UCLouvain

École polytechnique de Louvain

Convolutional Neural Networks for Oil Spill Detection in SAR Images

Author: **Guillaume LEGAT**
Supervisor: **Christophe DE VLEESCHOUWER**
Readers: **Benoît MACQ, Tahani MADMAD**
Academic year 2021–2022
Master [120] in Computer Science and Engineering

Abstract

Oil pollution in the seas and oceans is critical to the biological ecosystems of marine life and, in general, to the health of our earth. One of the sources of this pollution is the bilge dumping that generates long oil spills.

Satellite imagery and, in particular, SAR (Synthetic Aperture Radar) provide wide coverage of oceans and seas. The analysis of this large number of images can identify and locate these long oil spills using current machine learning methods. This information could be linked with the movements of the vessels, which makes it possible to identify and punish fraudsters.

However, many oceanic and atmospheric phenomena other than oil spills may modify the sea surface and create dark zones in SAR images. These dark zones are called look-alikes and represent a real difficulty in distinguishing from oil spills, even for a human operator.

Unlike many other applications, there is no dataset available to the scientific community for this type of situation. This results in limited research in this area and the inability to compare performances.

In this thesis, we develop a new spill oil dataset, including ground truth masks with training and test parts. We propose a CNN-based classification system that automatically detects oil spills from SAR images, including many other look-alikes. The architecture of the model and its hyperparameters are optimized and discussed.

Mean accuracy of 91.4% and a mean Intersection over Union (mIU) of 83.5% have been obtained on the test dataset.

Acknowledgements

My first thanks go to my supervisor, Professor Christophe De Vleeschouwer, for his patience, tolerance, and support. He did not spare any effort to encourage and motivate me during these long months. He gave me much scientific advice and was a real guide for me.

My thanks also go to Professor Benoît Macq and Tahani Madmad for agreeing to be part of the thesis jury. I hope that you will appreciate reading this document.

Moreover, my parents, sister, and two brothers have special thanks for their steady support and unfailing care. Your presence and help have been crucial throughout my years of engineering studies.

Contents

Nomenclature

ANN	Artificial Neural Network
$ASPP$	Atrous Spatial Pyramid Pooling
CNN	Convolutional Neural Network
DNN	Deep Neural Network
ESA	European Space Agency
FC	Fully-Connected Layer
FM	Feature Map
FN	False Negative
FP	False Positive
$IRSVRC$	ImageNet Large-Scale Visual Recognition Challenge
IU	Intersection over Union
MAC	Multiply-and-Accumulation
PQ	Panoptic Quality
$ReLU$	Rectified Linear Unit
RQ	Recognition Quality
SAR	Synthetic Aperture Radar
$SLIC$	Simple Linear Iterative Clustering
$SNAP$	Sentinel Application Platform
SQ	Segmentation Quality
TN	True Negative
TP	True Positive

List of Figures

List of Tables

Chapter 1

Introduction

Oil pollution in the seas and oceans is critical to the biological ecosystems of marine life and, in general, to the health of our earth. One of the sources of this pollution is the bilge dumping that generates long oil spills (Fig 1.1).

Most cargo or container ships and tankers use heavy oil as fuel. This generates a thick, oily residue that accumulates at the bottom of the vessel's tanks. Tanks must therefore be dumped and cleaned regularly. International maritime law strictly prohibits the dumping of these wastes into the sea. This operation must be carried out in specially equipped ports to process these residues. This naturally represents a high cost for shipping companies, and there are many pirate dumping, which causes significant pollution of seas and coasts. It is difficult to catch these polluters, who usually operate far from the areas monitored by the marine police.

Satellite imagery and, in particular, SAR (Synthetic Aperture Radar) provide wide coverage of oceans and seas. The analysis of this large number of images can identify and locate these long oil spills using current machine learning methods. This information is then linked with the movements of the vessels, which makes it possible to identify and punish fraudsters.

However, many oceanic and atmospheric phenomena other than oil spills may modify the sea surface and create dark zones in SAR images. These dark zones are called look-alikes and represent a real difficulty in distinguishing from oil spills, even for a human operator. Some of these look-alikes are illustrated in Fig 1.2.

Unlike many other applications, there is no dataset available to the scientific community for this type of situation. This results in limited research in this area and the inability to compare performances.

1.1 Objectives

The objectives of this thesis are to propose effective solutions to these challenges:

- The development of a new spill oil dataset, including ground truth masks with training and test parts that can be used to develop the classification system.

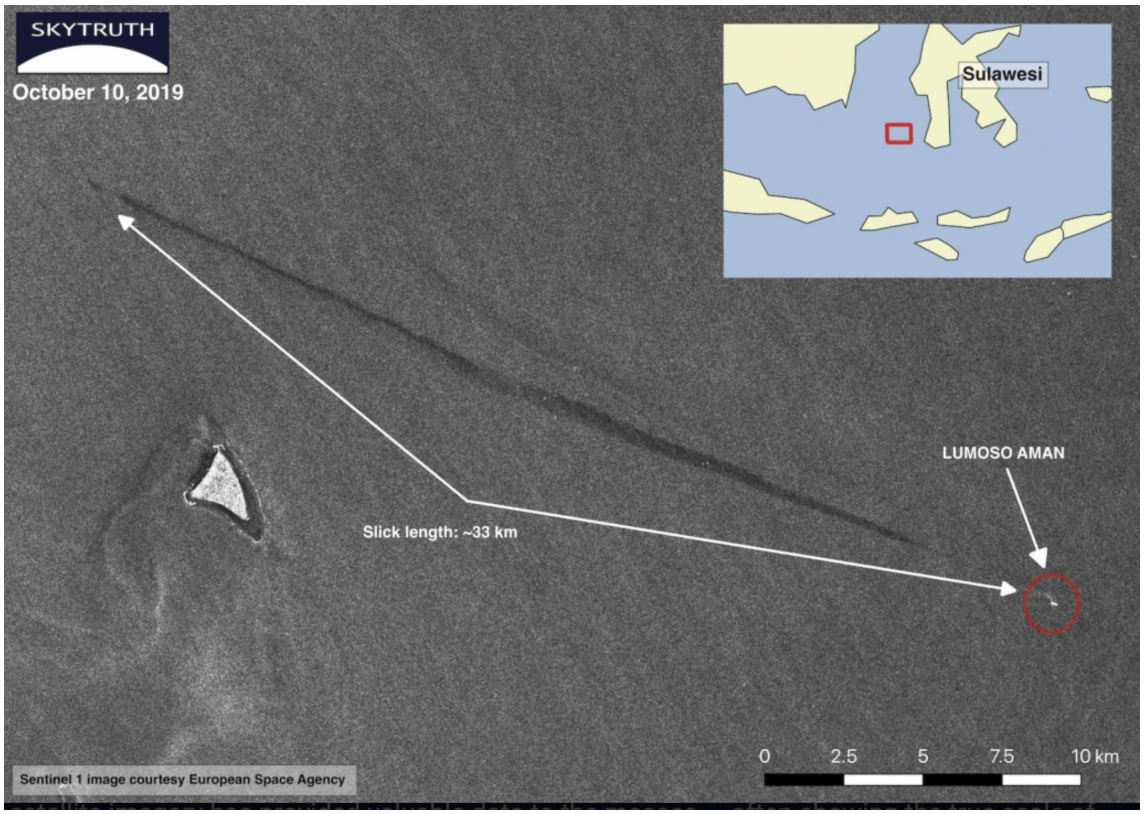


Figure 1.1: Example of bilge dumping (from [1])

- The development of a segmentation/classification system that can automatically detect oil spills from SAR images, including many other look-alikes.

1.2 Structure of the thesis

This thesis is organized as follows. After this introduction, Chapter 2 will present a state-of-the-art in the convolutional neural networks and advanced networks for image segmentation, representing the main theoretical concepts on which this thesis is based. The next chapter addresses the critical problem of the dataset. We will explain how we created a specific oil spill dataset for this application. In Chapter 4, the models for classification are presented, focusing on their parameters and implementations. Chapter 5 details and discusses the results of the simulation. This thesis ends with the conclusion and some perspectives that would be interesting to implement if we had enough time or if this work were to be continued.

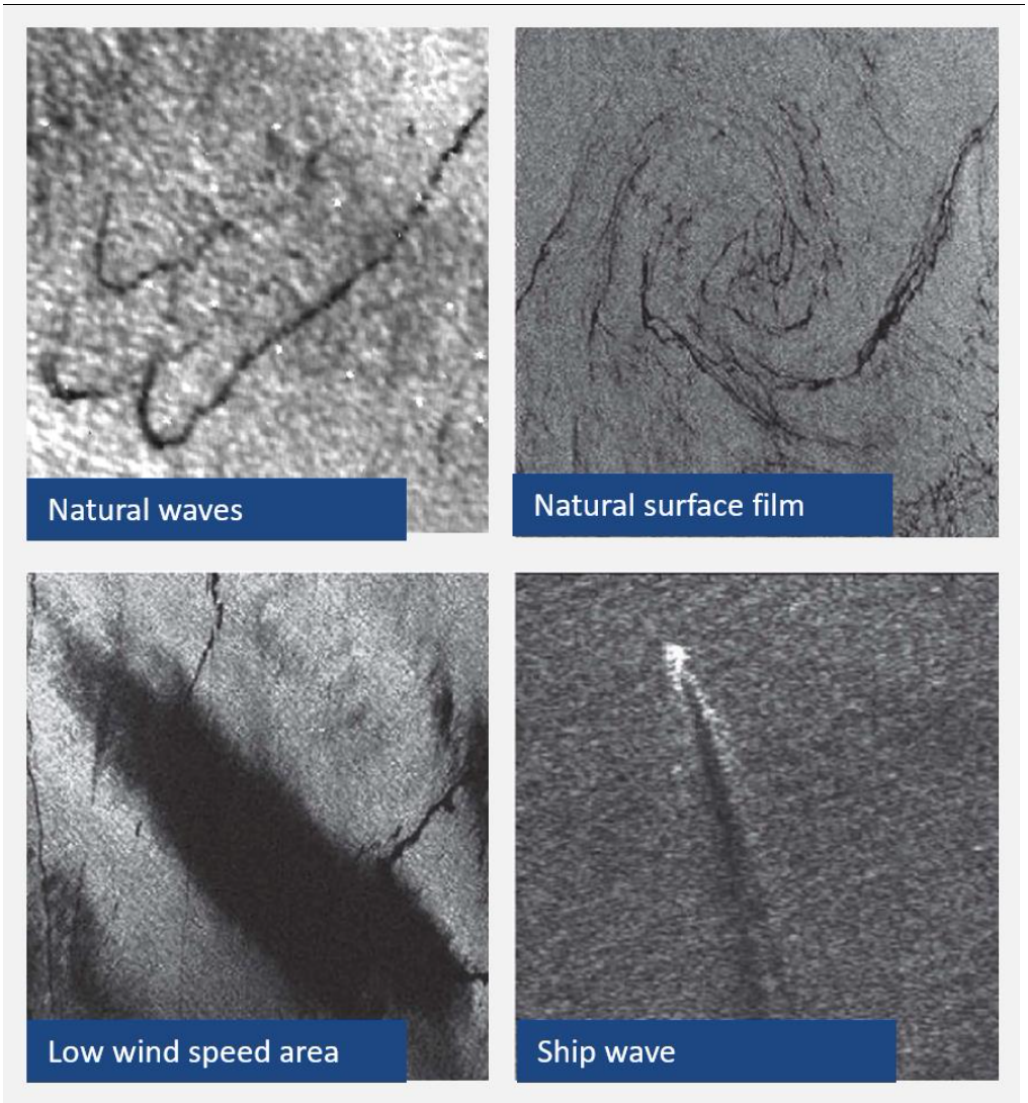


Figure 1.2: Example of typical look-alikes (from [2])

Chapter 2

State of the Art

This chapter presents the main concepts used in this thesis. We will also give the reader an overview of some advanced classification systems to put in perspective the results of our work. Section 2.1 provides an overview of the classical Neural Network. The Convolutional Neural Networks, which are the core of our thesis, are described in sections 2.2 and 2.3. Then, section 2.4 focuses on the training of the network. Some recent developments on segmentation are presented in section 2.5. Finally, section 2.6 presents a synthesis of published works in oil spill detection.

2.1 Introduction to Artificial Neural Networks

The concept of Artificial Neural Networks (ANNs) is linked to the biological nervous system. The ANNs try to emulate the basic operations of the biological neurons to efficiently perform tasks as humans do, for example, image recognition. An essential aspect of the ANNs is their ability to learn to handle new problems.

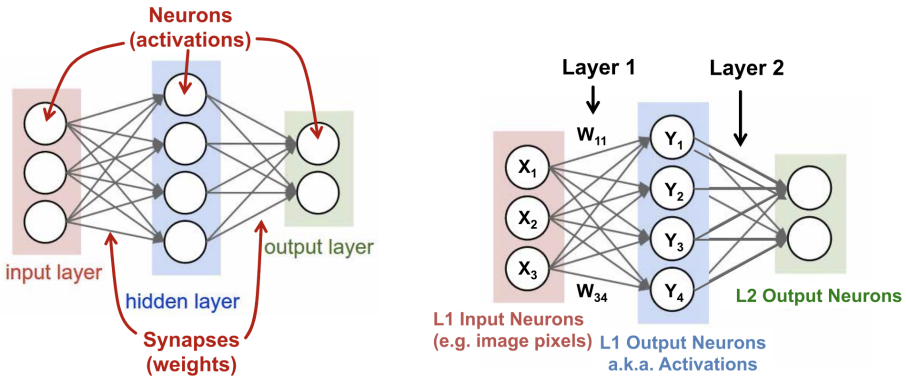
The artificial neuron operates in the following way: it computes the weighted sum of its inputs and then applies a non-linear function to the output of the sum.

$$y_j = f\left(\sum_{i=1}^{3} W_{ij} \times x_i + b\right)$$

where W_{ij}, x_i, and y_i are respectively the weights, inputs and outputs. The term b represents the bias.

Fig 2.1(a) shows the general organization of an ANN where we have three layers: the input layer, the middle layer, which is commonly called a hidden layer, and the output layer. Fig 2.1(b) represents the operation at each layer.

Different non-linear functions have been proposed in the literature. Fig 2.2 presents the most classic. The Rectified Linear Unit (ReLU) and its variations are often used due to its simplicity and its performance [22] [23] [24] [25].



(a) (b)

Figure 2.1: Simple ANN architecture and operation (from [3]).

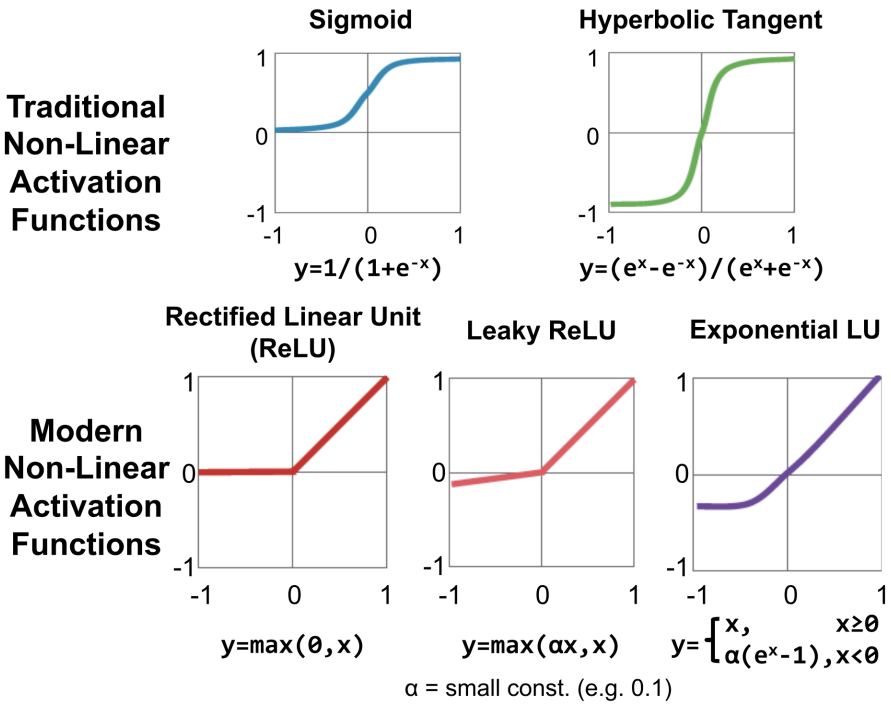


Figure 2.2: Main non-linear functions (from [3]).

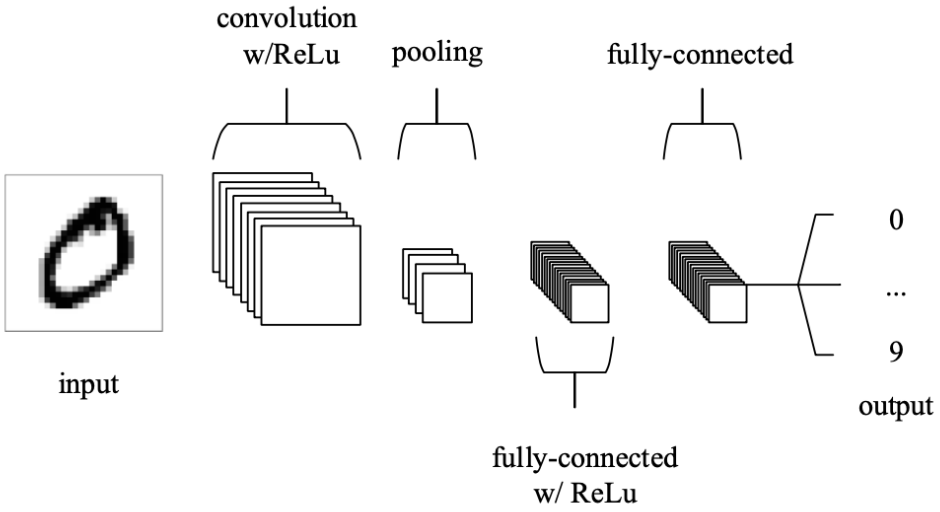


Figure 2.3: A simple CNN architecture (from [4]).

A recent area of research called Deep Learning refers to ANNs where the network has more than one hidden layer. These networks are generally noted as Deep Neural Networks (DNNs). They can perform more complex tasks than the traditional 3-layer network.

The learning process determines the value of the weights and biases and is referred to as the *training*. We use the term *inference* when the network is trained and no longer updates its weights.

2.2 Convolutional Neural Networks

For several years, Convolutional Neural Networks (CNNs) have been an enormous success for image or voice processing (e.g., detection, segmentation, and classification). A CNN is built on four types of layers:

- The input layer stores the input pixels (in case of image recognition).
- The convolutional layers compute some features.
- The pooling layers reduce the spatial dimensionality.
- The fully-connected layers (FC) classify and produce class scores.

For example, a simple CNN architecture with only five layers is given in Fig 2.3.

2.2.1 Convolutional Layer

The convolutional layer computes a 2-dimensional feature called the Feature Map (FM). Successively, the convolutional layers produce higher-level features.

A filter (also called Kernel) is applied to the input feature map. Each pixel of the selected window having the same size as the kernel is multiplied by its corresponding

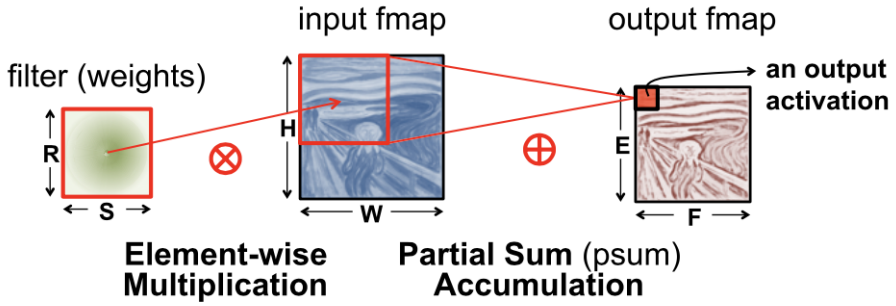


Figure 2.4: Illustration of the 2-D convolution (from [3]).

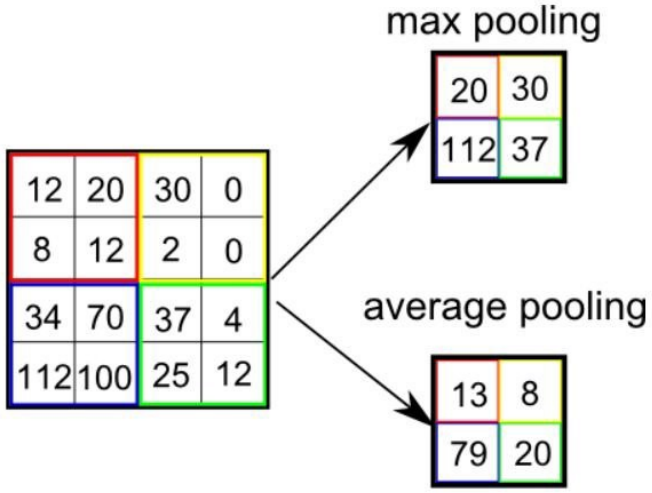


Figure 2.5: Example of Max and Average Pooling (from [5]).

kernel value. The output is then obtained by summing the product values. The 2-D convolutional operation is illustrated in Fig 2.4.

The stride parameter defines the amount of overlap between the windows. A higher value for the stride will produce an FM with lower spatial dimensions.

2.2.2 Pooling Layer

The objective of the Pooling Layer is to reduce the dimensionality of the output of the convolutional layer. It also makes the network more robust against small translation, rotation, or distortion. Classically, it is inserted between two consecutive convolutional layers.

Pooling computes an output pixel from a set of values from its inputs (e.g., 2x2 window). Many different operations of pooling have been proposed. Fig 2.5 illustrates two of them (Max and Average operations). Max Pooling is generally preferred because it eliminates noise together with dimensionality reduction [26].

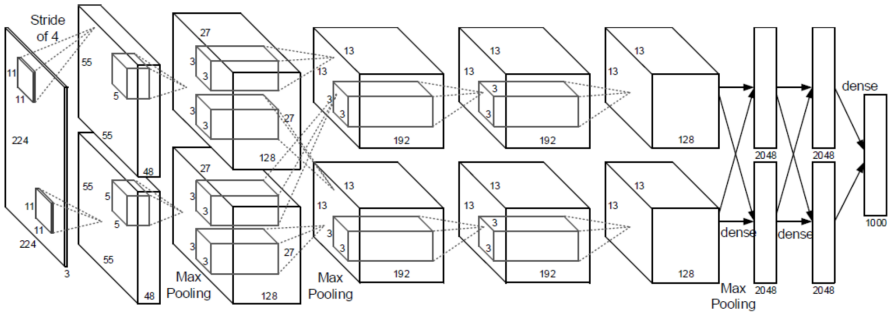


Figure 2.6: Architecture of the AlexNet (from [6]).

2.2.3 Fully-connected Layer

The CNN ends with fully-connected layers (FC) that take as input the various features computed by the convolutional layers. These FC layers act as a non-linear classifier and produce single value outputs.

2.3 Overview of some common CNNs

One of the first CNN is a network dedicated to handwritten digit recognition [27]. LeNet-5 [28] implements a multilayer neural network trained with a backpropagation algorithm. The NN was designed to recognize machine-print documents. The classifier was initially trained on 500.000 images, and a performance of 82% was obtained. The architecture of LeNet-5 is based on two convolutional layers with filters of size 5x5 with an average pooling of 2 x 2 after each convolution layer. LeNet-5 has a complexity of 60000 weights and requires 341000 multiply-and-accumulation (MACs) per image.

AlexNet [6] is a critical CNN because it was the first to win the ImageNet competition in 2012. ImageNet is a database of 15 million images with approximately 22000 categories. The architecture of AlexNet is represented in Fig 2.6. It implements five convolutional layers and 3 FC layers at the end. Each layer uses a ReLU nonlinearity that allows training several times faster than their equivalents with the tanh activation function (3). An error rate of 16.4% was obtained for IRSVRC-2012 (ImageNet Large-Scale Visual Recognition Challenge [29]). Compared with LeNet5-5, we can observe that the number of weights is much higher.

Overfeat [30] is very similar to AlexNet regarding the number of layers. However, the number of filters is higher, which induces an important number of weights and, consequently, the number of MACs. This network was applied to the ILSVRC-2013 challenge and ranked 4th in classification, 1st in localization, and 1st in detection [30].

The researchers continued to propose deeper CNNs while maintaining the cost (or the complexity) to an acceptable level. VGG-16 [31] has 16 layers (13 convolutional and 3 FC layers). Larger filters are used to compensate for the increasing number of convolutional layers). Based on their results, the authors confirm the importance of depth for image classification [31]. VGG obtained an error rate of 7.4% for classification for ILSVRC-2014 [29].

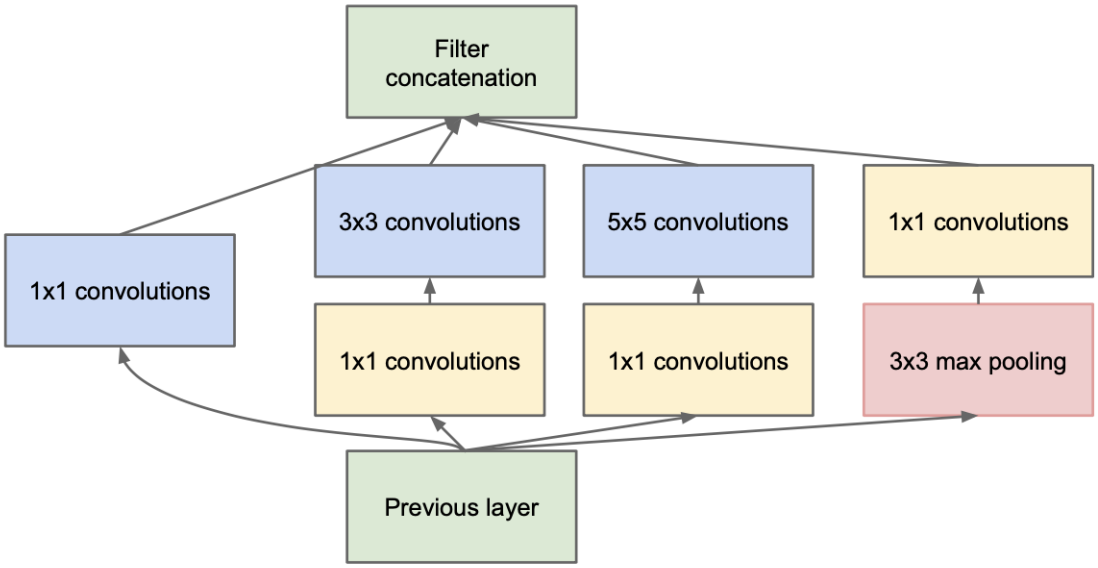


Figure 2.7: Inception module with dimension reductions (from [7]).

GoogLeNet (or Inception) [7] introduced for ILSVRC-2014 has 22 layers and includes a new module called inception module (Fig 2.7). This module is characterized by parallel connections, while the other networks were built on a serial connection. Different versions have been introduced to improve the error rate [32] [33].

ResNet [34] goes one step further with 50 or more layers. With such a high number, accuracy gets saturated and then decreases. This problem is related to the training, where it is difficult to maintain a sufficient gradient to update the weights correctly in the deep layers. ResNet introduces a shortcut module (Fig 2.8) that creates an identity connection to solve this issue. It works in the following way: at the beginning of the training, F(x) is equal to 0, and H(x) is equal to x. Then, progressively, the actual forward connection through the weight layer is used.

Various versions of ResNet with depths up to 152 have been proposed. Resnet-152 was the winner of ISLVRC-2015.

Table 2.1 summarizes the main features of the different CNNs presented in this section. The accuracies given in this table are based on Top-5 error on ImageNet [29].

2.4 Training

Training is an essential phase for the DNNs. It mainly comprises three steps:

- **Initialization**: the first task is to initialize the weights of the network. Different approaches are proposed and discussed in the literature [35] [36]. The most standard method is to initialize the weights randomly.
- **Forward propagation**: the training data is applied to the network inputs, and an output vector is computed.
- **Back propagation**: the output vector is compared with the target vector

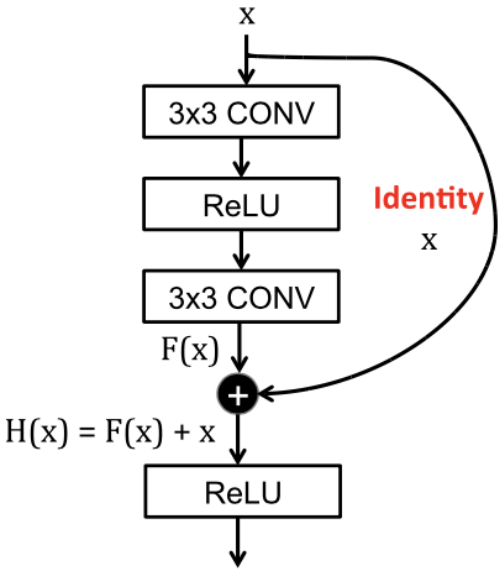


Figure 2.8: Shortcut module from ResNet (from [3]).

Metrics	LeNet 5	AlexNet	Overfeat fast	VGG 16	GoogLeNet v1	ResNet 50
Top-5 error†	n/a	16.4	14.2	7.4	6.7	5.3
Top-5 error (single crop)†	n/a	19.8	17.0	8.8	10.7	7.0
Input Size	28×28	227×227	231×231	224×224	224×224	224×224
# of CONV Layers	2	5	5	13	57	53
Depth in # of CONV Layers	2	5	5	13	21	49
Filter Sizes	5	3,5,11	3,5,11	3	1,3,5,7	1,3,7
# of Channels	1, 20	3-256	3-1024	3-512	3-832	3-2048
# of Filters	20, 50	96-384	96-1024	64-512	16-384	64-2048
Stride	1	1,4	1,4	1	1,2	1,2
Weights	2.6k	2.3M	16M	14.7M	6.0M	23.5M
MACs	283k	666M	2.67G	15.3G	1.43G	3.86G
# of FC Layers	2	3	3	3	1	1
Filter Sizes	1,4	1,6	1,6,12	1,7	1	1
# of Channels	50, 500	256-4096	1024-4096	512-4096	1024	2048
# of Filters	10, 500	1000-4096	1000-4096	1000-4096	1000	1000
Weights	58k	58.6M	130M	124M	1M	2M
MACs	58k	58.6M	130M	124M	1M	2M
Total Weights	60k	61M	146M	138M	7M	25.5M
Total MACs	341k	724M	2.8G	15.5G	1.43G	3.9G

Table 2.1: Overview of the main features for some common CNNs (from [3]).

using a loss function. The weights are then updated to minimize this loss function.[37] gives a good overview of the main algorithms with a compromise between performance and complexity.

- *Batch gradient descent* computes the gradient of the loss function for the complete training dataset. The weights are adjusted in the direction of the gradients according to a defined learning rate.
- *Stochastic gradient descent* updates a weight for each training data. This results in much less computation but also more fluctuation and makes convergence more difficult.
- *Mini-batch gradient descent* is a compromise between batch gradient descent and stochastic gradient. It updates the weights for every mini-batch of n training data (n between 16 and 256 typically).

2.5 Image Segmentation

Image segmentation has been a critical area of research for a long time, and recent developments in CNNs have enabled results that could not be achieved with traditional techniques such as thresholding, region-based methods, edge detection, clustering algorithms, ...

Image segmentation can be classified into three types depending on the output of the segmentation. We will use the images in Fig 2.9 to explain and illustrate these three kinds of segmentation.

- **Semantic segmentation**: the pixels of an image are classified into semantic classes. In Fig 2.9, we have four classes: the sky, the sea, the people, and the ground.
- **Instance segmentation**: the relevant information (in our example, the people) is extracted from the background (in our example, the background is composed of the sky, the sea, and the ground). This information is then segmented into different instances (e.g., each person)
- **Panoptic segmentation** is the combination of both semantic and instance segmentation. Each instance of an object is determined and classified according to its semantic class.

It is out of the scope of this work to give a complete overview of all the segmentation techniques recently published. We will focus on some architectures to illustrate the evolution of this domain.

In 2015, J. Long et al. showed that a fully conventional network could efficiently perform semantic segmentation [9]. The idea, illustrated in Fig 2.10, is to adapt standard CNNs previously described in this chapter (AlexNet, VGG net, and GoogLeNet) to the segmentation. FCN-VGG16 obtains the best score of mean IU of 56.0 (the mean of Intersection over Union) on the validation set of PASCAL VOC 2011 (dataset containing 2223 images for training with 5034 objects and 1111 images with 2028 objects for testing [38]). Based on these encouraging results, J. Long et al. proposed a new network (FCN) that combines coarse, high layer information with

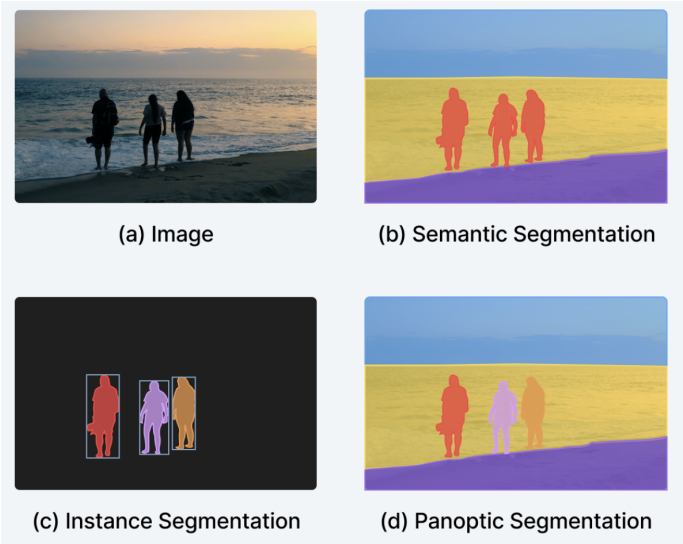


Figure 2.9: Three types of segmentation (from [8]).

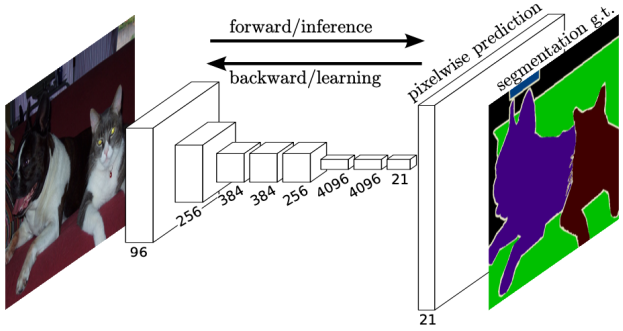


Figure 2.10: Fully conventional network for semantic segmentation (from [9]).

fine, low layer information [9]. The architecture of this network with its different alternatives is represented in Fig 2.11. FCN-8s improves the mean IU to 62.7 and 62.2 respectively for VOC-2011 and VOC-2012.

FCN-based networks showed limitations in handling complex scene segmentation. This is especially true for the new ADE20K dataset [39][40], where the number of objects and parts is much greater.A new pyramid scene parsing network (PSPNet) was proposed in 2017 to tackle these difficulties [10]. Fig 2.12 presents an overview of the PSPNet. First, a CNN generates the feature map; then, the pyramid pooling module extracts the representation of sub-regions. An upsampling and concatenation layers generate the final feature presentation, which a CNN converts to the final pixel-level prediction. PSPNet obtains a mean IU of 85.4 for PASCAL VOC 2012 and 80.2 for Cityscapes (a dataset of urban scenes [41]) which represents a significant improvement in comparison with FCN-8s performances.

In 2019, A. Kirillov et al. proposed the concept of *Panoptic segmentation*, as explained above [11]. They also define a new metric called PQ (Panoptic Quality) to represent performance segmentation for all classes (stuff and things). PQ measures the quality of the predicted panoptic segmentation relative to the ground truth. A. Kirillov et al. define that the condition for a predicted segment (p) to match a

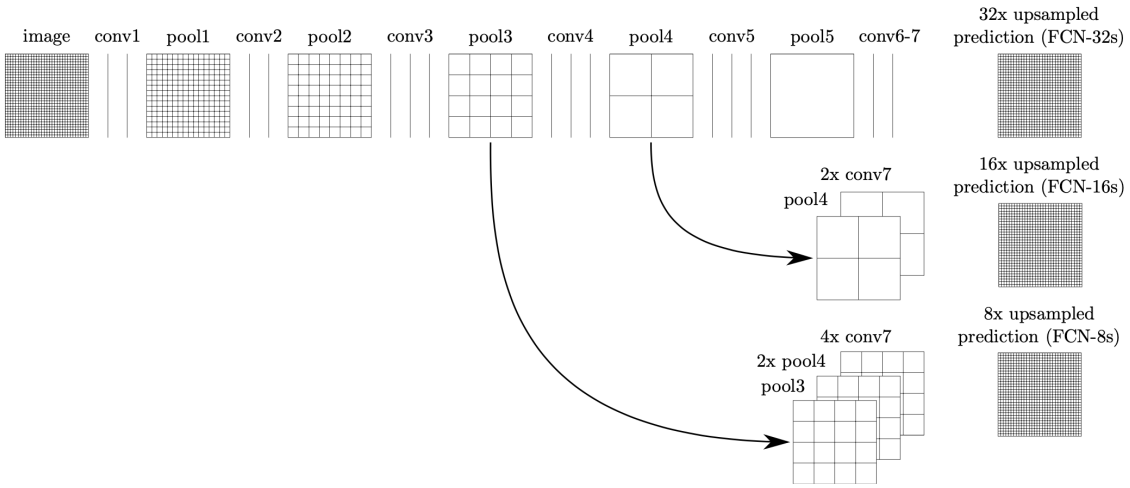


Figure 2.11: Architecture of the FCN network (from [9]).

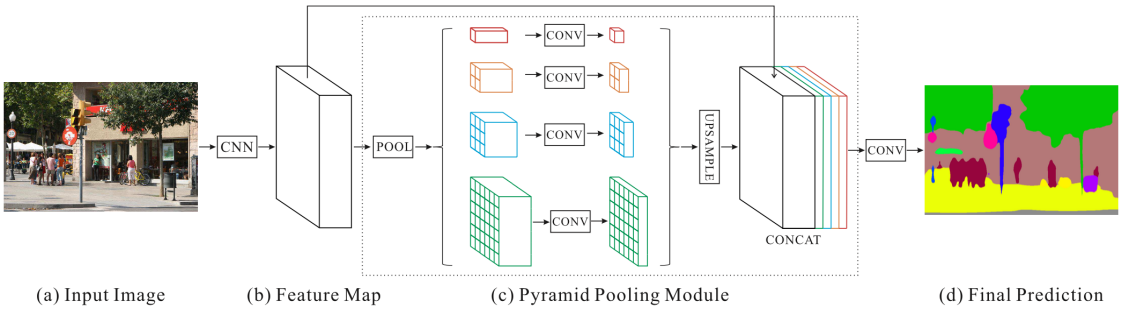


Figure 2.12: Architecture of the PSPNet (from [10]).

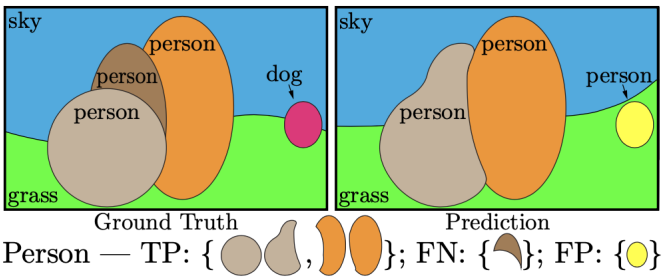


Figure 2.13: Illustration of the condition for pairs of segments to match. The pairs of segments for the class person having the same color have an IU greater than 0.5 (from [11]).

ground truth segment (g) is that their IU is greater than 0.5. PQ is computed for each class and then averaged. It is based on the determination of three parameters: TP (true positive), the number of matched pairs of segments, FP (false positive), the number of unmatched predicted segments, and FN (false negative), the number of unmatched ground truth segments. This concept is illustrated in Fig 2.13.

Given these definitions, PQ is defined as:

$$\mathrm{PQ} = \frac{\sum_{(p,g)\in TP} \mathrm{IU}(p,g)}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}$$

This equation can be rewritten to put in evidence the segmentation quality (SQ) and the recognition quality (RQ):

$$\mathrm{PQ} = \underbrace{\frac{\sum_{(p,g)\in TP} \mathrm{IU}(p,g)}{|TP|}}_{\text{segmentation quality (SQ)}} \times \underbrace{\frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}}_{\text{recognition quality (RQ)}}$$

Panoptic-DeepLab, introduced recently (2020), proposes a bottom-up panoptic segmentation [12]. Its block diagram is represented in fig 2.14 and includes a separate network for semantic and instance segmentation. While many instance segmentation systems are based on keypoint detection [42] [43], Panoptic-DeepLab only requires the prediction of centers.

The detailed architecture of Panoptic-DeepLab is given in Fig 2.15. Each segmentation branch has an ASPP (Atrous Spatial Pyramid Pooling [44]) to predict context. A the end, panoptic representation is obtained by a « majority vote ». Detailed results on different datasets can be found in [12]. In particular, for Cityscapes, they get a PQ of 65.5 on the test set which is a new state-of-the-art at that time.

In December 2021, Ghasemzadeh et al. presented DeepSportLab, a new unified framework for ball detection, player instance segmentation, and pose estimation in team sports scenes [13]. An overview of DeepSportLab is given in Fig 2.16. Two main features are a single CNN to detect the ball and predict the player's poses and 19 keypoint types: 17 for human body parts, 1 for ball centroid, and 1 for player centroid.

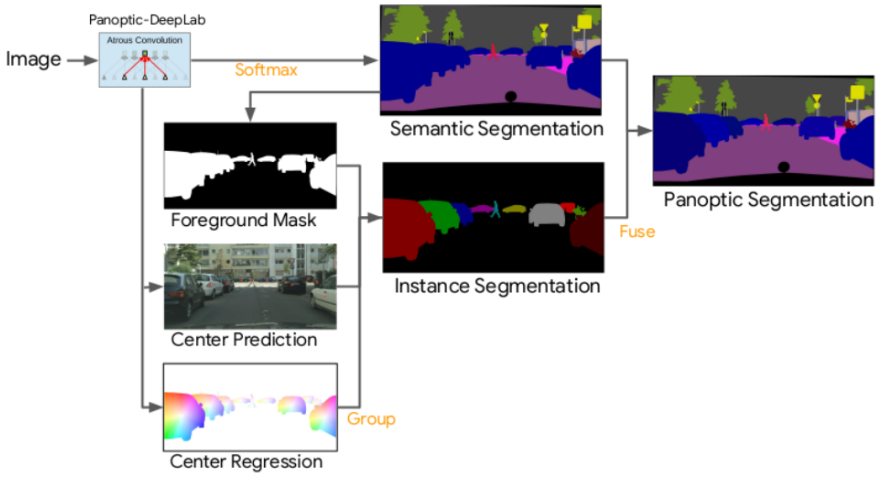


Figure 2.14: Panoptic-DeepLab block diagram (from [12]).

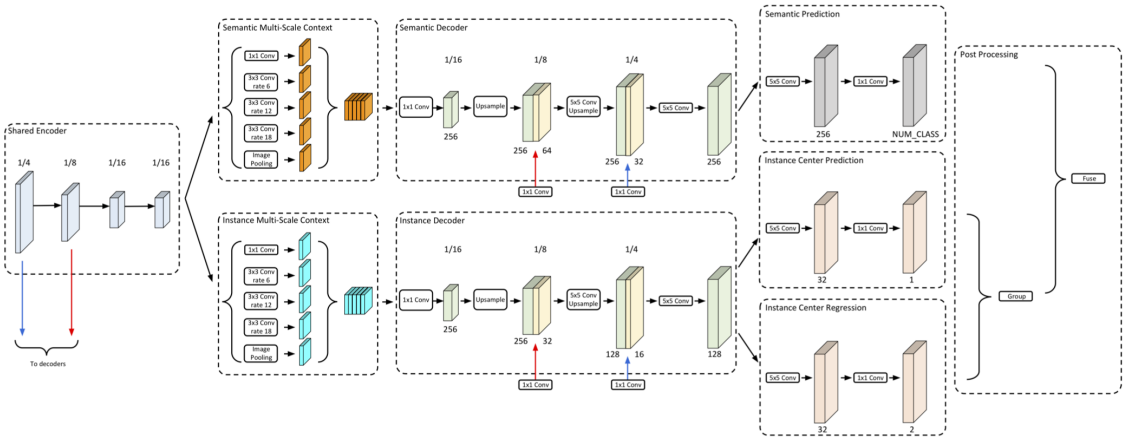


Figure 2.15: Panoptic-DeepLab Architecture (from [12]).

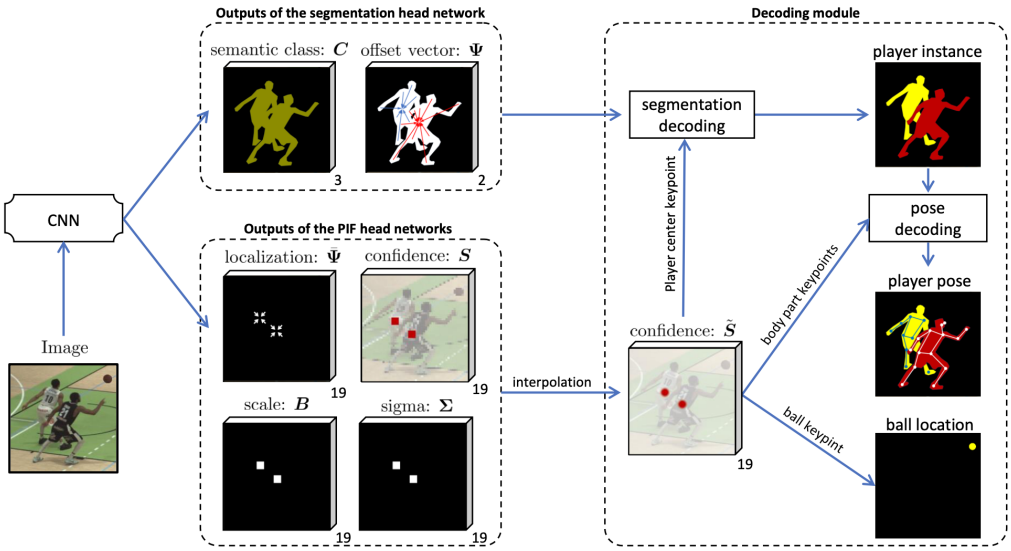


Figure 2.16: Overview of DeepSportLab (from [13]).

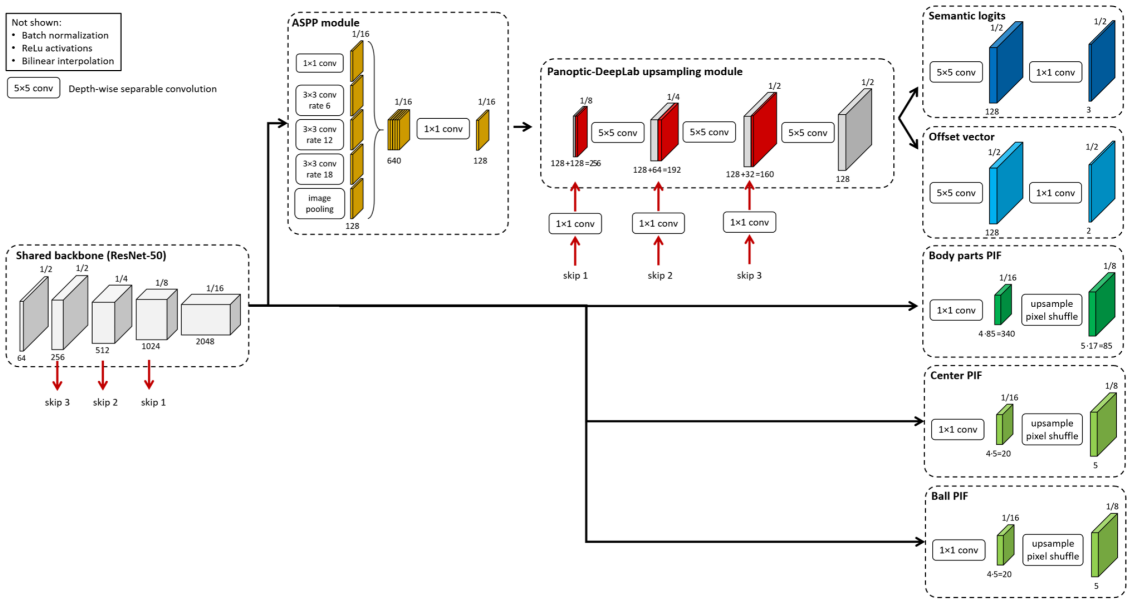


Figure 2.17: Architecture of DeepSportLab (from [13]).

The CNN architecture of DeepSportLab is presented in Fig 2.17. It has a ResNet-50 network connected to 4 head networks. The first is built from an ASPP followed by an upsampling module from Panoplic-DeepLab connected to two prediction modules for semantic logits and offset vectors. The three other head networks generate the PIF (Part Intensity Field) parameters for the 19 keypoint types. It is interesting to note that this system uses several concepts we have presented throughout this chapter.

DeepSportLab has been trained on the COCO [45] and DeepSport [46] datasets. The authors show that DeepSportLab compares favorably against other models such as Panoptic-DeepLab in terms of quality measures while reducing by a factor 3 the required memory [13].

2.6 Oil Spill Detection

The domain of oil spill detection is not new, and we have found some publications related to this topic, mainly in IEEE conferences and in the Remote Sensing journal. However, this field of research suffers from the fact that there is no public dataset from which researchers can compare their results. There are briefly presented in chronological order (from 2016 to 2021). For some of them, we provide the block diagram of the network to give the reader an idea about the complexity of the network without explaining any details.

- A classical approach is proposed [47] in with three main steps: the segmentation stage, the features extraction stage, and the classification stage using a Mahalanobis metric.
- The interest of Deep Neural Networks for quad-polarimetric SAR image recognition is explored in [48].
- A specific CNN-based model is proposed in [14]. This network is similar to

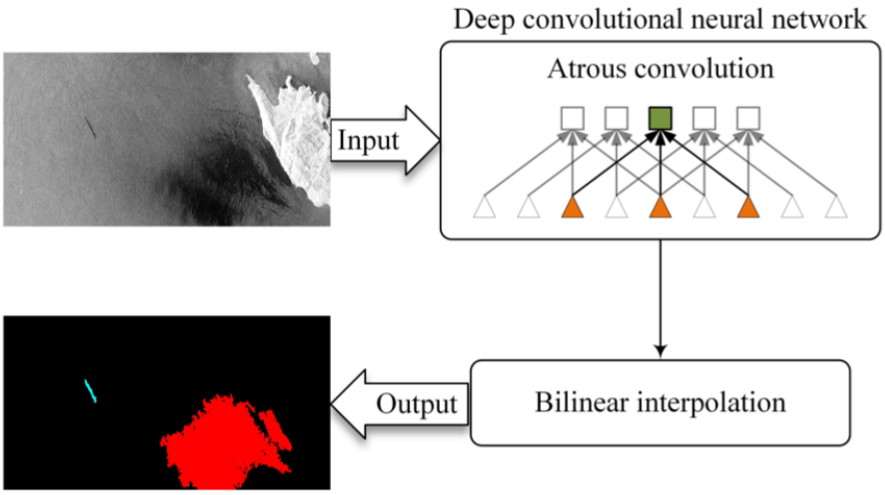


Figure 2.18: CNN with Atrous convolution (from [14])

DeepLab with atrous convolution and ASPP (Atrous Spatial Pyramid Pooling) (Fig 2.18).

- In [15], the SAR image recognition is based on a CNN with five layers and an FC classification (Fig 2.19).
- Five models for semantic segmentation are compared on their own dataset in [49]:
  - U-Net [16], which is an extension of a fully connected network with an encoder and a decoder parts (Fig 2.20)
  - Link-Net [17], which also includes an encoder-decoder structure (Fig 2.21)
  - PSPNet (already described above)
  - DeepLabv2 and v3 (already described above)

  DeepLabv3 achieves the best performance of accuracy on the test set, according to the authors.
- The oil spill detection method of [18] is again based on quad-polarimetric SAR images (Fig 2.22). A special feature is the use of SLIC (Simple Linear Iterative Clustering) Super-Pixel introduced by [50].
- OSCNet (Oil Spill Convolutional Network) is a deep convolutional neural network optimized for oil spill detection [19]. It is based on VGG-16, and its architecture and hyperparameters have been tuned to the SAR images dataset (Fig 2.23).
- A dedicated network called CBD-Net (for boundary-supervised detection network) is proposed in [20]. It includes residual modules, scSE (Parallel spatial and channel squeeze excitation [51]) attention blocks, multi-parallel dilated convolution module, and a four-decoder block (Fig 2.24). The performance of CBD-Net is compared with that of U-Net, Link-Net, and DeepLabv3, and a significant improvement is observed.

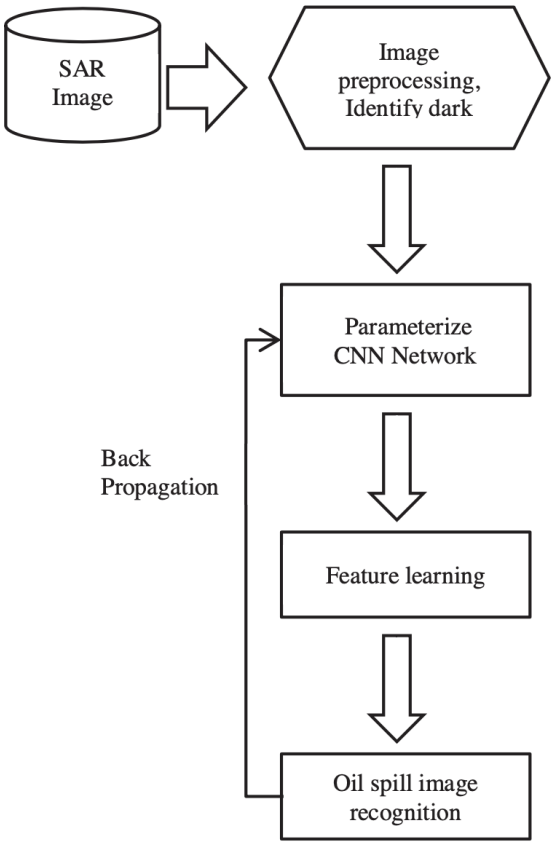


Figure 2.19: Block diagram of the oil spill detection (from [15])

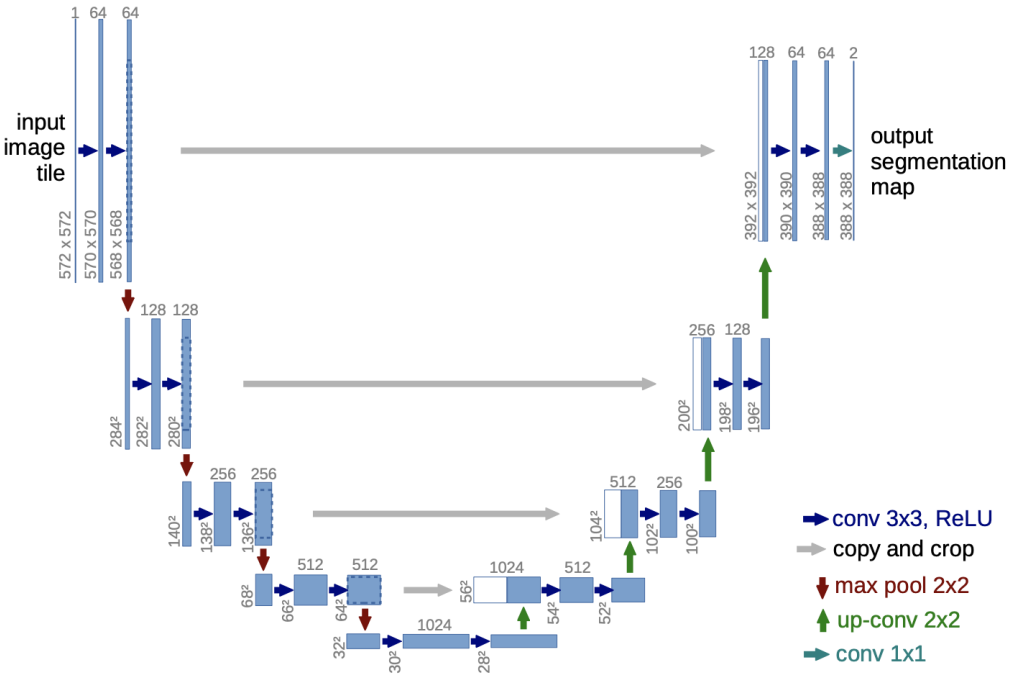


Figure 2.20: U-Net architecture (from [16])

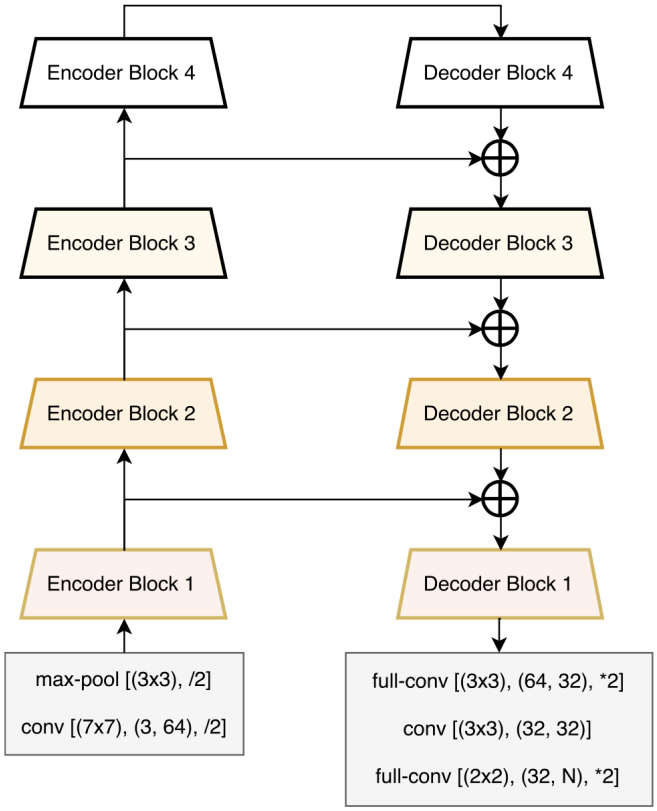


Figure 2.21: LinkNet architecture (from [17])

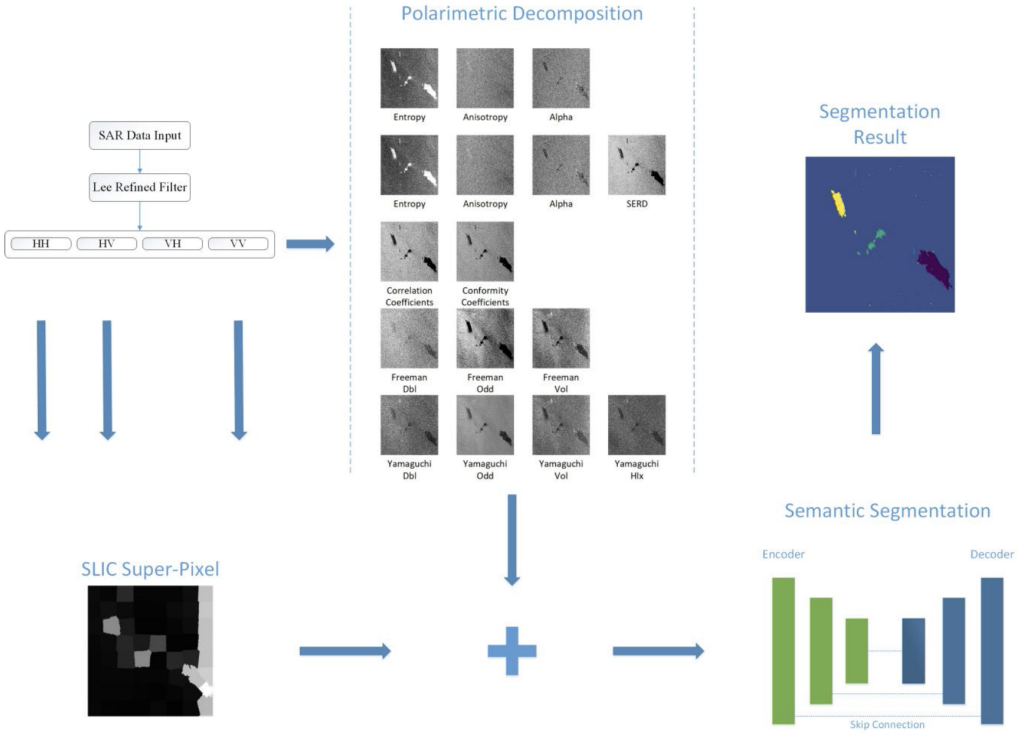


Figure 2.22: Overview of the oil spill detection system for quad-polarimetric SAR images (from [18])

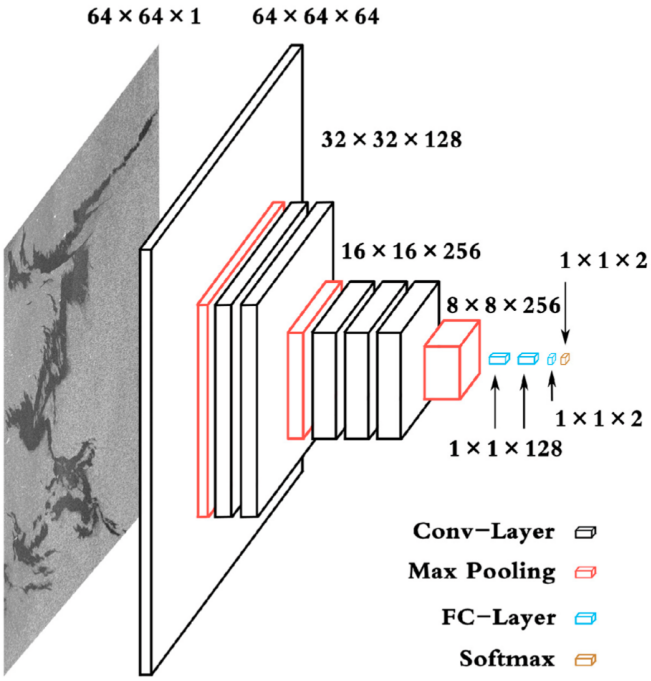


Figure 2.23: OSCNet architecture of OSCNet(from [19])

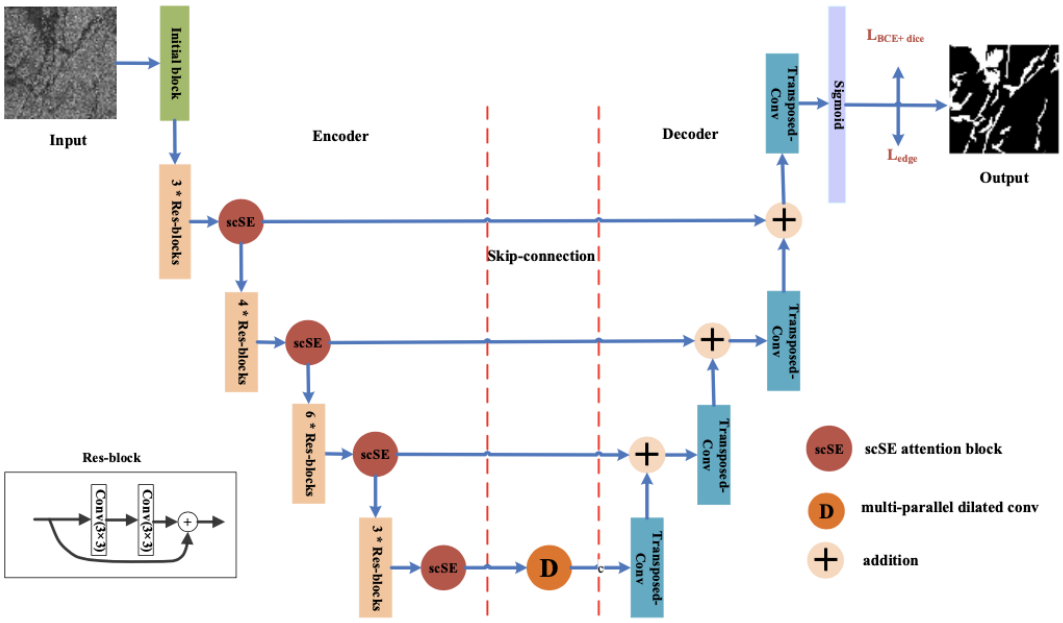


Figure 2.24: Architecture of CBD-Net (from [20])

Chapter 3

Oil Spill Dataset

To our knowledge, there is no dataset dedicated to oil spill detection accessible to the research community. The first task of this work was thus to create such a dataset with ground truth masks. We also decided to slice the images into tiles to facilitate the classification and localization of the oil spills. The output of slicing is the dataset we will use for the CNN classification.

3.1 Image Gathering

Aerospacelab worked before me on this problem and managed to find relevant information about oil spill pollution. They were able to access the Sentinel-1 image database and gather 67 images with oil spills.

3.2 Image Preprocessing

The SAR images provide from the Sentinel-1 satellite. These images cannot be used directly and must first be preprocessed.

The image preprocessing was realized at Aerospacelab. We only list the main blocks [2]:

- Orbit file correction
- Radiometric calibration
- Speckle filtering
- Land deleting
- Conversion for dB to real luminosity
- Downscaling
- Splitting and contrast adjusting

	Training	Test
Original images with oil spill	61	6
Data augmentation for images with oil spill	764	
Other images	2557	74
Total	3321	80

Table 3.1: SAR images dataset

The preprocessing was done using the SNAP (Sentinel Application Platform) tool from ESA [52] and output 8-bit grey level images of 1250 x 650 pixels. A detailed study of the SAR images preprocessing (Fig 3.1) is available in [21].

Examples of SAR images containing an oil spill and look-alikes are given in Fig 3.2 and 3.3.

3.3 Data Augmentation

As the number of images containing an oil spill is limited, increasing the training data set is necessary. The image transformations consist of rotation, horizontal and vertical shifts, up and downscaling, and flips in both directions (x). I worked on this task during my internship at Aerospacelab.

Table 3.1 summarizes the image dataset at the beginning of this thesis.

3.4 Image Annotating

The images do not include information on the oil spill localization, and a human identification has to be made for each image containing an oil spill. This represents 67 images to be annotated. We use the user-friendly tool COCO annotator [53] for this task. The output of the annotating is a ground truth binary mask. An example of such a mask with its original image is given in Fig 3.4.

3.5 Image Slicing

The size of the images is relatively large. Consequently, we decided to slice the original images into sub-images of size 125x130. We call these subimages tiles in this text.

The slicing operation is represented in Fig 3.5. We used two overlapping grids (10x5 blue grid and 9x4 red grid), and we got 86 tiles for each image.

For each tile, we have its corresponding ground-truth mask. We extract the number of pixels ($Npix$) representing an oil spill from this mask. If this number is higher than a threshold (Thr), we say that this tile contains a significant part of the oil spill. We will call this tile an *oil spill tile*. The other tiles will be called *No Oil Spill Tile*.

	Training	Test
Oil spill Tiles	792	61
No oil spill Tiles	4454	455

Table 3.2: Oil spill dataset for CNNs

Fig 3.6 gives some tiles with their parameter *Npix*.

The tiles will be submitted to the CNN, which will decide if it is an oil-tile or an empty-tile. The CNN will thus provide a binary decision.

Based on these decisions for each tile, we will be able to reconstruct the original image and localize the complete oil spill. The precision of the localization depends on the threshold *Thr*. Fig 3.7 gives the reconstructed image for three values for *Thr*. A too high value will crop the oil spill, and, on the opposite, a too-small value will reduce the precision of the localization.

A good compromise for this threshold is 100.

The slicing operation has only been applied to the original images containing an oil spill (61+ 6 images) to have a balanced dataset between the oil-tiles and the empty-tiles. Consequently, we will not use the original images that do not contain an oil spill and those coming from the data augmentation.

The dataset features that will be used subsequently are summarized in Table 3.2.

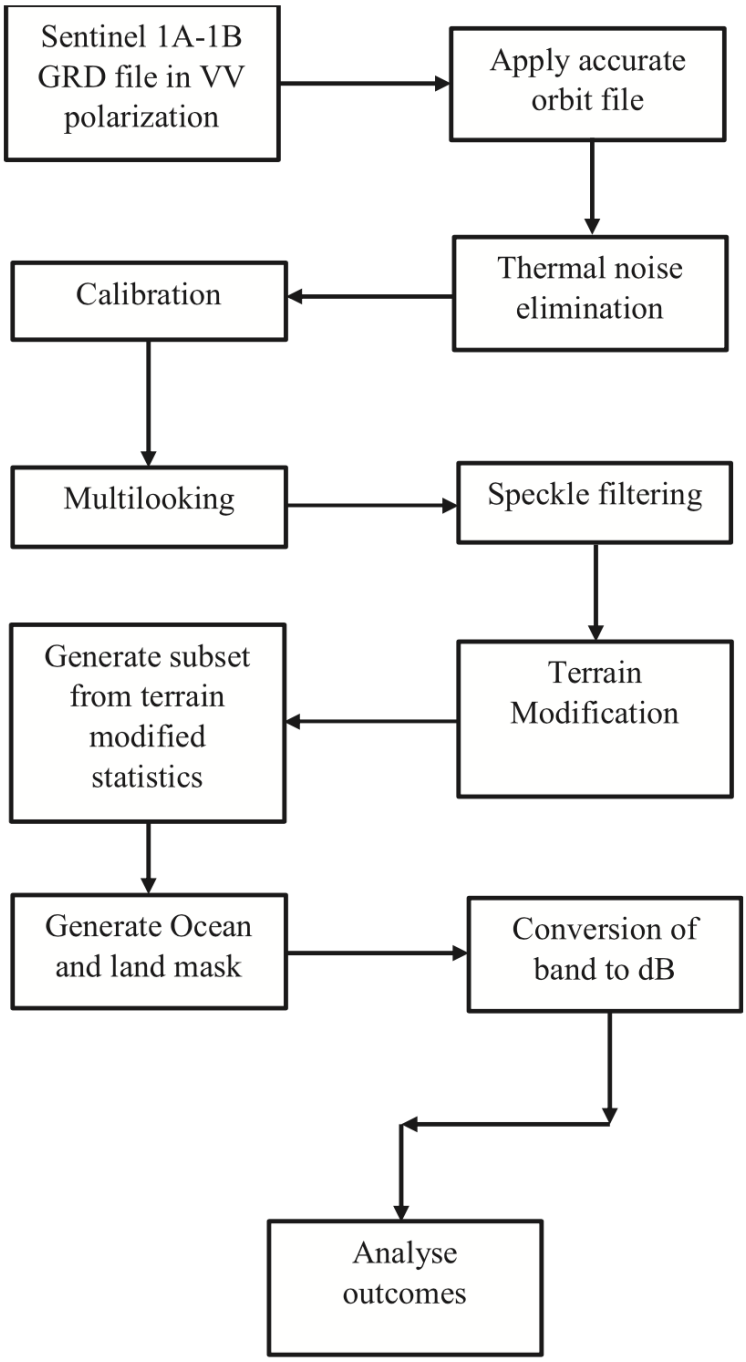


Figure 3.1: Block diagram of SAR images preprocssing (from [21])

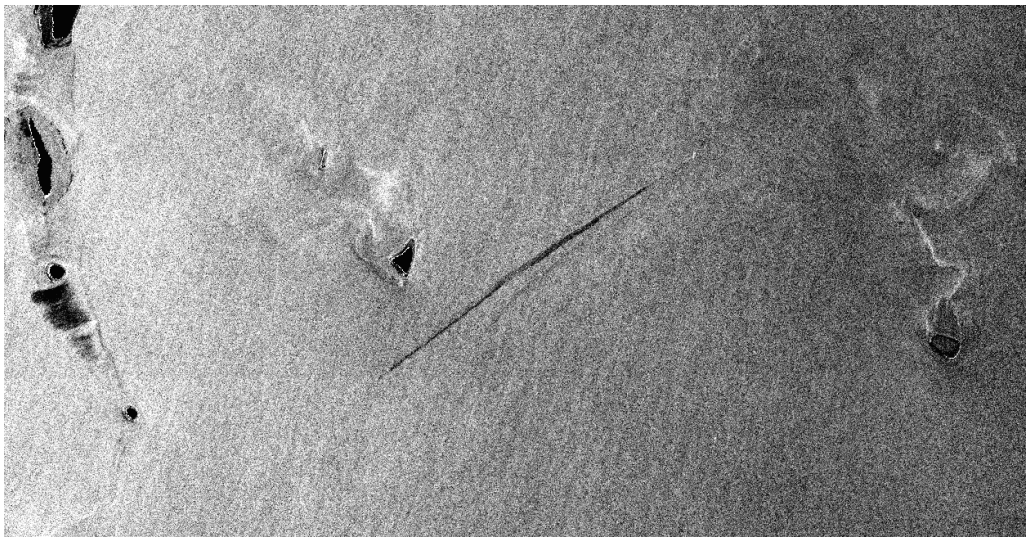


Figure 3.2: Example of SAR image with oil spill

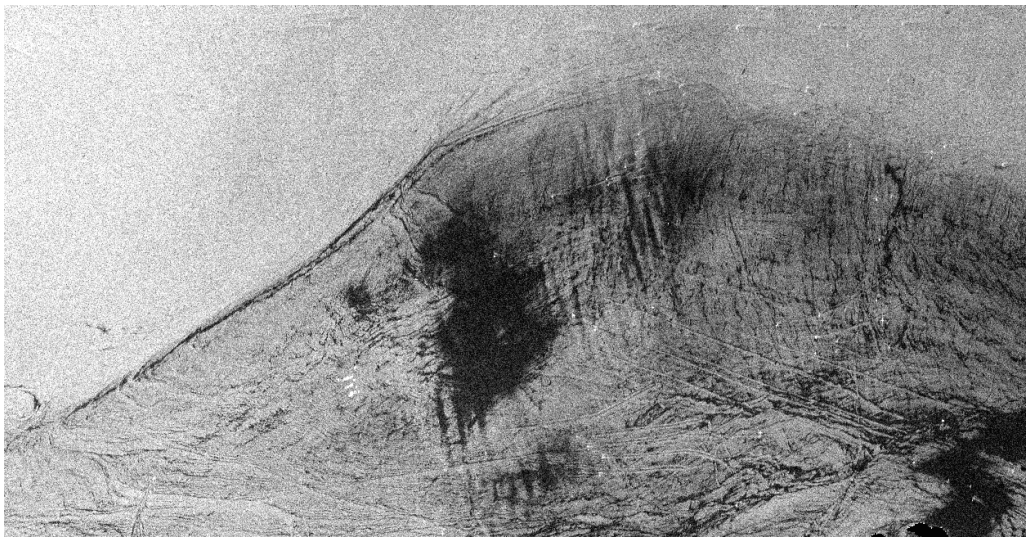


Figure 3.3: Example of SAR image with look-alikes

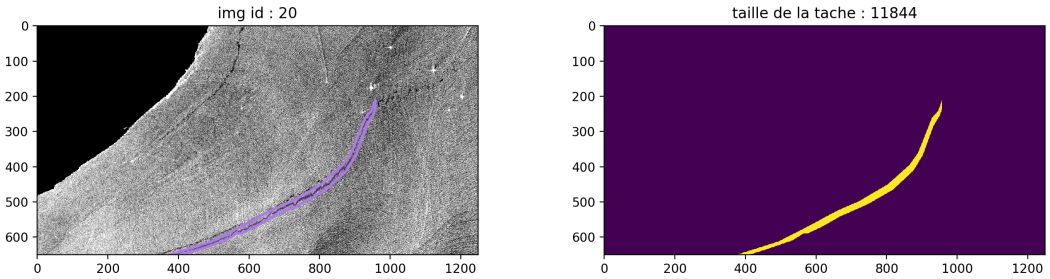


Figure 3.4: Illustration of the data annotating with at left the original and right the ground truth mask

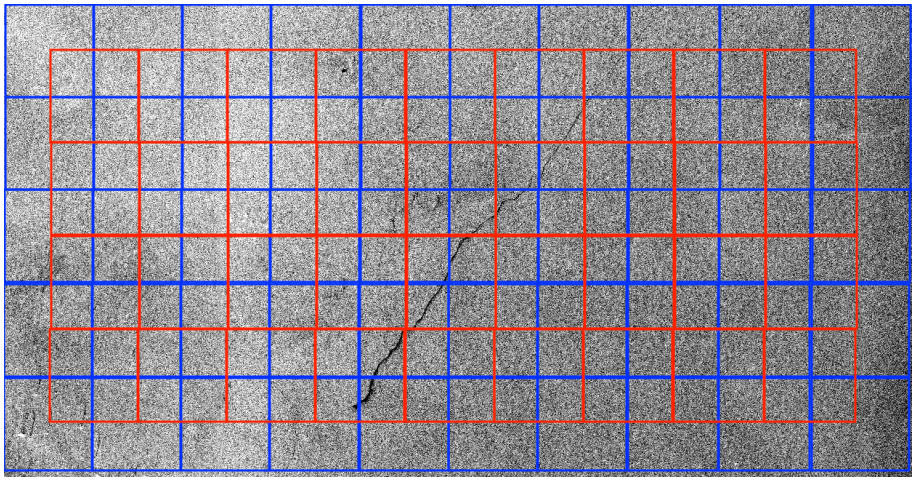


Figure 3.5: Illustration of Image slicing with overlapping grid

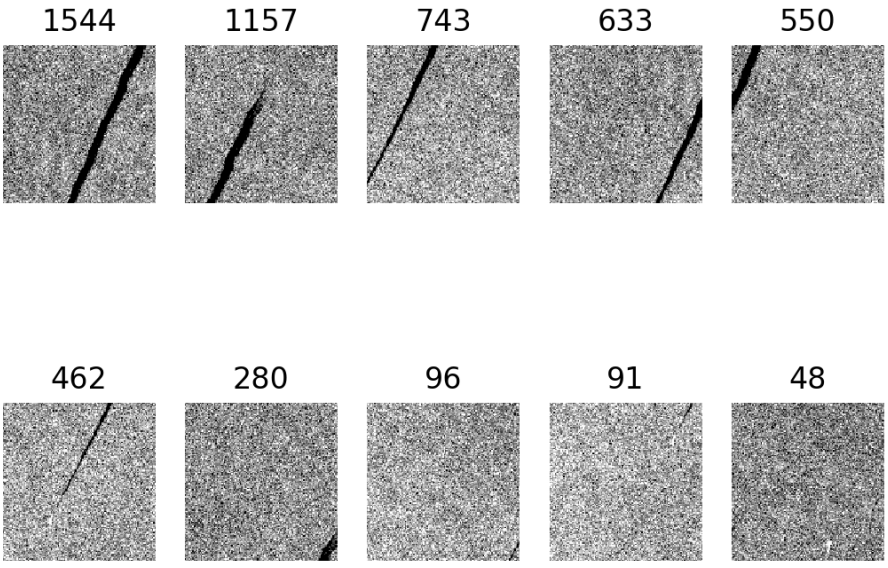


Figure 3.6: Some tiles including part of oil spill with their parameter *Npix*

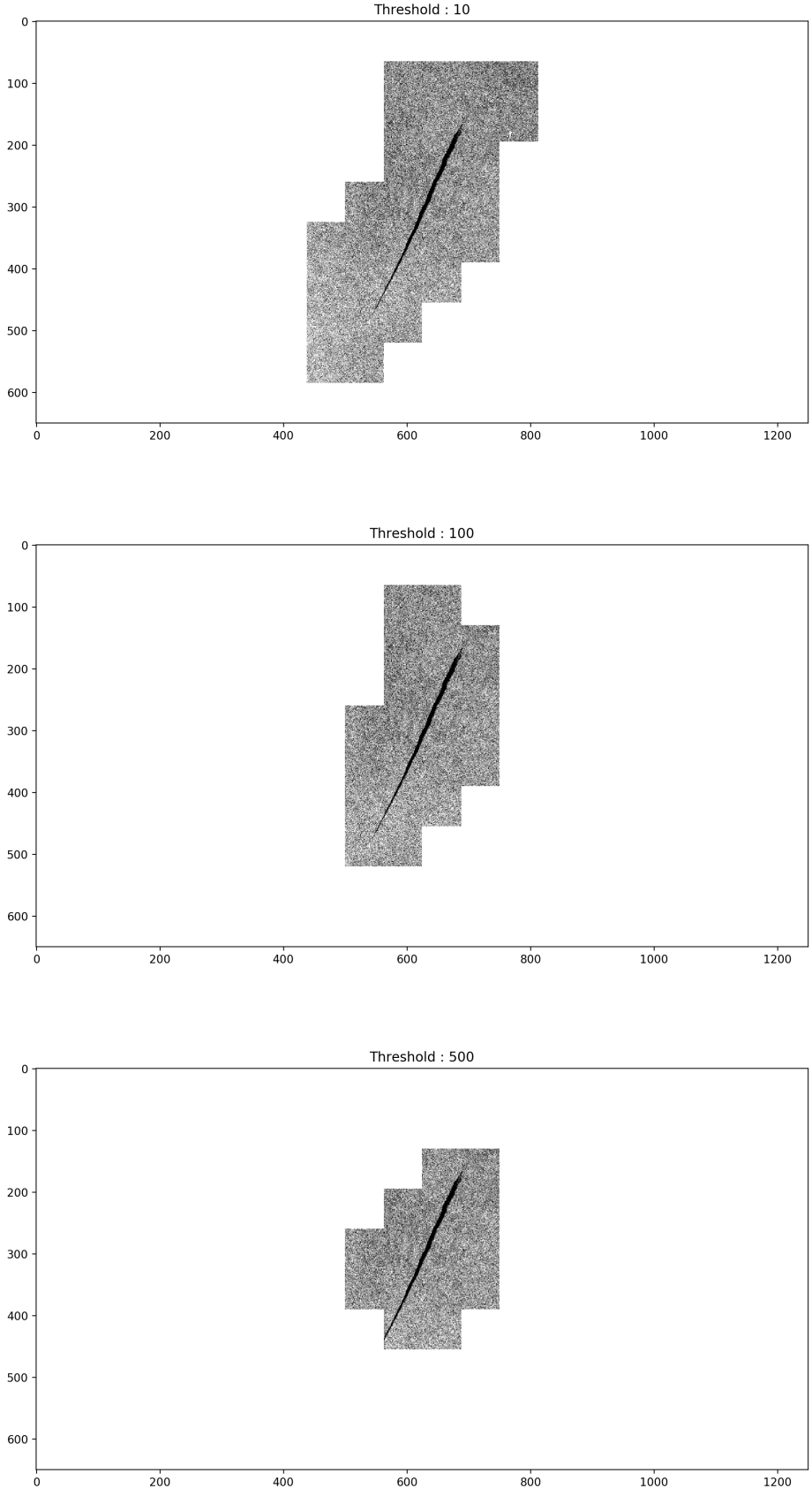


Figure 3.7: Reconstructed image for 3 different thresholds (10, 100, 500)

Chapter 4

Classification Model

This chapter will present our classification model and detail its general architecture. We will also introduce the main parameters to be optimized for the oil spill application.

The block diagram of our classification model is represented in Fig 4.1. The SAR image is first sliced into 86 slices with its corresponding ground-truth mask. Each tile follows then the same way. It goes through a configurable data generator block. The interest of this block and its impact on accuracy will be discussed in the next chapter. The output of the data generator is sent to the CNN-based classification that outputs a binary decision: this tile is an oil spill tile or not.

We can reconstruct the SAR image when all the tiles are processed, including only the oil spill tiles. This is a kind of coarse segmentation whose accuracy depends on the value of the *Thr* parameter (see section 3.5).

The classification itself is based on a convolutional neural network (CNN). This choice is rather obvious as CNN has shown its efficiency for image segmentation and recognition (see chapter 2).

The architecture of the classifier is given in Fig 4.2. It contains the following blocks:

- A CNN
- A flattening layer
- A set of fully connected (FC) layers
- A softmax layer

As a reminder, the input tile has a size of 125 x 130 8-bit pixels. The tile is connected to a CNN, which will extract some features.

This architecture has many options and hyperparameters that must be optimized for this application. This will be done by simulation and presented in the next chapter.

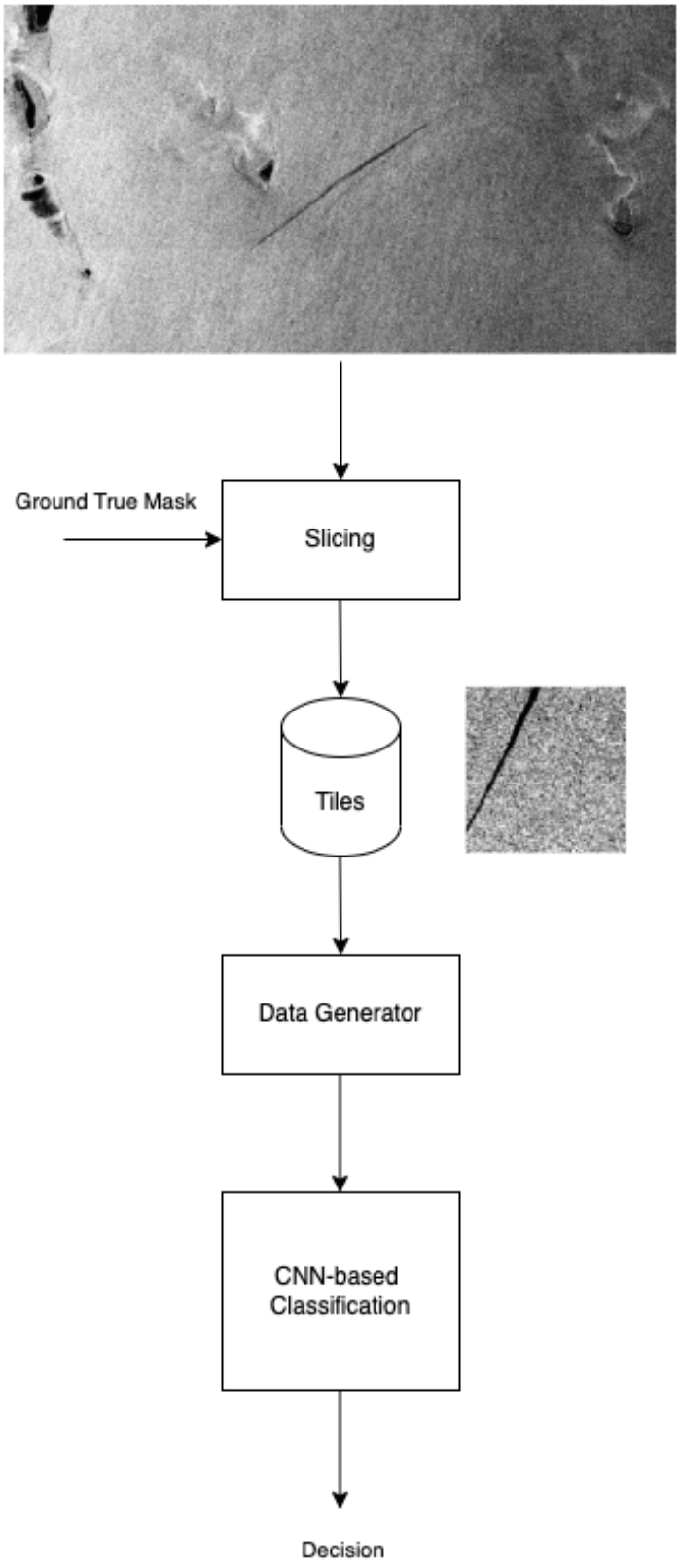


Figure 4.1: Block diagram of our model

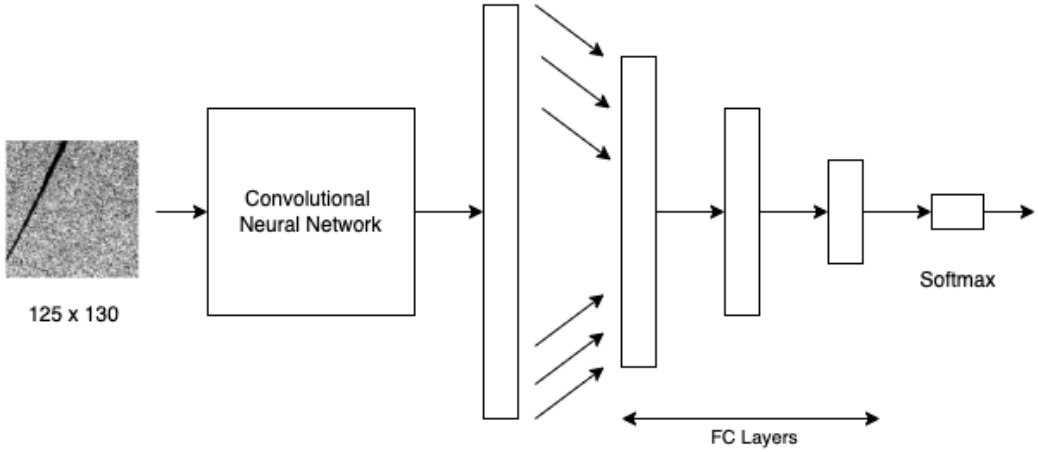


Figure 4.2: Architecture of our CNN-based classifier

The items that have to be tuned are:

- The type of CNN
- The type of pooling
- The batch size
- The learning rate
- The activation non-linear function
- The data generator (zoom, shear and flips)

Chapter 5

Results and Discussion

This chapter presents and discusses the main results we have obtained. This mainly concerns the choice of the CNN and the optimization of the hyperparameters of the classifier to have the best accuracy. This tuning will be done sequentially, parameter by parameter, and, in the end, the global performances of our model will be evaluated.

5.1 Evaluation metrics

Before proceeding with the evaluation, we must define the metrics carefully we will base on. The metrics include the recognition rate (accuracy), the detection rate, the precision, and the mean IU (Intersection over Union). These indicators can be evaluated from four simulation results: true positive (TP), true negative (TN), false positive (FP), and false negative (FN), as defined in the table 5.1 [19].

		Label	
		Oil Spill Tile	**No Oil Spill Tile**
Cassification	*Oil Spill*	TP	FP
	No Oil Spill	FN	TN

Table 5.1: Definition of TP, FP, FN, and TN

The mathematical definition of these indicators are:

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

$$\text{Detection Rate} = \frac{TP}{TP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{F-Measure} = \frac{2 * TP}{2 * TP + FN + FP}$$

$$\text{Intersection over Union (IU)} = \frac{TP}{TP + FP + FN}$$

5.2 Software framework

All our software development has been done under TensorFlow, a well-known open-source platform for machine learning. It provides most classical networks, learning rules, and tools to efficiently and quickly develop machine learning application [54].

We had access to two high-performance environments for running our programs:

- *Cassiopee*, which is a workstation dedicated to machine vision with 4 graphics card (GeForce GTx 1080 Ti) to accelerate the simulation and is available to the researcher group of my supervisor.
- *Google Colab Pro* which gives access to high-performance servers and GPU [55].

5.3 Initialization of our model

It is essential to start the optimization phase with a consistent set of parameters to ensure proper convergence. This set of initialization parameters has been chosen from our experience during this thesis and from the literature, particularly from [19], and is given in Table 5.2. In summary, we start from a VGG-16 convolutional neural network with max-pooling followed by an FC 3-layers.

	Parameters	**Initial**
CNN	Type	VGG-16
	Pooling	max
General	Batch Size	32
	Learning Rate	0.00005
FC Layers	Sizes	(64,32,16)
	Activation	ReLU
Data Gene	Zoom	0.2
	Shear	0.2
	Flips	Horizontal

Table 5.2: Initial parameters for our model

5.4 Selection of the CNN

Type

Five types of CNN have been tested : VGG-16, VGG-19, ResNet-50, ResNet-101 and DenseNet-121. VGG and Resnet are introduced in section 2.3 and DenseNet is presented in [56]. The evolution of the accuracy and the loss during 50 epochs are given in Fig 5.1 and 5.2.

We observe that VGG-16 outperforms the other networks while ResNet-101 has some delay before converging.

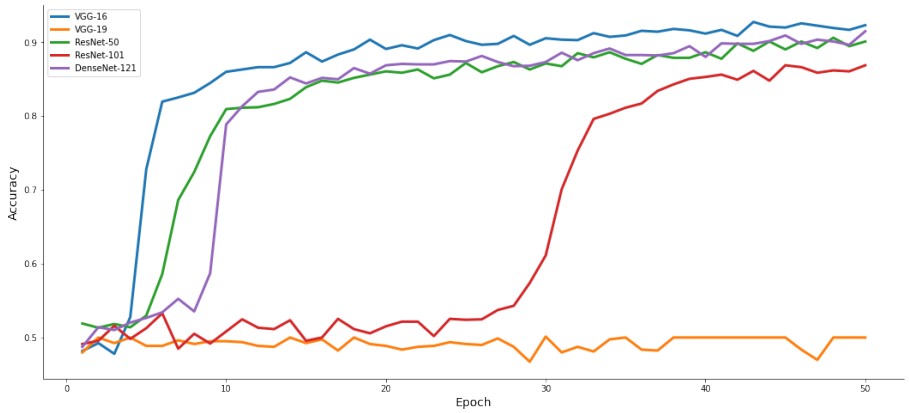


Figure 5.1: Evolution of the accuracy for the 5 types of CNN during 50 epochs

Pooling

The pooling layer decreases the dimensionality of the output of the convolutional layer and makes the network less sensitive to image variations (see 2.2.2 and Fig 2.5). Three types of pooling are tested (Max Pooling, Average Pooling and No Pooling). The evolution of the accuracy and loss in function of Epoch are drawn in Fig 5.3 and 5.4 for these three types of pooling. In agreement with the literature [26], Max Pooling is the best one.

FC Layers

The fully-connected layers (FC) take place after the flattening layer. The configuration parameters are the number of layers and their size (e.g., the number of neurons). Six configurations have been tested:

- One layer of 64 or 256 neurons
- Two layers of (128-64) or (256-64) neurons

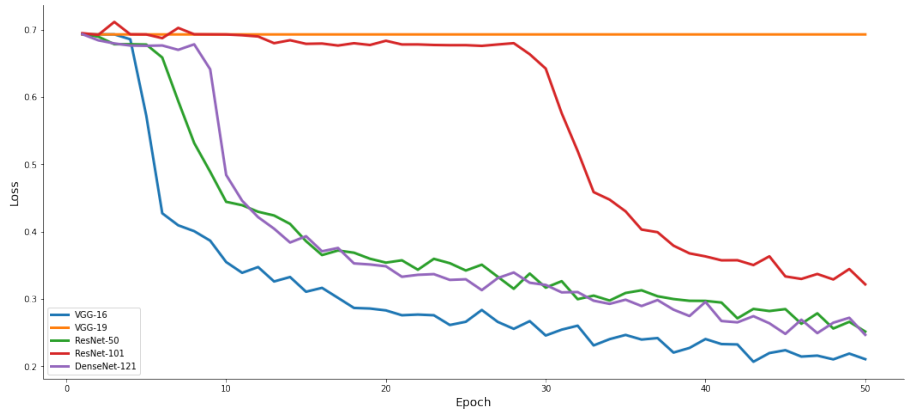


Figure 5.2: Evolution of the loss for the 5 types of CNN during 50 epochs

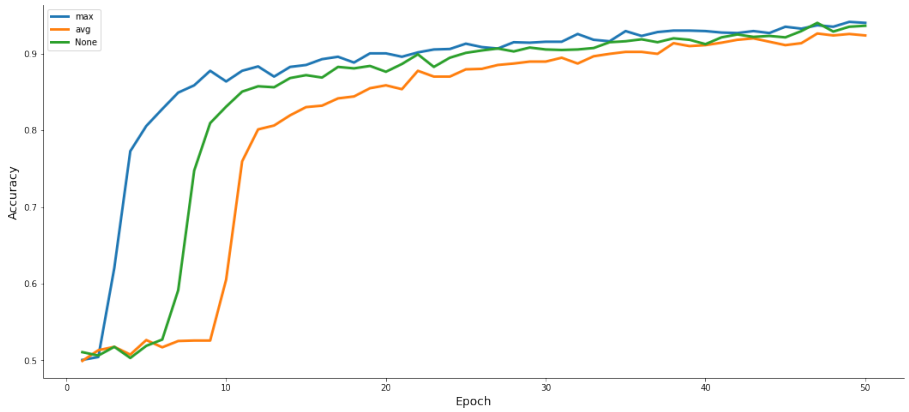


Figure 5.3: Evolution of the accuracy for 3 types of pooling during 50 epochs

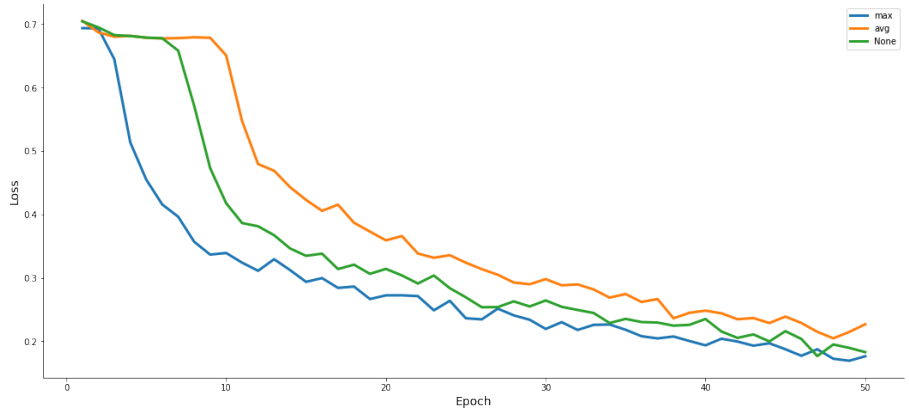


Figure 5.4: Evolution of the loss for 3 types of pooling during 50 epochs

- Three layers of (512,128,64) or (512,256,128) neurons

Fig 5.5 and 5.6 represent the performances of our model as a function of the number of epochs. The 3-layer configuration with (256,128,64) neurons gives the best results.

5.5 Optimization of the hyperparameters

To complete the configuration of our classification model, some parameters still need to be set by evaluating their impact on performance.

Batch Size

The batch size is a parameter of the gradient descent algorithm and specifies the number of training images from the training dataset to estimate the error gradient (see 2.4). Values of 16, 32, 64, 128, and 256 have been tested, and the lowest value gives the best results (Fig 5.7 and 5.8). This is consistent with the literature [10][13][49], where small batch sizes is used.

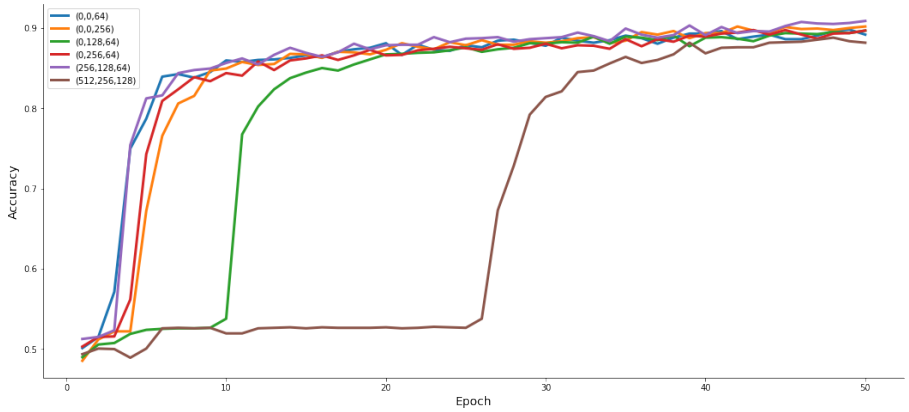


Figure 5.5: Evolution of the accuracy for different configuration of fully connected layers during 50 epochs

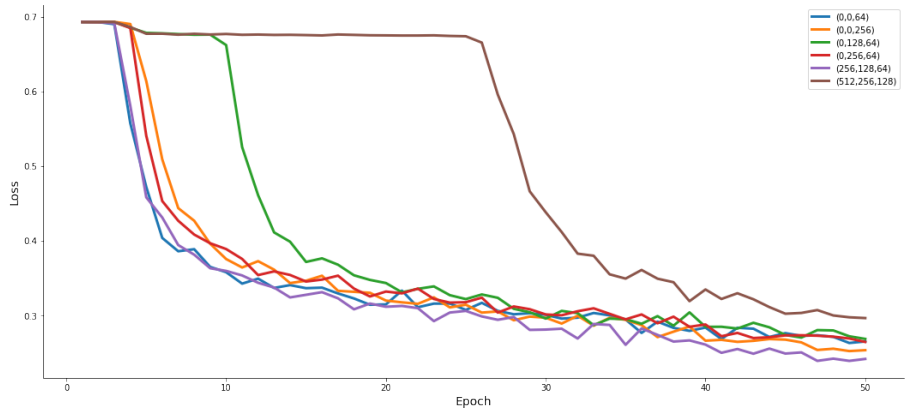


Figure 5.6: Evolution of the loss for different configuration of fully connected layers during 50 epochs

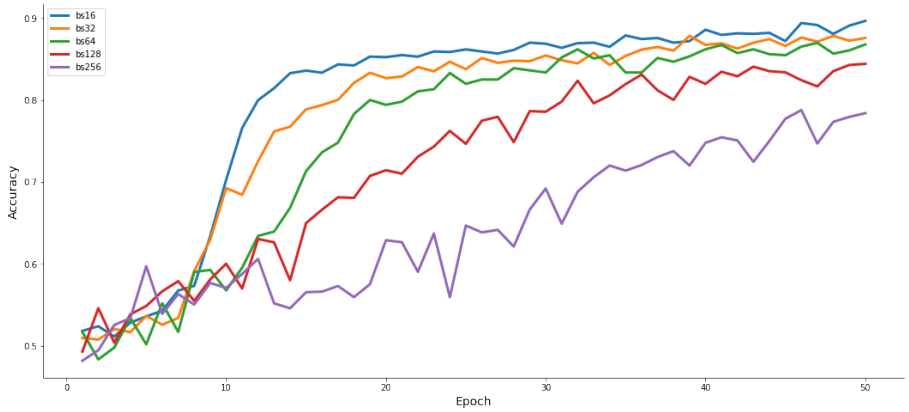


Figure 5.7: Evolution of the accuracy for different batch sizes during 50 epochs

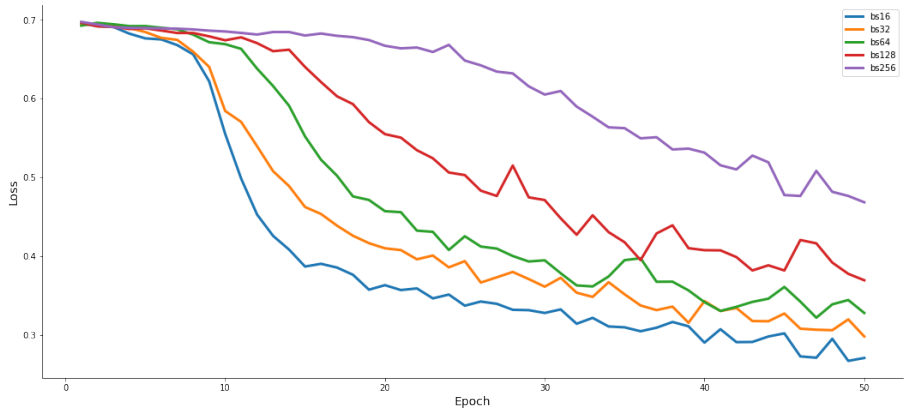


Figure 5.8: Evolution of the loss for different batch sizes during 50 epochs

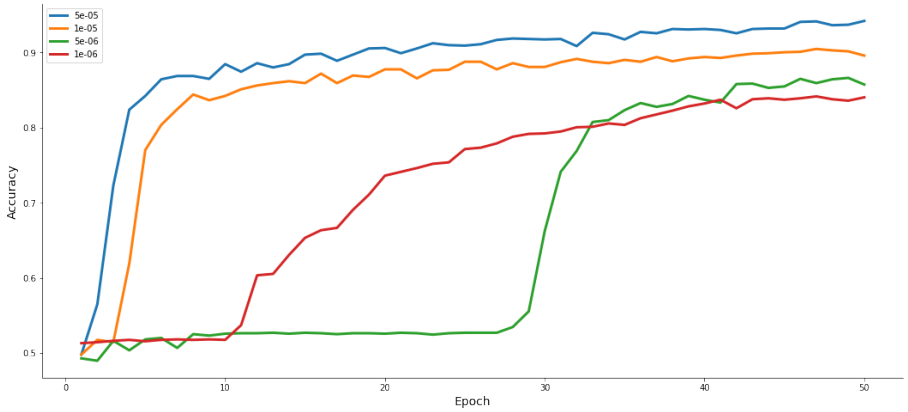


Figure 5.9: Evolution of the accuracy for different learning rates during 50 epochs

Learning Rate

The learning rate is an important hyperparameter that determines the size of the steps to reach a minimum in the gradient descent algorithm [37]. The impact of the learning rate on the accuracy and loss are represented in Fig 5.9 and 5.10. A learning rate of 0.00005 gives the best results.

Activation

The activation hyperparameter specifies the non-linear function to evaluate the output of the artificial neurons (see 2.3 and Fig 2.2). We tested Rectified Linear Unit (ReLU), hyperbolic Tangent (Tanh), Sigmoid, Exponantial (Etu), and Leaky ReLU (5.11 and 5.12). The most common ReLU has been selected due to its best performance.

Data Generators

TensorFlow provides an efficient way for data augmentation through the ImageDataGenerator class. The image transformation (zoom, shear, and flip) can be configurated, and it results from Fig 5.13, 5.14, and 5.15 in zoom of 0.1, shear of 0.3, and horizontal flips giving the best loss.

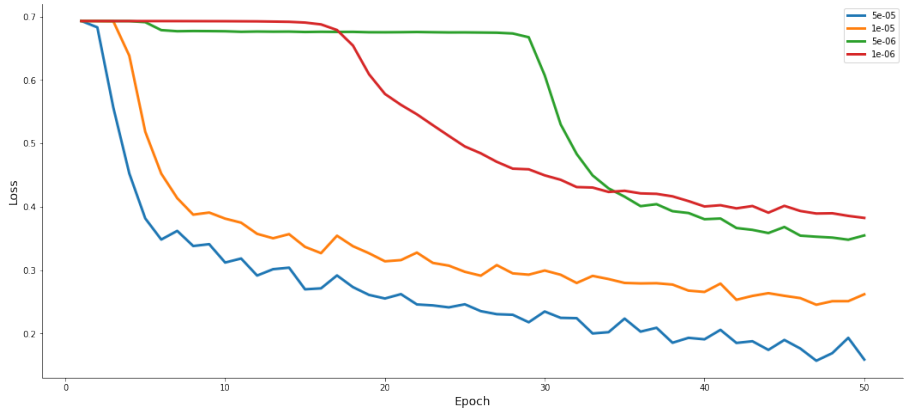


Figure 5.10: Evolution of the loss for different learning rates during 50 epochs

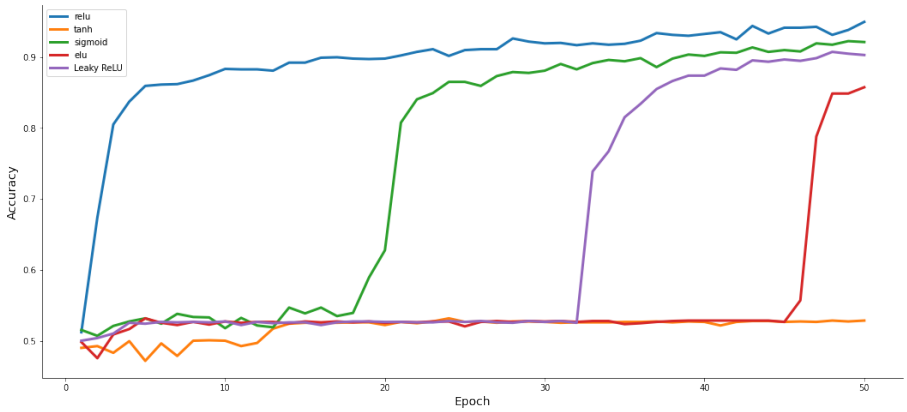


Figure 5.11: Evolution of the accuracy for different activation during 50 epochs

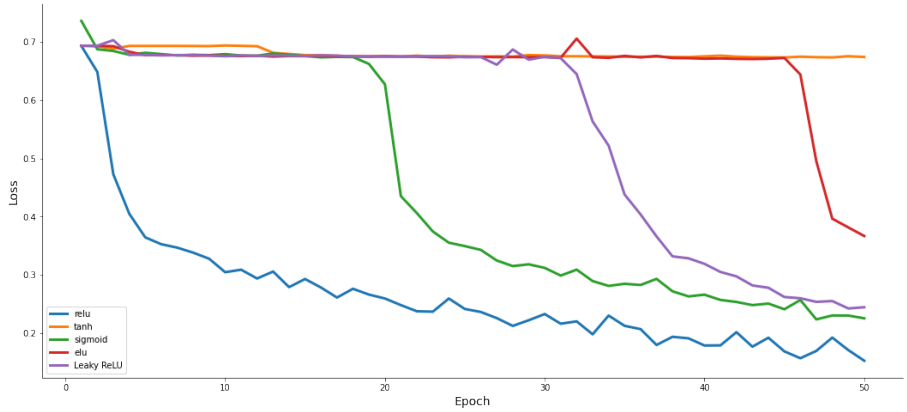


Figure 5.12: Evolution of the loss for different activation during 50 epochs

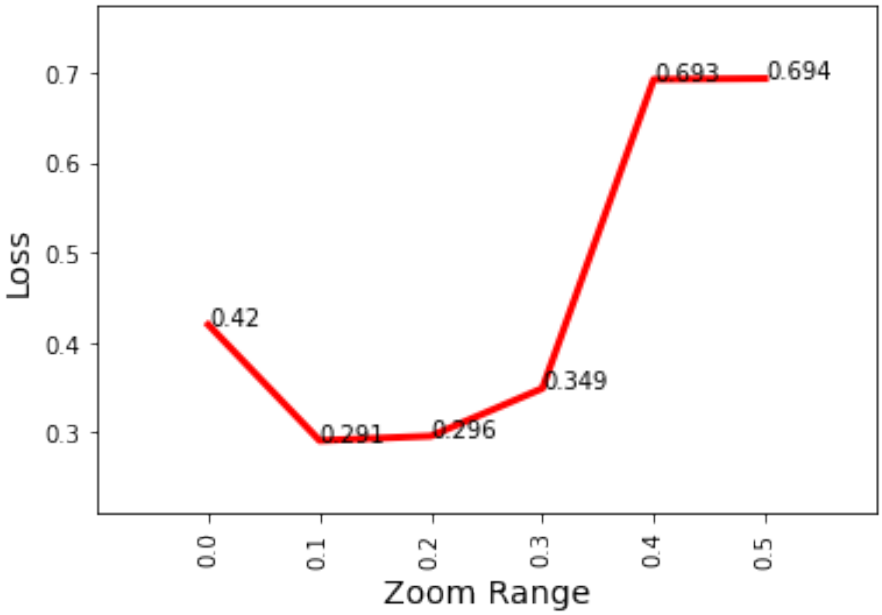


Figure 5.13: Evolution of the loss for different zoom range during 50 epochs

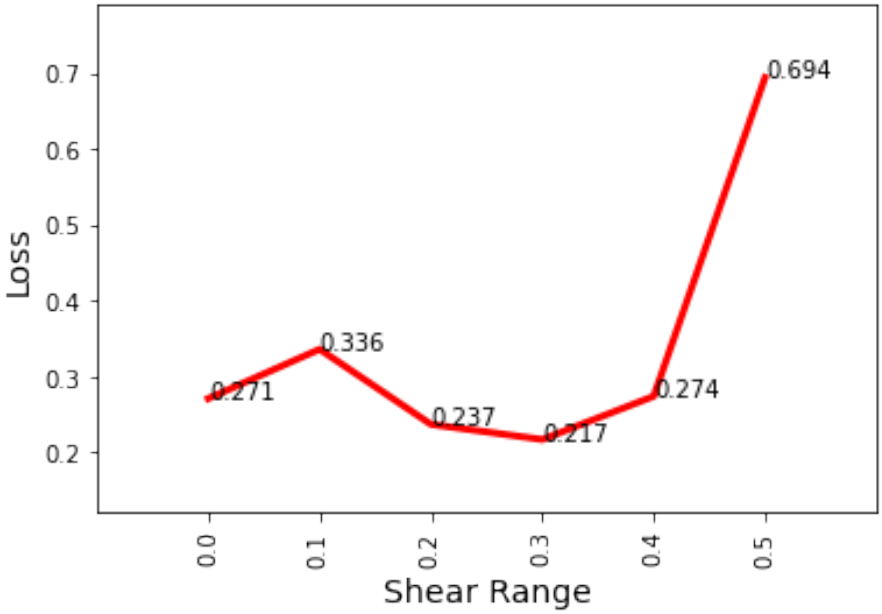


Figure 5.14: Evolution of the loss for different shear range during 50 epochs

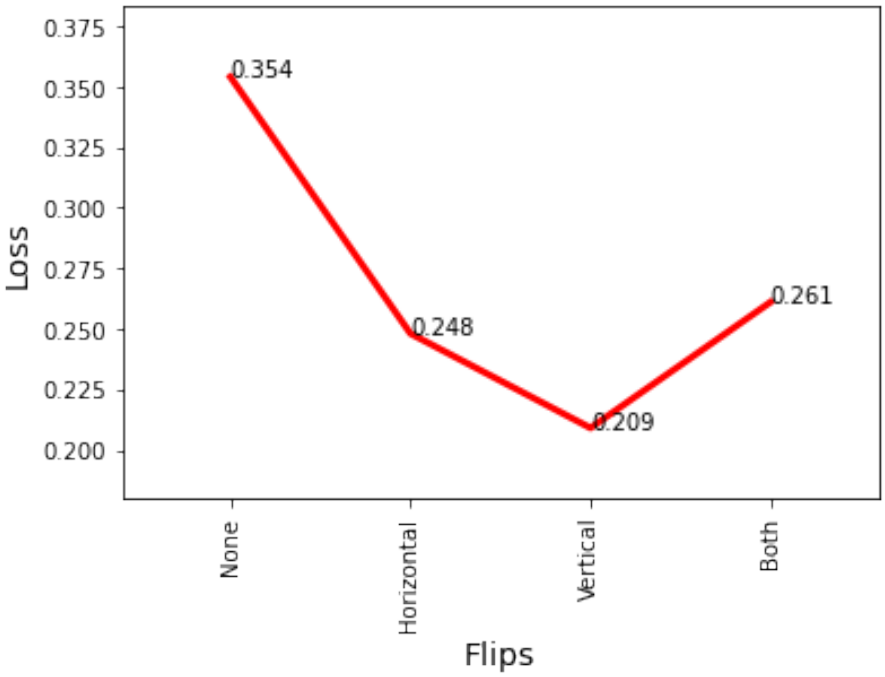


Figure 5.15: Evolution of the loss for different flips configuration during 50 epochs

5.6 Final Results

Table 5.3 summarizes the final configuration and the optimized values for the hyperparameters.

	Parameters	**Initial**
CNN	Type	VGG-16
	Pooling	max
General	Batch Size	16
	Learning Rate	0.00005
FC Layers	Sizes	(56,128,64)
	Activation	ReLU
Data Generator	Zoom	0.1
	Shear	0.3
	Flips	Vertical

Table 5.3: Configuration and hyperparameters of our model

Based on these parameters, we can now evaluate the performance of our classification model. This has been done on the training set several times. For each time, we give the recognition rate (or accuracy), the detection rate (or recall), and the precision (see 5.1). The best rates we obtained are 93.4% for the recognition, 90.2% for the detection, 98.2% for the precision, 94% for the F-Measure and 88.7% for the IU. The average values are 91.4%, 87.2%, 95.2%, 91% and 83.5%.

As already explained, we cannot compare these results with other similar work because there is no common dataset. However, it seems interesting to give the figures for recent oil spill recognition systems that we have found in the literature. This

Training Time	**Recognition Rate**	**Detection Rate**	**Precision**	**F-Measure**	**IU**
1	0.934	0.902	0.965	0.932	0.873
2	0.918	0.902	0.932	0.917	0.846
3	0.902	0.869	0.93	0.898	0.815
4	0.902	0.852	0.945	0.897	0.812
5	0.893	0.852	0.929	0.889	0.8
6	0.91	0.852	0.963	0.904	0.825
7	0.918	0.869	0.964	0.914	0.841
8	0.91	0.82	0.971	0.901	0.82
9	0.926	0.885	0.964	0.923	0.857
10	0.934	0.885	0.982	0.931	0.871
11	0.902	0.885	0.915	0.9	0.818
12	0.893	0.836	0.944	0.887	0.797
13	0.902	0.885	0.915	0.9	0.818
14	0.943	0.902	0.982	0.94	0.887
15	0.918	0.885	0.947	0.915	0.844
Average	0.914	0.872	0.95	0.91	0.835

Table 5.4: Performances of our model for 15 runs

comparison is given in Table 5.5 and shows that our model performs quite well. We have to be very careful not to draw too hasty conclusions, as there are very different realities behind these figures.

	Our Model	**DeepLabv3+** [49]	**OSCNet** [19]	**CBD-Net** [20]
Recognition rate	0.914	—	0.965	0.975
Detection rate	0.872	—	0.932	0.986
Precision	0.95	—	0.93	—
F-measure	0.91	—	0.945	0.976
mIU	0.835	0.651	—	0.832

Table 5.5: Comparison of different classifiers for oil spill detection

To conclude this analysis, we reconstruct the original SAR images by only putting the tiles that have been recognized as oil spill tiles with their corresponding prediction (output of the classifier). The TP (true positive) predictions are green and the FP (False positive) red. At this stage, postprocessing could be applied to enhance the oil spill detection and eliminate some FP tiles. An example of straightforward processing is eliminating the tiles with a prediction lower than a defined threshold. Fig 5.16 and 5.17 give two examples of such image reconstruction with a threshold of 0.5 and 0.7.

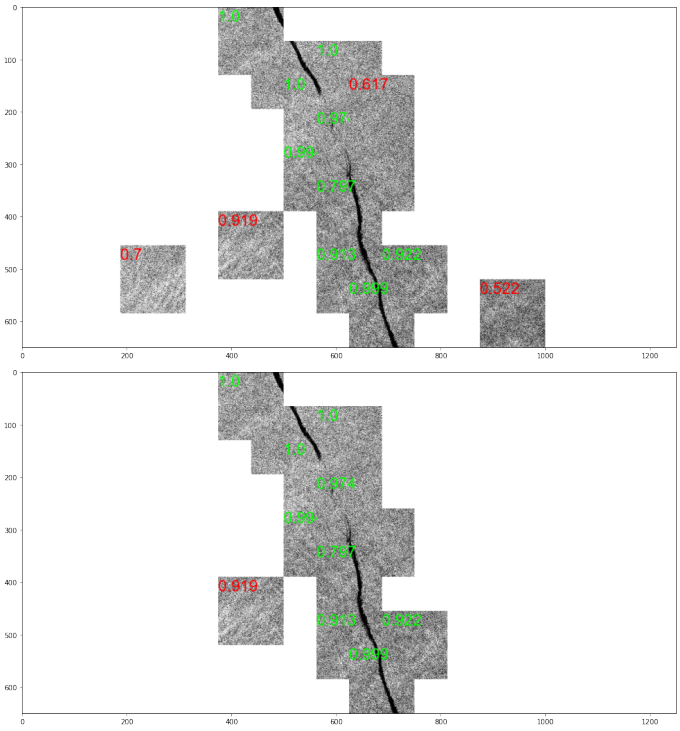


Figure 5.16: Example of reconstruction of an vertical oil spill with a threshold of 0.5 and 0.7

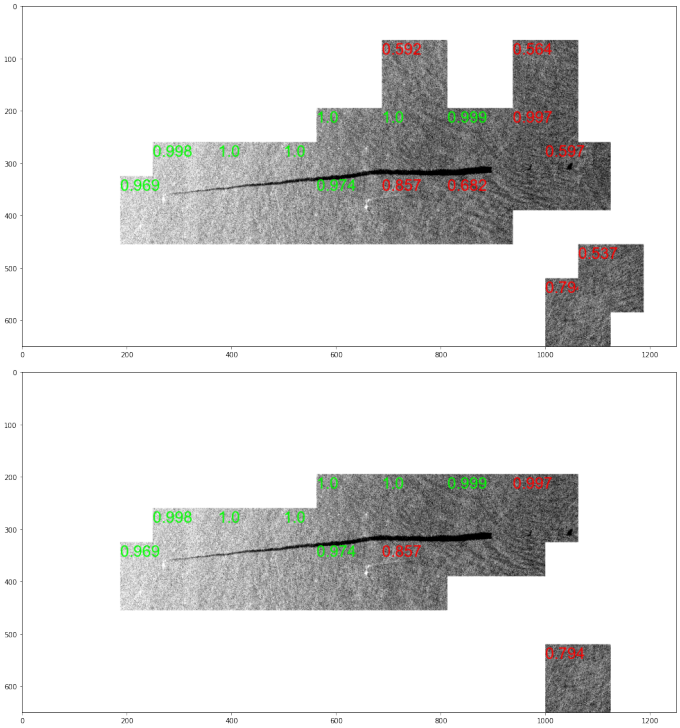


Figure 5.17: Example of reconstruction of an horizontal oil spill with a threshold of 0.5 and 0.7

Chapter 6

Conclusion

This thesis focused on detecting oil spills generated by illegal bilge dumping. This detection is based on the analysis of SAR images that also contain many look-alikes making the recognition task difficult.

In the first chapter of this thesis, a literature overview is realized in the domain of the convolutional neural networks with an emphasis on image segmentation and oil spill detection.

6.1 Main Results

The most important contributions of this thesis can be summarized as follows:

- The development of a new oil spill dataset with training and test parts.
- The manual annotating of the complete dataset
- The decision to slice the original images into tiles that will be used as input for the classifier
- The design of a CNN-based classifier.
- The optimization of its structure and its hyperparameters.
- The evaluation of its average performance on the test dataset which gives the following figures:
  - Recognition rate (accuracy) : 0.914
  - Detection rate (recall) : 0.872
  - Precision : 0.95
  - F-measure : 0.91
  - mIU : 0.835

In comparison with the literature and recently published oil spill detection systems, we can conclude that our model performs quite well.

6.2 Future Perspectives

This thesis has completed a first phase, but the work is far from over. We could extend the project in several directions:

- Currently, the weights and biases of our network are initialized randomly. A self-supervised preprocessing of our network could be applied before the learning phase. It involves accustoming our network to the unusual features of the SAR images. This should improve the learning phase.
- We compared different CNNs and found VGG-16 best suited for this application. It would be interesting to particularise the internal structure of VGG-16 to our application, following a similar approach to that adopted by [19]. Different structures would be tested on our dataset to identify the one that gives the best performance. We would then have a CNN fully dedicated to our application.
- We decided to cut out the image and work from the tiles. This allowed us to reduce the complexity of the detection system. In the light of other approaches [20] [18], we believe it would be interesting to design a semantic segmentation and classification system working directly on the SAR image. We could then compare the two approaches in terms of performance but also in terms of computational complexity.

* *
*

Bibliography

[1] Sar image of bilge dumping. https://skytruth.org/bilge/.

[2] Adrian Lauwers. Oil spills detection using sentinel-1 sar images. Aerospacelab - Internal document, 2020.

[3] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017.

[4] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. https://arxiv.org/abs/1511.08458, 2015.

[5] Sumit Saha. A comprehensive guide to convolutional neural networks — the eli5 way. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way, 2018.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. NIPS*, pages 1097–1105, 2012.

[7] C. Szegedy et al. Going deeper with convolutions. In *Proc. CVPR*, pages 1–9, 2015.

[8] An introduction to image segmentation: Deep learning vs. traditional. https://www.v7labs.com/blog/image-segmentation-guide.

[9] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE CVPR*, pages 3431–3440, 2015.

[10] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. https://arxiv.org/abs/1612.01105, 2017.

[11] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollar. Panoptic segmentation. In *Proceedings of the IEEE CVPR*, pages 9404–9413, 2019.

[12] Bowen Cheng, Maxwell D. Collins, Yukun Zhu, Ting Liu, Thomas S. Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. https://arxiv.org/abs/1911.10194, 2020.

[13] Seyed Abolfazl Ghasemzadeh, Gabriel Van Zandycke, Maxime Istasse, Niels Sayez, Amirafshar Moshtaghpour, and Christophe De Vleeschouwer. Deepsport-

lab: a unified framework for ball detection, player instance segmentation and pose estimation in team sports scenes. In *Proceedings of the British Machine Vision Conference*, pages 1–13, 2021.

[14] Georgios Orfanidis, Konstantinos Ioannidis, Konstantinos Avgerinakis, Stefanos Vrochidis, and Ioannis Kompatsiaris. A deep neural network for oil spill semantic segmentation in sar images. In *Proceedings of 25th IEEE International Conference on Image Processing (ICIP)*, pages 3773–3777, 2018.

[15] Yaohua Xiong and Hui Zhou. Oil spills identification in sar image based on convolutional neural network. In *Proceedings of 14th International Conference on Computer Science & Education (ICCSE)*, pages 667–670, 2019.

[16] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. `https://arxiv.org/abs/1505.04597`, 2015.

[17] Abhishek Chaurasia and Eugenio Culurciello. Linknet: Exploiting encoder representations for efficient semantic segmentation. `https://arxiv.org/abs/1707.03718`, 2017.

[18] Jin Zhang, Hao Feng, Qingli Luo, Yu Li, Jujie Wei, and Jian Li. Oil spill detection in quad-polarimetric sar images using an advanced convolutional neural network based on superpixel model. *Remote Sensing*, 12(6):1—-26, 2020.

[19] Kan Zeng and Yixiao Wang. A deep convolutional neural network for oil spill detection from spaceborne sar image. *Remote Sensing*, 12(6):1–23, 2020.

[20] Yanan Zhang, Qiqi Zhu, and Qingfeng Guan. Oil spill detection based on cbd-net using marine sar image. In *Proceedings of IEEE International Geoscience and Remote Sensing Symposium IGARSS*, pages 3495–3498, 2021.

[21] Sudhir Kumar Chaturvedi, Saikat Banerjee, and Shashank Lele. An assessment of oil spill detection using sentinel 1 sar-c images. *Journal of Ocean Engineering and Science*, 5(2):116–135, 2020.

[22] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proc. ICML*, pages 807–814, 2010.

[23] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, pages 1–6, 2013.

[24] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet. In *Proc. ICCV*, pages 1026–1034, 2015.

[25] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *Proc. ICLR*, 2016.

[26] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[27] Y. LeCun et al. Handwritten digit recognition: Applications of neural network chips and automatic learning. *IEEE Commun. Mag.*, 27(11):41–46, 1989.

[28] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Handwritten digit recognition: Applications of neural network chips and automatic learning. *Proceedings of the IEEE*, 86(11):2278—-2324, 1998.

[29] O. Russakovsky et al. ImageNet large scale visual recognition challenge. *Int. J. Comput. Vis.*, 115(3):211–252, 2015.

[30] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. OverFeat: Integrated recognition, localization and detection using convolutional networks. In *Proc. ICLR*, 2014.

[31] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proc. ICLR*, 2015.

[32] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Very deep convolutional networks for large-scale image recognition. In *Proc. CVPR*, pages 2818–2826, 2016.

[33] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Very deep convolutional networks for large-scale image recognition. In *Proc. AAAI*, pages 1–3, 2017.

[34] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. CVPR*, pages 770–778, 2016.

[35] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[36] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[37] Sebastian Ruder. An overview of gradient descent optimization algorithms. `https://arxiv.org/abs/1609.04747`, 2017.

[38] Visual object classes challenge 2011 (voc2011). `http://host.robots.ox.ac.uk:8080/pascal/VOC/voc2011/index.html`, 2011.

[39] Ade20k dataset. `https://groups.csail.mit.edu/vision/datasets/ADE20K/`.

[40] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. `https://arxiv.org/abs/1608.05442`, 2018.

[41] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE CVPR*, pages 1–11, 2016.

[42] George Papandreou, Tyler Zhu, Liang-Chieh Chen, Spyros Gidaris, Jonathan Tompson, , and Kevin Murphy. Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model. In *Proceedings of the ECCV*, pages 1–21, 2018.

[43] Xingyi Zhou, Jiacheng Zhuo, and Philipp Krähenbühl. Bottom-up object detection by grouping extreme and center points. In *Proceedings of the CVPR*, pages 1–10, 2019.

[44] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. `https://arxiv.org/abs/1606.00915v2`, 2017.

[45] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context. `https://arxiv.org/abs/1405.0312`, 2015.

[46] Deepsport data set. `https://sites.uclouvain.be/ispgroup/Softwares/DeepSport`, 2021.

[47] Abdul Duane Lawal, Gianmarco Radice, Matteo Ceriotti, and Abubakar Umar Makarfi. Investigating sar algorithm for spaceborne interferometric oil spill detection. *International Journal of Engineering and Technical Research (IJETR)*, 4(3):123–127, 2016.

[48] Guandong Chen, Yu Li, Guangmin Sun, and Yuanzhi Zhang. Polarimetric sar oil spill detection based on deep networks. In *Proceedings of IEEE International Conference on Imaging Systems and Techniques (IST)*, pages 1–4, 2017.

[49] Marios Krestenitis, Georgios Orfanidis, Konstantinos Ioannidis, Konstantinos Avgerinakis, Stefanos Vrochidis, and Ioannis Kompatsiaris. Oil spill identification from satellite images using deep neural networks. *Remote Sensing*, 11(15):1–22, 2019.

[50] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274—-2282, 2012.

[51] Abhijit Guha Roy, Nassir Navab, and Christian Wachinger. Concurrent spatial and channel squeeze & excitation in fully convolutional networks. `https://arxiv.org/abs/1803.02579`, 2018.

[52] Sentinel application platform (snap). `https://step.esa.int/main/toolboxes/snap/`.

[53] Coco annotator. `https://github.com/jsbroks/coco-annotator`.

[54] Tensorflow, an open source platform for machine learning. `https://www.tensorflow.org`.

[55] Google colab. `https://colab.research.google.com`.

[56] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. `https://arxiv.org/abs/1608.06993`, 2018.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**