Rennes 1

Research Internship Report

Automatic detection of oil spills in marine areas

Name Diane Maillot-Tchofo
Faculté des Sciences Economiques
Master 1 Mathématiques Appliquées, Statistique
Promotion 20/21
Dates 03/05/21 au 03/09/21
Tutor Isabelle Cadoret

Company Halias Technologies
57 Chemin du Vieux Chêne
38240 MEYLAN
Supervisor Laurent Testard (CEO)

Lab ERIC (Lyon 2)
Supervisor Stéphane Chretien
Researcher/Full professor

Contents

1 Introduction

I have carried out my summer internship at the ERIC (Entrepots, Representation et Ingenierie des Connaisances) laboratory in Lyon (France), with funding and support from HALIAS Technologies and Digital Services[1], a company specialized in scientific data management software, located near Grenoble (France). The latter, are focused in the energy market, including oil and gas markets, they are currently looking for new product to offer to clients.

It is in this perspective that the internship was proposed to me, namely, find or build an algorithm to automatically detect oil spills from satellite images; areas to explore further were also expected to be delivered. The end goal is obviously to be able to include the algorithm into a web application or software, so that clients, mainly oil companies, can monitor areas of their choosing and get an alert if an oil spill appear.

I had complete freedom during this internship, after reading research articles for around 2 months in order to single out a few methods, I dived in the various codes, using exclusively python in the process (via Jupyter Notebooks and Spyder). A large number of things were completely new to me, however the large resources available online, and the solid foundation I had to work on allowed me to overcome (almost) every obstacle encountered.

This report will be divided in 3 parts. Firstly an overview of the techniques, methods and algorithms I tried or went over ; then I explain the method I decided to apply, why and how. Finally, in the last section, the algorithm chosen to be implemented, with great success, is detailed along with the data used to deploy it, with a small discussion to explain what could be the future research in this area.

[1]See halias.fr

2 Methods & Attempts

In this first section I will go over most of the methods I encountered, tests I ran and the struggles I had before opting for a transversal approach.

2.1 Context

As it depends on very to extremely high resolution satellite images, research on automatic oil spills detection is relatively recent and niche. Indeed, different types of satellites bring different types of images. For this kind of problematic, images that are unaffected by climatic conditions are ideal, these are produced by Synthetic-Aperture Radar (SAR). SAR is an active sensor that measures the sea-surface texture and roughness, therefore it is not affected by cloud cover, or heavy rain Li et al. [15]. Which makes it a suitable sensor for oil spill detection because :

- petrol (or oil) dampens the sea-surface capillary waves so that they appear dark in SAR images,
- diesel (or refined oil/petrol) smoothen the sea-surface so that they change de texture of the relevant area.

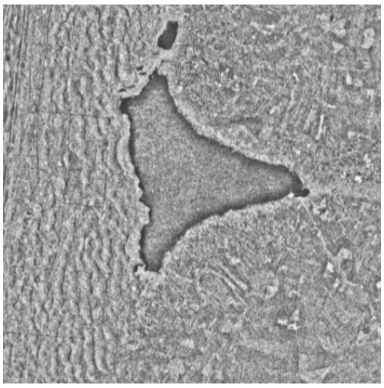
Above are shown an image for each situation; biological (or natural, lookalikes) slicks refer to fish bank, algae, corals, marine oils, etc...

Due to the cost of one SAR image (approximately 800€ for a 2000 pixel by 2500 and 1.5 k€ for 5500 pixels by 4000), most research article do not make their databases public. However, a thorough reading of many articles strongly suggests that most of the research (if not all), only refer to brut petrol when *oil slicks* are mentioned. We will see later that detecting indifferently brut and refined oil is one of the main challenges.

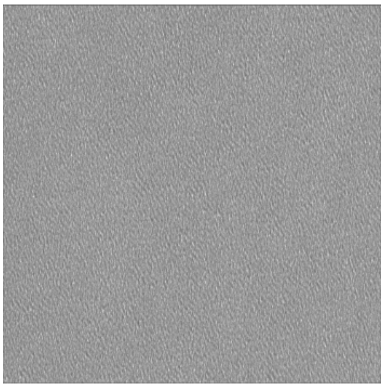
The published methods vary in objectives. Some only focus on oil slicks (brut petrol) versus lookalikes (natural slicks) classification Guo et al. [12]. Others use process in 3 steps : dark spot detection, feature extraction and classification between lookalikes and oil slicks.

2.2 Segmentation Methods

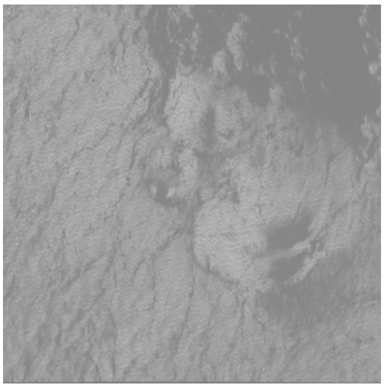
At the time of writing, not much research is being done on oil spill segmentation and, when it is, it only concerns *petrol leaks*, as in this recent comparative study Cantorna et al.



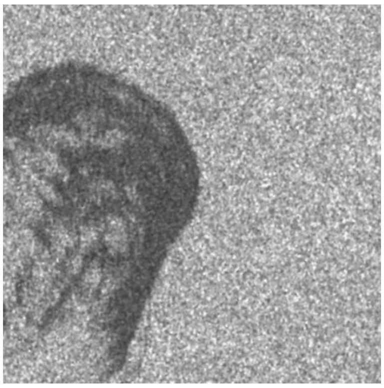
(a) Diesel leak



(b) Water



(c) Biological slicks



(d) Petrol leak

[4](2019) where only "dark spots" are mentioned, and as the illustrations showed, *diesel leaks* do not appear dark in SAR images. Thus, while researching for applicable methods, I was concentrated, at first, on petrol leaks. Semantic segmentation is basically a pixel-wise classification, therefore it mainly uses the same methods as classic classification with some tweaks.

The starting point of my semantic segmentation research, was an article on tensor reconstruction applied to the field Chen et al. [8]. This came at a time where more and more research focus on local or regional context features rather than global context. Which is also the case of Zhang et al. [22] (2019) and Zhao et al. [23] (2016). The innovation of Chen et al. [8] is to use tensor reconstruction (high-rank tensor are expressed as a combination of rank-1 tensors) to directly model contextual information without compression (from 3D to 2D data). Tensors are algebraic objects that describes a multilinear relationship between sets of algebraic objects (related to a vector space)[2], it generalizes the concepts of scalar, vector (the simplest tensor) and linear operator [3].

Along with reaching state-of the-art result on public datasets, their reconstruction net-

[2]See en.wikipedia.org/wiki/Tensor : Wikipedia definition

[3]See en.wikipedia.org/wiki/Category:Tensors : Wikipedia category

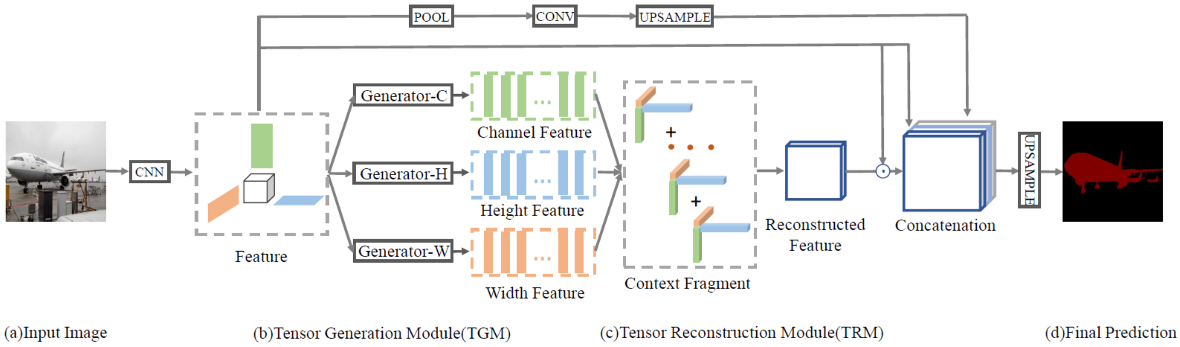


Figure 1: RecoNet framework as shown in the original paper by Chen et al. [8]

work (RecoNet) is computationally less consuming compared to global context segmentation methods. The basic idea is to use low-rank tensors to capture fragments of context feature and aggregate them to reconstruct fine-grained context features. The workflow of the model can be seen in Fig.1. Although their code is (partially) available online, the theory around it far exceeded my knowledge and skill set at the time. Furthermore, even though it is a state of the art, the evaluation metrics (mean Intersection-over-Union - mIoU [4]) is quite low on the vast majority of public datasets tests (between 41 and 55), mostly because there is a lot of categories and classes, but it means that we do not have a good vision of what it could give in our case.

While looking for simpler, alternative methods in semantic segmentation the only relevant comparative study or survey I found, Cantorna et al. [4] (2019) was very insightful. They compared multiple statistical methods, including the K-Means algorithm, Fuzzy C-Means (FCM[5]) or even logistic regression. They also tested neural networks, namely a fully convolutional network (FCN) with convolutional layers. Among all of these methods, the FCN came out on top, a difference of around 4 points between the best statistical method (logistic regression) and the implementation of the FCN (94.5% accuracy for the latter). Furthermore, their minimalist preprocessing steps also reassured me, as I was at the time under the impression that these steps were important but not crucial to obtain good results. Unfortunately, none of their code, neither statistical methods nor the deep learning ones, were available publicly (at the time of writing and need), and I was not confident in coding an entire architecture from the ground up by myself. Hence, I decided to search for deep learning methods which were open source.

The first article that ticked every boxes, except for the official open source code[6], was a

[4]See pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/ fotr Intersection over union definition

[5]See en.wikipedia.org/wiki/Fuzzy_clustering : FCM definition

[6]See github.com/preddy5/segnet, unofficial Python implementation

paper published in 2016 by Badrinarayanan et al. [3]. They bring a solution to the need of low-resolution mapping needed for pixel-wise classification, by using an encoder-decoder framework. The role of the decoder is exactly that, meaning, mapping the low-resolution feature maps from the encoder, into full input resolution feature maps designed for pixel-wise classification. The encoder role, is to map the input data, here, by using convolutional layers. This method demonstrated good results both visually and quantitatively on road scenes (i.e. the type of images that motivated SegNet). It outperformed FCN techniques by 5 to 10 points depending on the dataset used (CamVid[7], SUNRGB-D[8]) when the network goes through a lot of iterations (>140k & >80k in this case).

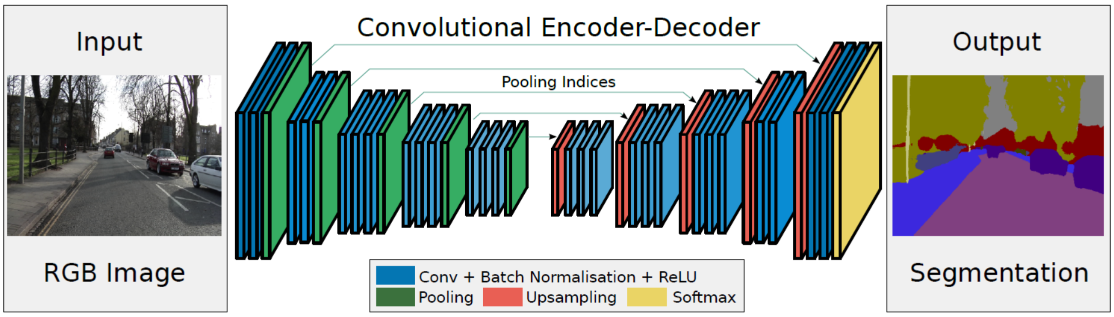


Figure 2: SegNet framework as shown in the original paper by Badrinarayanan et al. [3]

With the aim of testing out a reproduction of the SegNet architecture shown in Fig.2, I first needed a run-through of image analysis in Python. Using modules like Open-CV[9] and PIL[10]. In order to visualise what the machine & algorithms were supposed to "see", I had to have a precise idea as to what I could do and see *by hand*. As I had never worked with images before, when I tried to feed masked images of petrol leaks[11] to a simplistic classification network, the preliminary results seemed more random than anything. Hence, as this task revealed to be harder than predicted, a step back from segmentation to "classic" classification was needed.

I came back to segmentation methods nearing the end of my internship to explore what could be done in future research. As we will see later with classification methods, it is almost inevitable to use 2 types of networks/algorithms depending on the type of leaks. For instance, with diesel leaks a similarity with cells was suggested to me during the Interns Day event at the ERIC laboratory. Therefore, using the same tools as the medical world to detect and segment cancerous cells for example in tissues could be a

[7]See CamVid project's website
[8]See SUNRGB project's website
[9]See opencv.org
[10]See he-arc.github.io/livre-python/pillow
[11]It is simpler to take masks of petrol than diesel leaks, especially when diesel leaks are only partially visible

good idea, on the contrary, this approach would not make sense with petrol leaks.

Still with diesel leaks, we noted that a **texture change** announces the presence of a leak, thus, any type of method that is meant to detect specifically *when* the texture change occurs could be applied. The segmentation would not need to be visual as, what really matters is the surface total of the leak, not to be able to see it. Therefore, scattering transform[12] could be a good way forward. The latter can be used in neural network as a layer, on visual or sounds datasets. These layers will be specifically designed to detect/map the texture changes in the studied image (resp. sound). Scattering transforms layers capture the whole image (resp. sound), hence, depending on the dimension (or size) it can be quite computationnaly expensive. This is one of the reason why I didn't look further into it, another being that I did not find any article or resources sufficiently close to the task at hand to act as base. Like before this would not make sense either in the case of a petrol leak, as no texture change can be noted.

Finally, for petrol leaks, conventional algorithms could have been used, however the lack of data and diverse data was the main set-back. One of the data augmentation technique that was suggested to me by a doctoral student was to place modified leaks (e.g. in shape, size) on normal images (i.e. water or biological slicks) to substantially multiply the number of petrol leak images. However this still poses the question of variety as all modified images would come from the same scene. This is a technique to keep in mind if new images, even if small in numbers, are made available.

To conclude on segmentation methods, I encountered a lot of technical hurdles, whether it be my own skills limitation or a lack of clairvoyance within a growing field. Indeed, segmentation being a "subset" of classification, now that the latter is overwhelmed with techniques and algorithms reaching higher and higher accuracy levels ; more and more people turn their attention on segmentation, which is seen as much more challenging and rewarding. In the next few years, new methods, architectures and implementations in Python or other programming languages should be made public. For now, classification methods seemed more adequate with my current skill set[13] and the available resources.

2.3 Classification Methods

There are numerous techniques, methods and approaches when one desire to classify satellite images. Depending on the need, one method will be more appropriate than the other. In our case, what made it tricky, but also very interesting, is that, the problem can

[12] See di.ens.fr/data/scattering : scattering transform explanation

[13] Classification both supervised and unsupervised is something we have worked on in the first year of my master's degree

be tackled from many different angles. Oil leaks in marine areas can be seen as a *scene parsing*, *object detection*, or even *texture detection*.

Firstly, I was directed towards convolutional neural network (CNN) as the easiest place to start. CNN are a sub-class of artificial neural network, where layers of neurons are (fully or not) connected between each other, and each layer's "goal" is roughly to extract patterns; for example detect edges in specific places of an image. The deeper the network is, namely the more layers it contains, the more specific are the patterns extracted by the layers (e.g. an entire knife, tennis ball, etc..). CNNs are mainly used on image datasets (or video) because of the pattern analysis capacity.

CNNs require labelled data, as exhaustive as possible, in order to *generalize* well when encountering new data. This made the task trickier, because of the lack of images available. Nonetheless, I tried to adapt 2 available, recent code, to the data I had at my disposal, the result were encouraging at first, but the nature of CNN can turn them into black boxes, where it is almost impossible to know *how*, the network classify the images or how stable it can be. Thus, I started looking at methods that require the least amount of data possible while being more transparent then neural networks. Hence, statistical methods came forth.

The first statistical method I tested, came from my own deduction and was specifically made to classify water and petrol, ignoring diesel leaks : the K-Means algorithm. The latter's aim is to compute K clusters, by minimizing the variance within each cluster, for images clusters correspond to groups of pixels, thus the clustering takes the color value as data. I arbitrarily chose k value to be 4. Then, if one of the four means were below a certain local and/or global threshold, the algorithm would classify an image as containing a leak or not. This worked well (accuracy of over 0.97), but it quickly became evident that this was too specific to the data I had, and would not generalize at all. Moreover the computation time did not fit the high-resolution and high-dimensionality images I needed to work with.

Support vector machine (SVM), is a technique named in most surveys as a state of the art method for 2-class classifiers of oil spills; whether it is classifying spills from lookalikes or spills from water. Tree-based techniques, like bagging, bundling or boosting, have been found by a comparative study (Xu et al. [21], 2013) to outperform artificial neural network (ANN), by 15 points, in classifying oil spills from lookalikes, and the phenomenon persists when the data goes through a pre processing step.

One of the easiest tree-based techniques I knew at the time were random forests (RF). Hence, I tried a simplistic RF algorithm (150 estimators, depth of 150) in a 2-class classification problem : oil spill or not. Once again the results were interesting, as we can see

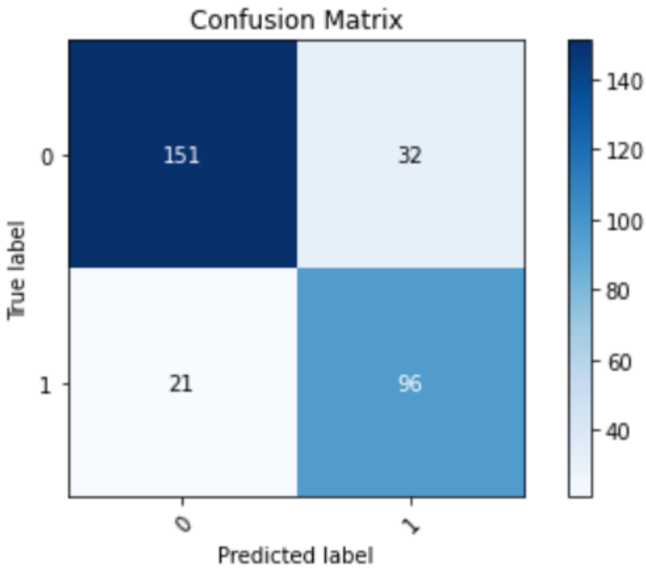


Figure 3: Confusion matrix of a simplistic RF algorithm

below with the confusion matrix (true labels versus given labels). The fact that a very very simple model could give such encouraging accuracy lead me to think that there was a lot on the table.

After noticing first hand how a simple tree-based technique could be suited, my tutor advised me to look into more recent variant : AdaBoost (2003) and XGBoost (2014). AdaBoost stands for adaptive boosting and XGBoost stands for extreme gradient boosting.

In contrast to RF, in AdaBoost each tree of the "forest" is usually one node and two leaves. As a "tree" with just two leaves is not adequate to make good decisions, every tree is weighted. Furthermore, the order in which trees are made is important, indeed, each subsequent tree takes the error of the previous one into account[14]. On the other hand, XGBoost, for classification (it can also be used for regression), is a much more intricate and complicated method, one of which I will not go over in this report, indeed to explain XGBoost, even in simple terms, an entire article or report is needed. See the original paper by Chen and Guestrin [6] for details[15][16].

With XGBoost being more recent, intricate and interpretable than AdaBoost, I decided to focus only on the former. The big advantages of those two random forests variants is the fact that they can be combined with other types of algorithms to improve the performance. Thus, I tried two different combinations of XGBoost with neural networks, both resulting in very good performances.

[14]See Youtube video : AdaBoost Clearly Explained - Josh Starmer
[15]Illustrated guide to XGBoost
[16]XGBoost explained simply by Josh Starmer

The first one I tried was XGBoost combined with VGG16 Simonyan and Zisserman [19] (2014), a very famous CNN for classification and detection. The big advantage of VGG16 is that, in Python (among others), the model also exists pre-trained. The pre-training means that the weights have been calculated on a specific dataset, in this case VGG16 is trained on the ImageNet dataset[17] (Deng et al. [9]). ImageNet consists of more than 14 millions images organized in more than 20k subcategories, and 27 high-level categories; there is all kinds of images and scenes, from wildlife to appliances and utensils.[18]

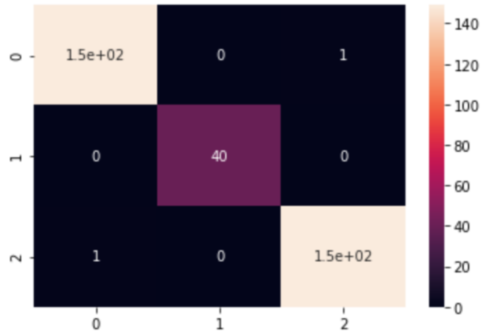
The second one I tried was XGBoost combined with Inception-ResNet (V2, Szegedy et al. [20] 2016), a more recent CNN also implemented with a pre-trained version on Python. Inception-ResNet V2 is also pre-trained on ImageNet. After adapting two scripts[19], to 3-class classification : water, biological spills and oil spills (petrol).

Using default XGBoost parameters for both models, the performance accuracy-wise, were 99.41% and 94.12% respectively for the Inception-ResNet and VGG16 implementations (on the tests sets). The confusion matrices for both model for testing and validation sets are presented below. The main takeaway from those are the fact that only one oil spill image, in 4 tests, was misclassified. And this is far more important than the misclassification of biological spills as water or vice-versa.

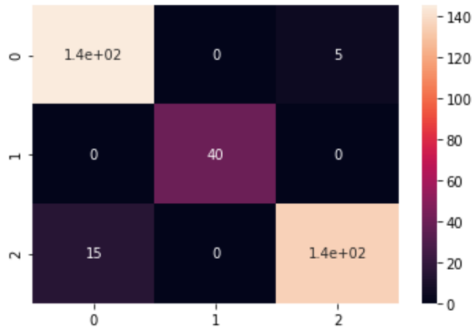
In this configuration, XGBoost act as the final (and only) decider and the VGG16 model as the trees weight's. When these methods were tested, I only had petrol images at my disposal and they all came from a single scene (Mexico's bay area). Hence, the lack of diversity , was a factor to consider. After receiving diesel leaks images, I tested the trained model on a single, pre-processed image of a diesel leak, the result was beyond disappointing. Indeed, in classification problems, when using XGBoost, the assumption, unless stated otherwise, is that there is equal probability between all classes. Thus with 3 classes, the probability of any image to be in the "water" class is $\frac{1}{3}$. And this is exactly the overall result I got for diesel leaks, *without retraining the model*.

In an another situation I would simply have retrained the model with an additional class and data for the training/testing sets. However, here, the lack of data of diesel leaks was a major hurdle. Therefore, after removing the diesel leaks from the test or validation sets, I put aside those two models, with the intention of coming back to them if I felt that the data received was sufficient for a new training phase. Unfortunately, this didn't happened in the duration of my internship. Instead, I reviewed and cleaned the code so that someone else is able to make it their own and use it if sufficient data becomes available. However, nothing guarantees that these networks, in their current states, will

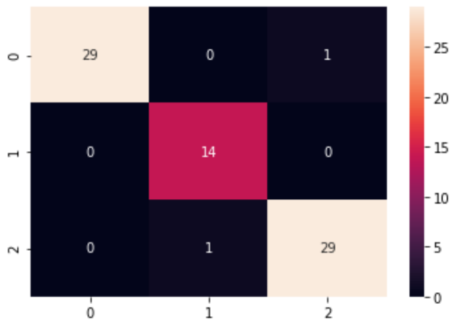
[17] See image-net.org
[18] See devopedia.org/imagenet
[19] See python for microscopists GitHub repository



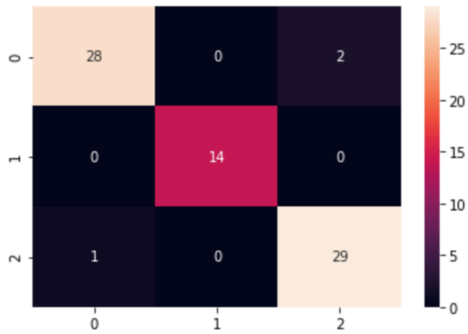
(a) Confusion matrix on the test set, Inception-ResNet model



(b) Confusion matrix on the test set, VGG16 model



(c) Confusion matrix on the validation set, Inception-ResNet model



(d) Confusion matrix on the validation set, VGG16 model

Figure 4: 0 : biological spills, 1 : oil spills, 2 : water

be able to properly classify diesel leaks *and* petrol leaks, it might be a case of having to separate both into distinct models.

Unsurprisingly, no algorithm tried was able to indifferently classify petrol and diesel spills from water and biological spills. Indeed, training a single model for two types of leaks revealed to be as effective as a role of a dice. Thus, I discarded the possibility of implementing and propose to HALIAS a classification model.

A completely different approach revealed to be the solution, namely, anomaly detection.

3 A Transversal Approach : Anomaly Detection

Anomaly detection is an interesting angle to the problem because with diesel leaks a *texture* change can be seen, however in the event of a petrol leak, a *color* change is noticeable. And this is without mentioning the difficulty and price of getting diverse oil leaks images.

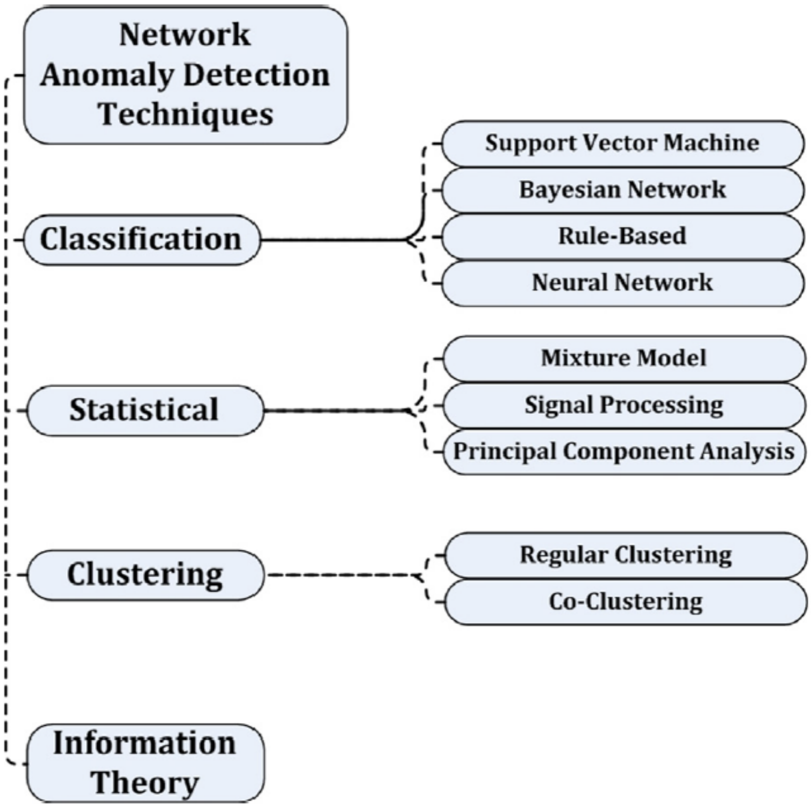


Figure 5: Anomaly detection as of 2016 by Ahmed et al. [1]

This is a theme that goes back to the late 90's-early 2000's, when more and more research were being done in banking intrusion or computer network traffic. The field generalized to time series and bank fraud (among others), but the highly-demanding nature of images, sound processing or videos made anomaly detection unsuitable for this kind of data. Therefore, even when neural network started to emerge in the mid 2010's, these techniques (like RNN) were often coupled with statistical approaches to lessen the computational burden Ahmed et al. [1]. The diagram above (5) details the state of anomaly detection research as of 2016.

An interesting side to statistical anomaly detection lie on the fact that they do not rely on training a dataset, PCA (principal components analysis) for example does not require any statistical distribution assumptions for the data. PCA is also a well-known dimensionality reduction method relatively low-cost computationally-wise. With PCA applied to anomaly detection the idea is when a "point" diverge too much from other conglomerate then it is flagged by a defined threshold. This holds thanks to the fair assumption that there is significantly more normal than anomalous data.

Classification-based techniques include standard algorithms like Support Vector Machine (SVM[20]) in both supervised and unsupervised learning forms, or its variant : Robust SVM (RSVM by Le et al. [14]) which ignores noise. Nonetheless, rule-based and neural network seemed to be the most promising way forward for high-dimensional data. Not only given the type of data but also theoretically. Indeed, the former method basic idea is to learn from only normal data and flag as an anomaly anything that is not considered similar to what has been learned.

This last idea lead me to generative adversarial networks for anomaly detection, those combine the efficacy of neural networks with the parsimony of a specific type of data when flagging anomalous images. Before directly diving in the core problem, I will try to clearly explain what is a generative adversarial network, then, give some practical exanples.

3.1 Generative Adversarial Network

In order to avoid the troubles of implementing 2 algorithms (one for diesel, the other for petrol leaks) with limited *anomalous* data, I decided to explore something relatively new namely anomaly detection using Generative Adversarial Networks (GAN). The main idea is to train a network only on "normal" data (here water and biological slicks images) to model a *normal behavior* and then detect anomalous images with the help of an *anomaly score* and a threshold.

Anomaly detection using GANs was first used for real-life problem where huge amounts of data (e.g images) are hard to get. For instance, luggage scanning in airports. There is enormous amount of images containing clothes, laptops, books, and so on, but if you want to detect the presence of knives, firearms and other prohibited items, it is next to impossible to gather the required amounts of those images for standard deep learning (or machine learning) classification algorithms.

GANs first appeared with Goodfellow et al. [11], at first they were usually made of 2 big blocks of layers, a generator G and a discriminator D trained simultaneously. The generator is supposed to capture the data distribution and generate new data with it, while the discriminator aims to compute the probability that the generated data is fake. During training, generator and discriminator are effectively two models competing against each other at the same time, trying to achieve high robustness and precision. Indeed, the

[20]See en.wikipedia.org/wiki/Support-vector_machine : SVM definition

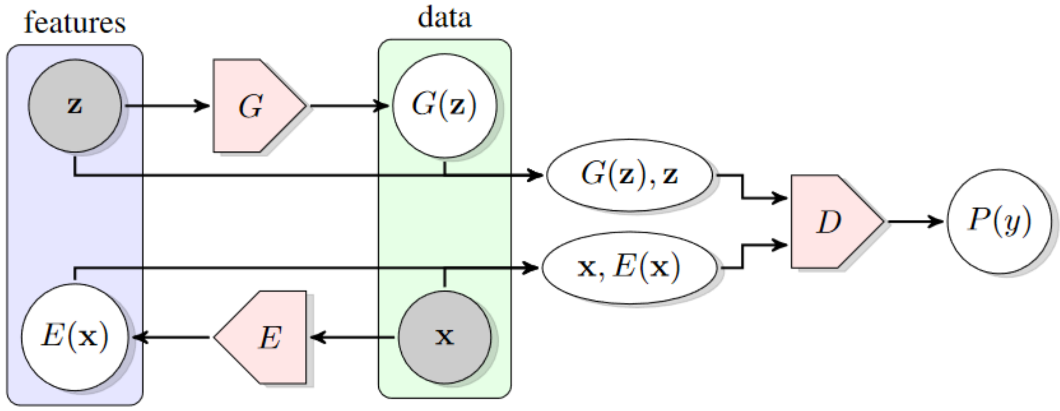


Figure 6: BiGAN proposed structure in Donahue et al. [10]

training objective of G is to maximize the error of D, and D training objective is to minimize the distance between generated and true data.

Since Goodfellow et al. [11] work, two major GANs extension emerged (Mattia et al. [16]), Conditional GAN (CGAN) and Bidirectional GAN (BiGAN):

- CGAN are conditioning either G or D to a new information y which can be some constraint or class labels. In the case of y being class labels, it transforms a previously unsupervised problem into a semi-supervised one; moreover the objective function to minimize during training is the same as the original GAN plus the condition on the input data and latent vector fed to the discriminator and generator. (Mirza and Osindero [17])

- BiGAN introduce an encoder E with $E = G^{-1}$, as well as G and D, E is a non-linear parametric function, the idea is to project data back into the latent space, hence the name bidirectional. Now, D discriminates not only x against $G(z)$ (where z are the latent representations, and x the original input data) but also tuples $(x, E(x))$ versus $(G(z), z)$. The new training objective is a minimax objective function, as before, but where both generator and encoder must (roughly) maximize the error of the discriminator. (Donahue et al. [10])

While CGAN enables semi-supervised learning (when the conditioning is a labelled class), BiGAN can effectively act has an implicit fully supervised model. Indeed, the latent representation z, which is put back in the latent space, can be considered as an implicit label for an input data x, thus without the need of labelled datasets.

Given that they are mostly human-made, large labelled datasets are known to be hard and/or expensive to get, especially for real-life problems. In this context, GANs provide a way of nullify this constraint. Moreover, GANs architecture do not depend on any assumptions regarding data type or structure, they are very generic and a single type of architecture (as we will see later) can be used on extremely varied datasets and problems.

However, in recent works (e.g. Akcay et al. [2]), high-level accuracy require either significant training time and/or extremely high-end piece of hardware (GPU and CPU). So this kind of "Swiss knife" does not come cheap computationally-wise. This issue was of one of the main concern of my internship as I needed to find both a "Swiss knife" and a computationally-cheap model.

3.2 Anomaly Detection using GANs

Anomaly detection using GANs first emerged in 2017 with Schlegl et al. [18] and their AnoGAN algorithm, based on a standard GAN framework, namely, only with two blocks G and D. The general idea is the following, given that a generator G only learns the mapping of normal data, it can only reconstruct (more or less) normal data. Therefore, during the testing phase, if anomalous data are given to the networks, the difference between the reconstructed data and the actual data will be notable.

There is various ways to detect said difference, from what I have read, the most commonly used are L1 distances. The latter are computed between the generated data and the actual data, pixel by pixel in the case of an image (or video), but dissimilarity measures and other types of combinations are also used. This helps to determine an *anomaly score* $A(x)$, finally a threshold on the anomaly score is needed to classify inputs as anomalous or normal.

As of 2021, there are only a few detailed architecture and open-source algorithms available. Roughly 10 to 12 methods are available, some are different only on the choice of the anomaly score function, or the way of computing the differences between generated and actual data (loss functions). Those algorithms can split in two groups : the single-pass methods and the iterative ones (one of which is AnoGAN).

As could be expected, the single pass methods reach significantly better accuracy than iterative methods which are extremely time-consuming during *both* training and testing phase. Two of the best published algorithms, based on the are under the curve results on the MVTec-AD open source dataset are ITAE (Inverse-Transform AutoEncoder -for anomaly detection) Huang et al. [13] (2019) and CBiGAN (Combining BiGANs and AutoEncoders) Carrara et al. [5] (2020). These algorithms have the particularity to both use autoencoders.

One of the original objective of autoencoders are dimensionality reduction. An autoencoder is an artificial neural network (ANN) made of an encoding and decoding phase, both phases mirroring each other in terms of layers. The encoder effectively compresses the input data, ignoring the noise, to extract the essential information, meanwhile the decoder aims to reconstruct the input from the core information extracted to produce an outcome.

Particularly used for visual data (images or videos), as a way of de-noising (the input image is simply reconstructed with as little noise as possible), coloring, etc.. This kind of dimensionality reduction attribute is extremely useful when one desire to lessen the computation time, either during training and/or testing.

To my knowledge this kind of neural network has not been applied to oil leaks yet.

3.3 Delving Into The Maths

In this section, the maths behind GANs for anomaly detection will be detailed through 2 specific architectures : GANomaly (Akcay et al. [2]) and CBiGAN (Carrara et al. [5]). These two were chosen because they are both very recent (2018 & 2020), they use the same kind of data and they represent two interesting approaches : one single pipeline (GANomaly) & double auto-encoder mapping (CBiGAN). For each big component of the networks I will give both models' take on the task.

Note : this section can be omitted without loss of information.

Hypothesis

Both networks have the same global hypothesis (these are common to most GAN models). These models are based on the following hypothesis.

- There are equally few abnormal images in the *test set* and *real life*. Meaning, anomalous images will always make up only a very small part of the dataset considered (bar the training set).
- The latent vector of a GAN represents the *true* distribution of the data.
- Parametrization should not be suitable for generating abnormal/anomalous data.
- The most important hypothesis may be that G is *not* able to reconstruct the abnormality, independently of the fact that E manages to map the input (x) to the latent vector (z).

Loss functions

- *GANomaly*. Three losses functions compose the objective function.

$$\mathcal{L}_a = \mathbb{E}_{x \sim p_X} ||f(x) - \mathbb{E}_{x \sim p_X}(f(G(x)))||_2$$

$$\mathcal{L}_c = \mathbb{E}_{x \sim p_X} ||x - G(x)||_1 \quad (\Leftrightarrow \mathbb{E}_{x \sim p_X} ||x - \hat{x}||_1)$$

$$\mathcal{L}_e = \mathbb{E}_{x \sim p_X} ||G_E(x) - E(G(x))||_2$$

  Where $\mathcal{L}_a$, $\mathcal{L}_c$ and $\mathcal{L}_e$ are respectively the adversarial loss, the contextual loss and the encoder loss; x is the original input, f is a function that outputs an intermediate layer of the discriminator D (for a given input), p_X represents the data distribution, G and E are respectively the generator and the encoder.

  The adversarial loss is the L_2 distance between the *feature representation* of the original image and the generated images. The contextual loss is the L_1 distance between the *original image* and the generated images. The encoder loss is the L_2 distance between the *bottleneck features* of the *input* and the encoded features of the generated image (thus $G_E(x) = z$ and $E(G(x)) = \hat{z}$).

- *CBiGAN*. Three losses functions are needed, during both training and testing phases, independently from the objective function.

$$\mathcal{L}_D = -\frac{1}{N}\sum_{i=1}^{N} D(x_i, E(x_i)) + \frac{1}{N}\sum_{i=1}^{N} D(G(z_i), z_i)$$

$$\mathcal{L}_C(x, z) = \mathcal{L}_R(x) + \mathcal{L}_{R'}(z)$$

$$\mathcal{L}'_{E,G} = (1-\alpha)\mathcal{L}_{E,G} + \alpha\mathcal{L}_C$$

  With, $\mathcal{L}_D$ the discriminator loss, $\mathcal{L}'_{E,G}$ the loss for the auto-encoder (E and G), α being a weight controller, if $\alpha = 0$ the double auto-encoder is *not* adversarially trained, on the contrary, if $\alpha = 1$ the network is reduced to a standard BiGAN model (Donahue et al. [10]). While x ans z are defined as for GANomaly, N is the total number of training set elements.

$$\mathcal{L}_{E,G} = \frac{1}{N}\sum_{i=1}^{N} D(x_i, E(x_i)) - \frac{1}{N}\sum_{i=1}^{N} D(G(z_i), z_i)$$

  Finally, $\mathcal{L}_C$ is a cycle consistency regularization term to enforce the alignment of learning, namely, it is a term to assure that $G = E^{-1}$, with,

$$\mathcal{L}_R(x) = ||x - G(E(x))||_1 \quad \& \quad \mathcal{L}_{R'}(z) = ||z - E(G(z))||_1$$

Where $\mathcal{L}_R(x)$ and $\mathcal{L}_{R'}(z)$ are the pixel-based reconstruction error (for both original and reconstructed inputs).

Objective function

- *GANomaly.* The objective function is then defined as follows.

$$\mathcal{L} = \alpha\mathcal{L}_a + \gamma\mathcal{L}_c + \beta\mathcal{L}_e$$

  Where, α, γ and β are weighting parameters ajusting the impact of each individual loss during training.

- *CBiGAN.* The objective function is a minimax function.

$$\min_{G,E}\max_{D} \quad \mathbb{E}_{x\sim p_X}[D(x, E(x))] - \mathbb{E}_{z\sim p_Z}[D(G(z), z)]$$

  As before, $E(x) = z$ and $G(z) = \hat{z}$.

Anomaly score

- *GANomaly.* The anomaly score, defined as follows, is then scaled to be interpretable within a probabilistic range.

$$A(\hat{x}) = ||G_E(\hat{x}) - E(G(\hat{x}))||_1$$
$$\mathcal{S} = \{s_i : A(\hat{x}_i), \hat{x}_i \in \hat{\mathcal{D}}\}$$
$$s'_i = \frac{s_i - min(\mathcal{S})}{max(\mathcal{S}) - min(\mathcal{S})}$$

  Where s'_i is the scaled (probabilistic) anomaly score for a given input and $\hat{\mathcal{D}}$ is a test set of the test sample $\hat{x}$.

- *CBiGAN.* The anomaly score uses the previously defined reconstruction loss ($\mathcal{L}_R$), but also a feature based discriminator loss ($\mathcal{L}_{f_D}$).

$$A(x) = (1 - \lambda)\mathcal{L}_R(x) + \lambda\mathcal{L}_{f_D}(x)$$

  With λ being a balancing hyper parameter and,

$$\mathcal{L}_{f_D}(x) = ||f_D(x, E(x)) - f_D(G(E(x)), E(x))||_1$$

Where $f_D(x, z) \in \mathbb{R}^d$ is a feature vector extracted from an intermediate output of discriminator D.

Evaluation Metrics

- *GANomaly.* Area under ROC (Receiver Operating Characteristic) curve (AuROC) was used, which is the area defined by the TPR as a function of FPR for each TPR-FPR pair given by a different threshold value.
- *CBiGAN.* AuROC was also used as a threshold-free metric, along with Balanced Accuracy (BalAcc), dependant on the threshold which is defined by :

$$BA = \frac{TPR + TNR}{2}$$

  Where TPR stands for true positive rate and TNR for true negative rate.

Threshold

- *GANomaly.* The treshold ϕ is determined iteratively during the testing phase, in which different value of ϕ are tested, the one yielding the best results (on average) regarding to the AuROC gets the nod.
- *CBiGAN.* The threshold is decided iteratively, as for GANomaly, but by maximizing the Youden's index, namely $J = TPR - TNR - 1$.

We saw 2 very different approaches, even though the initial hypothesis are the same and the idea are identical. If it was still needed, this demonstrates the variety of the available methods to a single problem.

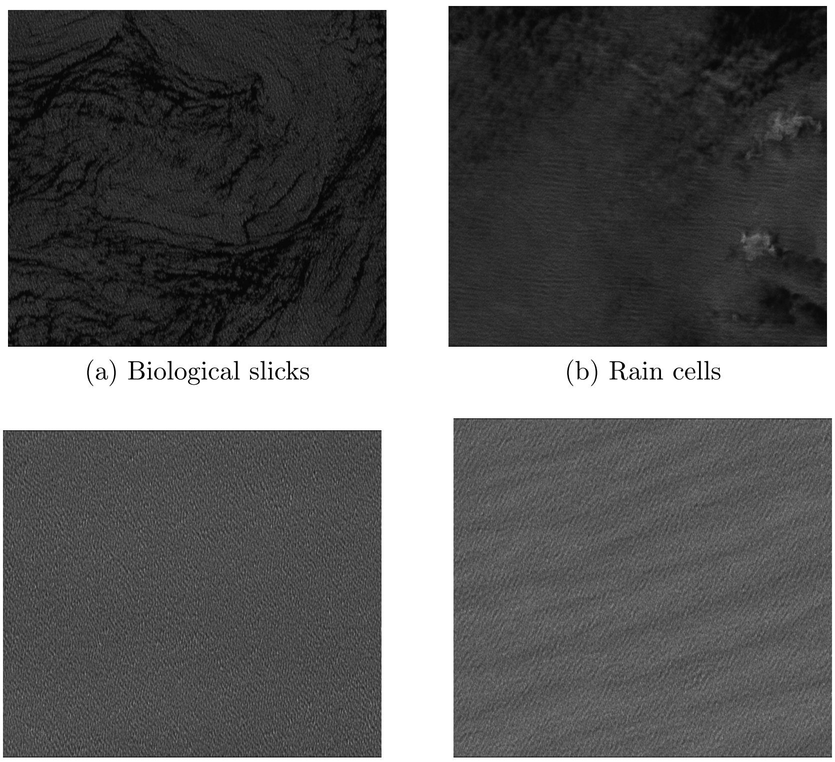
4 Algorithm & Performance

In this final section I will detail the algorithm I chose to implement, its goal, why I chose it and on which data I trained and tested it. Along with the procedure used on the data, the way I split it into sets, before being passed to the network.

4.1 Dataset description and constitution

I had at my disposal 3 sources of data, one with roughly water and biological slicks images, an other provided by the company HALIAS for diesel leaks, and the last one provided by my tutor for petrol leaks. All of those images vary in shape (e.g. not a rectangle), sizes, resolution and color (some in 2D, others in 3D).

The biological slicks and water SAR images came from an open-source labelled dataset Chen et al. [7]. They cover almost every possible situation with great variety as there is around 5 000 samples for each class (below some pre-processing examples). However I decided to keep only certain classes, namely: Biological Slicks, Rain Cells, Micro Convective Cells, Wind Streaks, Pure Ocean Waves. I discarded the others because they contain things such as ice banks and icebergs, which corresponds to locations where oil leaks are not really a concern.



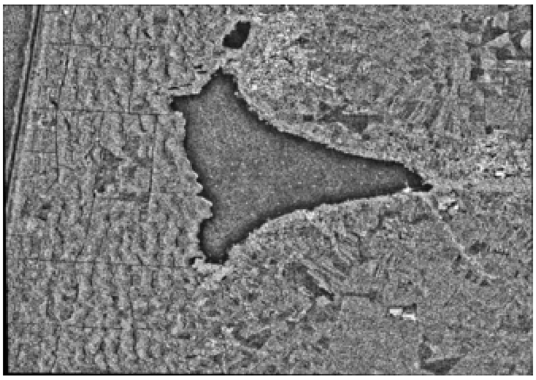
(a) Biological slicks

(b) Rain cells

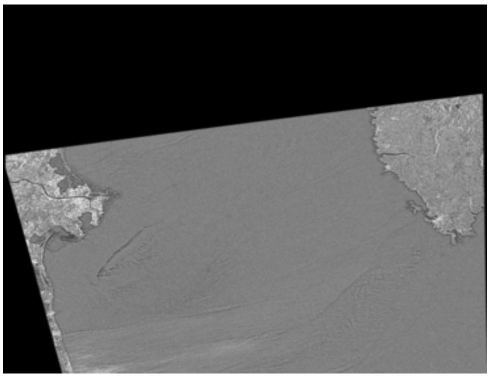
(c) Ocean waves

(d) Wind streaks

Diesel leaks vary in shape and color but they are noticeable thanks to the change in texture, as can be seen below, HALIAS provided us both partially and totally visible leaks. The types of shape available turned out to be a problem when slicing up the pictures. Due to this, there was quite a loss and extensive data augmentation was needed (unlike the previous set of data). I used standard data augmentation techniques including skewing, mirroring, rotation and zooming. Luckily the sizes of the images allowed us such a treatment without too much repetition, which could be armful to any model.

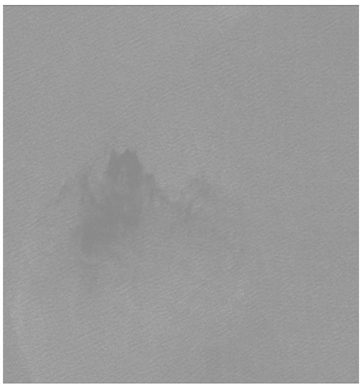


(a) Totally visible leak

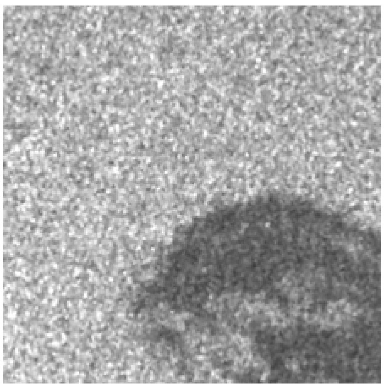


(b) Partially visible leak with "faulty" shape

Finally, the last source of data is derived of a single ridiculously large image (19010 $\times$ 14436 pixels), where both water-only and petrol leaks were extracted using a k-means algorithm. A global mean color for the entire image was computed, then after slicing the image into around 800 smaller ones, the 4 most present colors and the local mean color (here, shades of gray) were collected for each sample. As the leaks appear several steps darker than there surrounding areas; if the color level of one of the 4 colors was under, either the local *or* the global mean, then I considered the sample to contain petrol. That way, water and petrol were placed in separated sets. Below, examples of extracted samples (before processing).



(a) Water sample



(b) Petrol sample

Given the different nature of my 3 sets of data, I [applied] various types of pre-processing. We will see later that an another pre-processing step is part of the algorithm I used, thus the following steps goal was essentially to put every images on the same starting point.

For the water and lookalikes set, the main modifications were image sizes and brightness. I used an arbitrary color value (i.e. 153 GBR, also corresponds to the overall color mean of the third source of data), to either brighten or darken every images up to that level. Finally, I simply resized them to fit a 128 $\times$ 128 pixels square.

Next set, consisting of diesel leaks turned out to be the most challenging. I applied consecutively brightening (or darkening, depending on the color level), rotation and cropping to extract the biggest rectangular shape, Gaussian blurring (after some tests I opted for a 15 $\times$ 15 kernel of size 4), slicing (exclusively for the biggest samples) and finally resizing. The samples extracted from the last set only needed the adequate brightening (or respectively darkening) and resizing.

The constitution of train, test and valid set (for anomaly detection) was fairly straightforward. After gathering all normal images (i.e. water, biological slicks, rain cells, etc..) from every source of data, I split approximately 18 000 images into three sets, 80% (around 15 000) went in the training set, 18% in the test set (around 3 000) and 2% went into the validation set (around 350). Naturally, no oil leaks images are needed in the training set for the kind of algorithm I chose. Therefore, the gathered oil leaks images got separated in two sets : test and validation. In the end I only had around 250 usable oil leaks images (petrol and diesel added up). Hence, I decided to put them all in the validation set, and discard (at least for a while) the normal images sitting in the test set.

This limitation will hopefully be compensated by the number of batches and patches defined for the training phase of the chosen model.

4.2 Final Algorithm

After wandering between algorithms and architectures, I settled on a recent all-in-one code using up to date modules : the official code associated with the CBiGAN paper by Carrara et al. [5][21]. The CBiGAN architecture is one of the two best models at detecting anomaly using GANs Ahmed et al. [1]. The other best, ITAE (Huang et al. [13]) does not have an official (and open source) reproducible code as of july 2021, therefore the choice between these two was easy. Nonetheless, if the ITAE code ever becomes publicly available it could be interesting to try adapting it.

[21] See github.com/fabiocarrara/cbigan-ad , for the official code implementation

Entirely written in Python 3, the CBiGAN implementation available uses tensorflow back-end and allows one to reproduce the paper results on the MVTec-ad dataset[22]. The latter is a well-known benchmark dataset suited for unsupervised anomaly detection ; it is made of 10 categories of objects (e.g. toothbrush, screw,...) and various textures (e.g. leather, carpet). A detailed architecture with layers specifications and the algorithm's structure are presented below.

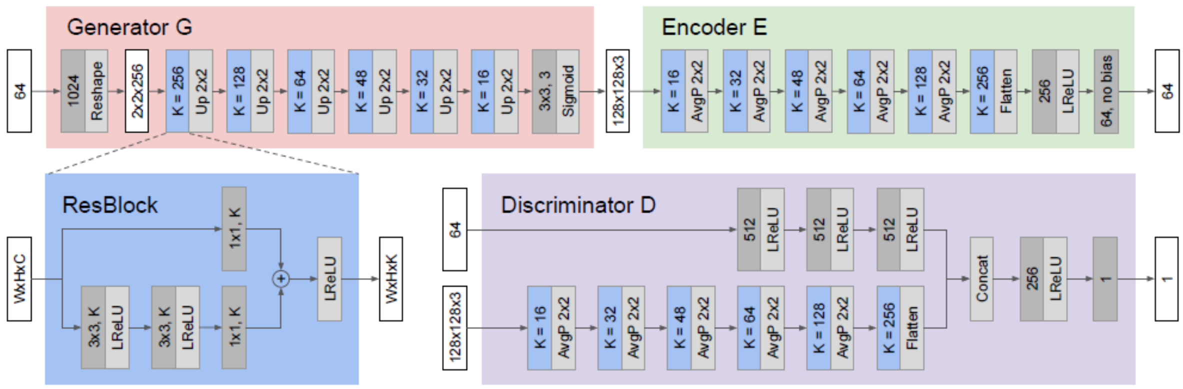


Figure 7: Implementation of E, G, and D (for the MVTec-ad dataset) by Carrara et al. [5]. White blocks specify input and output shapes, and dark gray blocks indicate learnable convolutions or fully-connected layers (Fig.3 in the original paper)

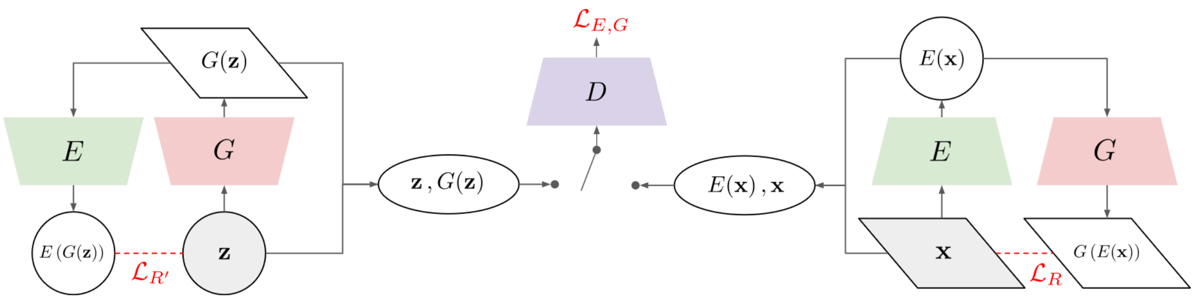


Figure 8: CBiGAN structure. (Inputs are in gray, parallelograms and circles respectively represent samples in the image and latent space, ellipses represent the inupt of the discrimintor, red dashed lines represent $\mathcal{L}_C$. Fig.2 in Carrara et al. [5])

Although I didn't alter the code's core, namely, losses, models and scoring methods I modified everything else in order to fit to the needs of both my datasets and the need of the company. Therefore, in the end one of the main challenge was to "skin" the code to the absolute minimum. Obviously at first I try to adapt it in the simplest way possible to test it and see the provisional results. As those were very good, I continued to make the code as little as possible.

[22]See mvtec.com/company/research/datasets/mvtec-ad/ for description

The first dive into the code were not easy as it was my first time working on a intricate deep learning algorithm. However, the courses we had this year in python for data science gave me both a solid base on which to work on, and the tools on how to get additional information.

Most of the structural modifications I made were on the scoring side of the code. I had to make changes in order to extract explicit information, this includes the anomaly score for a single image, the value of the different thresholds when some factor λ varies for one and multiple test images. I also added steps to extract csv files of this data and to save the models as json files (i.e. encoder, decoder and the discriminator feature extractor) in order to use them independently later on. Obviously for the need of production, I added two scripts that weren't needed in the paper, namely :

- single image pre-processing script based on the method explained earlier,
- single image anomaly detection script which is specific to the data and is making use of the saved G, E and D and a chosen threshold determined with the testing set evaluation.

In the end I removed quite a lot of scripts and chunks from the original code, and narrowed it down to 6 python scripts for the training and testing phases, with 2 more for single images use.

4.3 Results & Discussion

The training phase, with 100 batches, was actually a lot quicker than I (or even my tutor) expected for a <15k images training set. Indeed, each batch iteration takes about 1.2 to 1.4 seconds, all in all an entire training phase can be done in less than 1min45 to 2 mins. The same can be said for the testing phase, with the size (around 300 images), it does not take more than 10 to 20 seconds to extract a new threshold or accuracy measure. For reference, I ran these tests with a GeForce RTX 2060 GPU and an AMD Ryzen 7 CPU.

A lot of parameters are needed in this model, both for the training phase and the anomaly scoring specifications. At first I relied on the default parameters bar the number of batches and patches to lessen the computation burden. After successful testing I ultimately kept most of the default parameters. The number of batches is set at 100 and the number of patches at 128. Future changes are let to the discretion of the receiving company as every parameter's role is explained in a dedicated file, explication gathered from the original code and thorough study of the code's functionalities.

Two different metrics are available to measure the quality of the network on the final test/validation set.

- We got AuROC = 0.99940 ;
- We got BalAcc = 0.99306.

These results being extremely good, it is always interesting to go a bit deeper. Therefore, I looked at the scores given to each images per batches and λ, the weight value in the loss function, and the threshold, to see if there was a lot of images close to the threshold or not. It was not the case, except for the only two misclassified images, which were on both sides of the threshold, indifferently of the value of λ. This was hugely reassuring as that was where the previously tested models failed.

Of course, the results are really good and promising, nevertheless we must recall the nature of the network.

There is an important hypothesis for GANs to be efficient. Even before detecting anomalies a GAN's goal is to learn the data distribution of the input training data to be able to recreate it. Hence, the implicit hypothesis is that *there is* a data distribution to be learned. Meaning that, input data can be mapped. Which can be tricky in some cases, resulting in poor reconstruction, thus poor anomaly detection.

Moreover, even if the training set is large and contains a wide range of images and marine situations, we cannot forget the fact that other situations, type of images can occur. For example, very noisy images can be tricky to manage, the pre-processing steps I included do not vary much depending on the image considered, thus, one image might need more blurring than another, and vice versa.

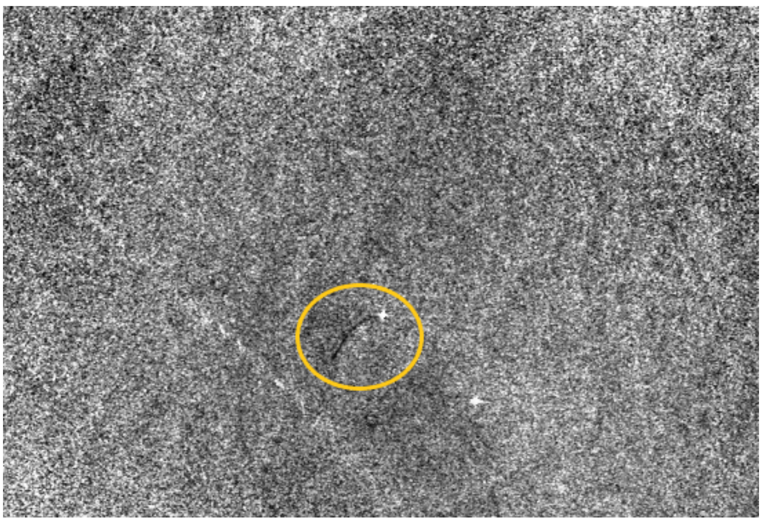


Figure 9: Satelite image with petrol leak (yellow circled)

For illustration, one of the satellite image HALIAS gave me Fig.9, was very noisy and the presence of the leak could only be noticed by a trained human eye. They provided me the image with (and without) a pointer so that I was able to see it, but without it I wouldn't have known. As can be expected, when the pre-processing step were applied, the leak was no longer visible by the naked eye. When passed into the network, it did not classify it as an anomalous image. The only difference was the resolution, the images I worked with previously were all *very high* resolutions, whether the size was actually big or not. Hence making the resolution of the passing image a very important factor to consider.

A GAN can only be efficient in very specific situations, regardless of the context, results accuracy-wise might be very interesting and encouraging, we must not hide the fragility of this type of network to outliers that are *not* anomaly.

This limitation has been passed on to the company that will be using the algorithm, so they are well aware of the issue. The nature of oil leaks (diesel or petrol) makes it safer to flag an image that is on the brink of being anomalous, than the contrary. Indeed, the human element cannot ignored, if an anomaly score is very close to either side of the threshold then we can, for example, add a precautionary note, so that the image can be checked by a living soul.

Hence, cooperation between us (my tutor and I) and HALIAS, the beneficiary company had fruitful conversations about what can and cannot be done, or what their preference were in terms of code architecture and the layout of automation. I handled the automation of the code at a computer level, to match to the best of my abilities, the need of the company. This code was then to be incorporated into a software and an online service by their software engineer.

In the future, it could also be possible to create a pipeline with multiple algorithms. For example the CBiGAN adaptation detection model first, then if an image is flagged as anomalous, it could be then pass through an another classification algorithms, (like the XGBoost model combined to VGG16) to distinguish diesel from petrol leaks, and finally to a segmentation algorithm (like the ones discussed earlier as future research), to identify the size or even direction of the leak. To the contrary, if an image is deemed normal, then the pipeline would stop there and keep the computation time and the server's burden low.

5 Conclusion

In this report I explained what lead me to choose an anomaly detection algorithm and why but also detailed the various possible approaches to the same problem. The final algorithm was delivered to the supporting company, HALIAS, along with every methods tried during the internship. This include failed, or partly failed, attempts like classification using the Inception-ResNet model coupled with XGBoost discussed in the first section. As we saw, numerous methods could not be applied due to the lack of (diverse) data or hardware limitations. Overall, solid statistical foundation provided by the first year of my master's degree, allowed me to explore different things as almost every method cited in this report derive from the statistical world.

In order to automatically detect oil spills (diesel and petrol) on satellite images, without having to design 2 distinct models (one for each type of leak), I took a transversal approach with anomaly detection using GANs. It gave surprisingly good results, balanced accuracy at 99.31%, without being too demanding computationally and hardware wise, and dodging the lack of anomalous data (i.e. leak images) hurdle.

Perspectives for future research are abundant. Some depend on new data being available, as with classification or segmentation using neural networks, others depend on parallel research as with scattering transform for semantic segmentation. The latter would be used to retrieve the total area of a leak and alert the relevant authorities so that they can deploy the necessary means to treat the area affected.

A pipeline with different level of alerts could also be an interesting way forward. First detecting the presence of an anomaly, if there is, a classification between diesel, petrol or event biological slicks is made, and finally if a client desires to know the extent of the leak he can opt for a final step in the pipeline, namely a segmentation algorithm that outputs the size of the concerned area, in real-life terms (i.e. meters or kilometers). On the other hand, if nothing is flagged as anomalous then the pipeline stops there, so that the whole process can be cheap computationally wise as leaks are not supposed to be frequent.

References

[1] Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60: 19–31, 2016. ISSN 1084-8045. doi: https://doi.org/10.1016/j.jnca.2015.11.016. URL `https://www.sciencedirect.com/science/article/pii/S1084804515002891`.

[2] Samet Akcay, Amir Atapour Abarghouei, and Toby P. Breckon. Ganomaly: Semi-supervised anomaly detection via adversarial training. *CoRR*, abs/1805.06725, 2018. URL `http://arxiv.org/abs/1805.06725`.

[3] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017. doi: 10.1109/TPAMI.2016.2644615. URL `https://ieeexplore.ieee.org/abstract/document/7803544`.

[4] Diego Cantorna, Carlos Dafonte, Alfonso Iglesias, and Bernardino Arcay. Oil spill segmentation in sar images using convolutional neural networks. a comparative analysis with clustering and logistic regression algorithms. *Applied Soft Computing*, 84:105716, 2019. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2019.105716. URL `https://www.sciencedirect.com/science/article/pii/S1568494619304971`.

[5] Fabio Carrara, Giuseppe Amato, Luca Brombin, Fabrizio Falchi, and Claudio Gennaro. Combining gans and autoencoders for efficient anomaly detection. *CoRR*, abs/2011.08102, 2020. URL `https://arxiv.org/abs/2011.08102`.

[6] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016. URL `http://arxiv.org/abs/1603.02754`.

[7] Wang Chen, Mouche Alexis, Tandeo Pierre, Stopa Justin, Longépé Nicolas, Erhard Guillaume, Foster Ralph, Vandemark Douglas, and Chapron Bertrand. Labeled sar imagery dataset of ten geophysical phenomena from sentinel-1 wave mode (tengeopsarwv), 2018. URL `https://www.seanoe.org/data/00456/56796/`.

[8] Wanli Chen, Xinge Zhu, Ruoqi Sun, Junjun He, Ruiyu Li, Xiaoyong Shen, and Bei Yu. Tensor low-rank reconstruction for semantic segmentation. 2020. URL `https://arxiv.org/abs/2008.00490`.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[10] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *CoRR*, abs/1605.09782, 2016. URL `http://arxiv.org/abs/1605.09782`.

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL `https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf`.

[12] Hao Guo, Danni Wu, and Jubai An. Discrimination of oil slicks and lookalikes in polarimetric sar images using cnn. *Sensors*, 17(8), 2017. ISSN 1424-8220. doi: 10.3390/s17081837. URL `https://www.mdpi.com/1424-8220/17/8/1837`.

[13] Chaoqing Huang, Jinkun Cao, Fei Ye, Maosen Li, Ya Zhang, and Cewu Lu. Inverse-transform autoencoder for anomaly detection. *CoRR*, abs/1911.10676, 2019. URL `http://arxiv.org/abs/1911.10676`.

[14] Trung Le, Dat Tran, Wanli Ma, Thien Pham, Phuong Duong, and Minh Nguyen. Robust support vector machine. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 4137–4144, 2014. doi: 10.1109/IJCNN.2014.6889587.

[15] Xiaofeng Li, Bin Liu, Gang Zheng, Yibin Ren, Shuangshang Zhang, Yingjie Liu, Le Gao, Yuhai Liu, Bin Zhang, and Fan Wang. Deep-learning-based information mining from ocean remote-sensing imagery. *National Science Review*, 7(10):1584–1605, 03 2020. ISSN 2095-5138. doi: 10.1093/nsr/nwaa047. URL `https://doi.org/10.1093/nsr/nwaa047`.

[16] Federico Di Mattia, Paolo Galeone, Michele De Simoni, and Emanuele Ghelfi. A survey on gans for anomaly detection. *CoRR*, abs/1906.11632, 2019. URL `http://arxiv.org/abs/1906.11632`.

[17] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014. URL `http://arxiv.org/abs/1411.1784`.

[18] Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. *CoRR*, abs/1703.05921, 2017. URL `http://arxiv.org/abs/1703.05921`.

[19] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. 2015.

[20] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016. URL `http://arxiv.org/abs/1602.07261`.

[21] Linlin Xu, Jonathan Li, and Alexander Brenning. A comparative study of different classification techniques for marine oil spill identification using radarsat-1 imagery. *Remote Sensing of Environment*, 141:14–23, 2014. ISSN 0034-4257. doi: https://doi.org/10.1016/j.rse.2013.10.012. URL `https://www.sciencedirect.com/science/article/pii/S0034425713003805`.

[22] Hang Zhang, Han Zhang, Chenguang Wang, and Junyuan Xie. Co-occurrent features in semantic segmentation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 548–557, 2019. doi: 10.1109/CVPR.2019.00064.

[23] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. *CoRR*, abs/1612.01105, 2016. URL `http://arxiv.org/abs/1612.01105`.