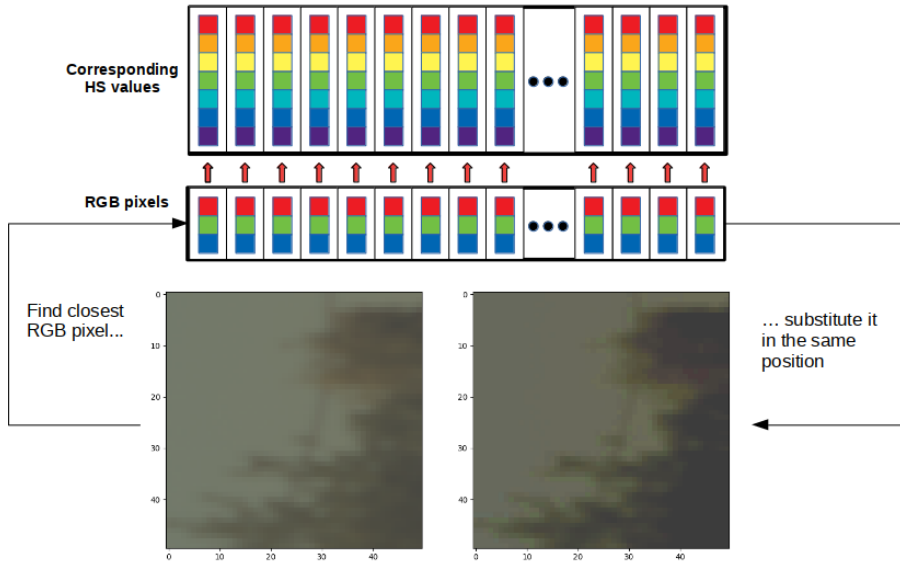


Figure 3.8: RGB image with sparse hyperspectral ground truth

employing deep learning, since it is not clear how it would be possible to train a model with only a few pixels at our disposal.

The main idea is to employ a preprocessing step where we *build* some realistic hyperspectral ground truth based on the RGB info at our disposal.

The procedure is based on the following hypothesis: pixels with similar RGB values also have similar spectra. As it is possible to imagine this statement is not always true, since it is indeed possible for very different spectra to correspond to similar if not equal RGB values (metameric spectra); anyway if the hyperspectral manifold is not too wide such an assumption turns out to be reasonable. Moreover, as previously mentioned, the network can also exploit the contextual information to discern between pixels with similar RGB values. For the sake of clarity let's assume our training data was enough for the network to recognize some objects. Were this the case, even if we had a tree and a car with the same exact RGB colour, then the network would be able to retrieve the correct spectra from the contextual information, as he would know from the training images how the spectrum of a car and that of a tree should look like. This example is of course a bit far fetched as the small training set at our disposal only allows the network to focus on a very small receptive field, but even with this limitations it can still exploit some contextual information (edges,



**Figure 3.9:** Creation of fake HS dataset with a set of ground truth pixels

corners etc.).

Based on the previous assumption we now want to build an approximated hyperspectral dataset. In order to do so, we follow the procedure shown in Fig. 3.9: given an input RGB patch, for each of its pixels we look for the closest neighbour amongst the few pixels we have with ground truth and assign the HS data to the original RGB pixel. In practice we are building a hyperspectral patch based on a lossy copy of the original RGB patch.

The reconstructed RGB image as we can see in figure 3.9 is not exactly identical to the input (we only have a few hundreds/thousands pixels after all) and the same we expect on the hyperspectral side, but as we will see in chapter 5 it turns out experimentally that this approximated hyperspectral ground truth proves to be very useful for the training.

In Fig. 3.10 we can visualize the proposed pipeline. We can briefly compare the proposed method to a naive approach where we employ the RGB loss for the pixels without ground truth and the HS for the remaining ones. This method would simply fail to train because the receptive field of our network is not even close to being big enough to cover a whole image. In the extreme scenario of having only 100 pixels with HS information, we end up on average with a single ground truth pixel for image,

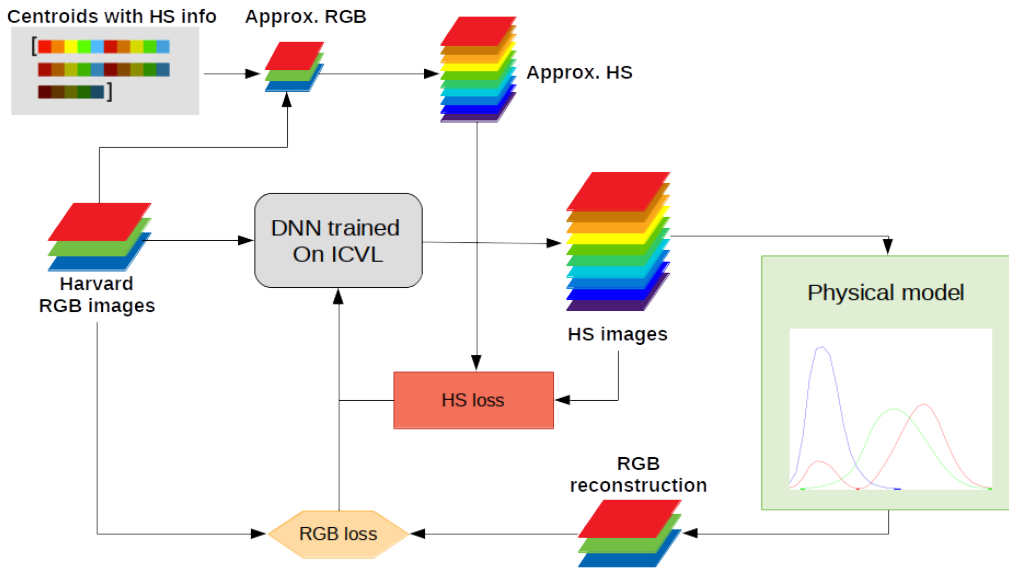


Figure 3.10: Training pipeline using only a few pixels as supervision

which will not have any impact during the training.

Another variable to keep into account, crucial for building a good approximated hyperspectral dataset, is the choice of the set of ground truth pixels. In the image in Fig. 3.9, it was simply done choosing the pixels at random amongst the whole pool of possibilities; this choice, far from reasonable, gives a high preference to pixels that are in a high amount inside of our database. For example, if the dataset is mainly composed of outdoor images, we can expect the sky to be a very meaningful component of our dataset, and for this reason we can expect to randomly select a high amount of pixels of the sky. This however would not provide additional information, since as their spectra will be very similar, we would just find ourselves with a lot of redundant info about the sky but at the same time small to no coverage on less represented areas.

What we want in practice is to provide each of the pixels in the dataset with a good enough neighbour and therefore find the set of pixels covering the RGB space of our images in the most regular way possible.

Before actually looking for an algorithm that performs such task, we need to define what we mean with similar pixels. The similarity of two pixels is inversely propor-

tional to their squared Euclidean distance. Therefore the distance between two RGB pixels  $p_1$  and  $p_2$  is:

$$d_{1,2} = \sum_{i=1}^3 (p_{1,i} - p_{2,i})^2 \quad (3.3)$$

The choice of the squared Euclidean distance is indeed debatable as in fact, what we would like to do in reality would be minimizing a measure on the hyperspectral domain; anyway, since the relation between RGB and HS values is far from obvious (otherwise spectrum reconstruction would not exist in the first place), focusing on the RGB domain is indeed logical.

In conclusion, given a set of three dimensional points, we want to find a partition of it, such that the mean squared distance of each point with respect to the centre of its cluster is minimized, which is the reason we decided to employ the K-means algorithm.

**K-means** The problem tackled by the k-means algorithm is the following: given a set of  $n$  points belonging to a space of dimension  $d$ , find the partition of the points in  $k$  clusters such that the within-cluster sum of squares is minimized. The previous metric can be shown to be equal to the mean squared distance of the points from the centre of their respective clusters.

The previously stated problem is *NP-hard* but the k-means algorithm succeeds in giving a close to optimal and efficient solution.

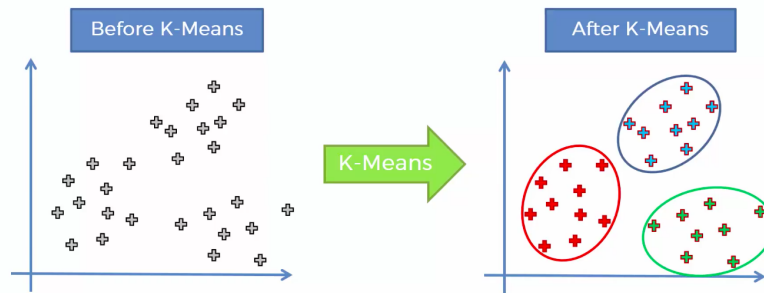
---

**Algorithm 3.1** K-means

---

- 1: Choose the number  $k$  of centroids
  - 2: Randomly choose  $k$  starting centroids
  - 3: Repeat steps 4 and 5 until a condition is met (e.g. number of steps or improvement under a threshold)
  - 4: For each point  $x_i$  find the closest centroid  $c_j$  and assign it to its cluster
  - 5: For each cluster compute the new centroid as the mean of all the point assigned to it
  - 6: End
- 

The algorithm in its simplicity presents three major limitations that have to be taken into account:



**Figure 3.11:** k-means algorithm on 3 clusters

1. The goodness of the solution strongly depends on the starting set of centroids
2. The number of clusters has to be chosen beforehand
3. The algorithm forces a spherical symmetry on the clusters

In our case these limitations do not cause any significant issue.

A poor initialization usually hinders the performance of the algorithm only when the number of clusters is small; in our case it is reasonable to expect its amount to be of at least 100 since the space of images we want to cover can be composed of millions of pixels. Moreover, it is also possible to deal with the issue using a smart way to initialize the clusters such as k-means++ [35].

As for the actual number of clusters, it turns out to be quite helpful to be able to control it, since while on one hand the paradigm “the more the better” surely makes sense (more centroids means a better coverage of the RGB space of our images), on the other hand this tool is meant to be a starting point for a later acquisition of hyperspectral data on the selected points. This means that while having a few hundred thousands centroids to build our dataset sounds pretty appealing, the idea of having to acquire each of them singularly makes it less so. Moreover, we also need to keep into account that for each of the pixels of the images, we will need to find the closest neighbour in the set of centroids, which can get out of hand pretty quickly if they are too many. What we need therefore is to decide beforehand for a good trade-off between the two measures and then apply it by selecting the desired number of clusters.

Finally, based on the starting definition of distance (with the due limitations), we can see how a spherical shape of the clusters can be fitting. The similarity is defined as



having a spherical symmetry, which goes along with the functioning of the algorithm.

**Limitations and further improvements** The previously described procedure, with its efficiency and ease of use, still shows some weaknesses.

One consideration is that in the pre-processing step we are not taking into account any spatial information, which could be useful for the reconstruction. Therefore, it could be sensible to put some additional description to each of the pixels (e.g. mean RGB values of his neighbours) and then run k-means. This possible improvement however leads to many complications, such as how to balance information of different kinds and the fact that we are not anymore in a three dimensional space but a bigger one, which can lead to very sparse information and bad clustering.

Another possibility would be to use a non-parametric clustering algorithm, such as mean shift[36]. The algorithm might give some problems when the number of pixels gets too high and it is not obvious how to choose the size of the window but overall seems to have a good affinity with the problem at hand.

It would also be interesting to see how much the reference colour space influences the centroids and therefore the overall performance. In our case we managed to verify that the results were actually the same, but we only managed to test this on the Lab colour space with a total amount of 10000 centroids. Intuitively we would expect some variations only when the number of centroids is small ( $\sim 100$ ).

On a different note it is important to relate this approach using k-means with a real scenario. In a practical case it would not be immediate to apply the algorithm, since even though it is very efficient, the images we are considering are composed of millions of pixels. This would require a prior acquisition of RGB images and only later the use of the spectrometer on the highlighted points. While the procedure can be applied (within some margin of errors), it is not ideal as it would be much slower than directly hand-picking the pixels.

Based on this consideration how can we compare the performance using pixels chosen by the algorithm and others chosen by a human? The immediate answer is of course that the only way to find out would be to train our pipeline with hand-picked points

and then compare the two, but it is still possible to say something in advance. The key thing is that it is very far from obvious that a human choice would perform worse than an algorithm guided one, since while the algorithm will of course have a very good performance on the metric it is built to optimize, that is not the same metric we want to optimize. In practice k-means only sees RGB data and while we believe it to have a good relation with the hyperspectral one, it is for sure not true in general and some of the pixels will end up mapped to hyperspectral values far from their ground truth. A human expert instead, while being worse on the computational side, has prior knowledge of the spectra of objects and knows which ones need to be acquired; moreover, he can also use contextual information and base his decision on the objects. In conclusion the two approaches should end up providing comparable performances and the best choice would be to combine the two.

## 3.4 Fine tuning

One last technique we decided to test was the possibility of training a model on a bigger database and then adapt it on a smaller one for which less hyperspectral info is available. The idea is to apply the previously described approaches for fine tuning.

Fine tuning [37] is a deep learning technique that consists in reusing previously learned information for a different task. The point is that if one particular database is too small to train a network with satisfying performances, then it is possible to adapt an already existing network previously trained on a much bigger dataset (figure 3.12).

As a starting point we used the HSCNN-R model previously trained on the ICVL dataset (full supervision). The training pipeline can be found in figure 3.13. Our focus was not simply centred on using the smaller dataset as it is for the tuning, but mostly on comparing the performances reached by a fake dataset built with the procedure shown in section 3.3.2 with the standard training of the network.

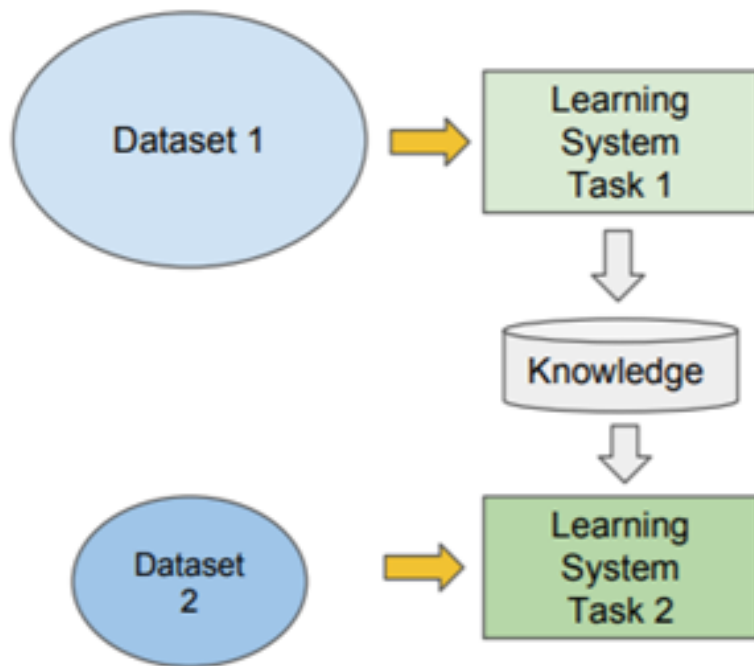


Figure 3.12: Fine tuning [37]

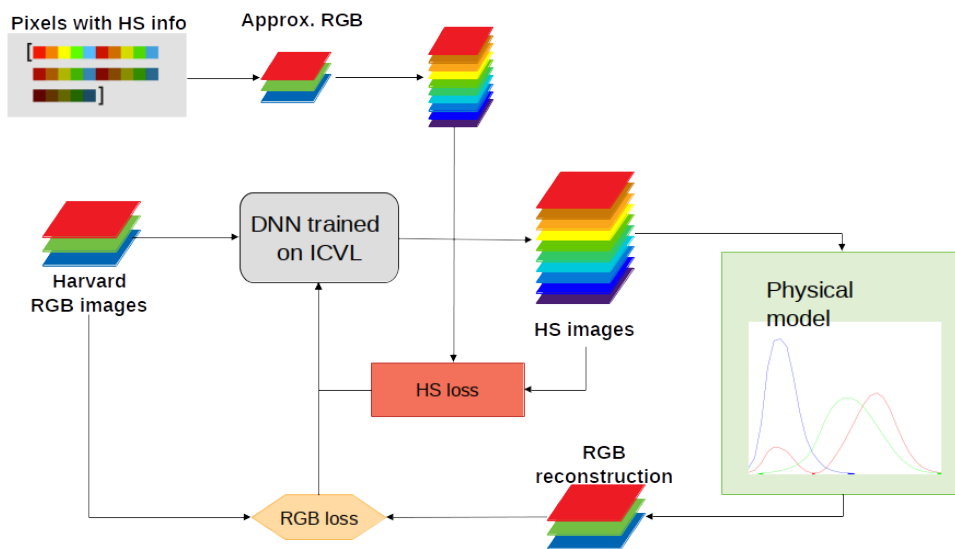


Figure 3.13: Fine tuning on the Harvard dataset



# 4

## Pipeline and training techniques

The network that we decided to employ for all of our approaches was the HSCNN-R architecture proposed by Shi et al. [8], which came in second place at the NTIRE 2018 challenge [7]. While not having as good of a performance as the HSCNN-D variant, the HSCNN-R network shows a much faster convergence time while still providing close to state of the art results.

In the next few sections we are going to describe the structure of the network, starting from the theory behind Convolutional Neural Networks (CNNs), then explaining the difference between them and Residual Neural Networks (RNNs), and finally focusing on the peculiarities of HSCNN-R and the slight modifications we did to the architecture.

### 4.1 Convolutional Neural Networks

The main issue when applying a fully connected neural network to an image concerns the sheer amount of parameters needed to make the structure work. The number of connections in the network is in fact directly related to the number of input elements, which in the case of an image it can range from a few thousands up to a few millions, generating a network which would simply be impossible to train.

Images however show a high degree of spatial correlation, a useful property that has been exploited in a deep learning framework under the shape of convolutional neural networks (Fig.4.1).

CNNs exploit the grid-like shape of images in various ways:

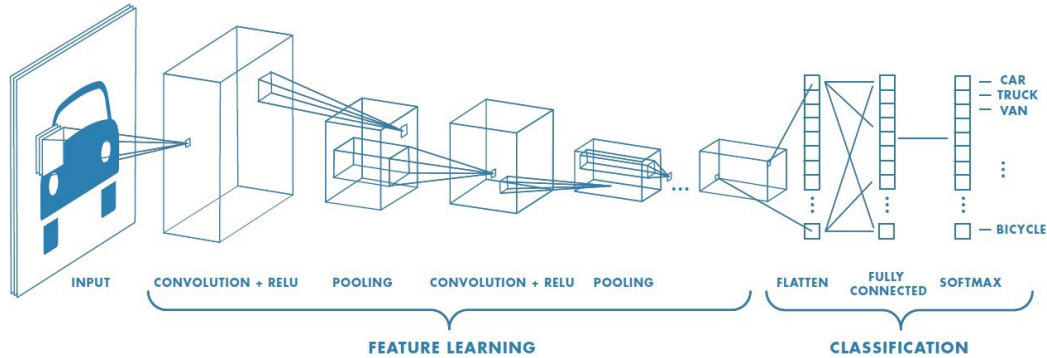


Figure 4.1: Structure of a generic CNN [38]

1. One crucial point is the idea that it is sufficient to connect the hidden units only to subregions of the image (as in Fig. 4.2), since pixels show a high correlation with their neighbours while proving to be less and less dependent as we stride farther away. In this way the hidden units are only focused on areas where we know we will have useful info and the number of parameters of the network is greatly decreased.

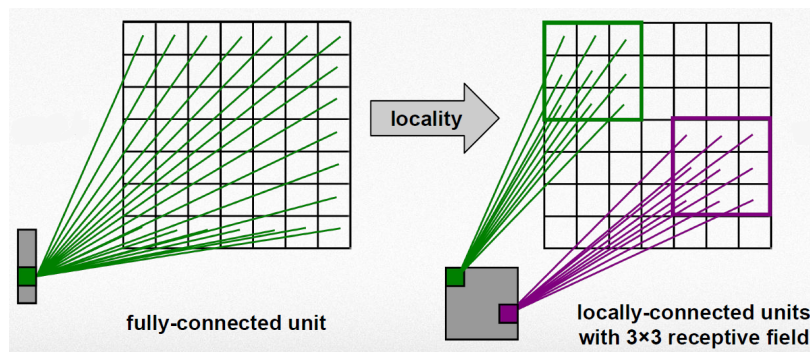


Figure 4.2: Comparison between fully connected networks and CNNs [39]

2. Another tool which helps reducing the computational burden is the concept of *parameter sharing*. It consists in sharing the parameters along the hidden units belonging to the same feature map. The idea is that if a filter is able to detect

a particular shape in a subregion of the image, then it is probably going to be able to do the same in a different subregion.

Parameter sharing manages to furthermore reduce the computational burden of the network and gives a particular structure to the CNN, since it assembles the hidden units into groups (the feature maps), and each feature map refers to a particular filter (the shared weights) as shown in figure 4.3.

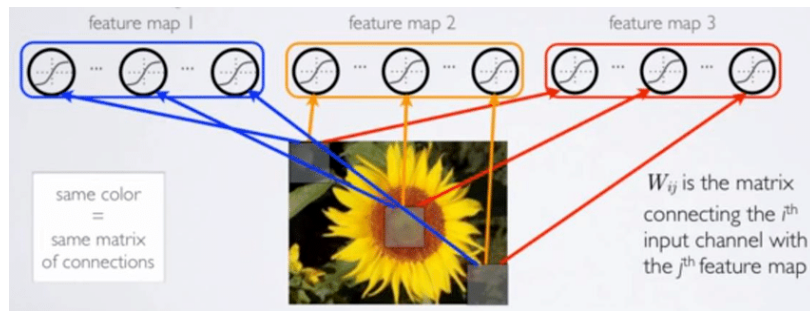


Figure 4.3: Parameter sharing in a CNN [40]

3. One last important element in a convolutional neural network is *pooling*. After each layer with feature maps, a pooling layer is commonly added where the local information gets summarized as we can see in figure 4.4. In this way the size of the network does not go out of hand (the number of feature maps tends to increase with depth), but at the same time we do not lose high amounts of useful info (due to the high correlation in the spatial dimension).

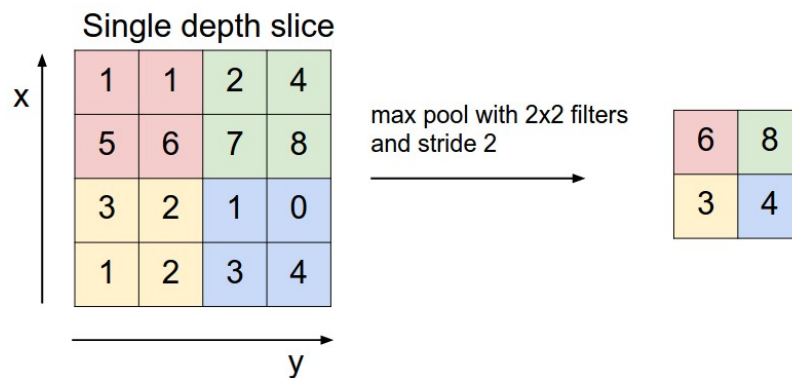


Figure 4.4: Example of max pooling with a 2x2 filter [41]

Convolutional neural networks have been successfully employed for a wide variety of tasks concerning images, such as classification [42], segmentation [43], object detection [44], face recognition [45] and many others.

The field has developed a lot throughout the years and some architectures have been establishing their trademark for their efficiency in specific tasks. A common trend that can be seen, which went along with the technological advances and an always more growing amount of training data, was the growing depth of the networks. Deep Convolutional Neural Networks (DCNNs) [42] showed extremely promising performances suggesting that the deeper the network was, the better the prediction would have been. This ended up being true up to a certain point: after a determined depth the performance of the networks stops improving [46].

### 4.1.1 Residual Neural Networks

The issue that we just introduced is surprisingly not linked to overfitting since it has been shown [22] that the training error degrades along with the test error. Very deep networks can even lead to the paradoxical event of the network becoming unable to reproduce an identity function.

The problem we are facing is called *vanishing gradients*: as the network gets deeper, the number of derivatives needed for the backpropagation algorithm to work grows getting to the point where the gradients become smaller and smaller and finally drop to 0.

A possible solution for this was proposed in [22] . In practice the main idea is to

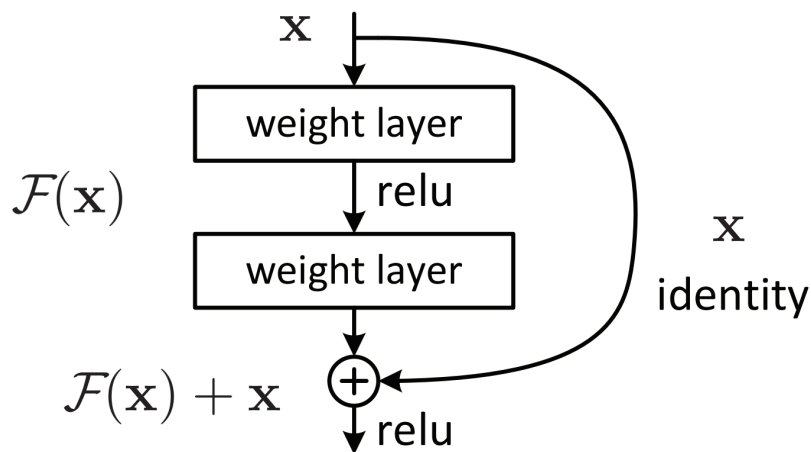


Figure 4.5: Example of a residual layer [22]

couple every layer with some skip connections whose only job is send a copy of the



input information of the layer directly to the input of the next one. In figure 4.5 we can see a residual layer;  $\mathcal{F}(\mathbf{x})$  represents the usual non linear transformation of the input  $\mathbf{x}$ , and in parallel we see the skip connection which simply propagates the input as it is; the two components are added together to provide the input to the following layer.

As a consequence it is now possible to train much deeper networks (hundreds of layers), without the fear of the gradients going to 0 since we created a “highway” that avoids the creation of vanishing gradients.

**HSCNN-R** The HSCNN-R network is a residual neural network proposed by Xiong et al. [8] during the NTIRE 2018 challenge [7]. The structure was adapted from the Res-Net architecture and can be seen in figure 4.6.

The network is composed of an initial convolutional layer that takes care of the

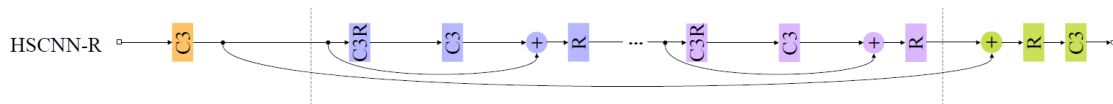


Figure 4.6: The HSCNN-R network [8]

upsampling (the network takes in input an image with three channels and returns one with 31), which is then followed by a set of residual blocks. All the filters of the network are of size 3x3 and the size of the intermediate layers is kept constant by padding. Finally, since spectrum reconstruction is a (very challenging) regression task, no batch normalization layers were introduced to avoid any loss of information.

Starting from the initial structure of Xiong, we made some small adjustments to aid the training. As for the spectrum of an image we only expect positive values smaller than 1, we added a ReLU after the last output layer to make sure to get only positive outputs and changed the initialization so that the range of starting values was closer to the desired one:  $[0, 1]$ . These two alterations, while not changing the long term behaviour of the network gave it a better starting point and provided a somewhat faster and more regular training.

## 4.2 Datasets

For all our trainings we made use of the dataset employed for the NTIRE 2018 challenge [7], which is an extension of a previous dataset by Arad and Ben-Shahar [5] and is the biggest one up to date. We focused in particular on the so called “clean track” of the challenge, which counts for 201 images out of the 256 total ones, all with 31 bands from 400 *nm* to 700 *nm*.

Almost all of the images are outdoor and in figure 4.7 we can see a few samples.

As we can see the colours are not exactly natural, so we decided to focus on the



**Figure 4.7:** *Clean track* images from the NTIRE challenge [7]

hyperspectral ones and from those compute the respective sRGB. Since the sensitivity function of the camera was not provided, we first scaled the images in the  $[0, 1]$  range, and then followed the procedure described in section 3.1 for the conversion to the sRGB domain. In particular the conversion to the  $XYZ$  colour space was done using the CIE1931 colour matching functions and in order to realistically portray the illumination for outdoor scenes we chose the D65 illuminant, which is a standard choice for that instance.

In figure 4.8 we can see the final result of the procedure with a clear improvement over the baseline in figure 4.7. For the fine tuning attempts on a smaller dataset we chose to use the Harvard dataset [27] as it showed similar characteristics to the previous one.

The dataset is composed of 77 images 39 outdoor and the remaining indoor. In practice we focused only on the outdoor images and discarded the others so that we could get a fairer comparison with the other dataset. All the images of this dataset have



**Figure 4.8:** Clean track images from the NTIRE challenge converted from hyperspectral values[7]

31 bands, which are a bit shifted in comparison to ICVL, going from 420  $nm$  to 720  $nm$

In this case the Camera Sensitivity Function (CSS) was provided with the images and was used in the pre-processing before the conversion to the sRGB domain. After applying the CSS another step was needed to deal with outliers a major issue as can be seen in figure 4.9; in said image we plotted the histogram of intensities over all images after a naive mapping to the  $[0, 1]$  interval (division by the overall maximum).

Applying the conversion starting from such low values would give us some very dark sRGB images; for this reason we decided to clip all values bigger than 0.3 to that threshold and remap all intensities to  $[0, 1]$  as shown in figure 4.10.

The pre-processed HS values were then converted to sRGB in the same way as done for the NTIRE dataset and in figure 4.11 we can see a few samples of the end result.

As a last remark, it is important to notice that the range of values of the hyperspectral images of the NTIRE dataset is not exactly  $[0, 1]$  but  $[0.0427, 1]$ ; this is needed as it would have otherwise lead to a division by 0 when computing the main metric for the challenge (which we will thoroughly describe in section 5). For this reason the HS information of the Harvard dataset was mapped to the same interval.

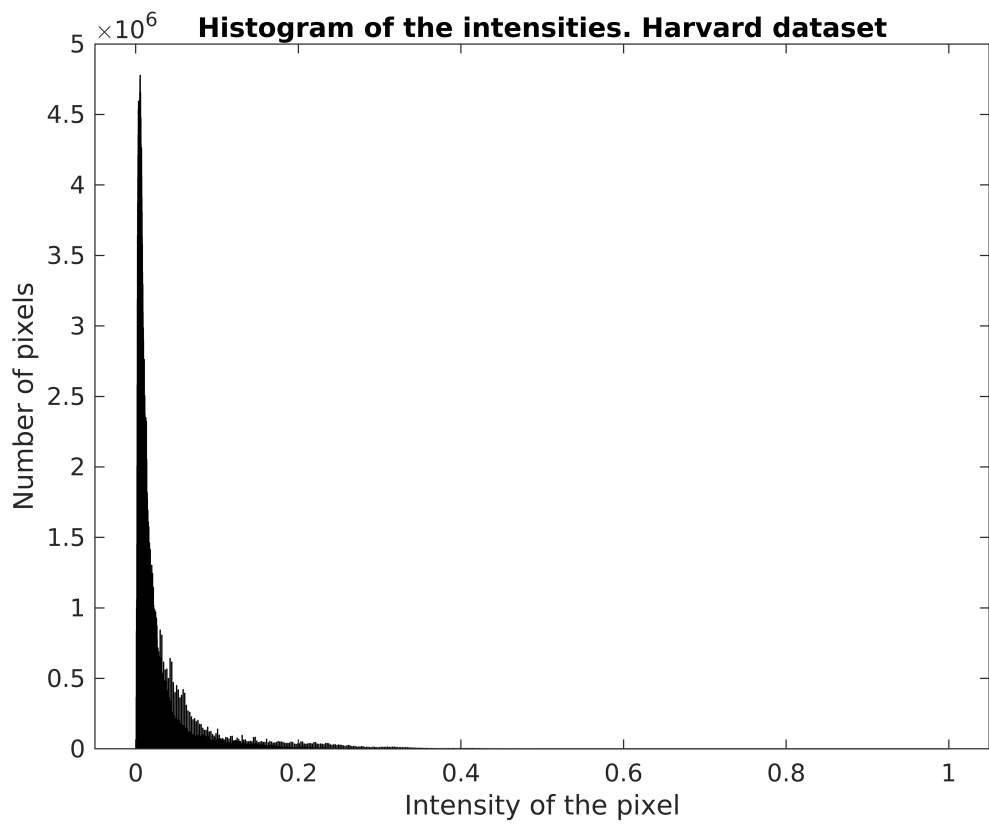


Figure 4.9: Mapping to the  $[0, 1]$  interval with outliers

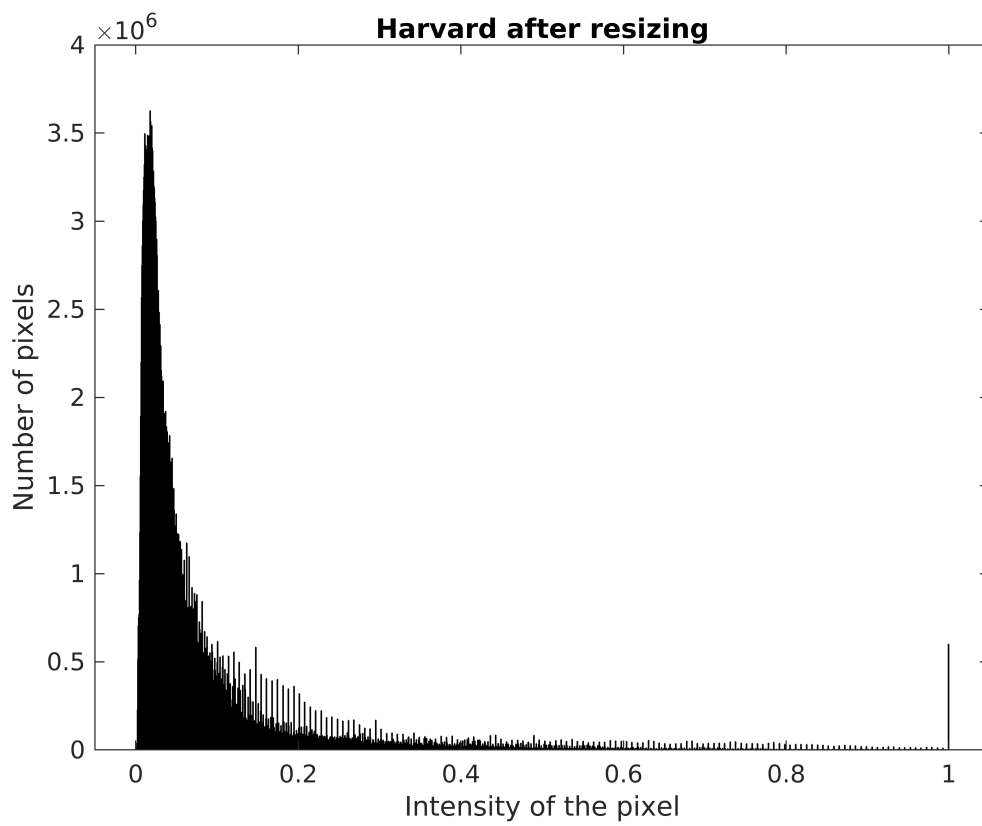


Figure 4.10: Mapping to the  $[0, 1]$  interval without outliers



Figure 4.11: sRGB images from the Harvard dataset



# 5

## Setup and experimental results

In this section we will review the framework and the hardware used for the trainings, then proceed in describing the error metrics employed for the evaluation and finally show in detail the performance of the various approaches introduced in chapter 3

### 5.1 Experimental setup

The HSCNN-R network from Shi et al [8] was implemented using the code provided by the authors, with the few modifications introduced in the previous chapter (ReLU and weight initialization). All the code was written in Python 2, using version 1.0.1 of Pytorch. The trainings have been run over four GTX 1080Ti and required around two days to reach convergence.

For all of our attempts we employed early stopping, basing our choice on the hyperspectral loss if any HS data was used, or on the RGB one otherwise.

The k-means algorithm was implemented from scratch in Pytorch to exploit the parallelization capabilities of GPUs, with a considerable 3x speed-up over the scipy version. Finally, the recovery of hyperspectral information via closest neighbour search was mainly performed offline to avoid slowing down the training; this was possible as given the patch and centroids in output of k-means the algorithm gave a deterministic mapping. In practice however this gave a clear edge only when using 10000 centroids,

while in the 1000 and 100 cases it was possible to perform the computations online without a noticeable time loss.

## 5.2 Error metrics

All the trainings have been done using the Mean Relative Absolute Error (MRAE) as the major metric to compare HS images. Its expression is given as follows:

$$MRAE = \frac{1}{N} \sum_{i=1}^N \frac{|pred_i - GT_i|}{GT_i} \quad (5.1)$$

where  $N$  is the number of pixels we are considering,  $pred_i$  is the prediction of our model for pixel  $i$  and  $GT_i$  is its real hyperspectral value.

The perk of using the MRAE as an error metric when compared to others such as Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE) is due to the different intensities of the channels of a spectrum. In practice according to the channel of the spectrum that we are considering, we can see some recurrent patterns with some bands having consistently a higher intensity than others. If we do not keep this into account, channels with higher values will be given more importance by metrics such as MAE or RMSE while lower intensities channels will be disregarded. The MRAE makes instead sure that the error of a channel gets computed with respect to the intensity of the ground truth, leading to an unbiased comparison.

Along with the MRAE values, we also report the performances according to two other metrics: the RMSE and the Relative Root Mean Squared Error (RRMSE).

$$RMSE = \sqrt{\frac{1}{N} \sum_i (pred_i - GT_i)^2} \quad RRMSE = \sqrt{\frac{1}{N} \sum_i \frac{(pred_i - GT_i)^2}{GT_i^2}} \quad (5.2)$$

As a last remark, for the physical model loss we decided to employ the MAE since some of the values ended up being equal to zero after the HS to sRGB conversion, which would lead to a division by 0. Since it would become quite crafty to solve this issue every round of training (the conversion is done online) and the impact of the different channel intensities is very limited in the RGB case, we simply decided to adopt another metric.



## 5.3 Results and discussion

For the attempts over the NTIRE dataset we performed our trainings over three different splits of the 201 images. For the first one we randomly chose 181 images for training, 10 for validation and 10 for testing. The second one was acquired in a similar way, but leaving more room for validation and test set with 30 images each. The third one instead was the same split used in [47] with 99 images for training, 8 for validation and 93 for testing (one of the images of the training set seemed to be missing).

One of our first attempts concerned the RGB only scenario. Even with the modifications on the network and the addition of the TVM loss, our results were far from optimal getting as far as two orders of magnitude from the fully supervised case. This of course was hardly a surprise since without hyperspectral information it was challenging to lead the network in the right direction; anyway, as previously mentioned in section 3.2, it proved to be possible to use the network trained without supervision as a starting point for other trainings. The reasons for that is that the pre-trained model, while still being far away from what we want, has an output such as that shown in figure 5.1, which presents some considerably useful properties:

1. Knowledge of the physical model; the conversion of the HS values gives us the input sRGB
2. Good range of values; our output is in the same order of magnitude as that of the ground truth one
3. Correlation in the spectral direction; the output is very smooth

The use of this pre-trained model managed to give a boost to all other approaches, helping some trainings that would otherwise have shown strong difficulties with their convergence, and overall reducing the time needed to reach the end.

In order to assess the performance of the alternated training approach described in section 3.3.1, we decided to train two fully supervised models, one over 181 images (first of the three mentioned splits) and the second over 40. Then we trained a third model keeping the same 40 images with the ground truth and using the physical model loss for the others. In the end the training was performed on cycles of 10 rounds,

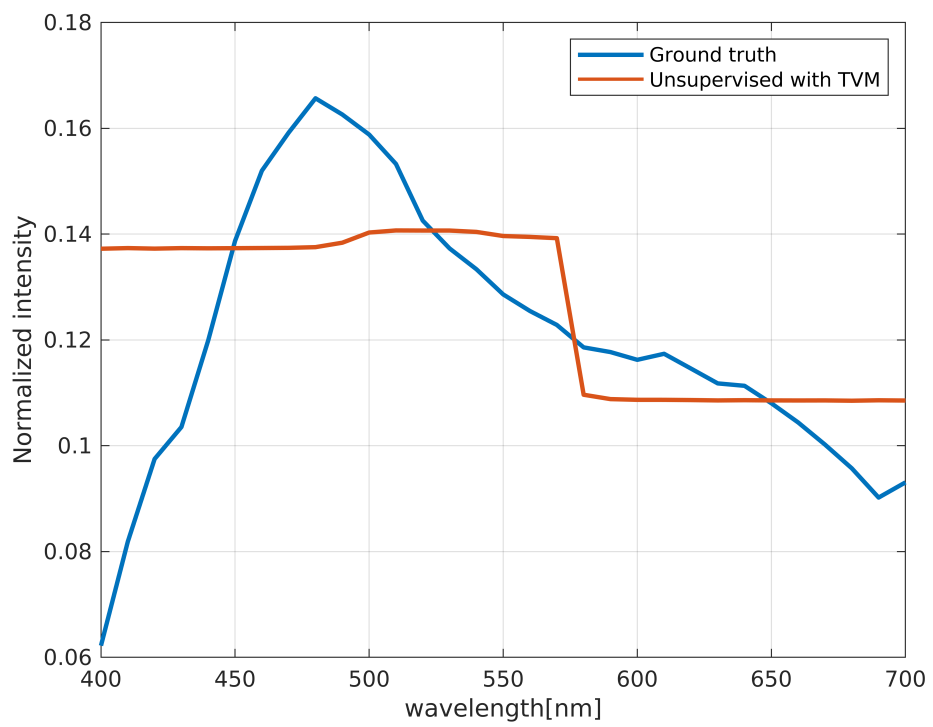


Figure 5.1: Output of the modified network after unsupervised training with TVM constraint

where in the first round we used the HS loss and followed with 9 other rounds of RGB loss on examples of the bigger set.

The performance of the alternated approach can be seen in figure 5.2, where it appears evident how the physical model improves the overall results while using the same amount of hyperspectral images.

A qualitative example of the result can be seen in figure 5.3, where we are comparing the prediction of the three models: full supervision over 181 images, alternated training and full supervision over 40 images.

We then proceeded to assess the viability of the pixel loss technique described in section 3.3.2. Since we wanted to make the scenario realistic, we studied the approach with three different amount of ground truth pixels: 100, 1000 and 10000. In a real case scenario it is reasonable to see the 100 and 10000 cases as a good representation of the minimum and maximum amount of ground truth we would be able to collect

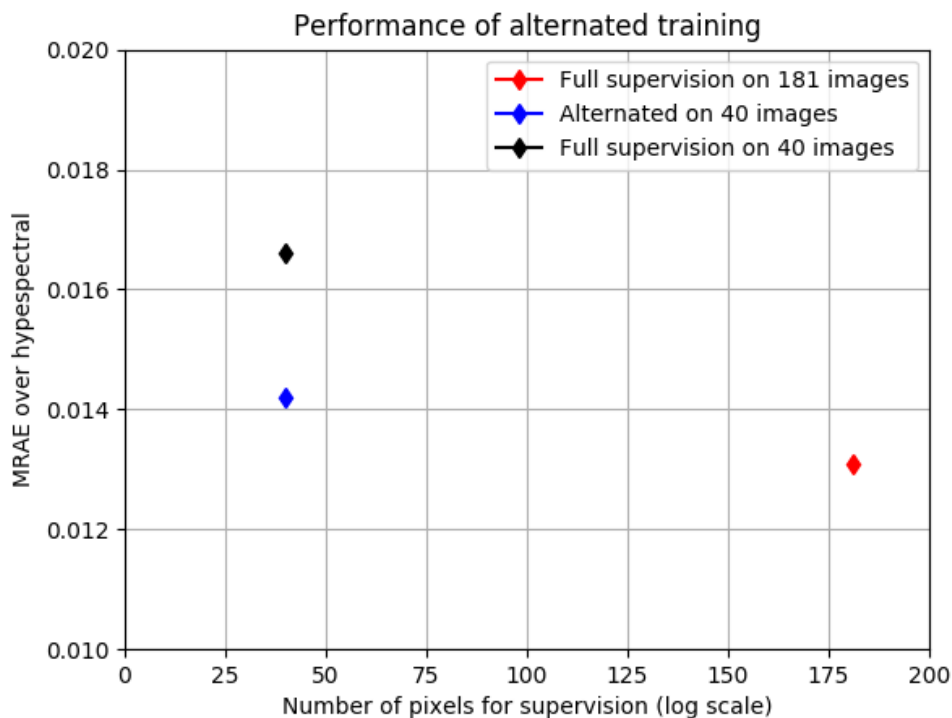
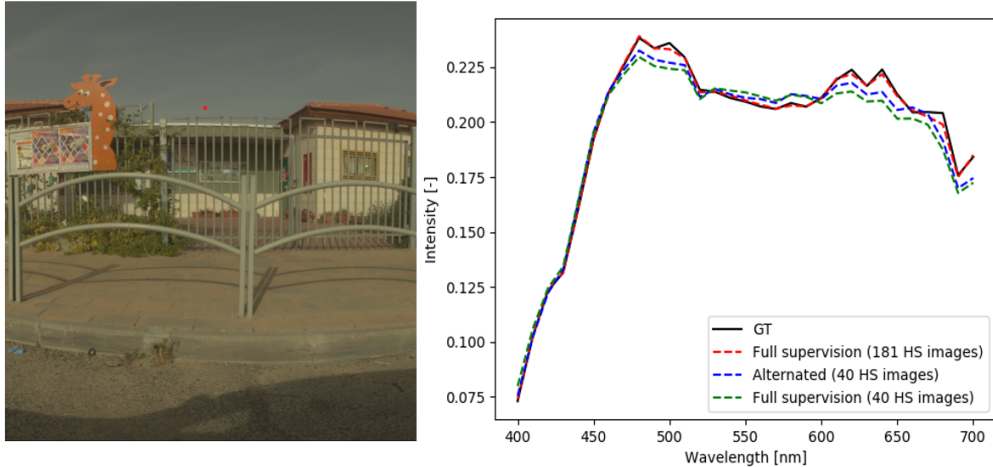


Figure 5.2: MRAE improvement with physical model



**Figure 5.3:** Prediction of the three models for the same pixel

with a spectrometer. Relating to the ICVL dataset, we would be respectively acquiring one pixel and 100 pixels per image.

We then ran the k-means algorithm for each of the three cases (100,1000,10000 centroids) for a total of 20 iterations using the k-means++ initialization [35]. The algorithm was initially employed over a total of 10 randomly chosen images from the training set.

Using the output centroids we then built offline an approximated training set, which we then used following a similar approach as the one employed for the alternated loss (one round HS loss, 4 rounds RGB loss). Finally, to make sure that the information we built was actually useful for the network, we compared the MRAE after the training to a direct use of the ground truth information over the training set. What we did was to apply the same procedure shown in figure 3.9 directly on the test set which practically consists in a lookup table.

We can see in figure 5.4 how using the network shows a definite edge over a direct reconstruction, justifying the use of our approximated dataset together with DNN architectures.

Later we tried to run the k-means algorithm over the whole training set to see how the number of images in input affected the performance. Again we could see a clear improvement (Fig. 5.5), confirming the intuitive idea that clustering on a greater amount of images would output centroids with better generalization capabilities.

Finally we tested how the starting colour space influenced the final result and ran

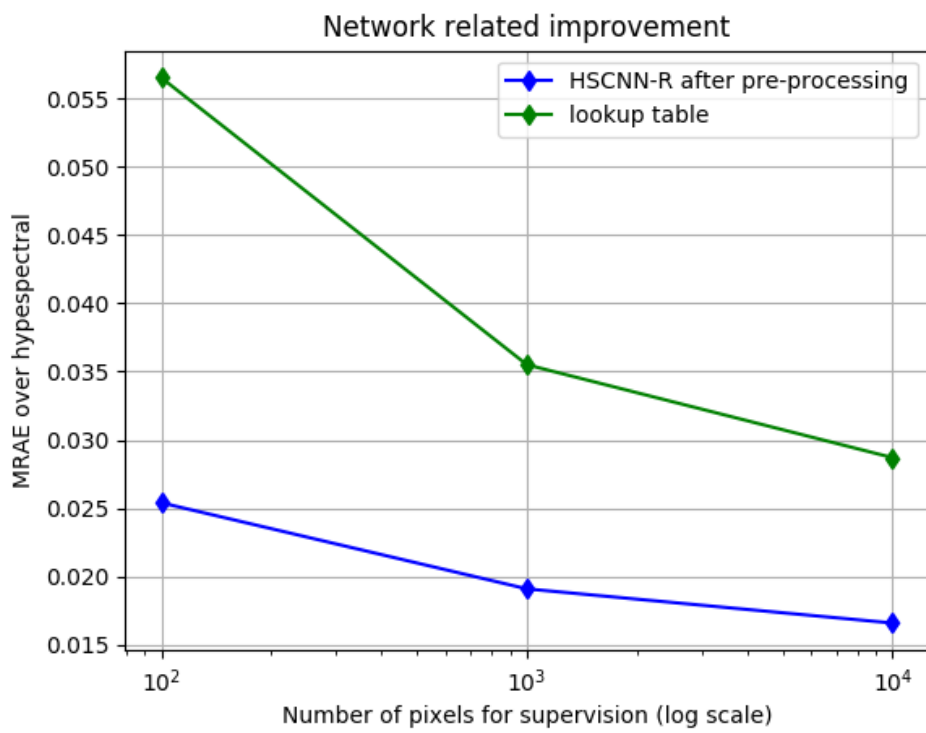


Figure 5.4: Improvement in MRAE due to the network

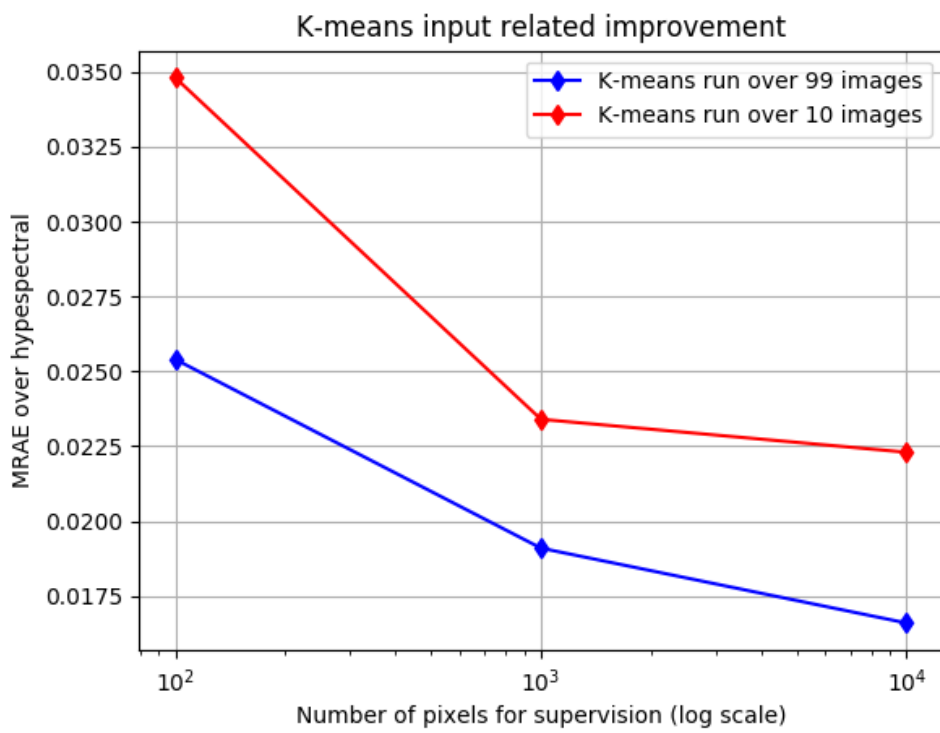


Figure 5.5: Improvement in MRAE due to the number of images in input to k-means

the algorithm after converting the HS images to the Lab colour space. The idea was based on the fact that in the sRGB space similar distances do not imply similar differences in colour, which is instead the case for Lab.

The experiment has been ran only on the case of 10000 pixels and gave the same performance to that of sRGB which suggests that the result does not strongly depend on the specific colour space where the clustering is performed. It is important to notice however that the difference could be more noticeable when the amount of pixel is small (e.g. 100).

The results of the different approaches over the ICVL dataset are summarized in figure 5.6. In the plot we are comparing the MRAE with the number of pixels used as supervision (for the techniques based on images too) reported in logarithmic scale.

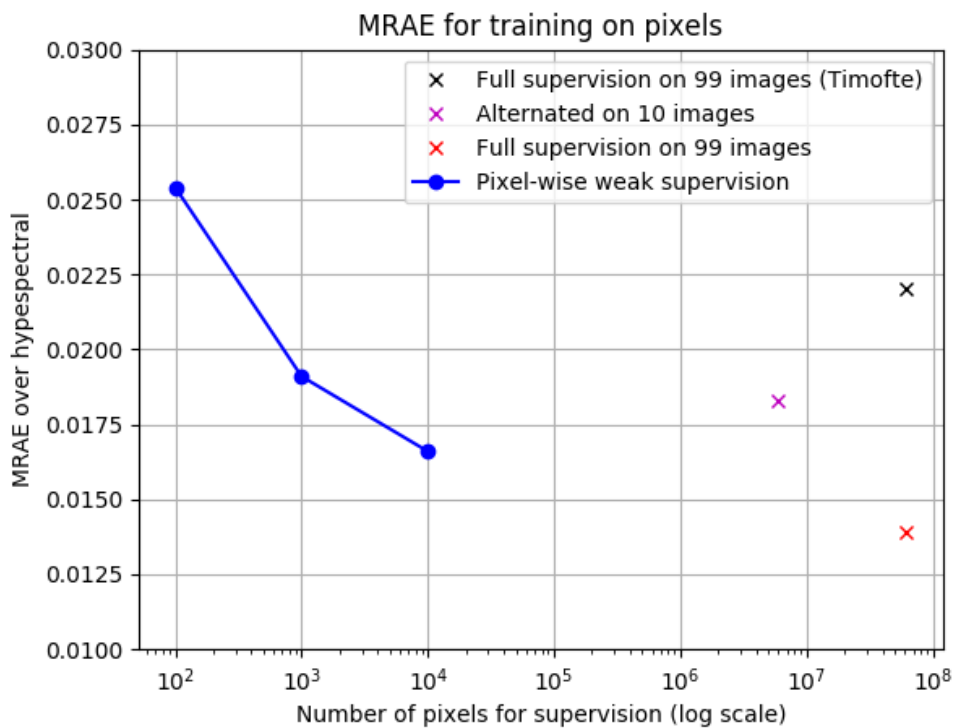


Figure 5.6: Summary of the results for the 99-8-93 split

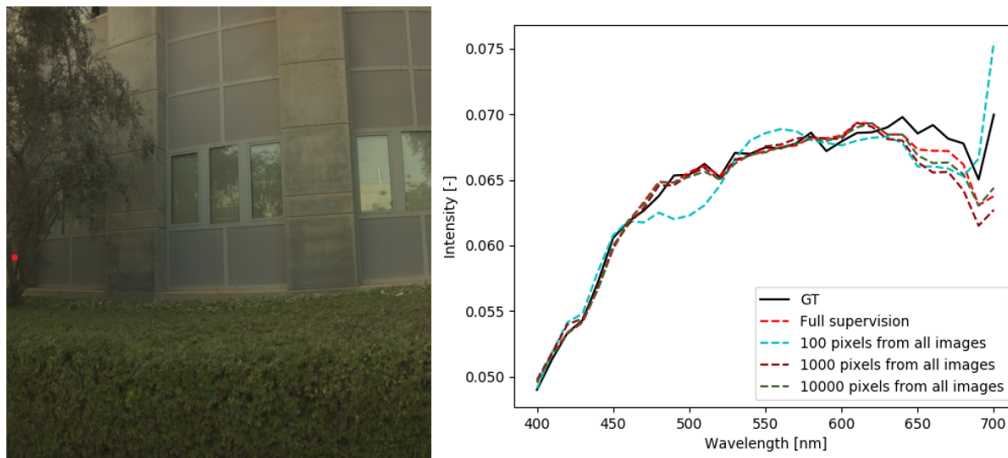
In the plot we also show the performance of another fully supervised architecture devised by Timofte et al. [47].

From said image it is clear how the proposed approaches still show competitive per-

performances to fully supervised methods while lowering by several orders of magnitude the amount of supervision needed for training. In particular, it is very interesting to notice how the pixel loss approach over 10000 centroids outperforms even the alternated training which has almost 1000 times the amount of supervision. The reason for this is that while 10 images provide several millions of pixels, they have a lot of redundant information and find it hard to represent spectra which are not in their space. Our centroids instead are chosen (based on the RGB values) over the whole set of training images and have therefore strong generalization capabilities. In conclusion, little but well chosen information is much more useful than a lot of redundant one.

In figure 5.7 we can see a qualitative comparison of the pixel loss approaches and the fully supervised one for the same pixel. In particular we can see how the approximated dataset in the 100 pixels case can lead to predictions very far from the ground truth. The network is not able to perform a good reconstruction as the few centroids are leaving too many holes in the hyperspectral manifold.

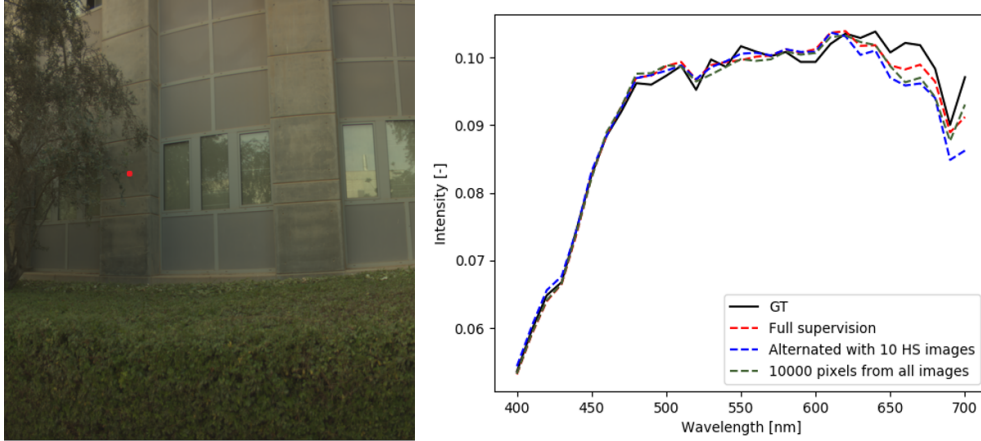
Finally in figure 5.8 we show the predictions of the overall best approaches (pixel loss



**Figure 5.7:** Prediction of the pixel loss models w.r.t. full supervision for the same pixel

10000 and alternated training), compared to the full supervised case. In the image it is possible to see how the pixel loss model, even with a much smaller amount of supervision, is able to perform a better reconstruction with respect to the alternated training one.





**Figure 5.8:** Prediction of the pixel loss 10000 model w.r.t. full supervision and alternated training

Training techniques	MRAE	RMSE	RRMSE
Full supervision	0.0139	0.0048	0.0273
Alternated	<b>0.0183</b>	0.0061	0.0351
K-means 10000	<b>0.0166</b>	0.0055	0.0309
K-means 1000	0.0191	0.0066	0.0357
K-means 100	0.0254	0.0072	0.0448
Timofte [47]	0.0224	0.0059	0.0343

**Table 5.1:** Experimental results for the benchmark split (99-8-93)

The numerical results of said approaches are shown in tables 5.1 and 5.2 for respectively the 99-8-93 and 141-30-30 splits. In particular in the first case we also report the performance of a reimplementation of the approach of Timofte [47], which is one of the best in the literature: top 5 on ICVL and state of the art over other datasets. Together with the MRAE we also report the other two metrics introduced at the beginning of this chapter: RMSE and RRMSE.

It is worth noticing that the performances on the 99-8-93 split are a bit worse with respect to the other, but this is reasonable as the test set is three times as big and it is therefore harder to represent it well.

As an additional remark, we can see that the RMSE and RRMSE metrics closely follow the behaviour of the MRAE.

Training techniques	MRAE	RMSE	RRMSE
Full supervision 141 images	0.0124	0.0043	0.0259
Alternated	0.0184	0.0061	0.0386
K-means 10000	0.0148	0.0051	0.0309
K-means 1000	0.0173	0.0057	0.0347
K-means 100	0.0256	0.0091	0.0479

**Table 5.2:** Experimental results for the 141-30-30

Finally as explained in section 3.4, we tested how the pixel loss approach could be employed in a fine tuning scenario. In order to do this, we split the 39 outdoor images in three sets: 29 for training, 5 for validation and 5 for testing. We then started from a model trained with full supervision over the ICVL dataset and then fine tuned it with information from the Harvard dataset.

The training dataset used for the fine tuning was acquired following the same steps as for the pixel loss: first we ran the k-means algorithm over the RGB values of our 29 training images and then built an approximated training set looking for the closest neighbour in the ground truth set for each of the RGB training pixels. The fine tuning was then performed using this set as ground truth alternating one round of HS loss with two of RGB loss. Finally, the performance was compared with a supervised model trained from scratch using all the 29 images as shown in figure 5.9.

As it can be seen in the plot both the fine tuning on 1000 and 10000 pixels have a performance which is in practice the same as that of full supervision. Moreover, we also trained one model with the approximated dataset built with 10000 centroids, starting the training from scratch and as we can see its performance is clearly worse than the fine tuned model, proving that fine-tuning improves our results. In table 5.3 we can find all the numerical results related to the previous plot.

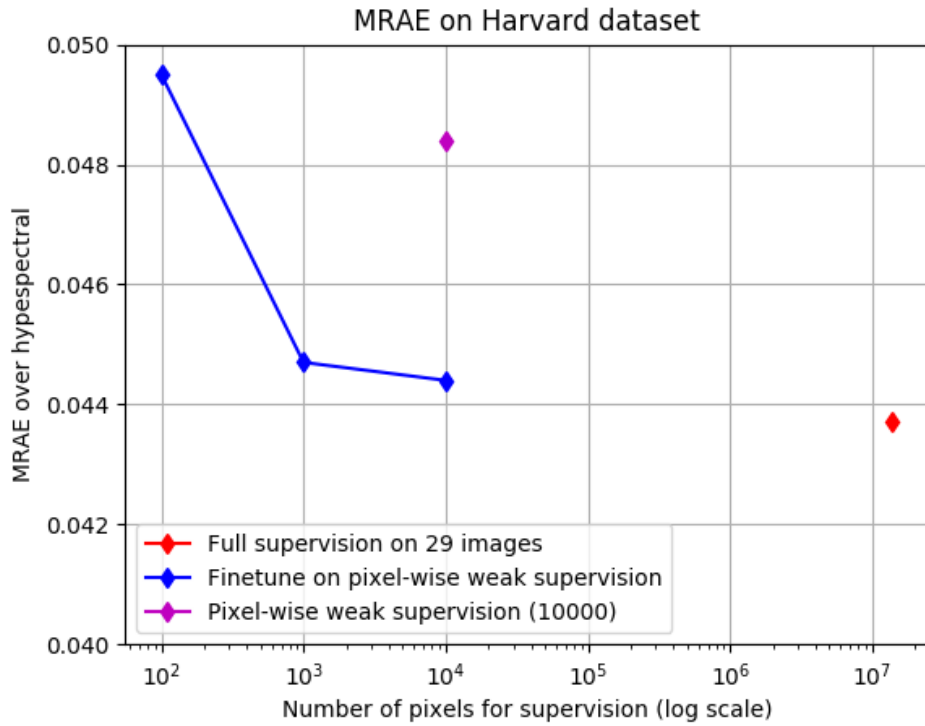


Figure 5.9: Summary of results for fine tuning and full supervision on Harvard dataset

Training techniques	MRAE	RMSE	RRMSE
Full supervision	0.0437	0.0118	0.0801
Fine tune K-means 10000	<b>0.0444</b>	0.0119	0.0799
Fine tune K-means 1000	0.0447	0.0119	0.0805
Fine tune K-means 100	0.0495	0.0129	0.0850
K-means 10000	0.0484	0.0124	0.0841

Table 5.3: Experimental results for the Harvard dataset



# 6

## Conclusion

In the hereby presented thesis we studied the topic of spectrum reconstruction and proposed several strategies devised to help with the training when the amount of hyperspectral data at our disposal is limited.

More precisely, we tried to consider some specific frameworks that could be linked to real case scenarios with limited means to acquire hyperspectral data.

One of the key tools we have introduced is the use of the physical model for hyperspectral images, which adds a crucial constraint and significantly eases the training of the models.

We then studied a framework where our training images are mostly RGB with just a few hyperspectral ones and showed an alternated training approach based on the physical model, which showed a very competitive performance with the fully supervised case.

Then we considered the configuration where our images were exclusively RGB with only a few pixels of hyperspectral ground truth. We proposed a clustering technique based uniquely on the RGB data to choose the most useful hyperspectral pixels as ground truth and then proceeded in building an approximated training dataset which was fed to the network. This approach showed some very promising performances and even proved to be able to beat the alternated method if the amount of HS pixels was high enough. Its overall performance was surprisingly good keeping into account

that the amount of hyperspectral data required is several orders of magnitude smaller than the fully supervised case.

Finally we demonstrated the potential of the same technique in a fine tuning scenario, adapting a model pre-trained on a bigger dataset with just the aid of a few hyperspectral pixels from a smaller one. With said approach it was possible to reach almost the same performance of a model trained from scratch with full supervision.

A possible extension of our work would be the study of different clusterization algorithms, as for example mean shift, to see which one suits best the problem we are facing. Another very interesting point would be considering the process according to which the approximated dataset from the set of hyperspectral pixels is created. In all of our attempts we just chose the closest centroid but it seems sensible not to focus only on it but also on the information provided by other close by points. On the same line it is also worth considering making said process fuzzy as given a set of centroids we would be always getting the same dataset. Such approach would be a very good step towards hyperspectral data augmentation which still remains a mostly unexplored field.

# Listing of figures

2.1	Use of hyperspectral imaging to monitor the growth of crops . . . . .	7
2.2	Hyperspectral remote sensing . . . . .	8
2.3	Mixture of rice and plastic . . . . .	8
2.4	Typical hyperspectral imaging approaches . . . . .	9
2.5	Spectrum reconstruction . . . . .	11
2.6	The Tiramisù network . . . . .	13
3.1	Summary of the devised approaches . . . . .	16
3.2	Structure of the training pipeline . . . . .	18
3.3	sRGB and Adobe RGB colour spaces . . . . .	19
3.4	CIE 1931 colour matching functions . . . . .	20
3.5	Training pipeline in an unsupervised scenario . . . . .	23
3.6	Spectrum before and after applying TVM compared to ground truth	24
3.7	Training pipeline using a limited amount of images as supervision . .	25
3.8	RGB image with sparse hyperspectral ground truth . . . . .	26
3.9	Creation of fake HS dataset with a set of ground truth pixels . . . . .	27
3.10	Training pipeline using only a few pixels as supervision . . . . .	28
3.11	k-means algorithm on 3 clusters . . . . .	30
3.12	Fine tuning . . . . .	33
3.13	Fine tuning on the Harvard dataset . . . . .	33
4.1	Structure of a generic CNN . . . . .	36
4.2	Comparison between fully connected networks and CNNs . . . . .	36
4.3	Parameter sharing in a CNN . . . . .	37
4.4	Example of max pooling with a 2x2 filter . . . . .	37
4.5	Example of a residual layer . . . . .	38
4.6	The HSCNN-R network . . . . .	39
4.7	Clean track images from the NTIRE challenge . . . . .	40

4.8	Clean track images from the NTIRE challenge converted from hyper-spectral values . . . . .	41
4.9	Mapping to the $[0, 1]$ interval with outliers . . . . .	42
4.10	Mapping to the $[0, 1]$ interval without outliers . . . . .	43
4.11	sRGB images from the Harvard dataset . . . . .	43
5.1	Output of the modified network after unsupervised training with TVM constraint . . . . .	48
5.2	MRAE improvement with physical model . . . . .	49
5.3	Prediction of the three models for the same pixel . . . . .	50
5.4	Improvement in MRAE due to the network . . . . .	51
5.5	Improvement in MRAE due to the number of images in input to k-means	52
5.6	Summary of the results for the 99-8-93 split . . . . .	53
5.7	Prediction of the pixel loss models w.r.t. full supervision for the same pixel . . . . .	54
5.8	Prediction of the pixel loss 10000 model w.r.t. full supervision and alternated training . . . . .	55
5.9	Summary of results for fine tuning and full supervision on Harvard dataset . . . . .	57



## Listing of tables

5.1	Experimental results for the benchmark split (99-8-93) . . . . .	55
5.2	Experimental results for the 141-30-30 . . . . .	56
5.3	Experimental results for the Harvard dataset . . . . .	57

## References

- [1] J. M. Bioucas-Dias, A. Plaza, G. Camps-Valls, P. Scheunders, N. Nasrabadi, and J. Chanussot, “Hyperspectral remote sensing data analysis and future challenges,” *IEEE Geoscience and Remote Sensing Magazine*, vol. 1, no. 2, pp. 6–36, June 2013.
- [2] B. F. Guolan Lu, “Medical hyperspectral imaging: a review,” *Journal of Biomedical Optics*, vol. 19, no. 1, pp. 1 – 24 – 24, 2014. [Online]. Available: <https://doi.org/10.1117/1.JBO.19.1.010901>
- [3] S. Matteoli, M. Diani, and G. Corsini, “A tutorial overview of anomaly detection in hyperspectral images,” *IEEE Aerospace and Electronic Systems Magazine*, vol. 25, no. 7, pp. 5–28, July 2010.
- [4] M. W. K. Nathan A. Hagen, “Review of snapshot spectral imaging technologies,” *Optical Engineering*, vol. 52, no. 9, pp. 1 – 23 – 23, 2013. [Online]. Available: <https://doi.org/10.1117/1.OE.52.9.090901>
- [5] B. Arad and O. Ben-Shahar, “Sparse recovery of hyperspectral signal from natural rgb images,” in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 19–34.
- [6] S. Galliani, C. Lanaras, D. Marmanis, E. Baltsavias, and K. Schindler, “Learned spectral super-resolution,” *CoRR*, vol. abs/1703.09470, 2017. [Online]. Available: <http://arxiv.org/abs/1703.09470>
- [7] B. Arad, O. Ben-Shahar, and R. Timofte, “Ntire 2018 challenge on spectral reconstruction from rgb images,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

- [8] Z. Shi, C. Chen, Z. Xiong, D. Liu, and F. Wu, “Hscnn+: Advanced cnn-based hyperspectral recovery from rgb images,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.
- [9] E. J. Bjerrum, M. Glahder, and T. Skov, “Data augmentation of spectral data for convolutional neural network (CNN) based deep chemometrics,” *CoRR*, vol. abs/1710.01927, 2017. [Online]. Available: <http://arxiv.org/abs/1710.01927>
- [10] L. Dale, A. Thewis, C. Boudry, I. Rotar, P. Dardenne, V. Baeten, and J. Fernández Pierna, “Hyperspectral imaging applications in agriculture and agro-food product quality and safety control: A review,” *Applied Spectroscopy Reviews*, vol. 48, p. 142, 01 2013.
- [11] D. Haboudane, J. R. Miller, E. Pattey, P. J. Zarco-Tejada, and I. B. Strachan, “Hyperspectral vegetation indices and novel algorithms for predicting green lai of crop canopies: Modeling and validation in the context of precision agriculture,” *Remote Sensing of Environment*, vol. 90, no. 3, pp. 337 – 352, 2004. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0034425704000264>
- [12] A. Gowen, C. O’Donnell, P. Cullen, G. Downey, and J. Frias, “Hyperspectral imaging – an emerging process analytical tool for food quality and safety control,” *Trends in Food Science and Technology*, vol. 18, no. 12, pp. 590 – 598, 2007. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0924224407002026>
- [13] “Food fraud detection with hyperspectral imaging.” [Online]. Available: <https://www.specim.fi/food-fraud-detection-with-hyperspectral-imaging/>
- [14] Y. Wang, N. P. Reder, S. Kang, A. Glaser, and J. Liu, “Multiplexed optical imaging of tumor-directed nanoparticles: A review of imaging systems and approaches,” *Nanotheranostics*, vol. 1, pp. 369–388, 08 2017.
- [15] J. M. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Q. Du, P. Gader, and J. Chanussot, “Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 5, no. 2, pp. 354–379, April 2012.

- [16] C. Lanaras, E. Baltsavias, and K. Schindler, “Hyperspectral super-resolution by coupled spectral unmixing,” in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.
- [17] D. H. Foster, K. Amano, S. M. C. Nascimento, and M. J. Foster, “Frequency of metamerism in natural scenes,” *J. Opt. Soc. Am. A*, vol. 23, no. 10, pp. 2359–2372, Oct 2006. [Online]. Available: <http://josaa.osa.org/abstract.cfm?URI=josaa-23-10-2359>
- [18] J. P. S. Parkkinen, J. Hallikainen, and T. Jaaskelainen, “Characteristic spectra of munsell colors,” *J. Opt. Soc. Am. A*, vol. 6, no. 2, pp. 318–322, Feb 1989. [Online]. Available: <http://josaa.osa.org/abstract.cfm?URI=josaa-6-2-318>
- [19] J. Y. Hardeberg, “On the spectral dimensionality of object colours,” *Conference on Colour in Graphics, Imaging, and Vision*, vol. 2002, no. 1, pp. 480–485, 2002. [Online]. Available: <https://www.ingentaconnect.com/content/ist/cgiv/2002/00002002/00000001/art00101>
- [20] A. Chakrabarti and T. Zickler, “Statistics of real-world hyperspectral images,” in *CVPR 2011*, June 2011, pp. 193–200.
- [21] J. Aeschbacher, J. Wu, and R. Timofte, “In defense of shallow learned spectral reconstruction from rgb images,” in *The IEEE International Conference on Computer Vision (ICCV) Workshops*, Oct 2017.
- [22] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [23] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham: Springer International Publishing, 2015, pp. 234–241.
- [24] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani,

- M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680. [Online]. Available: <http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf>
- [25] J. Kim, J. Kwon Lee, and K. Mu Lee, “Accurate image super-resolution using very deep convolutional networks,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [26] F. Yasuma, T. Mitsunaga, D. Iso, and S. K. Nayar, “Generalized assorted pixel camera: Postcapture control of resolution, dynamic range, and spectrum,” *IEEE Transactions on Image Processing*, vol. 19, no. 9, pp. 2241–2253, Sep. 2010.
- [27] A. Chakrabarti and T. Zickler, “Statistics of Real-World Hyperspectral Images,” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 193–200.
- [28] R. M. H. Nguyen, D. K. Prasad, and M. S. Brown, “Training-based spectral reconstruction from a single rgb image,” in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 186–201.
- [29] J. Deng, W. Dong, R. Socher, L. jia Li, K. Li, and L. Fei-fei, “Imagenet: A large-scale hierarchical image database,” in *In CVPR*, 2009.
- [30] Z. Li and N. Snavely, “Learning intrinsic image decomposition from watching the world,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- [31] H. Chen, J. Gu, O. Gallo, M. Liu, A. Veeraraghavan, and J. Kautz, “Reblur2deblur: Deblurring videos via self-supervised learning,” in *2018 IEEE International Conference on Computational Photography (ICCP)*, May 2018, pp. 1–9.
- [32] Y. Gandelsman, A. Shocher, and M. Irani, ““double-dip”: Unsupervised image decomposition via coupled deep-image-priors,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

- [33] A. Chambolle, “An algorithm for total variation minimization and applications,” *Journal of Mathematical Imaging and Vision*, vol. 20, no. 1, pp. 89–97, Jan 2004. [Online]. Available: <https://doi.org/10.1023/B:JMIV.0000011325.36760.1e>
- [34] R. Kawakami, Y. Matsushita, J. Wright, M. Ben-Ezra, Y. Tai, and K. Ikeuchi, “High-resolution hyperspectral imaging via matrix factorization,” in *CVPR 2011*, June 2011, pp. 2329–2336.
- [35] D. Arthur and S. Vassilvitskii, “K-means++: The advantages of careful seeding,” vol. 8, 01 2007, pp. 1027–1035.
- [36] K. Fukunaga and L. Hostetler, “The estimation of the gradient of a density function, with applications in pattern recognition,” *IEEE Transactions on Information Theory*, vol. 21, no. 1, pp. 32–40, January 1975.
- [37] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?” in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 3320–3328. [Online]. Available: <http://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks.pdf>
- [38] “A comprehensive guide to convolutional neural networks — the eli5 way.” [Online]. Available: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>
- [39] “Convolutional neural networks (cnn).” [Online]. Available: [http://i-systems.github.io/HSE545/machine%20learning%20all/Workshop/180208\\_COSEIK/06\\_CNN.html](http://i-systems.github.io/HSE545/machine%20learning%20all/Workshop/180208_COSEIK/06_CNN.html)
- [40] F. Abtahi and Z. Zhu, “Deep learning models for multimodal sensing and processing,” 12 2015.
- [41] “Cs231n convolutional neural networks for visual recognition.” [Online]. Available: <http://cs231n.github.io/convolutional-networks/>

- [42] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>
- [43] F. Milletari, N. Navab, and S. Ahmadi, “V-net: Fully convolutional neural networks for volumetric medical image segmentation,” in *2016 Fourth International Conference on 3D Vision (3DV)*, Oct 2016, pp. 565–571.
- [44] R. Girshick, “Fast r-cnn,” in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.
- [45] F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.
- [46] K. He and J. Sun, “Convolutional neural networks at constrained time cost,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.
- [47] Y. B. Can and R. Timofte, “An efficient CNN for spectral reconstruction from RGB images,” *CoRR*, vol. abs/1804.04647, 2018. [Online]. Available: <http://arxiv.org/abs/1804.04647>