POLITECNICO DI MILANO

Facoltà di Ingegneria Industriale

Corso di Laurea in
Space Engineering

DEEP LEARNING SEMANTIC SEGMENTATION FOR VISION-BASED HAZARD DETECTION

Relatore: Prof. Pierluigi Di Lizia

Co-relatore: Prof. Roberto Furfaro

Tesi di Laurea di:

Luca Ghilardi Matr. 875344

Anno Accademico 2018 - 2019.

Politecnico di Milano:
www.polimi.it

Scuola di Ingegneria Industriale e dell'Informazione:
www.ingindinf.polimi.it

University of Arizona:
https://www.arizona.edu/

Ringraziamenti

Questa tesi è stata sviluppata presso l'Università dell' Arizona a Tucson. Primi fra tutti ringrazio i miei genitori e mio fratello Daniele del grande supporto che mi hanno sempre offerto per questa esperienza.

Ringrazio di cuore anche il prof. Di Lizia per la grande occasione che ci ha offerto e per avermi seguito con interesse per le fasi più critiche di questa tesi.

Durante l'esperienza a Tucson vorrei prima di tutto ringraziare il prof. Furfaro e la sua famiglia che ci hanno sempre accolto a casa loro. Il professore sia dal punto di vista professionale che personale ci ha sempre seguito e offerto preziosi consigli durante tutta la nostra permanenza negli USA.

Ringrazio anche il mio collega di corso Emanuele e i ragazzi della stanza 36: Enrico, Mario, Andrea e Shain, con cui ho condiviso questa avventura. La loro amicizia mi ha convinto a voler svolgere il dottorato a Tucson.

Ringrazio inoltre i miei amici di Lucca che mi hanno fatto sempre sentire l'Italia vicina, non lasciandomi mai solo, e il mio amico Alessio a Milano per i suoi preziosi consigli per ogni cosa. Una menzione speciale va al mio eterno compagno di università Lorenzo, con cui ho condiviso tutte le giornate di studio per gli esami sia della triennale che della magistrale, il supporto che ci siamo dati l'un l'altro ha permesso di arrivare dove siamo oggi.

Milano, Luglio 2019 L. G.

Acknowledgements

This thesis was developed at the University of Arizona in Tucson. First of all, I thank my parents and my brother Daniele for the great support they have always offered me for this experience.

I also thank prof. Di Lizia for the great opportunity offered and for having followed me with interest for the most critical phases of this thesis.

During my experience in Tucson, I would like first of all to thank prof. Furfaro and his family who have always welcomed us to their home. The professor from both a professional and personal point of view has always followed us and offered valuable advice throughout our stay in the USA.

I also thank my colleague Emanuele and the boys from room 36: Enrico, Mario, Andrea, and Shain, with whom I shared this adventure. Their friendship convinced me to want to come back in Tucson for a doctorate.

I also thank my friends from Lucca, they have always made me feel close to Italy, never leaving me alone, and my friend Alessio in Milan for his valuable advice for everything. A special mention goes to my eternal university friend Lorenzo, with whom I shared all the study days for both the three-year and the master's exams, the support we gave each other allowed us to get where we are today.

Milano, Luglio 2019 L. G.

I do not fear computers. I fear the lack of them.
-Isaac Asimov-

Contents

List of Figures

[1] Convolutional layer

[2]Digital Terrrain Model

List of Tables

Sommario

Negli ultimi anni la computer-vision con il deep learning è sempre più utilizzata in applicazioni dove sono necessarie assoluta precisione e affidabilità. Specialmente nell'ambito medico e della guida autonoma. Essa si basa sul far capire al computer il contenuto di un immagine, frammentando i processi che noi facciamo involontariamente, come riconoscere la ditribuzione dei colori oppure dei contorni e le forme. Tra le varie tecniche di computer-vision la segmentazione semantica con il deep learning è tra le più studiate e promettenti. Essa permette di classificare un'immagine pixel per pixel, diventando lo stato dell' arte in molte applicazioni.

In questa tesi proveremo la deep learning segmentazione semantica per identificare caratteristiche territoriali pericolose come massi, crateri e aree scoscese. I casi studiati sono due: nel primo verrà investigato l'utilizzo di reti neurali su un corpo celeste di piccole dimensioni, come un asteroide. In particolare l'asteroide Itokawa, le cui immagini ci sono state fornite dalla sonda Hayabusa 1 lanciata nel 2014. Nel secondo caso, la rete neurale verrà validata sull'area di atterraggio lunare dell' Apollo 16. La rete verrà messa alla prova sul riconoscimento di crateri, e aree a inclinazioni non sicure per l'atterraggio. In questo caso le immagini saranno simulate partendo da modelli del terreno.
Parole chiave: PoliMi, Tesi, Deep Learning, Semantic Segmentation, Hazard Detection.

Abstract

In recent years, computer-vision with deep learning is often used in applications that require absolute precision and reliability. In particular, in medical applications and autonomous driving. The aim of computer vision is making the computer understand the content of an image, dividing the processes, that we do involuntarily, such as characterize the distribution of colors or edges and shapes. Among the various computer-vision techniques, the semantic sectioning with deep learning is among the most studied and promising. It allows classifying a pixel-by-pixel image, making it, state of the art in many applications.

In this thesis, we will investigate the use of deep learning semantic segmentation to identify hazardous territorial features such as boulders, craters, and steep areas. Two study cases are addressed: in the first one, the neural network will be tested on a small celestial body like an asteroid. In particular, the asteroid Itokawa whose images were provided by the Hayabusa 1 probe launched in 2014. In the second case, the neural network will be tested on the Apollo 16 lunar landing area. Here the network will be tested on the recognition of craters and areas with unsafe slopes for landing standards. In this case, the images will be simulated, starting from terrain models.
Keywords: PoliMi, Master Thesis, Deep Learning, Semantic Segmentation, Hazard Detection.

Chapter 1

Introduction

1.1 Work aim and Challenges

NASA and private companies like Blue Origin are aiming to expand their business and exploration by sending new probes and even humans on the moon surface. With this renewed interest in returning to the Moon, the delicate problem of the lunar landing is more important than ever. In the Apollo 11 mission, Neil Armstrong had to change with a manual maneuver the landing site due to an unexpected crater. That maneuver was incredibly risky and required much skill.

Today the engineers try to develop an efficient and robust algorithm for the autonomous planetary landing, but usually, those algorithms lack accuracy or computational efficiency for an On-Board use. On top of that, other machine learning classifiers (perceptrons) have been developed to tackle the problem of the safe landing hazard detection, but generally, they need the user to import the valuable features [Lunghi et al., 2016]. Our approach aims to automate the features import by applying the deep learning techniques of semantic segmentation to identify hazardous terrain characteristics.

The drawback is that the semantic segmentation is still an unresolved problem of computer vision. Even the state-of-the-art architecture do not reach human-level performance. The human vision can recognize objects in many different conditions of view-points, scale, illumination, or when they are translated or rotated. In the landing problem, the human stereoscopic vision is tricked by the large distances. We can distinguish a "safe" site from a "hazardous" one if obstacle like boulders and craters are in the area. However, for planetary landings, the requirements of a safe site rely

also on a maximum slope and roughness, which can not be easily identified by the human eyes.

Our objectives are to develop neural networks for visual-based hazard detection for two different scenarios:

Itokawa Safe Landing Site

On small celestial bodies, the detection of suitable landing sites is a challenging problem due to the many light conditions, and the amount of obstacle. In this thesis, we propose a deep learning approach to semantically segment the Itokawa asteroid using only the real images provided by the Hayabusa 1 mission.

Moon Safe Landing Site

As we said features as slope and roughness can not be easily detected only with camera images on big celestial bodies. We propose a network architecture that is capable of identifying those terrain features by analyzing the same scene in different illumination conditions.

1.2 State of the Art

On 3 February 1966, the Luna 9 was the first spacecraft to achieve a soft-land on the Moon. At that time, the landing site was previously chosen before launch. The only instrument used in the landing was the radar-altimeter to trigger the jettison of the side modules, the inflation of the airbags and the firing of the retro rockets. Back in our days, the most recent successful lunar landing was performed by the Chang'e 4 Chinese lander. It is not easy today to find information about the Chang'e 4 hazard detection system, but we can study its predecessor, the Chang'e 3 spacecraft.

The Chang'e 3 was the first robotic spacecraft that adopted autonomous hazard detection and avoidance technology but also is the spacecraft that achieved the highest landing precision on the Moon to date. The first Chang'e 3's hazard detection design included only the visual-based hazard detector, but it failed to pass validation tests. The final state of the system had a combination of two-dimensional (2-D) optical gray-image-based coarse hazard avoidance and three-dimensional (3-D) laser-elevation-image-based [Wei et al., 2018; S. Li et al., 2016].

The vision-based hazard detection was used to perform a coarse hazard detection from $\sim$ 2.4km to $\sim$ 100m. Coarse avoidance aimed to exclude hazardous large-scale obstacles (craters and boulders larger than 1 meter) and to provide potential safe landing regions for the following precise avoidance. At around 100 meters the lander performed a hovering maneuver, which is needed to scan the surrounding area with a LiDAR[1] instrument. Next, the on-board guidance-navigation and control (GNC) system detected craters larger than 20 cm and slopes steeper than 8 deg (10 m baseline) and determined the nearest safe landing site.

We already introduced that in this thesis, we are also going to asses the hazard detection problem on asteroids. Until now just a few missions tried to land payloads on such small bodies, and almost all of those just performed a touch and go. The Hayabusa 1 mission managed to land for a small amount of time on the asteroid Itokawa, and soon the OSIRIS-REx spacecraft will perform a touch and go to collect a sample from the Bennu asteroid. The Hayabusa 2 mission deployed two rovers on the Ryugu surface, but they were dropped from a safe altitude. All these spacecrafts mount a LiDAR to measure the distance from the objective and to build a 3-D map of the celestial body, coupled with an autonomous optical navigation system [Hashimoto et al., 2010; Lorenz et al., 2017]. The main aim of both instruments is to recover the spacecraft state.

The OSIRIS-REx autonomous optical navigation system is called Natural Feature Tracking (NFT) and was developed as a backup for the LiDAR [Lorenz et al., 2017]. It is designed to perform autonomous high-precision orbit determination by tracking the location of known features on the surface of small bodies such as Bennu. NFT predicts which features it expects to see in the image, renders the expected appearance of these features and finally matches these predictions to the actual image data. The locations of these matches are used to update the on-board knowledge of where the navigation camera is and how it is oriented. It is worth noticing that also the OSIRIS-REx laser altimeter (OLA) is used to build a shape model, with a resolution below 3 cm/pixel.

For lunar landers and asteroid landers, the vision-based systems are used as a backup for the LiDARs since their performances are strictly related to the illumination conditions of the objective. However, for vast distances from the target, they are still a reliable option to detect terrain features, thanks to the high resolution of the navigation cameras. Furthermore, the

[1] Light Detection and Ranging

LiDARs resolution drops drastically with the distance from the reflective surface.

In this work, we are going to present a deep-learning algorithm for vision-based hazard detection, instead of the current state of the art k-means clustering algorithms. The k-means clustering, which is a type of unsupervised learning (Section 2), aims to partition the observations into k-clusters. The observations are the features that characterize the hazards like the shape, the edges, and the shadows. In order to collect these features, filters are manually applied to the images. The deep learning network instead is based on supervised learning, and from the training set, it learns which features are essential for the inference. In this work, we do not directly compare our network with a k-clustering algorithm, but nowadays, for classification problems, supervised learning is the standard choice due to its superior performances.

1.3 Thesis Structure

This work is structured as follows:

Chapter 2 In this chapter, we build the basis for understanding the later chapters. The basis of the machine learning and deep learning will be explained with a focus on the computer vision problem.

Chapter 3 In this chapter, we define the architectures adopted in the thesis. In addition, we list the hyperparameters adopted for the training.

Chapter 4 In this chapter, we describe how we build the dataset used for the Itokawa and lunar problem, and we show how the labeling was done.

Chapter 5 In this chapter, we present the metrics adopted to evaluate the models, and we show the results for each model. The chapter is completed with a brief runtime analysis of the models.

Chapter 6 Finally, in the last chapter, we summarize our findings and give an outlook of the possible future development.

Chapter 2

Machine Learning

In this chapter there is a brief introduction to machine learning and its sub-category: the deep learning. The definition "without being explicitly programmed" is often attributed to Arthur Samuel, who coined the term "machine learning" in 1959. The term means the study of algorithms and statistical model which are able to perform specific tasks without any specific instruction given by the user. Instead, it relies on the patterns and inference based on sample data known as *training data*. How this mathematical model is built can be divided in three categories:

- **Supervised learning**: The training data are provided and correctly classified by the user. The principle behind this method is that the network should recognize patterns and features that link the raw training data to the respective labels.

- **Unsupervised learning**: No labels are given to the learning algorithm, therefore the system tries to learn without a teacher. In this case, the network tries to find structures in the inputs.

- **Reinforcement learning**: The learning system observes the environment through certain actions, and from this actions gets rewards or penalties in return and improves and learn the best strategy which is called policy.

Depending on the required output of the task, further categorizations of supervised learning algorithms can be made. Typically, we differentiate between *classification* and *regression* tasks. The goal of classification is to assign a category label, whereas in regression the targets are specified via real numbers.

2.1 Logistic Regression

The goal of the classification problem is to take an input vector $\mathbf{x}$ and to assign it to one of the K discrete classes ζ_k for $k = 1, \ldots, K$ [M.Bishop, 2007]. The input space is divided in *decision regions* delimited by *decision boundaries* or *decision surfaces*. In this thesis we consider the classifier known as *logistic regression*, and we will explain how to perform inference with this model.

The "classic" application of logistic regression model is binary classification. However, we can also use variants of logistic to tackle multi-class classification problems, e.g., using the One-vs-All or One-vs-One approaches. It is used when classes are linearly separable, but as we shall see, many data sets can not be discriminated linearly.

2.1.1 Inference

In the context of machine learning the inference is the task to predict unknown data based on some data samples. In the classification problem it means assigning a category label to a given feature. The features might be images or any other kind of data that can be represented in a feature space.

As we said the logistic regression makes use of hyper-planes as decision boundaries in the feature space to distinguish between classes. In the case of linear classification we can define a linear discriminant function called *score function*.

2.1.2 Score Function

To compute the *score function* our input data are a set of images $\mathbf{x_i} \in \Re^D$ where each element of the vector ($[D \times 1]$) represents the value of each pixel of the image flatted out, and the associated labels $\mathbf{y_i}$. The index $i = 1 \ldots N$ means the number of images in input, while $y_i \in 1 \ldots K$ the number of labels. We will now define the *score function* as $f : \Re^D \rightarrow \Re^K$:

$$\mathbf{f}(\mathbf{x_i}, \mathbf{W}, \mathbf{b}) = \mathbf{W}\mathbf{x_i} + \mathbf{b} \tag{2.1}$$

The matrix $\mathbf{W}$ ($[K \times D]$) and the vector $\mathbf{b}$ ($[K \times 1]$) are the parameters of the function, respectively the weights and the bias vector. The value of the function for a specific class $f_k = \mathbf{W_k}\mathbf{x} + b_k$ represents the point distance from the hyper-plane k [M.Bishop, 2007]. So the decision boundary between

the two classes y_k and y_i is given by $f_k = f_i = 0$ and hence corresponds to a hyper-plane defined by:

$$(\mathbf{W}_k - \mathbf{W}_i)\mathbf{x} + (b_k + b_i) = 0 \tag{2.2}$$

For a more intuitive point of view in the Figure 2.1 we can consider each image as a point in the graph and since we defined the score of each class as a weighted sum of all image pixels, each class score is a linear function over this space, which represents a graphical interpretation of the decision boundaries. Tuning the rows of the parameter $\mathbf{W}$ we can rotate the different lines while modifying the bias $\mathbf{b}$ we translate them.

Unfortunately most of the times the input data can not be linearly classified, to solve this problem we introduce the *basis functions* $\mathbf{\Phi}(\mathbf{x})$.

$$\mathbf{f}(\mathbf{x_i}, \mathbf{W}, \mathbf{b}) = \mathbf{W}\mathbf{\Phi}(\mathbf{x_i}) + \mathbf{b} \tag{2.3}$$

These functions perform a non-linear transformation of the input features into the *feature space*, where the features are linearly separable, (Figure 2.2). In this thesis we use convolutional neural networks (CNN[1]s) to perform a non-linear transformation.

2.1.3 Loss Function

Once the score function is computed, we need a measure of how different the results are from those desired, for this purpose in the literature two main classifiers are used which their own loss function, the *Multiclass Support Vector Machine* (SVM[2]) and the *Softmax classifier*. In this thesis the latter is adopted because of the small performance difference between the two and the easier probabilistic interpretation that we will explain later. The *Softmax classifier* is the generalization of the binary logistic regression in multiple classes, the score function stays unchanged $\mathbf{f}(\mathbf{x_i}, \mathbf{W}, \mathbf{b}) = \mathbf{W}\mathbf{x_i}$, but we now interpret these scores as the un-normalized logaritmic probabilities for each class and then we introduce a `cross-entropy loss` that has the form:

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right) \tag{2.4}$$

where the function $g_i(f) = \frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$ is called *softmax function*. In the notation adopted f_j is the j-th element of the class score vector $\mathbf{f}$, while the term f_{y_i}

[1] Convolutional Neural Network
[2] Support Vector Machine

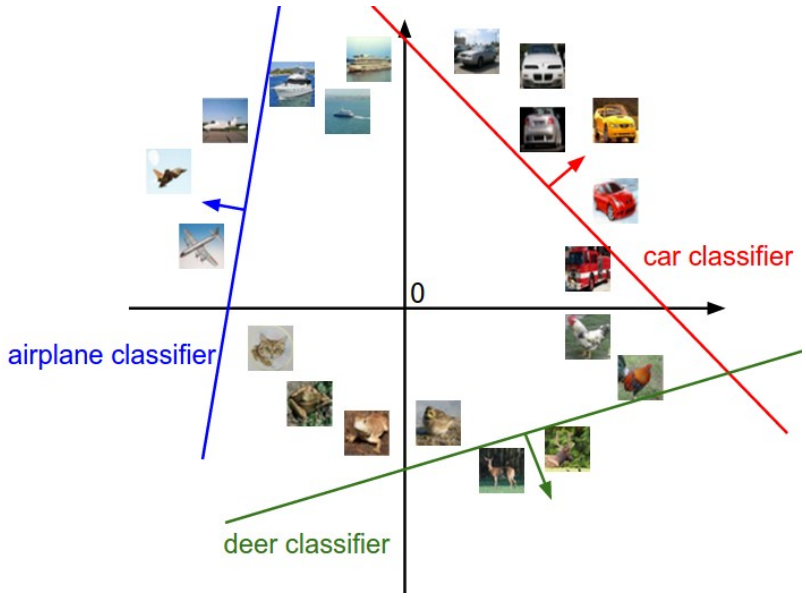


Figure 2.1: Input space representation with decision boundaries of tree classes with some data samples [F.-f. Li et al., 2017].

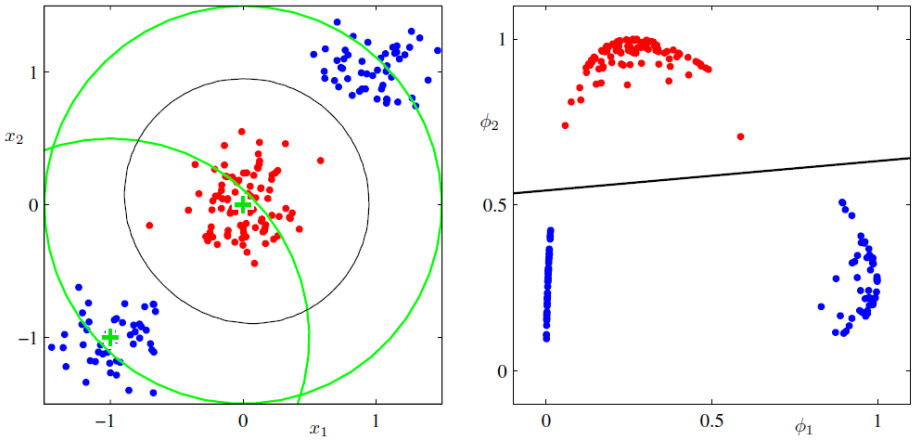


Figure 2.2: On the left plot is represented the input space **x** and data samples from two classes (red and blue). On the right plot shows the same data points but in the feature space after a non-linear trasformation. The linear decision boundary is the black line, which is a curve in the input space [M.Bishop, 2007].

is the score for the correct label. As can be seen, the *softmax function* is a normalized probability assigned to the label y_i; we are therefore minimizing the negative log likelihood of the correct class.

2.1.4 The Optimization Problem

We saw that the loss function measures how good the correct classes are classified, to maximize the performances we need to minimize the loss function $L(\mathbf{W})$ by changing the weights $\mathbf{W}$. To find the best path to reach the minimum, the most common way is the *gradient descent* method, which involves taking steps in the negative direction of the gradient $\nabla L(\mathbf{W})$. Than the models parameters are iteratively update with the following equation:

$$\mathbf{W}^{\tau+1} = \mathbf{W}^{\tau} - \alpha \nabla L(\mathbf{W}^{\tau}) \tag{2.5}$$

Where α represents the step size of the update step. Convergence occurs when the estimated loss function does not decrease anymore.

In hiking analogy [F.-f. Li et al., 2017], this approach roughly corresponds to feeling the slope of the hill below our feet and stepping down the direction that feels steepest. The steps that we take in that direction are called *learning rate*; it is one of the fundamental hyper-parameters for the neural network training. A small learning rate allows us to find the minimum of the function with high precision, but it comes at the cost of the training time, which increases considerably. At the opposite side, an high learning rate allows a faster training but with the risk to "overstep" missing the minimum of the function. This method is called *gradient descent* and can be computationally expensive with large models to compute the loss function for thousands of examples.

This optimization phase is the training of the neural network. Usually, it requires much time, especially for big networks. To save computational power, we divide the training set in batches. Then the parameters update is done computing the loss function and gradients on the batch, instead of the whole training set. In this case, we talk about Stochastic Gradient Descent (SGD[3]), which is a variant of the classic gradient descent method, that computes on a small subset of a random selection of data examples.

In the literature, there is another popular optimization algorithm, the Adaptive Moment Estimation (ADAM[4]), which is used in this thesis. It combines the advantages of two SGD extensions: the root mean square propagation and the adaptive gradient algorithm, and then computes individual adaptive learning rates for different parameters [Kingma and Ba, 2014]. ADAM is known to achieve good results fast.

[3] Stochastic Gradient Descent
[4] Adaptive Moment Estimation

2.1.5 Training Flow

In the previous section 2.1.4 we said that the optimization phase is the training of the network, here we show the work-flow. The objective of the process is to find a set of weights that minimize the loss function, and it is done by an iterative procedure called backpropagation, (Figure 2.3).

Initially, the weights are randomly chosen, then there is what is called *Forward Pass*. The input data batch, in our specific case, a set of images, is cast through the network to the output. The output is compared with the respective labels to compute the loss function.

The *Backward pass* where the weights that have contributed the most to the loss are determined. Lastly, the weights are updated in the negative direction of the loss function gradient.

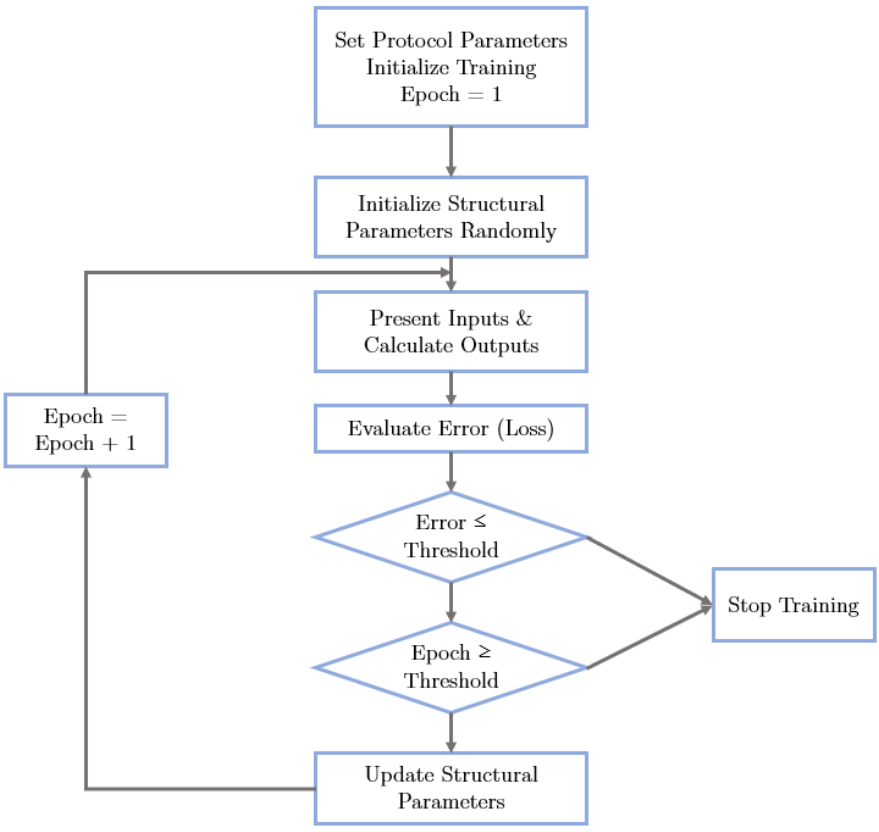


Figure 2.3: Flowchart of the general backpropagation algorithm to update structural parameters [Fan et al., 2017].

2.1.6 Overfitting

Ideally when the loss functions reaches its minimum the network should be at the maximum performances, but it is not always true. The really common problem is the overfitting of the learnable parameter to the training set, especially if it is small and with a small variety of data. When this happens the network behaviour is really good with the training data, but if the test data are a little different the network accuracy drops drastically. This happens because the network does not learn the more general features that bonds the training set and test set, but more specific ones to the training set. To avoid this problem the common approches are early stopping, regularization and data augmentation.

- **Early stopping**: we use a validation dataset to provide an unbiased evaluation of a model fit on the training dataset while tuning the model's weights. Then we stop training when the error on the validation dataset increases, as this is a sign of overfitting to the training dataset.

- **Regularization**: it is a term added to the loss function to avoid extreme parameters value usually by including in the loss function the norm of the weight values, known as L2 regularization [F.-f. Li et al., 2017].

$$L = \frac{1}{N} \sum_i L_i + \lambda ||\mathbf{W}||^2 \tag{2.6}$$

  N is the number of training samples and λ is a weight hyperparameter. In other words, we wish to encode some preference for a certain set of weights $\mathbf{W}$. The most appealing property is that penalizing large weights tends to improve generalization, because it means that no input dimension can have a very large influence on the scores all by itself.

- **Data augmentation**: it is a series of transformations applied to the training set to increase the number of training data. In the case of a data set of images they can be randomly rotated or flipped.

2.1.7 Biological Interpretation

The names *neural networks*, *neurons* are adopted to describe the architecture of the machine learning and its mathematical operations, because they resemble the human brain. In biology, the neurons receive inputs

from their `dendrites`, and then an output signal is transmitted through a `axon`, which is single for each neuron. As is shown in Figure 2.4 we can compare the mathematical flow to compute the activation function, explained shortly after, as the *firing rate* of the neurons, which is their embedded filter of the inputs signals. All these neurons combined create a network, similar to our brain. However, the artificial neural networks are not even close to the computational power of a real brain, as explained in [Brunel et al., 2014]. The real dendrites are not just a single weight; they are a complex non-linear dynamical system.

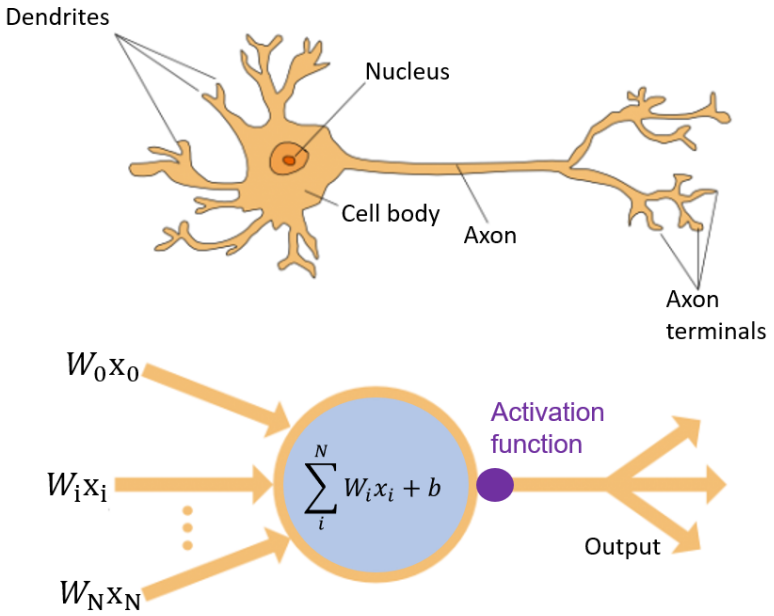


Figure 2.4: A real neuron and a graphical representation of a machine learning neuron [Gupta, 2017].

2.2 Deep Learning

Nowadays many of the devices of our daily life use some deep learning algorithms, such as image/sound recognition [Claesson and Hansson, 2014] features that we can find in the modern smartphones or self-driving cars. Deep learning is a branch of machine learning that tries to mimic the structure and the activity of the brain, it was developed in the 80s, but due to its high demands of computational power, it has only recently been actively used. Today state of the art deep learning models [L.-C. Chen

et al., 2015; He et al., 2015; Krizhevsky et al., 2017] can classify objects for specific tasks equally or better than humans.

These nets are denominated "deep" because their structure is made of many layers, in Figure 2.5 is shown the architecture of one of the first deep learning methods to recognize digits. The layers between the input and the output layer are called *hidden layers*, and their role is to project the input data into the feature space where they can be linearly classified.

The feature can be distinguished in three main categories: low, medium and high level features [Zeiler and Fergus, 2014], Figure 2.6 are shown how the features change during the training. Each category has been taken at different "depths" of the network. The low-level features are extracted by the firsts convolutional layers while the higher level features comes from deeper layers.

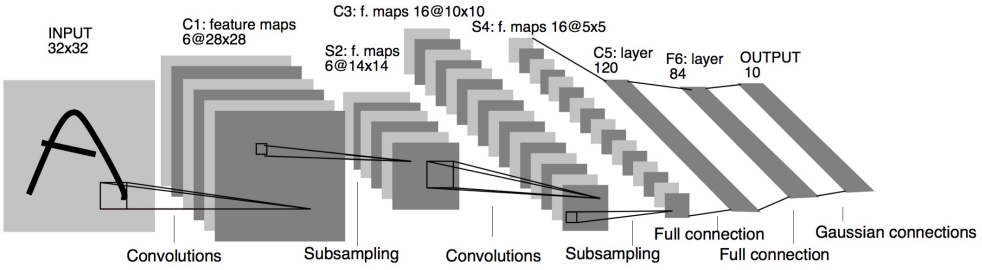
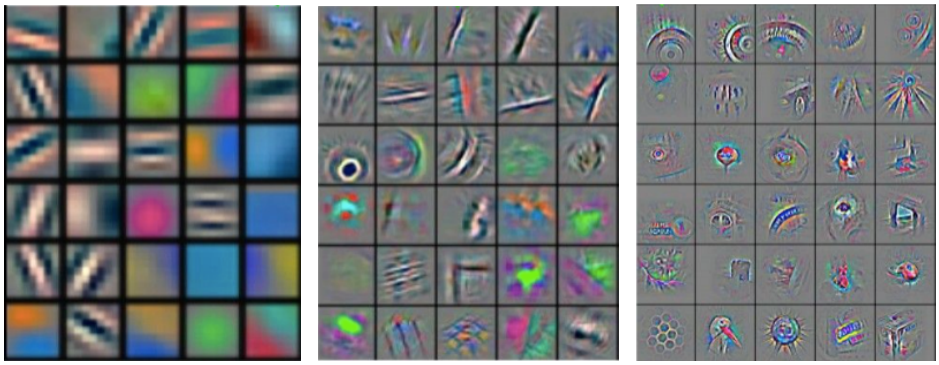


Figure 2.5: One of the first CNNs which was used to perform digits recognition [Lecun et al., 1998]

2.3 Convolutional Neural Network

The CNN is the most common algorithm used in the fields of object recognition. It has many similarities with other deep learning networks. It is based on neurons with learnable weights and biases depending on the loss function [Lecun et al., 1998]. The main reasons why it is so largely spread are:

- Automatic features detection and extraction
- State of the art recognition results
- Flexibility of the models to suit different tasks



(a) Low-level feature **(b)** Mid-level feature **(c)** High-level feature

Figure 2.6: Low (a), mid (b) and high (c) features taken from the network during training [Zeiler and Fergus, 2014]

The last point means that we can build our custom CNN easily from scratch or use what is called *transfer learning*. The transfer learning is a machine learning method, where a model developed for a task is reused as the starting point for a model on a second task. The selected model trained on a large and challenging data-set has to be tuned by modifying the last layers to suit the new task. At this point, the model is subjected to a brief re-training with our data. The advantage is the precision of a large and complex model with the minimum training effort.

If instead, we decide to design a custom model from scratch, the basic architecture is made of blocks. The first one is the `input layer`, which has the role in holding the image for the next layers and in performing a data normalization. Then the convolutional part of the network follows, where the information is extrapolated. Usually, this block is made of a sequence of three organic layers: the `convolutional layers`, the `activation layers` and the `pooling layers`. These layers are labeled as the "hidden" part of the network, and each one of them will be described more deeply. The last block of the chain is the one where the information processed are collected and classified. The main layer here is the `fully connected layer`, which deals with different combinations of features in order to make the final decisions, but other layers can be added depending on the application and performance desired.

2.3.1 Convolutional Layer

The convolutional layers (CONV) are the one responsible for the most computationally intensive part of the network. The input data of a CONV layer can be a 2 or 3-dimensional array of values. The learnable parameters consist of a set of filters, that will extend through the full depth of the input volume. A set of hyper-parameters characterizes the filters:

- the spatial dimensions of the filters: define how many pixels are processed by the filter, also called *receptive field*
- the number of filters: characterize the depth of the output
- the stride: characterize the sliding of the filters.

As we slide the filter over the width and height of the input volume, we will produce a 2-dimensional activation map that gives the responses of that filter at every spatial position. Intuitively, the network will learn filters that activate when they see some visual feature such as an edge of some orientation or a blotch of some color on the first layer, or different kind of patterns on higher layers of the network. From the Figure 2.7 can be seen that there is another hyper-parameter called *zero-padding* (the gray contour of the input volume). This parameter helps us to control the spatial size of the output volume by adding rows and columns of zeros. The output volume size follows this formulation:

$$\frac{W - F + 2P}{S} + 1 \tag{2.7}$$

The symbols W, F, S are respectively the spatial input volume size, the filter receptive field and the stride of the filter. Besides the symbol, P are related to the zero-padding and can be related to the other parameters with $P = (F-1)/2$ for $S = 1$ to ensures that the input volume and output volume will have the same size spatially. For example with an input size of $W = 10 \times 10$, a receptive field of $F = 3 \times 3$ and a stride $S = 1$, without any zero-padding, the output spatial volume would be $(10 - 3)/1 + 1 = 8$. Adding a two column and rows at the beginning, and at the end of the matrix, the spatial size of the output would be the same as the input.

2.3.2 Activation Layer

If we consider a neuron, the activation function has the role in deciding if the score should be "fired" or not, or rather let us say "activated" or not.

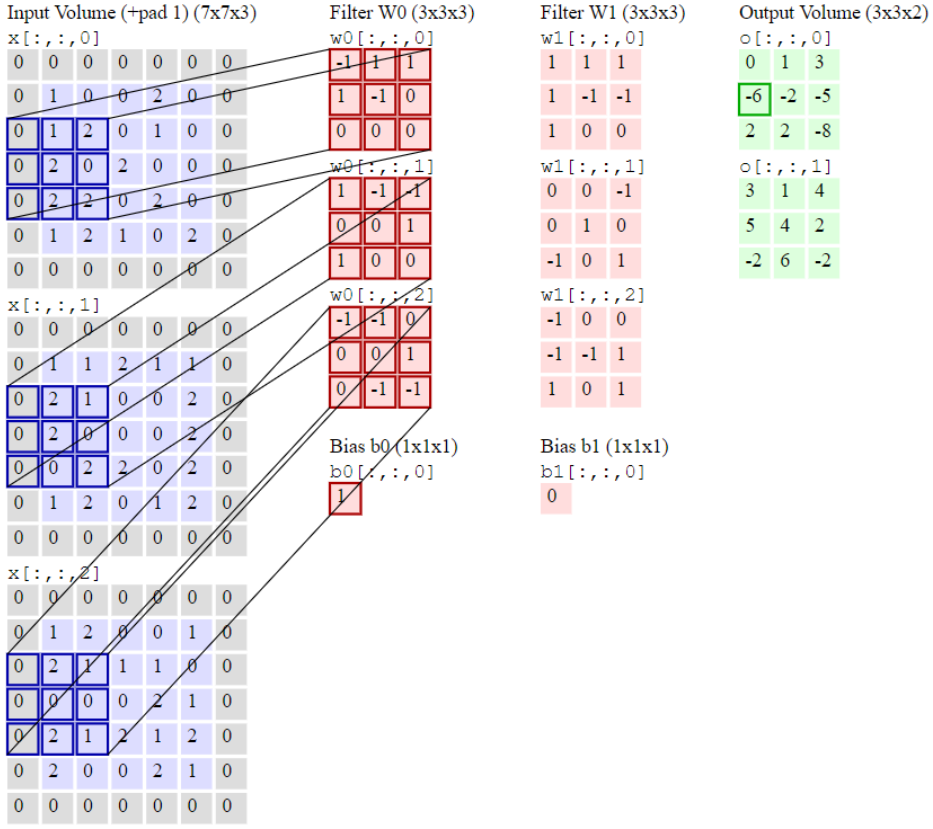


Figure 2.7: In this convolution demo, it is shown the iteration over the output activations (green) and shows that each element is computed by multiplying elementwise the highlighted input (blue)(i.e., an RGB image) with the filter (red), summing it up, and then offsetting the result by the bias. It is worth noticing that the third dimension of the output volume is directly connected to the number of filters [http://cs231n.github.io].

Several activation functions can be used, but all of them must have a non-linear nature. A linear activation function like $f(x) = cx$ looks promising to discriminate the score values, but there are two problems. Because the derivative of the function is a constant, the weight update due to back-propagation does not depends on the change of the input x. The second reason is that linear functions make multiple layers structures completely useless because at the end the final activation layer is nothing more than a linear function of the input of the first layer. We can demonstrate it

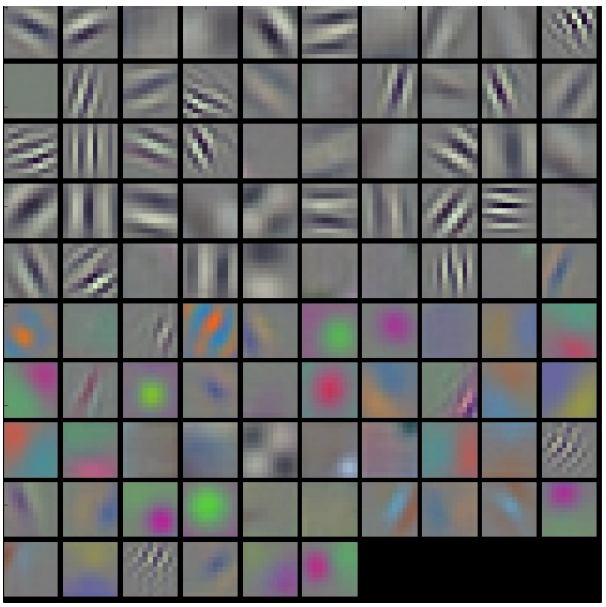


Figure 2.8: Filters activation maps on the first CONV layer of a trained AlexNet [http://cs231n.github.io]. Some filters are tuned for high-frequency grayscale features and the other low-frequency color features.

mathematically:

$$
\begin{aligned}
f(X) &= c_1(W_1X + b_1) \\
g(X) &= c_2(W_2f(X) + b_2) \\
g(X) &= c_2(W_2c_1(W_1X + b_1) + b_2) \\
g(X) &= c_2c_1W_2W_1X + c_2c_1W_2b_1 + c_2b_2
\end{aligned}
$$

collecting the terms: $W = c_1W_2W_1$ and $b = c_1W_2b_1 + b_2$. Now we can rewrite $g(X)$ as $g(X) = c_2(WX + b)$, which is practically the same of the first activation layer. Now that we understood why the functions should be non-linear, we can introduce the most common activation functions. In Figure 2.9 are shown the most common activation functions. In this thesis, we use the `ReLU`, because it can speed up the network training. The gradient computation is straightforward (either 0 or 1 depending on the sign of x, as shown in Figure 2.9). Also, the computational step of a ReLU is easy: any negative elements are set to zero; no exponentials, no

multiplication or division operations.

$$f(x) = max(0, x) \tag{2.8}$$

$$\frac{df(x)}{dx} = \begin{cases} 0, & \text{if } x < 0 \\ 1, & \text{if } x \geq 0 \end{cases} \tag{2.9}$$

Gradients of logistic and hyperbolic tangent networks are smaller than the positive portion of the ReLU. This means that the positive portion is updated more rapidly as training progresses. However, this comes at a cost. The zero gradient on the left-hand side has its problem, called "dead neurons," in which a gradient update sets the incoming values to a ReLU such that the output is always zero; modified ReLU units such as ELU (i.e., Leaky ReLU, or PReLU) can reduce this problem. We still chose the ReLU because, in our tests, the not relevant difference was found.

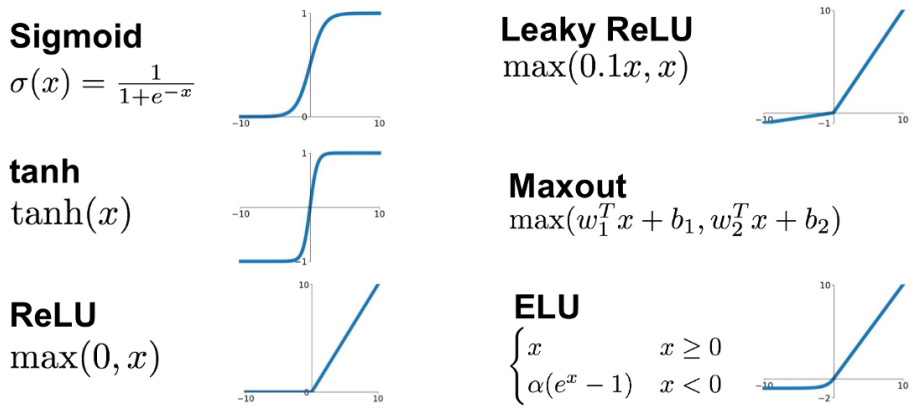


Figure 2.9: Most common activation functions.

2.3.3 Pooling Layer

Pooling layers are common after one or more CONV layers. Their role is to reduce the number of parameters in the network, by reducing the spatial size of the features map, but at the same time maintaining the important features. The resizing can be done with a fixed dimension sliding window on the features map and selecting only the maximum value of the window (Figure 2.10), or with an average of them. It is worth noticing that this operation works at the cost of the spatiality of the information, which could be a problem for pixel level classifications.

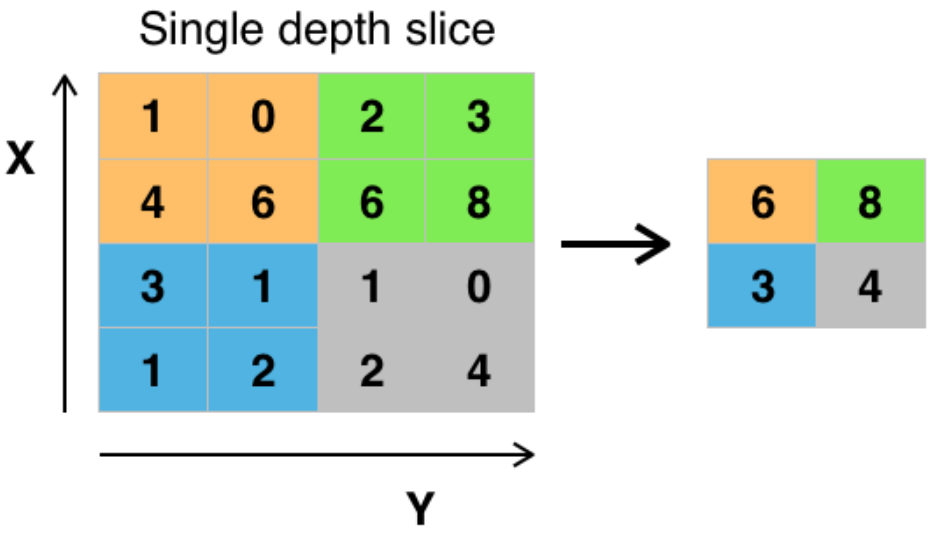


Figure 2.10: Example of max pooling.

2.3.4 Fully Connected Layer

Fully connected layers (FCL) connect every neuron in one layer to every neuron in another layer. Its role is to collect the information of the previous layers in order to evaluate them. The FCL is similar to a CONV layer, the main difference between the two is that in the CONV layer the neurons are connected only to a local region in the input and that many of the neurons in a CONV volume share parameters.

2.4 Semantic Segmentation

Looking at the big picture, semantic segmentation is one of the high-level task that leads towards complete scene understanding. Semantic segmentation with deep learning algorithm associates a label or category with every pixel in an image. Someone can confuse it with an object detection algorithm, but the semantic segmentation networks (SSN[5]) can detect objects of irregular shapes or that span in multiple areas. Instead in the object detection networks, the targets must be inside a bounding box, so the precision of the SSN is overall superior.

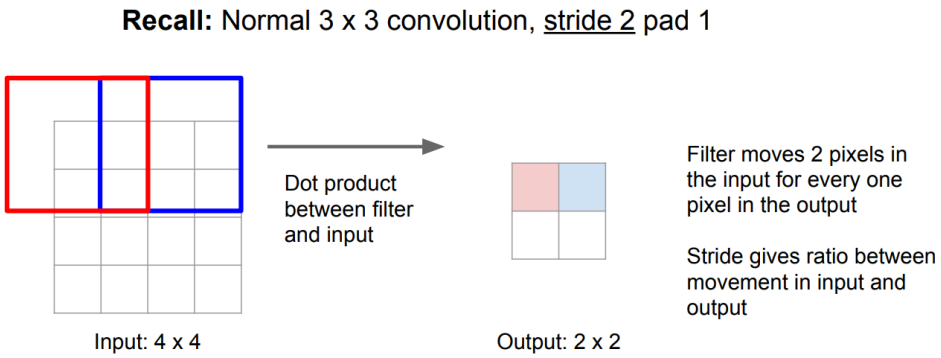
The most common architectures for SSN are based on CNN, divided

[5]Semantic Segmentation neural network

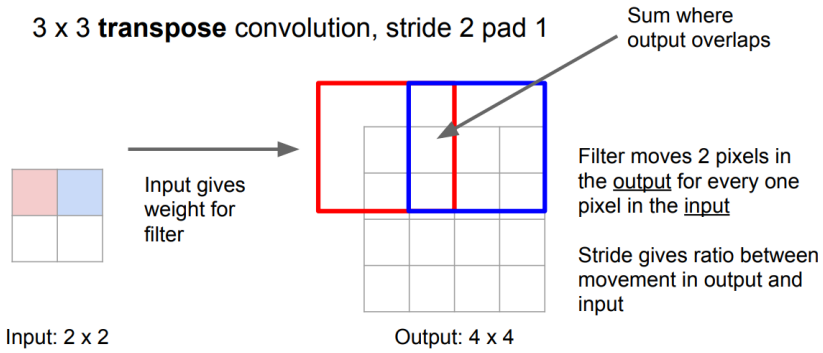
in an *encoder* part, and a *decoder* one, this kind of architecture is called *SegNet*. The Encoder part of the network has the task to extract the important features from the input image. Like in a normal CNN, this operations are also called *Down-Sampling*. This usually comes at the cost of the spatiality of the information, so to classify the image pixel by pixel, we must retrieve those information. The decoder part of the network basically is the reflection of the encoder, but instead of CONV layers, there are up-convolutional (UP-CONV) layers, in this phase we have the operation of *Up-Sampling*.

The architecture of the decoder may be very different in the various networks, but usually the UP-CONV layers are always present. The UP-CONV layers are based on the *transposed convolution* operation, which takes the each single value coming from the input volume and multiplying it for an array of weights similar to filters. After this operation the values are disposed spatially, taking into account the padding and striding parameters like in Figure 2.11. Also the UP-CONV layers have learnable parameter, so the network try to re-build the image in order to minimize the loss function.

The encoder-decoder architecture is many times more efficient than a sliding window architecture that classifies pixel by pixel, but it suffers of a drawback. In the Figure 2.11b we can notice that in the overlapping area of the receptive fields, the values that come out from this convolution operation are summed up. This can cause what is called "chess board" artifacts. In case of information in really small neighborhood this problem may introduce errors in the training and classification of the network.



(a) Convolution



(b) Transposed convolution

Figure 2.11: In this figure we have the representation of how the CONV and UP-CONV layers works. The red and blue boxes represent a sliding 3×3 filter. http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf

Chapter 3

Neural Network setup

In the computer vision problem, the infer subject is of primary importance, because it reflects on the whole design of the architecture. In this chapter will be present the neural network architectures adopted with relative parameters specifically tuned to solve the specific landing problem.

3.1 Architecture

Both models are based on the *U-Net* [Ronneberger et al., 2015] architecture, due to its good behaviour with small training sets and flexibility to be modified accordingly to the task. The name comes for the characteristic "U" shape as can be seen in Figure 3.1. We can distinguish two main parts of this architecture: the contracting path and the expansion path. This series of CONV layers and pooling layers define what is called "depth" of the network. It determines the number of times an input image is downsampled or upsampled as it is processed.

In the contracting path, the image will pass through convolutional layers and max-pooling layers that will extract the features of the image, but as explained in the Section 2.4 the spatial size of the data changes, with consequent loss of spatial information. The semantic segmentation should give an accurate spatial classification of the image, so to solve this problem we also have a concatenation of feature maps (gray arrows), that are with the same level to communicate the spacial information to the up-sampling part of the network, before any max-pooling layer. The expansive path recovers the original size of the image, but now the features are assigned to the corresponding pixels, which will be classified by a final softmax layer.

In this thesis, a 4 step encoder/decoder depth U-Net has been chosen

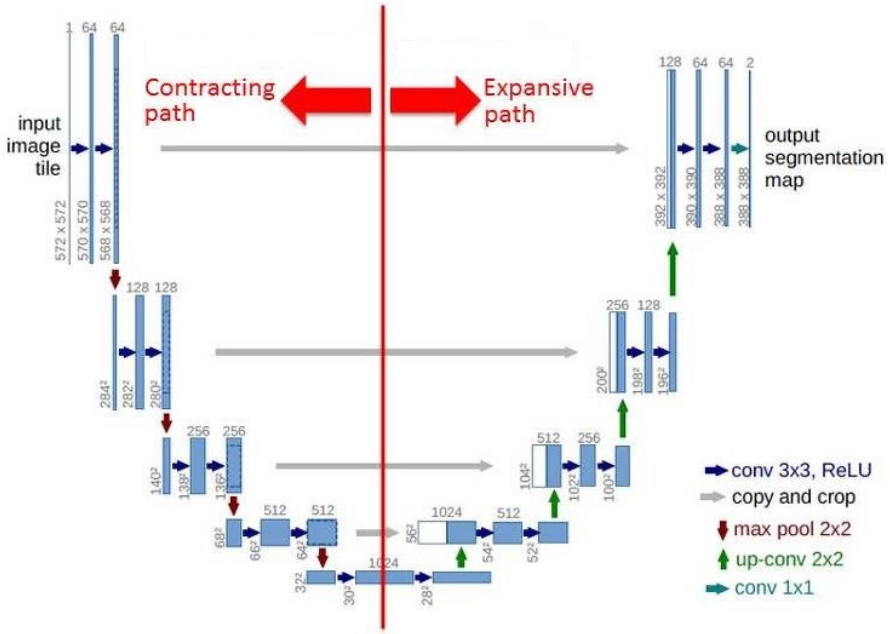


Figure 3.1: U-Net layers architecture

because it gives the best compromise between performances and training time. Higher depths are generally used for tasks with many classes because the model can collect more high-level features from the images. In our case we found that the accuracy improvement is negligible, since the hazard and safe areas are discriminable with low-level features, such as edges and pixels intensity standard deviation.

In Table 3.1 we have a list of the layers in the model. The role of the convolutional layers, maxpooling layers and transposed convolutional layers has been already explained in the Sections 2.3 and 2.4.

The drop-out layer means dropping out some neurons of the neural network to avoid over-fitting. It happens when many parameters are concentrated in a few layers, and hence, neurons develop co-dependency among each other during training, which curbs the individual power of each neuron leading to over-fitting of training data. Shutting down some neurons, we force the network to learn more robust features that are useful in conjunction with many different random subsets of the other neurons.

The concatenation layers are characteristic of the encoder/decoder structures. Their role is to concatenate the information on a specific dimension. Combining the location information from the downsampling path with the contextual information in the upsampling path is useful to

Table 3.1: Feature map, with output size per patch. If we want the output size for batch we must multiply it by the number of patches in a batch. `X` and `Y` are the spatial size of the input volume, while N_c is the number of channels. The abbreviations `Batch Norm`, `UP-Conv` and `Concat.` stands for batch normalization, transposed convolution and concatenation respectively. The indexes used represent [*E: encoder, B: bridge, D: decoder*]-[*depth number*]-[*layer number*].

Layers	**Output Size**	**Layers**	**Output Size**
Input	$(X \times Y \times N_c)$	UP-Conv(D1)	$(X/8 \times Y/8 \times 512)$
Batch Norm	$(X \times Y \times N_c)$	Concat.(C4.2)	$(X/8 \times Y/8 \times 1024)$
Conv (E1.1)	$(X \times Y \times 64)$	Conv (D1.1)	$(X/8 \times Y/8 \times 512)$
Conv (E1.2)	$(X \times Y \times 64)$	Conv (D1.2)	$(X/8 \times Y/8 \times 512)$
MaxPooling	$(X/2 \times Y/2 \times 64)$	UP-Conv(D2)	$(X/4 \times Y/4 \times 256)$
Conv (E2.1)	$(X/2 \times Y/2 \times 128)$	Concat.(C3.2)	$(X/4 \times Y/4 \times 512)$
Conv (E2.2)	$(X/2 \times Y/2 \times 128)$	Conv (D2.1)	$(X/4 \times Y/4 \times 256)$
MaxPooling	$(X/4 \times Y/4 \times 128)$	Conv (D2.2)	$(X/4 \times Y/4 \times 256)$
Conv (E3.1)	$(X/4 \times Y/4 \times 256)$	UP-Conv(D3)	$(X/2 \times Y/2 \times 128)$
Conv (E3.1)	$(X/4 \times Y/4 \times 256)$	Concat (C2.2)	$(X/2 \times Y/2 \times 256)$
MaxPooling	$(X/8 \times Y/8 \times 256)$	Conv (D3.1)	$(X/2 \times Y/2 \times 128)$
Conv (E4.1)	$(X/8 \times Y/8 \times 512)$	Conv (D3.2)	$(X/2 \times Y/2 \times 128)$
Conv (E4.2)	$(X/8 \times Y/8 \times 512)$	UP-Conv(D4)	$(X \times Y \times 64)$
Dropout	$(X/8 \times Y/8 \times 512)$	Concat (C1.2)	$(X \times Y \times 128)$
MaxPooling	$(X/16 \times Y/16 \times 256)$	Conv (D4.1)	$(X \times Y \times 64)$
Conv (B1)	$(X/16 \times Y/16 \times 1024)$	Conv (D4.2)	$(X \times Y \times 64)$
Conv (B2)	$(X/16 \times Y/16 \times 1024)$	Fully Connected	$(X \times Y \times 2)$
Dropout	$(X/16 \times Y/16 \times 1024)$	Softmax	$(X \times Y \times 2)$

finally obtain a general information combining localization and context, which is necessary to predict a good segmentation map.

3.1.1 Classes Balance

Due to the large difference of number of pixels associated to each class, to avoid a biased training we used the inverse class frequency weighting [V. Wang et al., 2012] to balance the two classes. This method promotes the underrepresented classes by computing the class frequency over the training set and then set the weights as the inverse of the relative frequency. These updated weights must be substitute to the ones in the pixel classification function in the softmax layer.

3.2 Deep Learning Framework

For both problems we implemented the U-Net in the `MATLAB` deep learning framework. The framework supports GPU acceleration, which lower the training time significantly when running on NVIDIA GPUs.

3.3 Itokawa Hazard Detection

In case of the Itokawa landing problem we explore two different algorithms to feed the network, which consequently reflects on the architecture. The first approach is based on resizing the images at a lower resolution. While the second one is a tile based approach. It consists in extracting crops of the images. Both methods are required to save GPU memory.

3.3.1 Hyperparamentes

In this section we discuss all parameters characterizing the training data set, there are certain parameters required for setting up a network and training it. In Tables 3.2, 3.3 are listed the hyperparameters adopted, many of which were already explained in the Section 2.1. The gradient clipnorm is a gradient threshold to avoid that the gradient explodes, which can occur when the weights update in one step is very large. As can be seen from the last layer in Table 3.1 we used a softmax classifier with cross-entropy loss.

Table 3.2: Hyperparameters settings for the Itokawa *Tile-Based algorithm*

Hyperparameter	Setting	Hyperparameter	Setting
Input Resolution	256×256	Optimizer	ADAM
Number of Channels	3 (RGB)	Initial Learning Rate	10^{-4}
Activation	ReLU	Batch Size	8
Stride	1	Epochs	50
Kernel Size	(3×3)	Weights Regularization	L2 - 0.005
Dropout Rate	50%	Gradient Clipnorm	0.1

Table 3.3: Hyperparameters settings for the Itokawa *Resizing algorithm*

Hyperparameter	Setting	Hyperparameter	Setting
Input Resolution	512×512	Optimizer	ADAM
Number of Channels	3 (RGB)	Initial Learning Rate	10^{-4}
Activation	ReLU	Batch Size	4
Stride	1	Epochs	200
Kernel Size	(3×3)	Weights Regularization	L2 - 0.005
Dropout Rate	20%	Gradient Clipnorm	0.1

Table 3.4: Hyperparameters settings for the lunar hazard detection algorithm

Hyperparameter	Setting	Hyperparameter	Setting
Input Resolution	512×512	Optimizer	ADAM
Number of Channels	4	Initial Learning Rate	10^{-4}
Activation	ReLU	Batch Size	4
Stride	1	Epochs	50
Kernel Size	(3×3)	Weights Regularization	L2 - 0.005
Dropout Rate	50%	Gradient Clipnorm	0.1

3.4 Lunar Landing Hazard Detection

For the lunar landing problem due to the size of the maps, only a tile-based approach has been adopted. This allows us to have a more various dataset with the implementation of data augmentation techniques, but it will be described more in detail in the next chapter.

The backbone of the network architecture is the same as Table 3.1, but this time, we use four channels instead of three. We noticed that the performances of the model improve if we add at the three color channels (RGB) another layer with the sun inclination angle in degrees [R. Liu et al., 2018].

We also explored more exotic architectures, in order to feed the sun angle directly into the features space of the network. In addition to that, we tried new loss functions based on the IoU[1] metric (Section 5.1.2) with promising results, but not near the performances of the more simple, yet effective, model chosen in this thesis.

3.4.1 Hyperparamentes

The list of parameters for this model are in Table 3.4. The batch size is small to respect the GPU memory requirements, due to the large dimensions of the patches. Because we still use the same U-Net backbone for this model too, we use a softmax classifier with cross-entropy loss.

[1] Intersection over Union

Chapter 4

Dataset Setup and Training

This chapter is organized as follows. The dataset used for the training is described. We continue by explaining how the labeling was performed. A description of the training process with the adopted algorithms follows.

The first model analyzed is the Itokwa hazard detection model. For this case, we will see the image processes adopted to ensure a more uniform and consistent training set. Then we show the two algorithms adopted for the training.

The last model described is the lunar model. We will explain how the dataset images were simulated and which parameters were taken into account for the labeling process. To complete the chapter, the training procedure is explained.

4.1 Itokawa Hazard Detection

The dataset adopted is based on real images of the asteroid Itokawa, provided by the Hayabusa 2 mission launched in 2014. Then we modify the dataset following two different approaches. In the first, we re-size every image to feed the network for the training, while in the second one, we take crops of every image to enlarge the dataset.

4.1.1 Data Set

The provided images refer to the approach phase and orbit phase of Hayabusa 2. They were taken with the spacecraft on-board camera. The illumination source is the sun, and the images are RGB and with normalized illumination. The whole data-set counts more than 1500 images taken

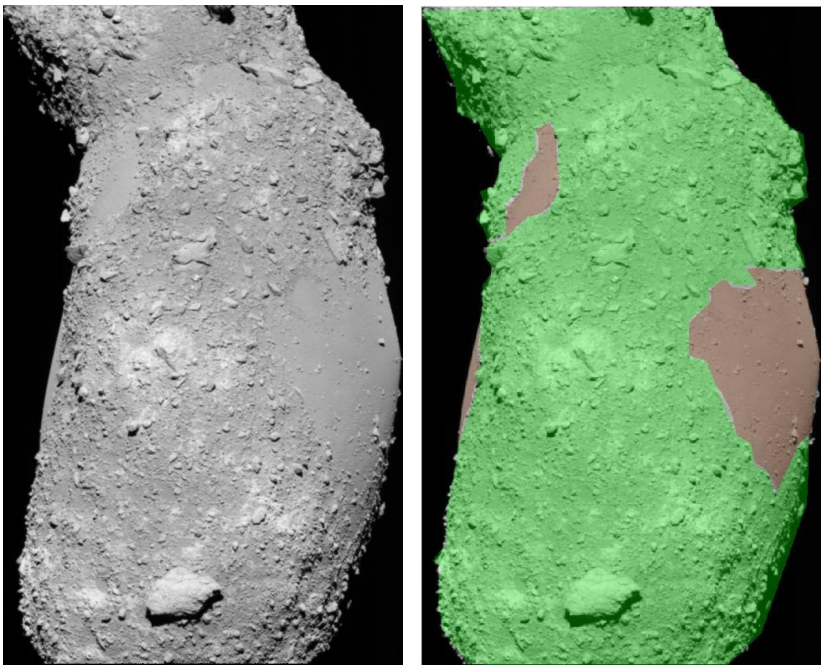
at various distances and illumination conditions, but for our purposes a selection of around 40 images has been adopted. Because the images are at the fixed resolution of 1024×1024 pixels, only the ones taken at a certain distance from Itokawa can be used, to avoid that the meaningful information are collected only in few pixels.

The limited number of images for the dataset is given by the large amount of time require to correctly label the images. The size of the dataset has a big impact on multiple aspects of the deep learning approach. With a small training-set the network may over-fit easily due to the lack of differentiation between the data. This can cause serious classification errors if the test-set is slightly far from the training one. Besides to avoid overfitting the training time can not be very long with a consequent impact on the overall accuracy of the network.

4.1.2 Labeling

Two labels have been taken into account: *smooth terrain - rocky terrain*. The *smooth* label identifies all the terrain portion visible in the images with fine sand, while the *rocky* label is assigned to the terrain with many boulders and irregularities, (Figure 4.1). The slopes on such small bodies are usually defined as the angle between the total gravitational acceleration plus centripetal acceleration vector and the surface normal. However, due to the lack of a shape model of the asteroid, the terrain slope was not taken into account.

The image labeling has been done manually with the `Image Labeler` app in the `MATLAB` suite; this method is not ideal because it suffers from the human error, but for this application is acceptable given that it doesn't have much influence in the learning of the characteristic, if it is limited to a few pixels. It means that not every pixel in the image is classified correctly in the ground truth label, but automatic methods, with the application of filters, fail to give a satisfactory result. The main reason is due to the small dimensions of the body; the brightness of the illuminated areas varies a lot with the body curvature. Besides, the parameters update in the network is dependent on the number of pixels of each class, so small errors have a low impact on the network training.



(a) Image **(b)** Ground truth

Figure 4.1: Example from the training dataset. The *rocky terrain* label is represented in green, while the *smooth terrain* label in the rust color.

4.1.3 Image Pre-Processing

To achieve better model learning, the training set of images has been modified by removing as much as possible the background and the shadow cast by the spacecraft solar panels. The main reason to remove as much background as possible is that many pixels are without class so they can bias the training. Especially in a tile-based algorithm, that will be introduce shortly after, some patches could contain only the background, which would be a waste of computer resources and time. To remove it first, we convert the image in HSV[1] where the third channel gives the value of the lightness of each pixel. We filter out any value under a certain threshold to remove the darkest regions. Then the image has been cropped around the region of interest. With this process, we also removed the part of the asteroid shadowed by the spacecraft.

[1]Hue, Saturation, Value

4.1.4 Training

The entire set of images has been divided in training set, validation set, and test set. Respectively: 22 images for the training set, 5 for the validation and 11 for the test set.

Two different training algorithms have been tested: the *Resize Algorithm* and the *Tile-Based Algorithm*. The respective training hyperparameters are in Section 3.4.1.

Resize Approach

This script takes the images and resize them at a fixed resolution of 512×512 pixels. This was done to fulfill the GPU memory limit. The downside of this method is the loss of resolution and therefore of information, but the network was still capable of extracting the important features from the images. Due to the small number of images for the training, we also applied on the training set a data augmentation procedure, which modified the images in order to make them look different to the network. This is a good method to partially overcome the lack of data for the training. The data augmentation algorithm randomly rotates the images between -20 and $+20$ degrees, and it randomly mirrors the image with respect to the X axis.

Tile-based Approach

In this variant, the images instead of being re-sized are kept at their original size, then the data set is built on fixed size random crops of the images. This method is often applied for medical semantic segmentation or satellite imaging segmentation, and it allows us to have a large dataset with a few images. For the training set, we get 20 patches of size 256×256 per image. Instead for the validation set, we extract 10 patches for image. On both datasets, data augmentation has been applied with the same options described above.

4.2 Lunar Landing Hazard Detection

In the case of moon maps we simulate the images using digital terrain models (DTMs) found on (http://lroc.sese.asu.edu), instead of using real images. We choose this approach because it allows us to change the resolution and illumination conditions. One more advantage is the possibility to easily make the DTM and the rendered image coincide, by modifying the

camera position in the rendering software, which is fundamental to create precise masks to label the model.

4.2.1 Data Set

The images are generated with a ray-tracing program, POV-Ray[2]. The program traces each light ray generated by the light sources. Then it computes the rays reflections on every object inside the considered domain. In our case, the object is a DTM of the Apollo 16 landing site. This area has been selected for a large number of suitable landing sites.

The DTM is a grayscale representation for the elevation of the terrain. The terrain elevation is measured with respect to the mean value of the Moon radius. It is worth noticing that the terrain models have been taken with a Lunar Orbiter Laser Altimeter (LOLA[3]) with a root mean square error (RMS) of 3.93m at 12 different LOLA orbit tracks. This value can give a measurement of the vertical and horizontal accuracy of the model.

Figure 4.2 illustrates the two squares from which the training set and the test set have been taken. The red one has been used to train the network, while the orange one to test it. The red box covers an approximate area of 4×4 km, instead the orange box around 16×16 km. Given that the images have a fixed resolution of 4096×4096 pixels, it means that in the train set the resolution is 1 m/pixel while in the test set is 4 m/pixel. This resolution is acceptable for most landers to detect a safe landing site, and because it is easier for the network to recognize hazard terrain characteristics. In our tests, higher resolutions did not bring any real advantages but instead had a big impact on the GPU memory. We can notice that the same area of the DTMs does not have the same brightness in the two cases, that is because each image is normalized on its maximum elevation excursion. For the training square, the elevation varies between around 10 and -160 m, while for the test square, between around 300 and -160. Both test set and training set images are rendered at four different sun inclinations: 25°, 35°, 45° and 55°. This gives us a total of four 16 megapixel images for the training set and for the test.

The other important role of the DTMs is to compute the safe index parameters. In [Ge et al., 2016] is shown that the safe index can collect many constraints, like the fuel consumption expressed as a propellant mass

[2]The Persistence of Vision Ray Tracer
[3]Lunar Orbiter Laser Altimeter

fraction (PMF[4]). However, we just considered landing safety. The landing safety index gathers the maximum slope and roughness requirements for the landing. We adopted a maximum of 8 degrees for the slope and 5% as maximum terrain roughness. Both values are similar to the requirements of the *Chang'e 3* lander [X. Liu et al., 2019].

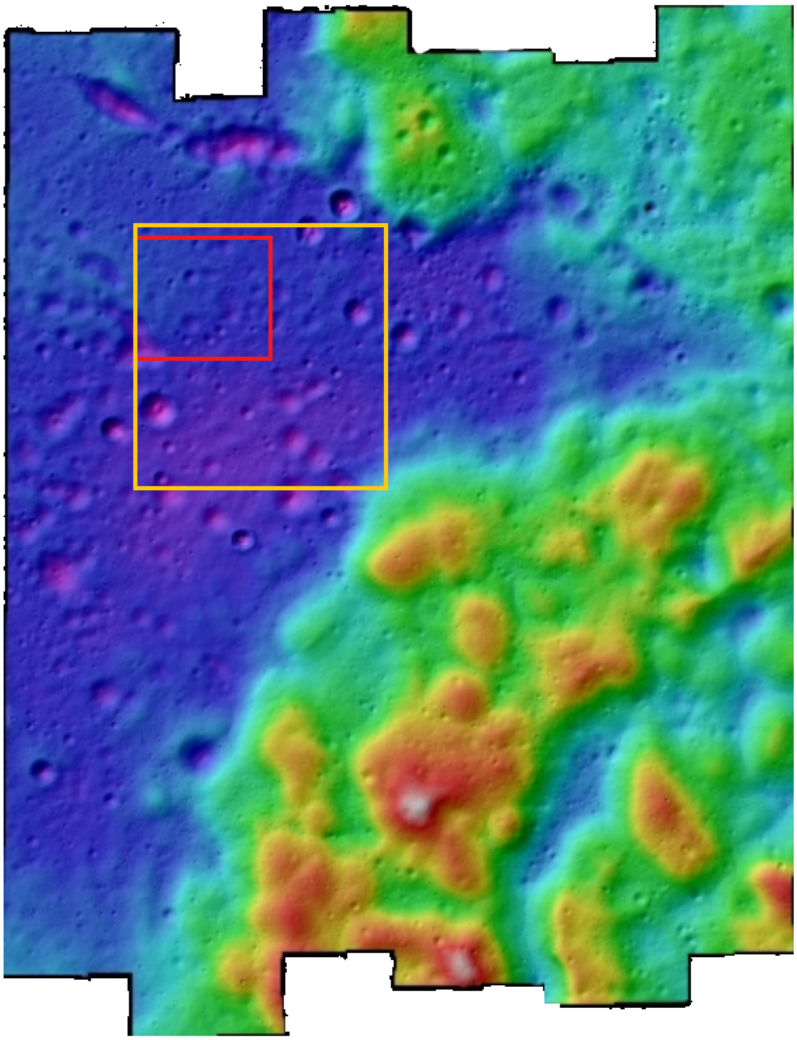
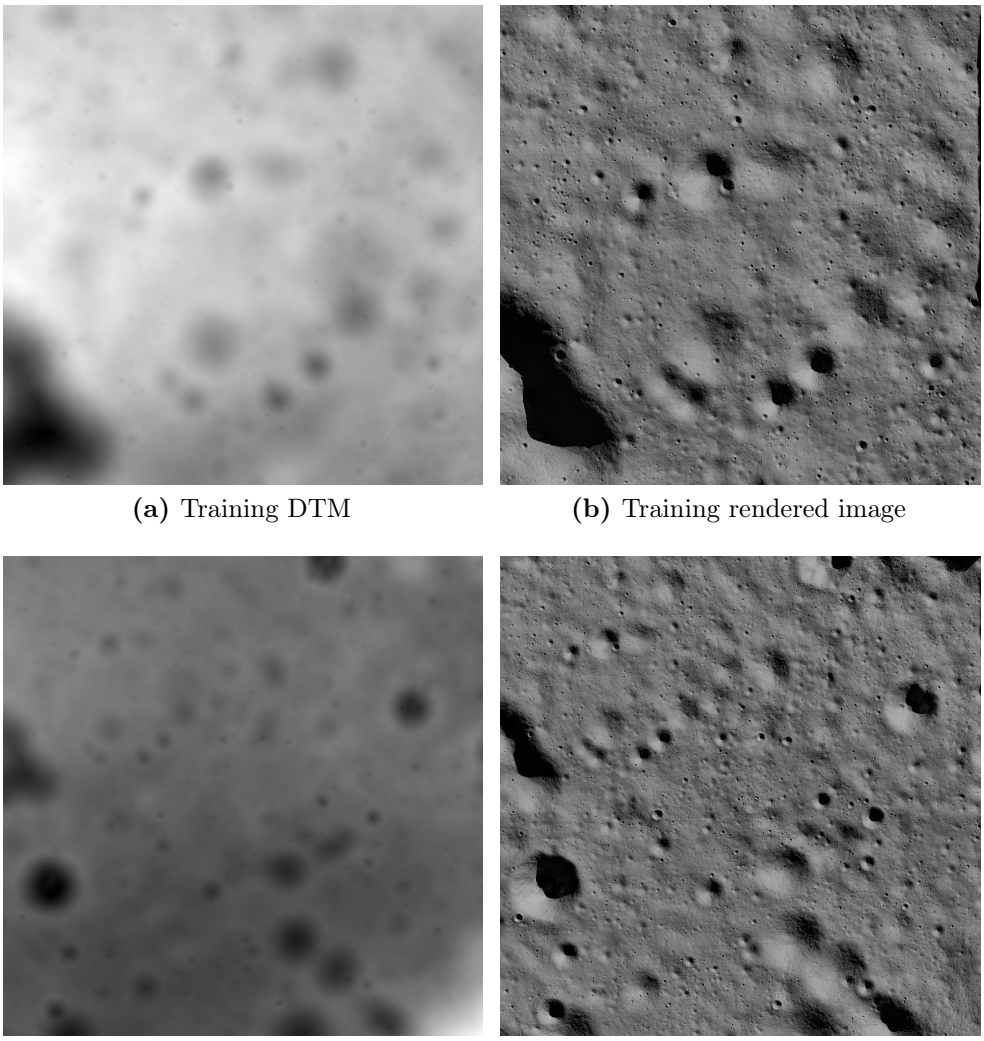


Figure 4.2: Elevation heat map of the Apollo 16 landing site. The red square represent the part of the map used for the training, while the orange square is the crop used for the test.

[4]Propellant Mass Fraction



(a) Training DTM **(b)** Training rendered image

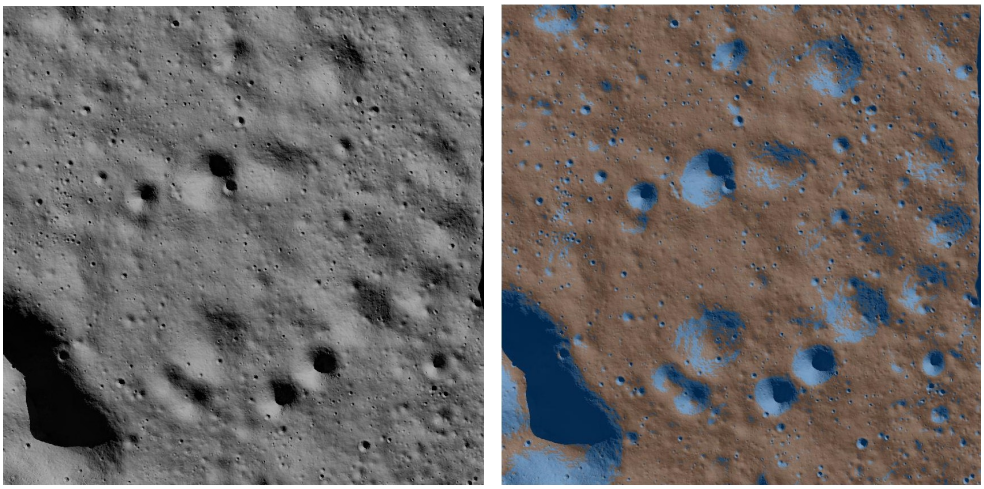
(c) Test DTM **(d)** Test rendered image

Figure 4.3: On the first row there are both DTM and rendered image crops of the map used for the training, while in the second row there are the crops used for the test.

4.2.2 Labeling

Differently from the Itokawa case, where the images were manually labeled, here, we develop an algorithm to have an high accuracy labeling. The first step is to import the DTM of the selected area and compute the slope pixel-wise as the gradient of the DTM. Also the roughness can be computed by finding the normalized standard deviation of the pixel intensity with respect the nearby pixels [Höfle and Hollaus, 2010]. At this

point we defined the classes: *Safe terrain* and *Hazardous terrain*. A pixel to be classified as *Safe* must meet three requirements: both slope and roughness must be less or equal of the respective thresholds, and the point must not be in shadow. If one of these points is not respected, the pixel is classified as hazard. In Figure 4.4 we can see the superimposition of the label on the camera image.



(a) Image **(b)** Ground truth

Figure 4.4: Example from the training dataset. The class *Hazard* is represented in blue, while the class *Safe* in the rust color.

4.2.3 Training

The training procedure is similar to the one described in Section 4.1.4. We used the area in the red square shown in Figure 4.2, and then we extracted from it random crops of fixed dimension: 512×512 pixels. The patches have such large size to be able to fit the larger craters inside it, which helps the network to detect terrain characteristics. Because the crops are taken randomly in such large area, they are used also for the validation of the network. For the training set we used 100 patches per image, while for the validation 30 per image, which gives us a total of 400 patches for the training and 120 for the validation. Then each dataset is subject to a data augmentation process like in the cases already described. Besides random rotation and mirroring we also used a random image re-scaling between 0.5 and 2.0. The re-scaling allows the network to see terrain features with different size, like craters, more easily. The hyperparameters adopted for

the training are listed in Section 3.4.1

Chapter 5

Results

This chapter is organized as follows. The metrics adopted for the evaluation are described. We continue by studying the quantitative and qualitative results of the suggested approaches. At last, the chapter is completed with a runtime analysis of the CPU and GPU implementation.

The first experiment discussed herein is about the Itokawa asteroid hazard detection. The performances will be evaluated on both algorithms: image crops and re-sized images. Furthermore, we present the results of the pre-processed dataset with haze reduction filters.

The last experiment in this chapter examines the performances of the network on the lunar landing hazard detection problem. Then we compare those results with a random forest classifier trained on specific image features.

5.1 Evaluation Metrics

Before diving into the evaluation methods we need to introduce some terminology:

- TP = true positive, data points labeled as positive that are actually positive.
- TN = true negative, data points labeled as negative that are actually negative
- FP = false positive, data points labeled as positive that are actually negative

- FN = false negative, data points labeled as negative that are actually positive

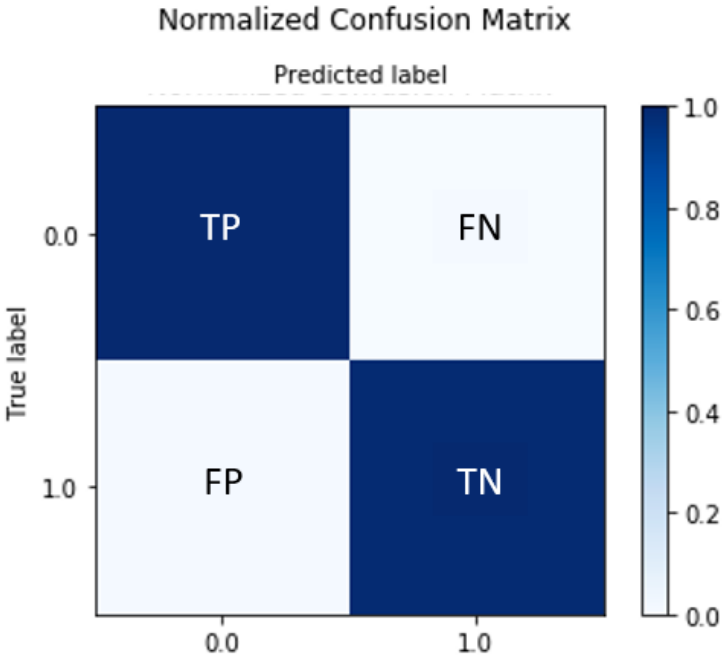


Figure 5.1: Normalized confusion matrix.

This nomenclature is often used in confusion matrices, (Figure 5.1), which is an intuitive graphical representation of algorithm performance in a semantic segmentation problem.

In this thesis we refer to TP for the safe pixels correctly classified as safe, on the other hand TN are those hazard pixels correctly classified as hazardous. Consequently, FP are hazard pixels incorrectly classified as safe, and at last FN are the safe pixels classified as hazardous.

The metrics adopted for the proposed experiments of semantic segmentation are the precision vs. recall and the intersection over union (IoU).

5.1.1 Precision Vs. Recall

This method is more suitable than the global accuracy metrics for semantic segmentation because it will yield misleading results if the data

set is unbalanced. The accuracy takes into account only the correct classifications, not the relevance of each class:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

In our case is fundamental that the hazardous terrain is correctly classified rather than the safe one, because any hazard can lead to the mission failure. So in case of an image with a vast safe area, if the model classifies everything as safe, the resultant global accuracy will be high.

Recall refers to the percentage of total relevant results correctly classified by the algorithm, and it is computed as:

$$Recall = \frac{TP}{TP + FN} \tag{5.2}$$

Precision refers to the percentage of the results that are relevant:

$$Precision = \frac{TP}{TP + FP} \tag{5.3}$$

As we said earlier the FP are extremely important, because they can lead to mission failure, so we need high precision. Nevertheless the recall can not be extremely low, otherwise we can not identify any safe landing site.

However, in cases where we want to find an optimal blend of precision and recall, we can combine the two metrics using what is called the F1 score [Goutte and Gaussier, n.d.]. The F1 score is the harmonic mean of precision and recall taking both metrics into account in the following equation:

$$F1 = 2\frac{Recall * Precision}{Recall + Precision} \tag{5.4}$$

5.1.2 Intersection Over Union

The intersection over union score, also known as Jaccard index, is often used to compare multi-class semantic segmentation methods. It is defined as follows:

$$IoU = \frac{TP}{TP + FP + FN} \tag{5.5}$$

Compared to precision and recall, the IoU is a strict metric, because it considers TP, FP and FN all together. Which means that the IoU score results in lower scores, since every missclassified pixel has a considerable impact on the overall score. We compute the IoU score for each class separately and, afterwards, compute the mean of all class scores.

5.2 Experimental Results: Itokawa Dataset

In this section, we evaluate our proposed algorithms on the Itokawa test dataset. For the experimental evaluation, we selected eleven unique images from the dataset of all the available Itokawa images, making sure that they are not present in the training set. The main challenge of this test is the small size of the objective, which leads to a high pixel intensity standard deviation even on smooth terrains, and a un-uniform distribution and shape of the shadows cast by the boulders.

The algorithms proposed are two: a *images resize algorithm* and a *tile-based algorithm*. A *manual features extraction algorithm* is also showed to make a comparison with non-deep-learning models. As for the training data, the resize algorithm fixes the resolution of the test images at 512×512 pixels.

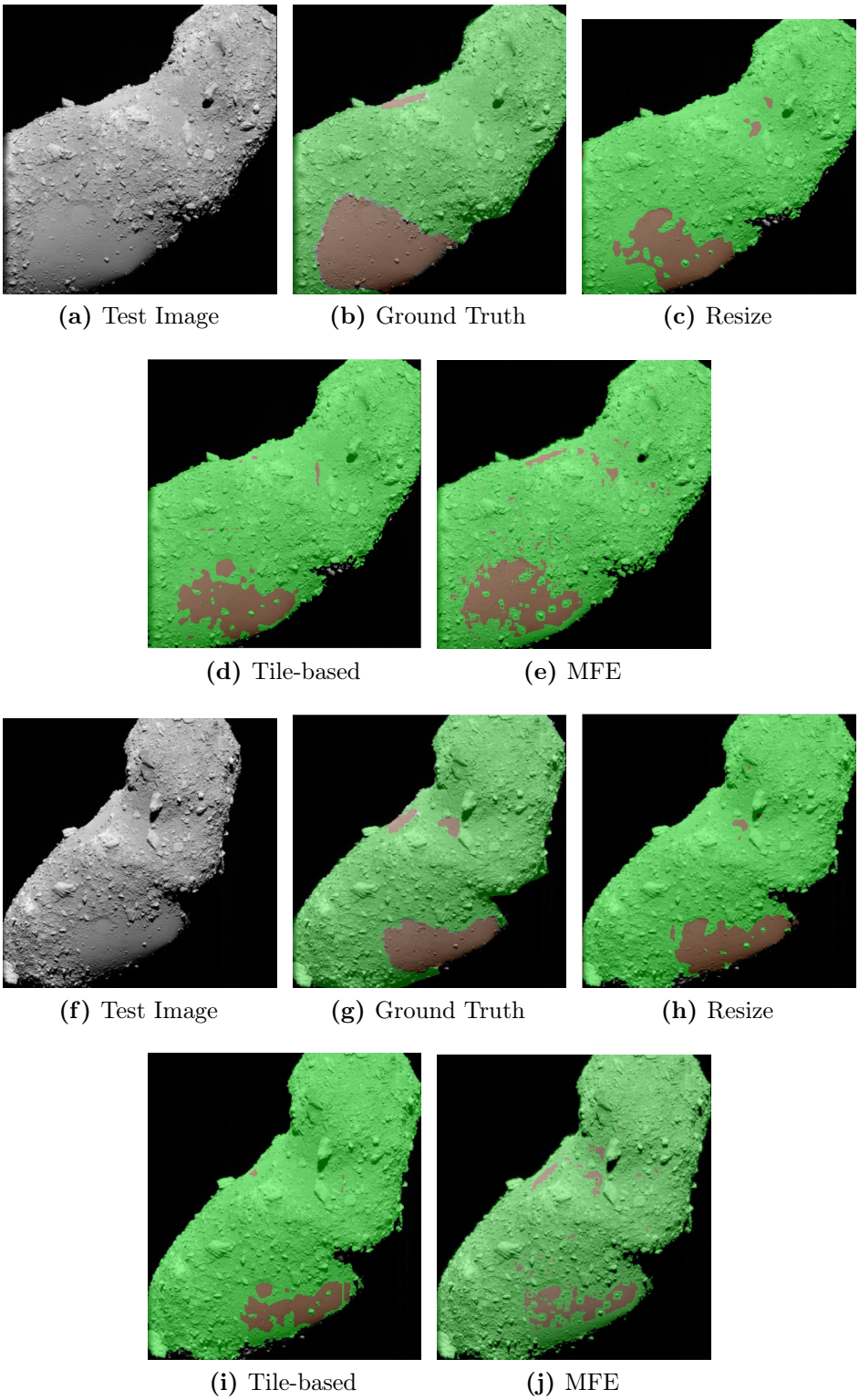
In the tile-based algorithm each image is cropped into tiles of size 256×256. Afterward, each patch is processed by the network. After the class labels have been inferred, the images are reassembled. Padding is used to reduce artifacts at tile-borders, so the patches overlap each other.

In the manual feature extraction algorithm (MFE[1]), the features are extracted from the images applying defined filters, which is similar to how CNN works, but with 3 diffent window sizes: 3×3, 9×9 and 15×15. The main difference is that in the MFE, we define a set of filters: mean value, mean, standard deviation, Laplacian of Gaussian filter, and Prewitt filter [Roushdy, 2006]. With these filters, we project the dataset features into the features space, then we use a random forest classifier to infer each class label. To train the classifier, we used the same dataset used to train the neural network.

We want to stress that the metrics results on Table 5.1 can not be used as a reliable source to assess the absolute performance of the models because the ground truth was manually labeled. Which means that the ground truth itself suffers from the human error. They can be used instead to compare the behavior of the different models.

In Table 5.1 are collected the scores for each model, and we can notice that the resize algorithm seems to outperform the other ones. But as we said in Section 5.1.1, the scores must be interpreted according to the scenario. In Figure 5.2 the resize algorithm confidence area covers most of the smooth class, but it also gives more false positive results. The

[1] Manual Features Extraction algorithm



(a) Test Image **(b)** Ground Truth **(c)** Resize

(d) Tile-based **(e)** MFE

(f) Test Image **(g)** Ground Truth **(h)** Resize

(i) Tile-based **(j)** MFE

Figure 5.2: Two examples results of the Itokawa dataset, the *rocky terrain* label is represented in green, while the *smooth terrain* label in the rust color.

Table 5.1: Performance comparison of the proposed algorithms.

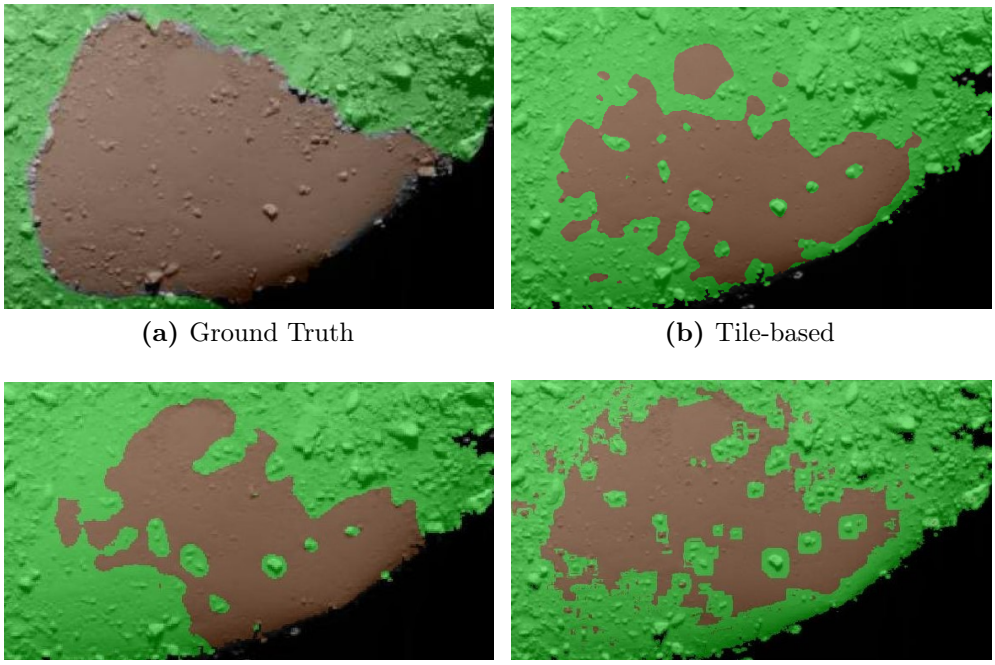
Algorithm	Precision	Recall	F1 Score	Mean IoU	Class	IoU
Resize	0.6988	**0.6831**	**0.6908**	**0.72212**	Rocky Smooth	0.91657 **0.52768**
Tile-based	**0.9325**	0.4493	0.6064	0.68119	Rocky Smooth	0.92719 0.43518
MFE	0.7788	0.4372	0.5600	0.6583	Rocky Smooth	**0.9276** 0.3889

tile-based approach on the other hand is more "cautious". It recognizes correctly almost all the rocky terrain as can be seen in Figure 5.4b. The tile-based even worst image result have a precision of 0.9176 and a IoU on the *rocky* class of 0.88734. It means that even in the worst-case scenario tested the model unlikely misclassified a hazardous terrain as safe.

If we look at Figure 5.4 separating the impact of each class on a real mission; the resize model is overall superior to the tile-based. The FP are not significantly more than the tile-based one; the TP and the FN difference is significant.

The resize approach was trained with the whole pictures of Itokawa in different light conditions, due to sun positioning or objective curvature. It explains why in Figures 5.3 the smooth area is correctly classified to the very edge of the asteroid. The patches have the advantage that they did not lose information by scaling the resolution, so the network can analyze even the smaller boulders, which edges can be blurred at lower resolutions. We can now explain the excellent performance of the tile-based approach to classify the rocky class. Both models also classify as rocks smaller boulders in the smooth terrain which are incorrectly in the ground truth smooth area, so their real performance metrics are likely superior to the ones shown in this thesis.

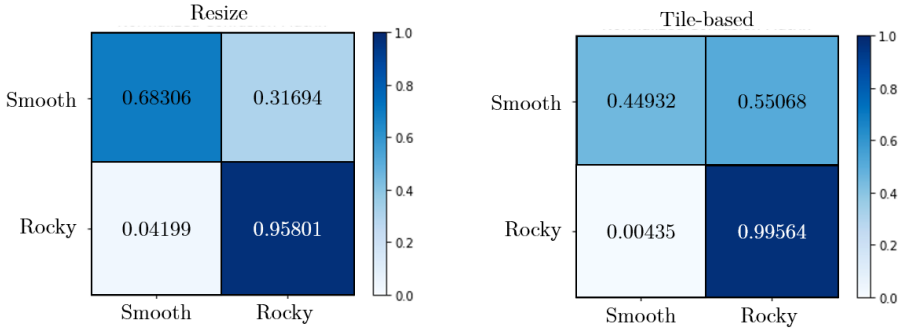
In Table 5.1 and Figure 5.2 it is also shown the clear performance improvement of both models with respect to the MFE algorithm. This is also due to the fact that the MFE model used was not originally designed for this kind of classification problem. Anyhow, it helps to understand what are the features that the deep learning models are considering for classes inference.



(a) Ground Truth **(b)** Tile-based

(c) Resize **(d)** MFE

Figure 5.3: Focus on the smooth area of the example in Figure 5.2a. The *rocky terrain* label is represented in green, while the *smooth terrain* label in the rust color.



(a) Normalized confusion matrix Resize algorithm **(b)** Normalized confusion matrix Tile-based algorithm

Figure 5.4: Normalized confusion matrix comparison.

5.3 Lunar Landing

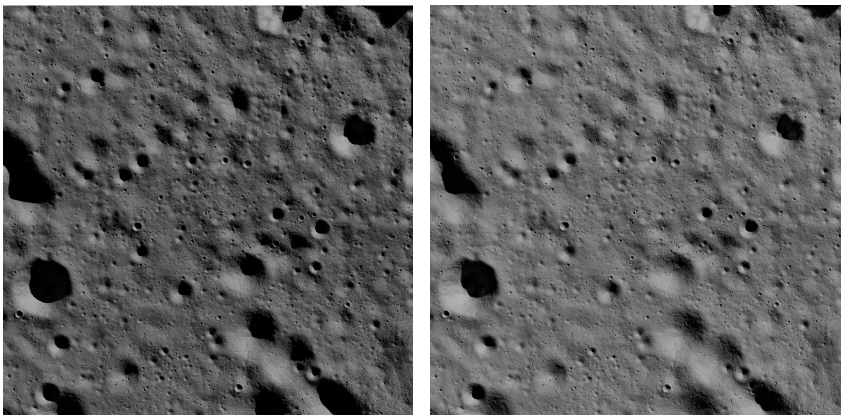
In this section, we use our model to recognize possible safe landing sites, on a simulated Apollo 16-like landing area. The experiment dataset has four images of the test area is represented in Figure 4.3d at different sun inclination conditions: 25°, 35°, 45° and 55°. The sun angle is provided as an additional channel for each test image because it has been noticed a clear improvement in the network performance. In Itokawa hazard detection problem, the human eye could easily distinguish between the two classes, but in case of a lunar vertical landing terrain features like slopes can not be easily recognized. This test aims to prove that the neural network can collect features not visible to humans, and elaborate them efficiently.

Due to the large size of the area, 16×16 km, with a resolution of 4096×4096 pixels a resize algorithm would lose too many information. The conventional approach in the semantic segmentation satellite imagery field is the tile-based approach, similar to the Itokawa tile-based approach already described in the previous section. We crop from each image patches of size 512×512, and then, we feed them into the network. The patch dimensions selected to fulfill the GPU memory requirements, and it allows to collect entirely also the more prominent craters inside it, since also the shapes of the terrain characteristics are essential features for the classification.

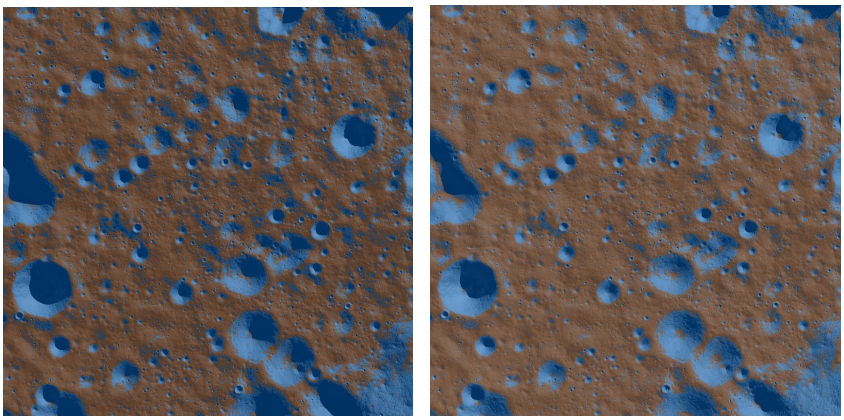
In this case, no MFE algorithm has been used to assess the network performances because of the total inability of the model to classify hazards in this scenario. Another limitation is the extended run-time required for the elaboration of an image at this resolution. The MFE requires hours while the neural network can do it in less than ten seconds.

Similarly to the training ground truth, the test ground truth has been made by computing the slope and roughness for each pixel from the DTM of the selected area. The safe class must respect all the three conditions already used for the training set: $slope \leq 8°$, $roughness \leq 5\%$, and not in shadow. If one of these conditions is not respected, the pixel is labeled as a hazard.

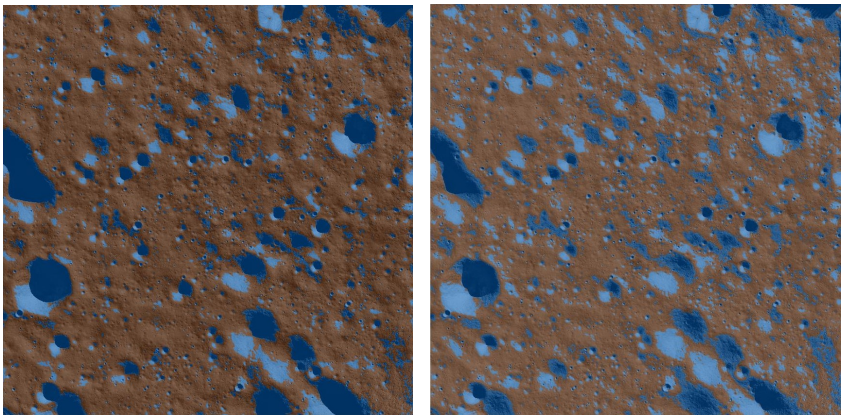
In Figure 5.5, 5.6 are represented the network experiment results. Since the ground truth take into account the shadow areas it changes with the sun angle. The ground truth maps and the network confidence maps shown are superimposed at the test images, (Figure 5.5a, 5.5b, 5.6a and 5.6b). This helps us to visualize which terrain characteristic the network is looking



(a) 25° Image **(b)** 35° Image

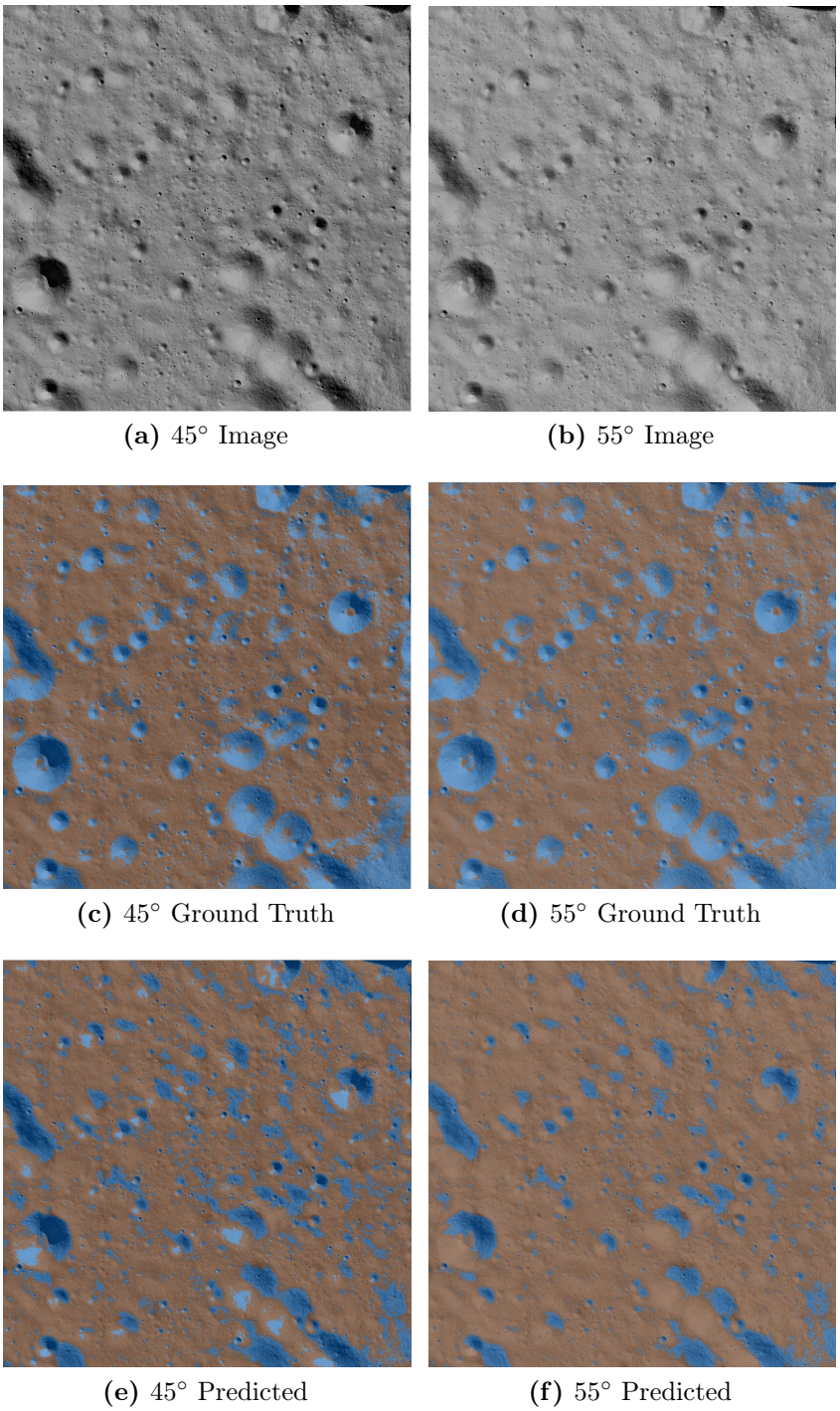


(c) 25° Ground Truth **(d)** 35° Ground Truth



(e) 25° Predicted **(f)** 35° Predicted

Figure 5.5: Experiment results for lunar landing with the sun inclination at 25° and 35°. The class *Hazard* is represented in blue, while the class *Safe* in the rust colour. The colors are not uniform because the confidence map have been superimposed to the test image.



(a) 45° Image **(b)** 55° Image

(c) 45° Ground Truth **(d)** 55° Ground Truth

(e) 45° Predicted **(f)** 55° Predicted

Figure 5.6: Experiment results for lunar landing with the sun inclination at 45° and 55°. The class *Hazard* is represented in blue, while the class *Safe* in the rust color.

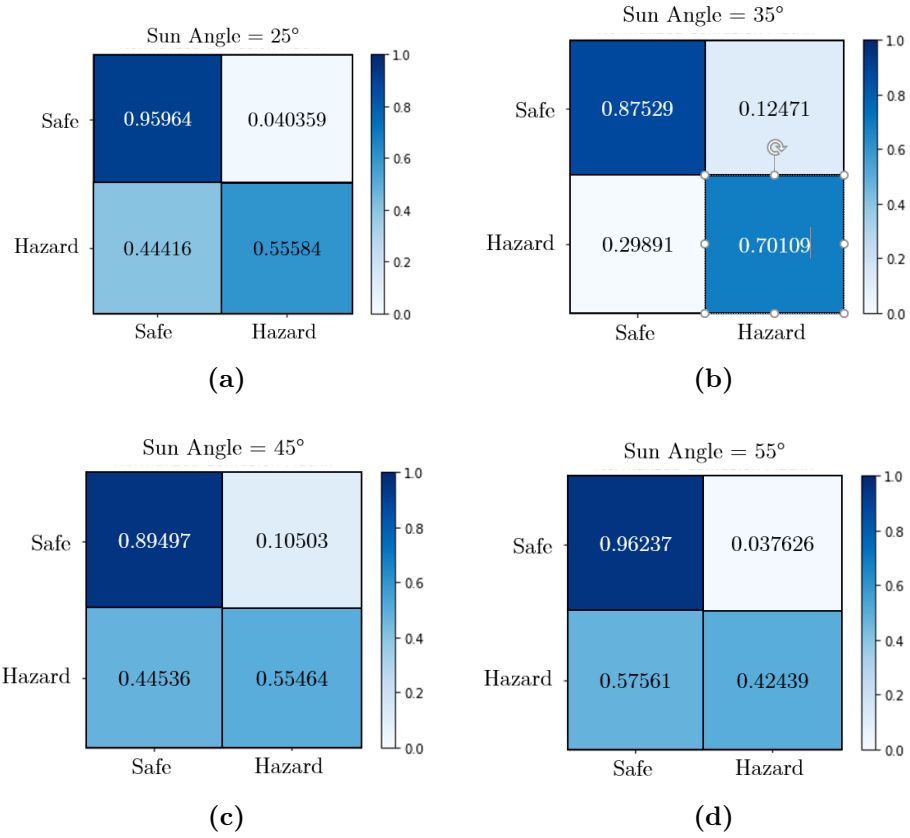


Figure 5.7: Normalized confusion matrix comparison.

for.

As we can clearly see in Figure 5.7 and Table 5.2 the results change significantly with the sun angles. We can notice that in brighter scenes like in Figure 5.6e and 5.6f only the dark side of the alrger craters is correctly predicted. Instead, with darker scenes like in Figure 5.5e and 5.5f also the brighter side is correctly classified. We can deduce that the network infer the slope and roughness from the pixels intensity with respect to the mean value of the map. A proof of this is visible in Figure 5.8. With the sun inclination at 35°, the confidence map is more accurate outside the craters, with respect to the scene with a sun inclination at 45°.

Table 5.2: Experimental results.

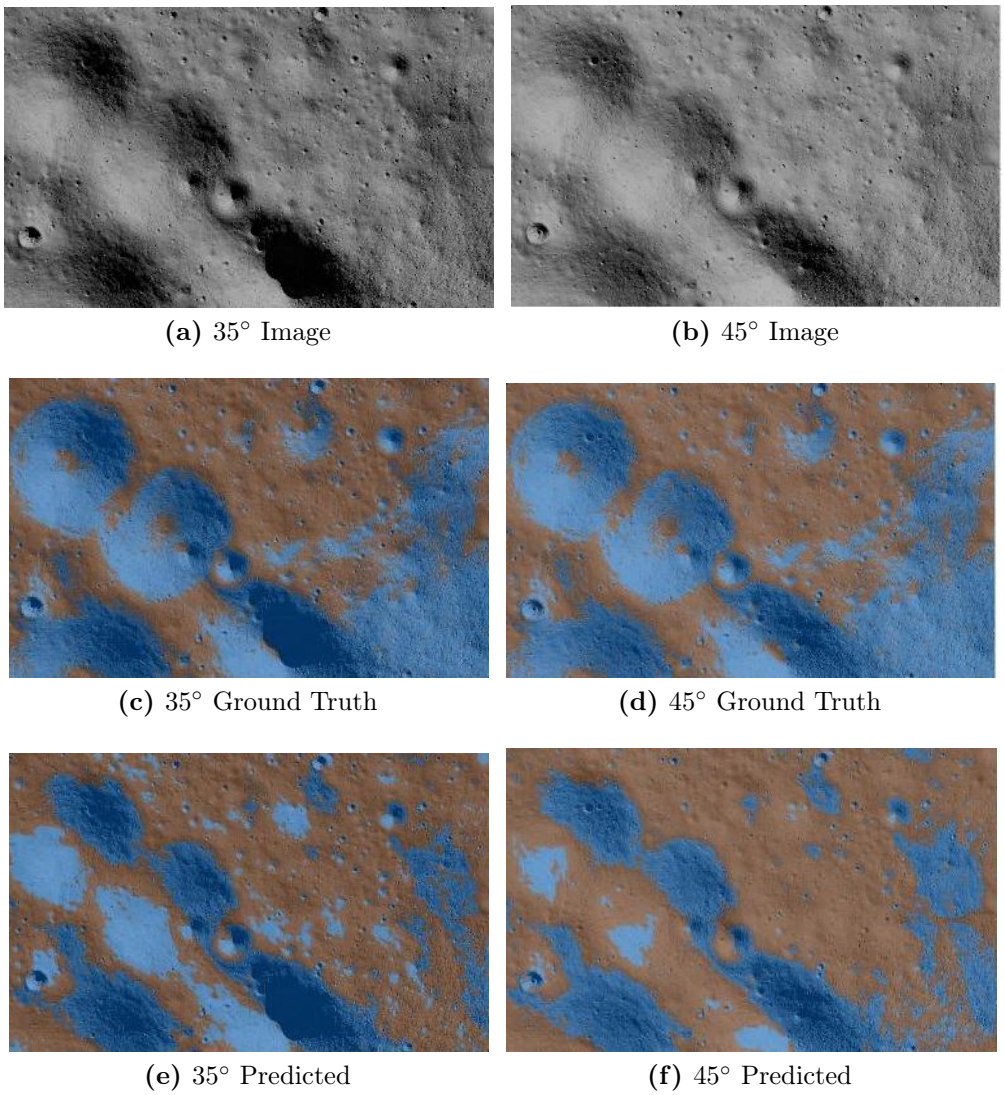
Sun Angle	Precision	Recall	F1 Score	Mean IoU	Class	IoU
25°	0.8173	**0.9596**	0.8828	0.65155	Safe	0.79013
					Hazard	0.51297
35°	**0.8868**	0.8753	0.8810	**0.65652**	Safe	0.78729
					Hazard	**0.52575**
45°	0.8514	0.8950	0.8727	0.60044	Safe	0.77407
					Hazard	0.4268
55°	0.8279	0.9624	**0.8901**	0.59244	Safe	**0.80196**
					Hazard	0.38292

5.4 Runtime Performance

Runtime performances for an autonomous hazard detection model is critical since if we want to use high-resolution cameras, the amount of data to process is enormous. For real space applications, the standard resolution for the cameras are around 1024×1024 pixels. Yet, cameras are not used for a complete vision-based landing.

Performances were recorded from a desktop PC with a GPU NVIDIA GTX 1070 with 8 GB of memory, while the CPU is an AMD Ryzen 5 2600 with 6 cores and 12 threads. The RAM is a dual-channel DDR4 with 16 GB. All the test were done on `MATLAB` with the GPU acceleration enabled.

In the Itokawa dataset, the resize algorithm requires 0.101 seconds for a 512×512 image, while the tile-based algorithm requires 0.303 seconds to



(a) 35° Image **(b)** 45° Image

(c) 35° Ground Truth **(d)** 45° Ground Truth

(e) 35° Predicted **(f)** 45° Predicted

Figure 5.8: Crop of the right bottom corner of the test map. The class *Hazard* is represented in blue, while the class *Safe* in the rust color.

process and rebuild the image. Each patch of size 256×256 is processed in around 0.0183 seconds. For tile-based algorithm in the Lunar landing case the time to process and rebuild an image is high (around 8 seconds) due to the large density of pixels (4096×4096 pixels). Each patch of size 512×512 is processed in around 0.110 seconds.

Even if this work was not done with the intention of optimizing the runtime performances, we can see in Figure 5.9 that the deep neural network with encoder/decoder architecture is several orders of magnitude

faster than the MFE algorithm. The reason is the long time required to extract the features from the images with 3 different window sizes. While, as we said in Section 2.4 the encoder-decoder architecture is really efficient.

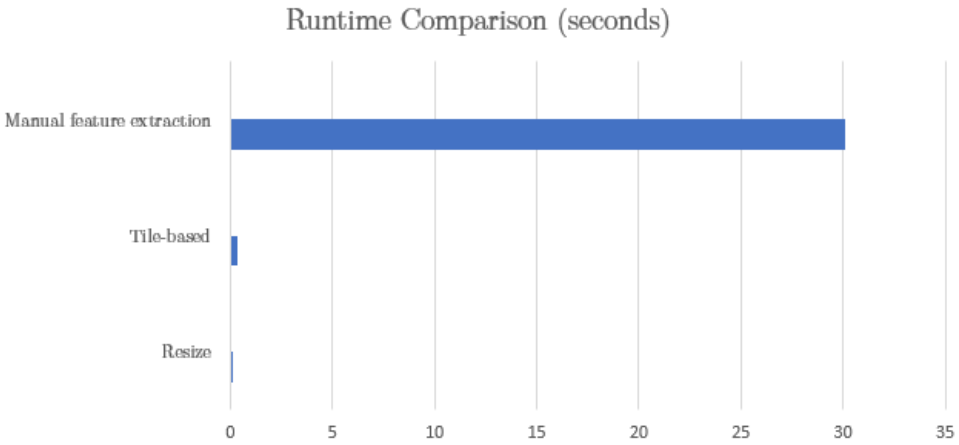


Figure 5.9: Runtime comparison between the differnt models for the Itokawa dataset.

Chapter 6

Conclusions and Future Works

In this work, we tested the latest deep learning semantic segmentation models applied to the autonomous landing hazard detection problem. Because this technique is based on camera images, it only works in certain illumination conditions. It means that this method was never meant to substitute the current state-of-the-art hazard detection systems, which involves the use of LiDARs. Instead, this new method can be adopted as a backup option or used on very small and light landers in special conditions, thanks to its reduced computational cost and the lightness of modern cameras.

In Section 5, in this preliminary test, we showed that to detect hazards, like boulders, on small celestial bodies, the deep learning approach is a reliable option. The most concerning false positive errors are limited in the tile-based approach. Also, in the worst possible scenario, the precision with this model is above 91%. It is worth noticing that the test was done on a network explicitly trained on the Itokawa asteroid. Many asteroids like Bennu of the OSIRIS-REx mission does not have bright, smooth areas like Itokawa. Luckily the latest missions, Hayabusa 2 and OSIRIS-REx took many pictures of the individual asteroids allowing us to build more massive and improved datasets. For what concern the computational cost as we already wrote, the model was not designed with efficiency purposes, so much improvement can be made to enable the real time execution of all the processes.

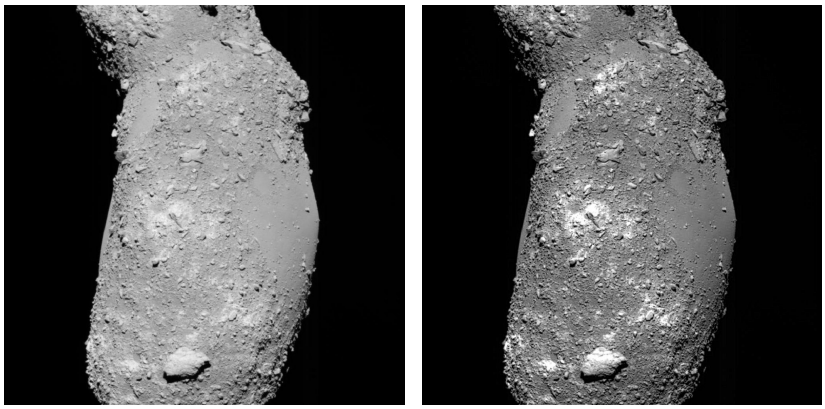
The lunar landing hazard detection is a challenging task for the deep learning semantic segmentation. Terrain features like the slope are highly dependent on the illumination conditions, as we can see from the results in Table 5.2, the metrics vary very much with the sun angle. Under 45°

of sun inclination, the model was able to recognize craters consistently, even the smaller ones. However, large areas with a hazardous slope were not correctly classified due to the lack of recognizable features. Above the 45° angle, the majority of the classification errors are linked to the overexposure of the scene, which saturates the colors.

It is worth mentioning that the model used is in a preliminary development phase for the deep learning lunar landing hazard detection. In fact, we did not perform neither the training nor the test on real images. The quality of the dataset has a fundamental impact on the model performances. In the next section, we are going to discuss some improvements that can be implemented.

6.1 Future Works

As we anticipated, our models are not optimized in terms of runtime; there is room for improvement. We think that changing the framework environment form `MATLAB` to Keras will boost the runtime of all our models. Thanks to the larger machine learning community working with python, it is easier to find better optimized functions. Another option is to use lossless data compression like the run-length encoding to store the labels as a single data value and count, rather than as the original run.



(a) Image un-processed **(b)** Image with haze reduction

Figure 6.1: Example of image processing in a train image.

For the Itokawa hazard detection model, some image processing techniques were tested. The best results were obtained using haze reduction filters on the images, (Figure 6.1), which granted a slight improvement

in the overall precision. However, it comes at a high cost of the recall metric, and it has a significant impact on the computational cost of the model, around 0.3 seconds more per image. That is because this process is based on the same method of the MFE algorithm. The idea is to try to feed the network with both datasets, processed and unprocessed, to make the network update the filters inside the convolution layers to do this image processing automatically. A fundamental step is to improve the dataset with many other images of different asteroids now that more high definition images are available, like the Ryugu asteroid of the Hayabusa 2 mission and the Bennu asteroid of OSIRIS-REx.

For future works on the lunar landing hazard detection, a couple of possible improved models were found. A promising model is a U-Net built on the backbone of a pre-trained ResNet50 classifier, with the training divided into two parts. In the first, we could train with the standard cross-entropy loss; then there can be another training phase using a Lovàsz Loss [Berman et al., 2018]. This technique is the state-of-the-art for many semantic segmentation challenges that can be found on [https://www.kaggle.com].

Another possible option is in the work of Cheng Huang and Hongmei Wang [Huang and H. Wang, 2018]. They used spatiotemporal convolutional networks for a vision based hazard detector. The spatial part of the two-stream branch carries information about contents and scenes, and the temporal part conveys the motion and edge information.

In the previous section, we said that the dataset is the foundation on which the network is built. For the future works, we will try to build a dataset with real high-quality images, or to render pictures as much photo-realistic as possible.

Acronyms

POV-Ray The Persistence of Vision Ray Tracer

The Persistence of Vision Ray Tracer, or POV-Ray, is a ray tracing program which generates images from a text-based scene description, and is available for a variety of computer platforms.
https://en.wikipedia.org

DTM Digital Terrrain Model

A digital elevation model (DEM) is a 3D CG representation of a terrain's surface – commonly of a planet (e.g. Earth), moon, or asteroid – created from a terrain's elevation data. A "global DEM" refers to a Discrete Global Grid. DEMs are used often in geographic information systems, and are the most common basis for digitally produced relief maps. While a DSM may be useful for landscape modeling, city modeling and visualization applications, a DTM is often required for flood or drainage modeling, land-use studies, geological applications, and other applications, and in planetary science.
https://en.wikipedia.org

LOLA Lunar Orbiter Laser Altimeter

The Lunar Orbiter Laser Altimeter (LOLA) provides a precise global lunar topographic model and geodetic grid that serves as the foundation of essential lunar understanding. This aids future missions by providing topographical data for safe landings and enhance exploration-driven mobility on the Moon. LOLA also contributes to decisions as to where to explore by looking at the evolution of the surface. LOLA fully achieves three LRO measurement objectives and addresses two other. LOLA will provide all the data

necessary to select intriguing, safe landing sites, while providing the reference system needed to navigate to those sites. LOLA builds on extensive spaceflight heritage, including the Mercury Laser Altimeter (MLA) and the Mars Orbiter Laser Altimeter (MOLA). The LOLA measurement team has 15 years of altimetry experience that includes providing MOLA data to the Mars Exploration Rover site-selection teams.
https://lola.gsfc.nasa.gov

LiDAR Light Detection and Ranging

LiDAR is a surveying method that measures distance to a target by illuminating the target with pulsed laser light and measuring the reflected pulses with a sensor. Differences in laser return times and wavelengths can then be used to make digital 3-D representations of the target. LiDAR is commonly used to make high-resolution maps, with applications in geodesy, geomatics, archaeology, geography, geology, geomorphology, seismology, forestry, atmospheric physics,laser guidance, airborne laser swath mapping (ALSM), and laser altimetry. The technology is also used in control and navigation for some autonomous cars.
https://en.wikipedia.org

PMF Propellant Mass Fraction

In aerospace engineering, the propellant mass fraction is the portion of a vehicle's mass which does not reach the destination, usually used as a measure of the vehicle's performance. In other words, the propellant mass fraction is the ratio between the propellant mass and the initial mass of the vehicle. In a spacecraft, the destination is usually an orbit, while for aircraft it is their landing location. A higher mass fraction represents less weight in a design. Another related measure is the payload fraction, which is the fraction of initial weight that is payload.It can be applied to a vehicle, a stage of a Vehicle or to a Rocket Propulsion System.
https://en.wikipedia.org

CNN Convolutional Neural Network

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing

visual imagery. Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.
https://en.wikipedia.org

ADAM Adaptive Moment Estimation

An algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments.
Kingma and Ba, 2014

SGD Stochastic Gradient Descent

Stochastic gradient descent (often abbreviated SGD) is an iterative method for optimizing an objective function with suitable smoothness properties (e.g. differentiable or subdifferentiable). It is called stochastic because the method uses randomly selected (or shuffled) samples to evaluate the gradients, hence SGD can be regarded as a stochastic approximation of gradient descent optimization.
https://en.wikipedia.org

SVM Support Vector Machine

In machine learning, support-vector machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.
https://en.wikipedia.org

HSV Hue, Saturation, Value

HSL (hue, saturation, lightness) and HSV (hue, saturation, value) are alternative representations of the RGB color model, designed in the 1970s by computer graphics researchers to more closely align with the way human vision perceives color-making attributes. In these models, colors of each hue are arranged in a radial slice, around a central axis of neutral colors which ranges from black at the bottom to white at the top. The HSV representation models the way paints of different colors mix together, with the saturation dimension resembling various shades of brightly colored paint, and the value dimension resembling the mixture of those paints with varying amounts of black or white paint.
https://en.wikipedia.org

FCN Fully Convolutional Network

The Fully Convolutional Network is a typology of convolutional neural network that is used for semantic segmentation tasks. It consist in several convolutional layers and a de-convolutional layer that retrieve the spacial information of the image from the input layer.
https://towardsdatascience.com

IoU Intersection over Union

The Jaccard index, also known as Intersection over Union and the Jaccard similarity coefficient (originally given the French name coefficient de communauté by Paul Jaccard), is a statistic used for gauging the similarity and diversity of sample sets. The Jaccard coefficient measures similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets.
https://en.wikipedia.org

MFE Manual Features Extraction algorithm

The algorithm is explained in Section 5.2.

CONV Convolutional layer

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery.

CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. However, CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme.
https://en.wikipedia.org

SSN Semantic Segmentation neural network

Semantic segmentation neural network is a general nomenclature used for deep neural network architectures for image segmentation. Image segmentation is one of the fundamentals tasks in computer vision alongside with object recognition and detection. In semantic segmentation, the goal is to classify each pixel of the image in a specific category
https://sergioskar.github.io

Bibliography

References cited

Publications and Manuals

Berman, Maxim, Amal Rannen Triki, and Matthew B. Blaschko

2018 *The Lovasz-Softmax Loss: A Tractable Surrogate for the Optimization of the Intersection-Over-Union Measure in Neural Networks*, tech. rep., pp. 4413-4421. (Cit. on p. 57.)

Brunel, Nicolas, Vincent Hakim, and Magnus Je Richardson

2014 "Single neuron dynamics and computation", *Current Opinion in Neurobiology*, 25, pp. 149-155, http://dx.doi.org/10.1016/j.conb.2014.01.005. (Cit. on p. 14.)

Chen, Liang-Chieh, George Papandreou, Iasonas Kokkinos, Kevin P. Murphy, and Alan Loddon Yuille

2015 "Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs", *undefined*, https://www.semanticscholar.org/paper/Semantic-Image-Segmentation-with-Deep-Convolutional-Chen-Papandreou/0690ba31424310a90028533218d0afd25a829c8d. (Cit. on p. 14.)

Claesson, Linnéa and Bjorn Hansson

2014 "Deep Learning: Methods and Applications [Preview]", *Foundations and Trends® in Signal Processing*, 7, 3-4, pp. 197-387, ISSN: 1932-8346, http://nowpublishers.com/articles/foundations-and-trends-in-signal-processing/SIG-039. (Cit. on p. 14.)

Fan, Fenglei, Wenxiang Cong, and Ge Wang

2017 "General Backpropagation Algorithm for Training Second-order Neural Networks" (Aug. 2017), http://arxiv.org/abs/1708.06243. (Cit. on p. 12.)

Ge, Dantong, Ai Gao, and Pingyuan Cui

2016 "Online Landing Site Selection Considering Maneuverability Constraint during Mars Powered Descent Phase", 5, pp. 1-9. (Cit. on p. 35.)

Goutte, Cyril and Eric Gaussier

n.d. "A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation Deep k-means clustering View project BioASQ View project A Probabilistic Interpretation of Precision, Recall and F-score, with Implication for Evaluation", https://www.researchgate.net/publication/226675412. (Cit. on p. 43.)

Gupta, Disha Shree

2017 *Fundamentals of Deep Learning - Activation Functions and their use*, https://www.analyticsvidhya.com/blog/2017/10/fundamentals-deep-learning-activation-functions-when-to-use-them/. (Cit. on p. 14.)

Hashimoto, Tatsuaki, Takashi Kubota, Jun'ichiro Kawaguchi, Masashi Uo, Kenichi Shirakawa, Takashi Kominato, and Hideo Morita

2010 "Vision-based guidance, navigation, and control of Hayabusa spacecraft - Lessons learned from real operation -", *IFAC Proceedings Volumes*, 43, 15, pp. 259-264, ISSN: 14746670, https://linkinghub.elsevier.com/retrieve/pii/S1474667015318504. (Cit. on p. 3.)

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun

2015 *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*, tech. rep., pp. 1026-1034, https://www.cv-foundation.org/openaccess/content_iccv_2015/papers/He_Delving_Deep_into_ICCV_2015_paper.pdf. (Cit. on p. 15.)

Höfle, B and M Hollaus

2010 *ISPRS Technical Commission VII Symposium, 100 Years ISPRS – Advancing Remote Sensing Science, Vienna, Austria, July 5–7, 2010, IAPRS, Vol. XXXVIII, Part 7B*, tech. rep., pp. 281-286, http://www.biosphaerenpark-wienerwald.org. (Cit. on p. 37.)

Huang, Cheng and Hongmei Wang

2018 "Vision-Based Hazard Detection with End-to-end Spatiotemporal Networks for Planetary Landing", p. 12023. (Cit. on p. 57.)

Kingma, Diederik P. and Jimmy Ba

2014 "Adam: A Method for Stochastic Optimization", pp. 1-15, http://arxiv.org/abs/1412.6980. (Cit. on pp. 11, 61.)

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton

2017 "ImageNet Classification with Deep Convolutional Neural Networks", *COMMUNICATIONS OF THE ACM*, 60, 6, http://code.google.com/p/cuda-convnet/.. (Cit. on p. 15.)

Lecun, Y., L. Bottou, Y. Bengio, and P. Haffner

1998 "Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, 86, 11, pp. 2278-2324, ISSN: 00189219, http://ieeexplore.ieee.org/document/726791/. (Cit. on p. 15.)

Li, Fei-fei, Justin Johnson, and Serena Yeung

2017 "Lecture 11 Detection and Segmentation", *CS231n: Convolutional Neural Networks for Visual Recognition*. (Cit. on pp. 10, 11, 13.)

Li, Shuang, Xiuqiang Jiang, and Ting Tao

2016 "Guidance summary and assessment of the Chang'e-3 powered descent and landing", *Journal of Spacecraft and Rockets*, 53, 2, pp. 258-277, ISSN: 00224650. (Cit. on p. 2.)

Liu, Rosanne, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski

2018 "An Intriguing Failing of Convolutional Neural Networks and the CoordConv Solution" (July 2018), http://arxiv.org/abs/1807.03247. (Cit. on p. 30.)

Liu, Xu, Shuang Li, Xiuqiang Jiang, and Xiangyu Huang

2019 "Planetary landing site detection and selection using multi-level optimization strategy", *Acta Astronautica*, ISSN: 00945765, https://doi.org/10.1016/j.actaastro.2019.01.004. (Cit. on p. 36.)

Lorenz, David A., Ryan Olds, Alexander May, Courtney Mario, Mark E. Perry, Eric E. Palmer, and Michael Daly

2017 *Lessons learned from OSIRIS-REx autonomous navigation using natural feature tracking*, tech. rep., https://ntrs.nasa.gov/search.jsp?R=20170002016. (Cit. on p. 3.)

Lunghi, Paolo, Marco Ciarambino, and Michèle Lavagna

2016 "A multilayer perceptron hazard detector for vision-based autonomous planetary landing", *Advances in the Astronautical Sciences*, 156, pp. 1717-1734, ISSN: 00653438. (Cit. on p. 1.)

M.Bishop, Christopher

2007 *Pattern Recognition and Machine Learning*, tech. rep. 3, pp. 366-366, http://users.isr.ist.utl.pt/~wurmd/Livros/school/Bishop%20-%20Pattern%20Recognition%20And%20Machine%20Learning%20-%20Springer%202006.pdf. (Cit. on pp. 8, 10.)

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox

2015 "U-net: Convolutional networks for biomedical image segmentation", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9351, pp. 234-241, ISSN: 16113349. (Cit. on p. 25.)

Roushdy, Mohamed

2006 *Comparative study of edge detection algorithms applying on the grayscale noisy image using morphological filter 3D Reconstruction of Brain Tumors using MRI View project Vision-Based Topological SLAM for Autonomous Robots View project*, tech. rep., http://cis.shams.edu.eg. (Cit. on p. 44.)

Wang, Vivienne, Elias G. Carayannis, Vivienne Wang, and Elias G. Carayannis

2012 *Inverse-Category-Frequency based Supervised Term Weighting Schemes for Text Categorization*, tech. rep., pp. 1-15, https://arxiv.org/ftp/arxiv/papers/1012/1012.2609.pdf. (Cit. on p. 28.)

Wei, Ruoyan, Jianwei Jiang, Xiaogang Ruan, and Jianke Li

2018 "Landing Area Selection Based on Closed Environment Avoidance from a Single Image During Optical Coarse Hazard Detection", *Earth, Moon and Planets*, 121, 3 (July 2018), pp. 73-104, ISSN: 15730794. (Cit. on p. 2.)

Zeiler, Matthew D. and Rob Fergus

2014 "Visualizing and understanding convolutional networks", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8689 LNCS, PART 1, pp. 818-833, ISSN: 16113349. (Cit. on pp. 15, 16.)

Additional sources consulted

Publications and Manuals

Badrinarayanan, Vijay, Alex Kendall, and Roberto Cipolla

2017 "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation." *IEEE transactions on pattern analysis and machine intelligence*, 39, 12, pp. 2481-2495, ISSN: 1939-3539, http://www.ncbi.nlm.nih.gov/pubmed/28060704.

Chen, Liang Chieh, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille

2018 *DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs*, tech. rep. 4, pp. 834-848, https://arxiv.org/pdf/1412.7062.pdf.

Csurka, Gabriela, Diane Larlus, and Florent Perronnin

2013 *What is a good evaluation measure for semantic segmentation?*, tech. rep.

Cui, Pingyuan, Dantong Ge, and Ai Gao

2017 "Optimal landing site selection based on safety index during planetary descent", *Acta Astronautica*, 132, July 2016, pp. 326-336, ISSN: 00945765, http://dx.doi.org/10.1016/j.actaastro.2016.10.040.

Dumoulin, Vincent and Francesco Visin

2016 "A guide to convolution arithmetic for deep learning", pp. 1-31, http://arxiv.org/abs/1603.07285.

Glasmachers, Tobias, Yung-Kyun Noh, and Min-Ling Zhang

2017 *Limits of End-to-End Learning*, tech. rep., pp. 17-32, http://proceedings.mlr.press/v77/glasmachers17a/glasmachers17a.pdf.

Krähenbühl, Philipp and Vladlen Koltun

2011 *Efficient inference in fully connected crfs with Gaussian edge potentials*, tech. rep., http://papers.nips.cc/paper/4296-efficient-inference-in-fully-connected-crfs-with-gaussian-edge-potentials.pdf.

Monien, Burkhard, Robert Preis, and Stefan Schamberger

2007 "Approximation algorithms for multilevel graph partitioning", *Handbook of Approximation Algorithms and Metaheuristics*, pp. 60-1.

Nolan, Michael C., Christopher Magri, Ellen S. Howell, Lance A.M. Benner, Jon D. Giorgini, Carl W. Hergenrother, R. Scott Hudson, Dante S. Lauretta, Jean Luc Margot, Steven J. Ostro, and Daniel J. Scheeres

2013 "Shape model and surface properties of the OSIRIS-REx target Asteroid (101955) Bennu from radar and lightcurve observations", *Icarus*, 226, 1 (Sept. 2013), pp. 629-640, ISSN: 00191035.

Schmidhuber, Jürgen

2015 "Deep Learning in neural networks: An overview", *Neural Networks*, 61, pp. 85-117, ISSN: 18792782, http://dx.doi.org/10.1016/j.neunet.2014.09.003.

Serrano, Navid

2006 "A Bayesian framework for landing site selection during autonomous spacecraft descent", *IEEE International Conference on Intelligent Robots and Systems*, pp. 5112-5117.

Silburt, Ari, Mohamad Ali-Dib, Chenchong Zhu, Alan Jackson, Diana Valencia, Yevgeni Kissin, Daniel Tamayo, and Kristen Menou

2019 "Lunar crater identification via deep learning", *Icarus*, 317, July 2018, pp. 27-38, ISSN: 10902643.

Sugawara, Etsuko and Hiroshi Nikaido

2014 *978-0387-31073-2*, 12, vol. 58, pp. 7250-7, ISBN: 978-0387-31073-2.

Wang, Qiong and Jizhong Liu

2016 "A Chang'e-4 mission concept and vision of future Chinese lunar exploration activities", *Acta Astronautica*, 127 (Oct. 2016), pp. 678-683, ISSN: 00945765.

Wurm, Michael, Thomas Stark, Xiao Xiang Zhu, Matthias Weigand, and Hannes Taubenböck

2019 "Semantic segmentation of slums in satellite images using transfer learning on fully convolutional neural networks", *ISPRS Journal of Photogrammetry and Remote Sensing*, 150, May 2018, pp. 59-69, ISSN: 09242716, https://doi.org/10.1016/j.isprsjprs.2019.02.006.

Zhang, Yiheng, Zhaofan Qiu, Ting Yao, Dong Liu, and Tao Mei

2018 "Fully Convolutional Adaptation Networks for Semantic Segmentation", *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 6810-6818, ISSN: 10636919.