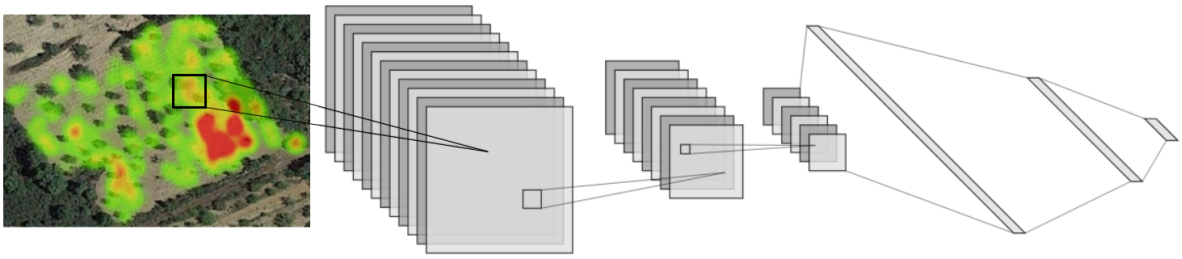
TÉCNICO LISBOA



Radioactive hot-spot detection using machine learning algorithms

Miguel Tomé Ramos e Barros

Thesis to obtain the Master of Science Degree in

Engineering Physics

Supervisors: Dr. Bruno Miguel Soares Gonçalves
Dr. Alberto Manuel Martinho Vale

Examination Committee

Chairperson: Prof. Horácio João Matos Fernandes
Supervisor: Dr. Bruno Miguel Soares Gonçalves
Member of the Committee: Prof. Diogo Manuel Ribeiro Ferreira

January 2021

Acknowledgments

Firstly, I would like to thank my supervisors, Dr. Alberto Vale and Dr. Bruno Gonçalves, for their guidance, insight, shared knowledge and support throughout the realization of this thesis.

A word of appreciation to Instituto Superior Técnico and Instituto de Plasmas e Fusão Nuclear, for all the material and the physical space made available, thus contributing to the improvement of this work.

I would also like to thank Yoeri Brouwer, for his great help with the simulator, and Filipe Mendes for being a great working partner, discussing many ideas and helping with the sensors.

Finally, I would like to thank my parents and my family for their unconditional support with love, understanding and patience, and my friends for keeping me alive, providing moments of fun and distraction during these hard months.

Resumo

A detecção de *hot-spots* radioativos tem sido um desafio para o setor de segurança, especialmente em situações que envolvem ameaças químicas, biológicas, radiológicas e nucleares (CBRN). Este trabalho propõe uma solução baseada em técnicas de Aprendizagem Automática, com foco em Redes Neuronais (NNs), de forma que observações de contagens de intensidade radiológica e respetivas localizações possam ser usadas para estimar o número de fontes radioativas desconhecidas presentes num determinado cenário, e ainda a sua localização e atividade ao mesmo tempo. Para isso, um simulador é utilizado para gerar um conjunto de dados de treino para o processo de treino, e assim, usando o modelo já treinado através do algoritmo de Divisão e Conquista, obtém-se estimativas rápidas e precisas, garantindo a confiabilidade de tal abordagem baseada em NNs.

A solução proposta é testada em cenários com múltiplas fontes, com obstáculos incluídos, e com fontes não pontuais. Ao contrário da maioria dos algoritmos existentes, que começam a falhar em cenários com essas condições, as NNs mostram ser capazes de realizar uma detecção precisa de *hot-spots*, com um baixo número de limitações. Ademais, resultados experimentais, feitos em ambientes de laboratório e em cenários reais localizados em antigos depósitos de minério radioativo, demonstram que o algoritmo é escalável tanto para regiões de grandes como de reduzidas dimensões.

Assim, as NNs demonstram ser capazes de serem uma ferramenta emergente com potencial para fazer a diferença na área nuclear, auxiliando no desenvolvimento de novas técnicas e novas soluções que visam salvaguardar vidas humanas.

Palavras-chave

Hot-spots Radioativos, Aprendizagem Automática, Redes Neuronais, Divisão e Conquista

Abstract

The detection of radioactive hot-spots has been a challenge for the security sector, especially in situations involving chemical, biological, radiological and nuclear (CBRN) threats. This work proposes a solution based on Machine Learning techniques, with a focus on Artificial Neural Networks (ANNs), so observations of radiological intensity counts and corresponding localizations can be used to estimate not only the number of unknown radioactive sources present in a given scenario, but also their location and activity at the same time. For this, a simulator is used to generate a training data set for the training process, and so, using the model already trained through a Divide and Conquer algorithm, fast and accurate predictions are achieved, ensuring the reliability of such an ANN-based approach.

The proposed solution is then tested in scenarios with multiple sources, with obstacles included, and with non-point-like sources. Unlike most existing algorithms, which begin failing in scenarios with those conditions, ANNs have shown that are capable of performing an accurate hot-spots detection, with a low number of limitations. Additionally, experimental results, done in lab environments and real scenarios located at old deposits of radioactive ore, have shown that the algorithm is scalable for very large regions, as well as for very short scenarios.

Thus, ANNs have demonstrated the capability of being an emerging tool with the potential to make a difference in the nuclear field, by helping in the development of novel techniques and new solutions in order to safeguard human lives.

Keywords

Radioactive Hot-spots, Machine Learning, Artificial Neural Networks, Divide and Conquer

Contents

List of Figures

List of Tables

Acronyms

ANN	Artificial Neural Network
CBRN	Chemical, Biological, Radiological and Nuclear
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DAC	Divide and Conquer
GMC	Geiger-Müller Counter
GPS	Global Positioning System
GPU	Graphics Process Unit
IPFN	Instituto de Plasmas e Fusão Nuclear
MLP	Multilayer Perceptron
NORM	Naturally Occurring Radioactive Material
PET	Positron Emission Tomography
SIPRI	Stockholm International Peace Research Institute
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
USB	Universal Serial Bus

1

Introduction

Contents

1.1 Motivation

Radioactive decay is the natural physical process by which an unstable atomic nucleus decays into another element(s), losing energy by radiation through particles emission, such as alpha, beta and gamma particles [13]. Radioactivity was discovered in 1896, by the french scientist Henri Becquerel, and from there on it has been used for applications in different fields: medicine (also called nuclear medicine), with an extensive use in diagnosis like Positron Emission Tomography (PET) scanning and therapy, being very useful in the identification of malignant tumours, in the study of the anatomic structure of organs and even in the cancer treatments; industry, with focus on power generation based on the release of the fission energy of uranium; and science, with important benefits regarding the chronological sequence of geologic past events [14].

However, there are dangers associated to the ionizing radiation that involves many problems for the human beings and the environment. These threats are related to Chemical, Biological, Radiological and Nuclear (CBRN) materials or weapons, that are used with the intention of causing significant harm and disruption in a certain target [15], although, in some cases, these threats can also be released accidentally. The chemical threats correspond to the poisoning or injury caused by chemical substances and, in the case of biological threats, to illnesses caused by the deliberate release of dangerous bacteria or viruses or by biological toxins. These dangerous materials can be acquired relatively easily in nowadays by agents with bad intents, leading to a complex challenges to security organizations. Radiological threats are related with the exposure to harmful radioactive materials that leads to illnesses, while the nuclear ones are associated to life-threatening health effects caused by exposure to harmful radiation, thermal or blast effects arising from a nuclear detonation [16]. According to Stockholm International Peace Research Institute (Stockholm International Peace Research Institute (SIPRI)) publication in [17], these threats related to CBRN are growing rapidly alongside changes and developments with regard to politics and technology. For instance, with the continued use of chemical weapons as well as toxic chemical and radioactive materials in recent armed conflicts suggest new improvements and agreements to be incorporated into national security policy. As such, it is really important to explore CBRN detection technologies, taking into account that is necessary to be in a security context. There is an example of a protection against CBRN threats in Turkey developed by Indra [18], using a commercial van as a platform, that in case of accident where the habitability would be affected, it has a mask system coupled to the filtration equipment allowing the operators to breathe clean air. The system presents graphically every operations and every relevant CBRN information, giving at same time all this information to the various operators present in the area (such as the police force, the fire brigade and health authorities). Also, the operators have a measuring and Gamma-ray spectroscopy system available, which can take samples of suspended particles to detect nuclear and radiological materials.

Besides that, we know that Radiation exists everywhere in our daily life, since it arises from Naturally

Occurring Radioactive Materials (NORMs), that are found in air, soil and building materials [19]. Besides the existence of natural radioactive places, the concentration of NORM radionuclides can significantly increase due to human activities and technological processes such as fossil fuel burning, mineral extraction, water treatment and purification, generation of geothermal energy and fertilizers application. The type and quantity of radioactive materials vary considerably from one industrial process to another, and the level of individual exposure from NORM is usually trivial. However, in some cases NORM can be dangerous and classified as radioactive waste, so that both public and industrial workers can be exposed to ionizing radiation at certain levels that require surveillance and regulation [20]. There may be also unexpected scenarios resulting from possible accidents (e.g. Chernobyl disaster in 1986), or somewhere with dangerous radioactive material being hidden for terrorism attacks, which can be associated with the so-called dirty bombs [21].

Therefore, the anomalous radiation source detection task is to localize and quantify the radioactive hot-spots in a certain area, so that humans and the environment can be mitigated from all those possible dangerous scenarios related to CBRN threats and NORMs, as well as facilitating the search of radioactive sources that are useful for applications in medicine, industry and science. Since the radioactive intensity prediction of entire scenarios cannot be made by measuring every point directly, some technological solution is needed to solve this problem, so it can provide a quick and accurate remote inspection of the scenario. Moreover, there may be scenarios where multiple sources are installed in different places, like in coordinated terrorism attacks, for instance. In such scenarios, we need a solution also able to compute the number of sources, besides the strength and location of each one. To perform an inspection in such radioactive scenarios, portable devices, such as radiological sensors, can be handled, in a backpack, or even in mobile robots, such as Unmanned Ground Vehicle (UGV) and Unmanned Aerial Vehicle (UAV). For example, in [22, 23] the UAV helps in detecting radioactive material through search strategy methods within a specific area. In [24], the UAV is used to build a 2D radioactive heatmap, where a Global Positioning System (GPS) receiver is used to measure different positions, along with a Geiger-Müller Counter (GMC) to quantify the corresponding levels of radiation. It is assumed that the whole scenario is plain, i.e. with a constant distance to the ground, and the mapping process is made through a data interpolation process. In other works, such as in [25], a rangefinder sensor is also used (with the GPS receiver) to make altitude corrections with the radioactivity radiation propagation model and, therefore, the acquired data from the rangefinder and GPS are combined in order to perform a triangulation between the points and thus obtain an elevation map.

Earlier detection systems are designed for a single source, or separated distributed sources [26, 27], where the radiation field could be described by several unimodal distributions and the cumulative effect of multiple sources could be neglected. Whereas, more recently, many approaches have also been adopted to locate multiple sources, taking into account the cumulative effect in statistic models

[28–31]. Most of them assume that the number of sources N is known *a priori*, being in this case a parameter estimation problem, where the goal is to find the most probable source parameters (location and strength) of each source. Thus the number of estimated parameters increases with N, and then the algorithm complexity also increases. On the other hand, in works where the number of sources N is unknown, the parameters of the sources are estimated for a range of possible values for N, using in the end the ones with highest probability. This can lead to inaccurate predictions, since a large range of possible N values and a superposition of signal strengths from different sources are examples of factors that make existing algorithms more complicated, and even more when these factors are jointly considered. For instance, if we consider the problem of non-point-like sources, a sensor can record a high measurement induced by a single strong source or by a combined multiple weak sources. This means that we may get similar results that seem to be valid for distinct values of N.

Another possible scenario is when obstacles are present, and so creating an accurate algorithm can be infeasible when the shapes, materials, sizes and positions of all the obstacles must be taken into account *a priori*. Besides the fact that these data are not easy to obtain, to incorporate them into an accurate model is a challenge that has not yet been overcome in the literature found.

This is where another emerging tool comes in: Machine Learning, and more precisely the Deep Learning sub-field, which can be explored in order to try to solve some of the previous challenges, although in recent years approaches related to the task of detecting radioactive sources have not been found in the literature by the author of this thesis (which can also be seen by the works [32–34]). Due to the recent improvements in computational hardware and software, Machine Learning has shown great success in handling a broad set of complex problems, with many applications in Artificial Intelligence, such as playing video games, self-driving vehicles, lethal autonomous weapons systems and controlling robots [35]. Hence, also in the nuclear world, namely the radioactive hot-spot detection, the application of advanced Machine Learning techniques, such as Deep Learning, has the potential to make a difference, by providing fast estimations for better informed decisions, as well as ensuring reliability and reproducibility of the models. So, demonstrating this is the final goal of this thesis.

1.2 State of the Art Algorithms

For the detection of hot-spots or signal sources, various formulations of related problems have been studied extensively. The detection of radioactive sources [1, 27–31, 36–48], brain electric sources [49–51], lithography hot-spots [52] and cancer indicator hot-spots [53] are just some examples. In order to see the existing state-of-the-art in the field of radioactive hot-spots detection, follows a subsection about the major approaches that are commonly used to estimate the location and intensity of such hot-spots, from radiation measurements. Then, another subsection is presented about the use of Machine

Learning techniques in problems of hot-spot detection in general, not only in the radiological field, but also in medicine and science.

1.2.1 Radioactive Hot-Spot Detection Approaches

As mentioned before, for radiation sources, there are formulations of localization of both single and multiple sources scenarios. Most of the first developed algorithms were concentrated on solving the classical problem of single source localization (with known source activity), where maximum likelihood estimation (MLE) [30, 36] and least square error [37, 38] are the typical methods used to search for the most likely source parameters, by fitting the sensor measurements. In [27, 39], the authors adapt a time difference of arrival (TDoA) algorithm from [40], to log-space, in order to exploit the logarithmic differences in source intensity measurements from three sensors to figure out the source position. Although these single source localization models provide good results, they are not yet able to be applied in scenarios with multiple sources.

Regarding the localization of multiple radiation sources, most the algorithms estimate the number of sources N and then the corresponding parameters. For instance, in [28], the authors start by using Gaussian mixture model based selection to compute N, and then estimate the source parameters through the MLE method. In this approach, the accuracy of the model degrades and the runtime explodes with increasing N. Similarly, in [41], the signal sources (targets) are modeled with Gaussian mixture models, followed by Akaike's or Bayesian Information Criteria to compute N. Then, a simple expectation maximization (EM) and clustering are used to estimate the parameters of the N sources. These MLE and EM based algorithms are not able to compute more than four sources (as reported in [28]), since the estimation of source parameters is computationally expensive when there are many parameters, and each additional source increases the total number of parameters by three. Beyond MLE and least square error approaches, in [37], Duarte has also considered the robust localization with clustering technique [42], since they could be adapted to the collimator model.

In [43], it is performed the gradient descent optimization with respect to a class of non-convex cost functions, assuming that the sources are located in the convex hull formed by the sensor positions. In [44, 45], the authors propose a technique where the sources are located in a grid of sensors under the region of interest, and then the source parameters are computed through convex optimization. That is, there is a discretization of the search space and then the algorithm localizes the sources in the discretized locations. This grid-based method allows to determine the number of sources N and respective characteristics simultaneously and efficiently, not having any restriction about the maximum number of sources (although the runtime increases with N and may also explode). However, the accuracy of the model is limited by the resolution of the grid and the number of sensors.

The localization of radioactive sources within environments with obstacles has also been considered.

In the work described in [46], the author assumes that the placement of obstacles is known. This algorithm makes a discretization of the search space (whereas, in our case, the scenario may not be known) and finds the probability of a source being located inside each cell, which can also be easily calculated with obstacles, once their location is known. In [47], the author proposes hybrid formulation of particle filter and mean-shift techniques to efficiently localize point-like sources, and can scale to moderately large regions. With no previous information about obstacles, the algorithm is tested in realistic scenarios with obstacles, in order to evaluate how they may affect its performance. It is concluded that the obstacles may improve the accuracy for some sources and degrade for other ones, depending on the placements of the obstacles relative to the sources and sensors, because the shielding by the obstacles may reduce the interference between two sources. Limitations of this method are associated with the need for a densely populated sensor network.

1.2.2 Machine Learning in Hot-Spot Detection

Machine Learning is a research field that recently has gained a lot of interest in many different areas, such as in image and speech recognition, natural language processing, spam detection and filtering, and advertising [54]. The nuclear world is clearly one of the main fields, and so is the Radiological, where several applications have been developed. Also, we can say that the new Machine Learning research has the potential to disrupt the nuclear field [35]. This idea can be supported by several reasons: the reduction of computing time through recent improvements in both software and hardware, the huge amount of current available open source frameworks, and also the fact that, although there is still not much nuclear data, according with the World Institute for Nuclear Security [55], it is intended that, in the near future, there will be large amounts of nuclear data.

There are three basic Machine Learning paradigms: Supervised Learning, Unsupervised Learning and Reinforcement Learning. The latter is about the training of models where software agents ought to take actions in an environment so as to maximize some notion of cumulative reward, becoming capable of making a sequence of decisions. Due to its generality, there are many applications based on this Machine Learning technique, such as game theory, operations research and simulation-based systems [56]. Regarding the problem of radiation source localization, it can be used to find the path that takes us to the source, with no human intervention. For instance, in [57], a convolutional neural network-based double Q-learning algorithm is built and tested, and it can reliably navigate the detector and reduce the searching time by at least 44% compared with traditional uniform and gradient search methods.

Unsupervised Learning techniques identify commonalities in a set of data and react based on the presence or absence of such commonalities in each new input data. The main method used in this type of algorithm is Clustering, which groups the data that has not been labelled, classified or categorized. It is present in works already mentioned, such as [37, 41, 42].

Focusing on Supervised Learning, this is the Machine Learning task of learning a function capable of predicting outputs from new inputs based on example input-output pairs (called training data set). For instance, Support Vector Machines (SVM) are supervised learning models associated to algorithms in order to do a classification and regression analysis regarding a certain data set, which belongs to two different classes [58]. After using a training data set to build the model, a non-probabilistic binary linear classifier is obtained, where the data points are mapped in space with a clear gap so the two classes are clearly identified. Thus, at the end it is possible to assign new data to the correspondent class, based on its side of the gap. The works already developed in the radiological area involve several approaches that divide the area, through a kind of heatmap with different radioactive levels. In [48], the author uses multi-scale support vector regression (SVR) with a Gaussian radial function as kernel function, getting then a heatmap that provides the location of the hot-spots present (namely Cs^{137}), in the region of Briansk following the Chernobyl nuclear power plant accident of April 1986.

A sub-field of Supervised Learning is Deep Learning, which is based on Artificial Neural Networks (ANNs). They are inspired by the information processing and distributed communication nodes in biological systems, that enables a machine to learn from observational data. The use of ANNs in problems of hot-spot detection (not radioactive) has received much attention, mainly in the recent years, in areas like medicine and science. In [52] a deep learning technique for lithography hot-spots detection is used in order to do the control quality of semiconductor manufacturing. In [53] we have an example of a deep learning approach to recognize candidate hot-spots, and subsequent proliferation rate scoring, by quantifying Ki-67 appearance in breast cancer immunohistochemical images. Finally, in [49–51] we have ANNs approaches for source localization in the human brain, with multilayer perceptron (MLP) techniques. These approaches could be easily applied in the problem of radioactive source localization if the goal was to localize only one source as well.

According to [33], ANNs are the dominant learning-based algorithm used in nuclear and radiological science, mainly in applications of reactor health and monitoring, radiation protection (characterization and identification of radionuclides), and optimization. However, regarding the concrete problem of radioactive hot-spot detection, the author of this thesis have not found approaches in the literature of the last decade, which can be confirmed by the reviews [32] and [34]. This may be related to the fact that, as we will see later, to build an ANN model, a training process has to be performed and some conditions must be fixed, such as the conditions of the scenario (e.g. its dimensions and the strength of the sources) and the way the observations are acquired (sensors distribution or the path taken by the UAV). Otherwise, new training processes must be performed and the time necessary for this may be a barrier to the development of such ANN-based approaches. Besides that, the computational costs in relation to both software and hardware to train the model, and the need to get a huge and varied training data set, can also be some of the constraints to obtain accurate results. Then, only if we go back almost two

decades ago, more precisely in 1995, the author of [1] uses an ANN optimization method for a single radioactive source localization, in a two-dimensional environment (treated as a box with cells) with known properties and dimensions. The algorithm is based on the recurrent Hopfield network with fixed weights, and then the output signals of the neurons indicate the spatial cells addresses of the unknown radioactive sources. Although this model shows accurate and quick solutions in large-scale scenarios, there are some limitations associated: the number of sources that can be detected (only one), the total intensity of the radioactive material present in the box is known *a priori*, and the results are still dependent on the noise and on the number of detectors.

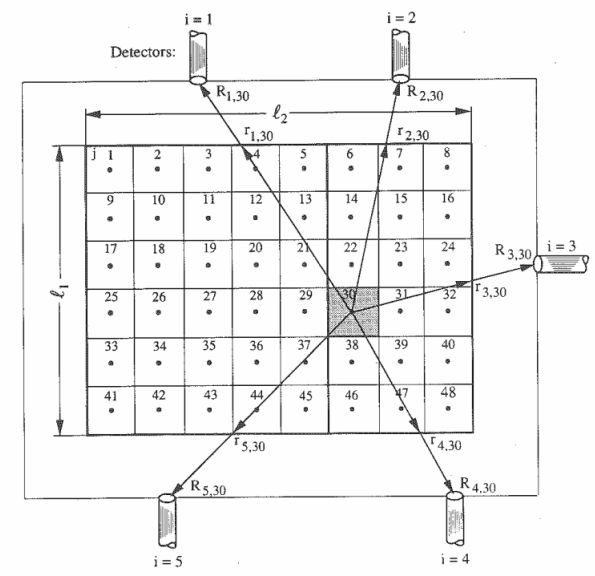


Figure 1.1: Schematics of radiation medium and detectors layout in [1].

1.3 Objectives

The main goal of this thesis is to find an approach, from a Machine Learning perspective, to solve the problem of radioactive hot-spot detection (which includes its localization and quantification of the radiation level), by using an UAV with a GMC and a GPS receiver to get observations and then compute the number, location and activity of the source(s) present in the area of interest. Therefore, there are some objectives so this goal can be achieved:

1. Models formulation: by defining the objective functions to be optimized, the sources model, the setting of the scenarios and the sensors to be used, and by selecting the best Machine Learning approaches capable of solving the problem. The models are supposed to be applied in scenarios like the one that followed the Chernobyl accident, where it is extremely important to measure the

radioactivity throughout the area involved and even to monitor its evolution over time. Nevertheless, depending on the detector capacity, this approach can be considered not only in these kind of scenarios with high levels of radioactivity, but also the ones that, although with less levels of radioactivity, still may have a certain importance, such as old ore mines, NORM-related places, landfills and heaps.

2. Data acquisition (real and simulated): the measurements with different positions and levels of radiation are made by sensors like GMC and GPS receivers. A simulator is also used to build a training data set, as well as to test the algorithm with complex simulated scenarios.

3. Algorithm application: by using the selected machine learning techniques, and by developing the models in such a way that it is possible to detect radioactive hot-spots in complex and realistic scenarios. That is, an algorithm capable of handling multiple sources (with low limitations about the maximum number) and not requiring previous information regarding the number of sources, as well as detailed specifications about obstacles that may be present in these scenarios.

4. Performance analysis: by quantifying the quality of the results, through a certain validation criteria, and by analysing and comparing them to existing solutions. Finally, it will be possible to verify if Machine Learning can really provide better models, so they can be applied not only in this kind of radiological problem but also in other related problems, like in chemical or biological fields.

1.3.1 Contributions

The main contributions of this work are as follows:

- A novel approach for detecting radioactive hot-spots, that is efficient in (i) handling multiple sources, without requiring any information about the number of sources; (ii) predicting the parameters of the detected sources (location and activity); (iii) considering scenarios with obstacles included, without any information given in advance and without training data composed of such scenario conditions; (iv) handling regions of the scenario without any observation and still get accurate predictions; (v) approaching non-point-like sources with interesting results; and (vi) handling both large and short-scale scenarios, through the application of an algorithm based on the Divide and Conquer method.

- Quantitative results are provided and show that the selected approach is capable of getting accurate predictions, with a low number of false negatives (missed detections) and false positives (false alarms) rates, as well as precise estimations regarding the exact location and activity of the detected sources.

- Finally, a paper entitled ”Radioactive Hot-spot Localization and Identification using Deep Learning” (in press) is about to be published in the journal Sensors, in section ”Physical Sensors” with the special issue ”Remote Navigation and Guidance for CBRNe Defense (Chemical, Biological, Radiological, Nuclear and Explosive Materials)”. This work aims to explore the capacities of ANNs in both tasks of localization and identification of radioactive sources. For this, it is done a combination with the main results obtained in this thesis and in the MSc Thesis of Filipe Mendes entitled ”Real-time radionuclides detection using artificial intelligence”, which aims to detect and identify in real-time the radionuclides contained in gamma-ray spectrometric data acquired during expeditions in outdoor scenarios, by using ANNs as well.

1.4 Thesis outline

The remainder of this thesis is organized as follows:

- Chapter 2 presents the problem statement and the assumptions made to show the position that is intended to argue within this work, followed by a description of the selected approach to solving the problem and the background needed to understand it. And finally, it is provided the quality metrics used to evaluate the different models, as well to assess the simulated and real test scenarios.
- Chapter 3 describes the setup and data assumptions for this thesis, with respect to the sensor modelling, the simulator used to build the training data set, as well as to acquire testing data of different scenarios, and the way the real scenarios are approached.
- Chapter 4 starts by detailing how the selected approach is implemented, taking into account how the acquired data is processed and how the algorithm can be modelled. Then, there is a description of all the training process, and the results associated, which involves the comparison among different training conditions. To conclude, different tests are made with the test data set, so the best models can be selected.
- Chapter 5 provides the performance results of the proposed solution with real and simulated scenarios, where the number of sources, the presence of obstacles and the shape of the source are tested.
- Chapter 6 concludes the work by detailing what was accomplished and what can still be explored based on this thesis.

2

Proposed Solution

Contents

2.1 Problem Statement

It is assumed an unknown number of radioactive sources, $N \geq 0$, with no information regarding the respective activities and locations, within an area of interest treated as a two-dimensional environment (since it is assumed that the distance to the ground is constant). Let $\mathcal{S} = [\boldsymbol{s}_1^T \dots \boldsymbol{s}_N^T]^T$ denote the set of radiations sources. Each source is considered as isotropic point-like source and located on the ground, being parameterized by the vector parameter $\boldsymbol{s}_i = [\boldsymbol{p}_{s_i} a_{s_i}]^T \in \mathrm{R}^2 \times \mathrm{R}^+$, for $i = 1, ..., N$. The position of the i^{th} source is given in Cartesian coordinates, with $\boldsymbol{p}_{s_i} = (x_{s_i}, y_{s_i})$ in meters (m), and the activity, a_{s_i}, is measured in units of becquerel (Bq), which means the number of disintegrations per second. Also, the activity is assumed to be stable, i.e., the duration of any measurement is negligible when compared to the half-life of the source.

Then, the radiation intensity measurements are assumed as independent random variables each following a Poisson distribution [59], with an equal exposure time. Having a known number of measurements, M, let $\mathcal{O} = [\boldsymbol{o}_1^T \dots \boldsymbol{o}_M^T]^T$ denote the set of observations, with $\boldsymbol{o}_j = [\boldsymbol{q}_{o_j} c_{o_j}]^T \in \mathrm{R}^2 \times \mathrm{R}_0^+$, for $j = 1, ..., M$, where $M >> N$. Similarly, $q_{o_j} = (x_{o_j}, y_{o_j})$ represents the j^{th} measurement position, and its intensity (number of particles detected within the exposure time interval), measured in units of becquerel per squared meter (Bq/m^2), is given by

$$c_{o_j} = c_b + \sum_{i=1}^{N} \frac{a_{s_i} \exp(-\alpha d_{o_j, s_i})}{d_{o_j, s_i}^2} \tag{2.1}$$

where both the coefficient α, which is related to the absorption of radiation particles in the air, and the background intensity, c_b, are known *a priori*, and

$$d_{o_j, s_i} = \sqrt{(x_{o_j} - x_{s_i})^2 + (y_{o_j} - y_{s_i})^2} \tag{2.2}$$

is the Euclidean distance between positions $\boldsymbol{q}_{o_j}$ and $\boldsymbol{p}_{s_i}$.

The mentioned background intensity, c_b, is universally present as consequence of cosmic rays and the decays of naturally occurring radio-isotopes like Carbon-14 and Potassium (the so-called NORMs, detailed in section 1.1, when it is not anomalous radiation) [60].

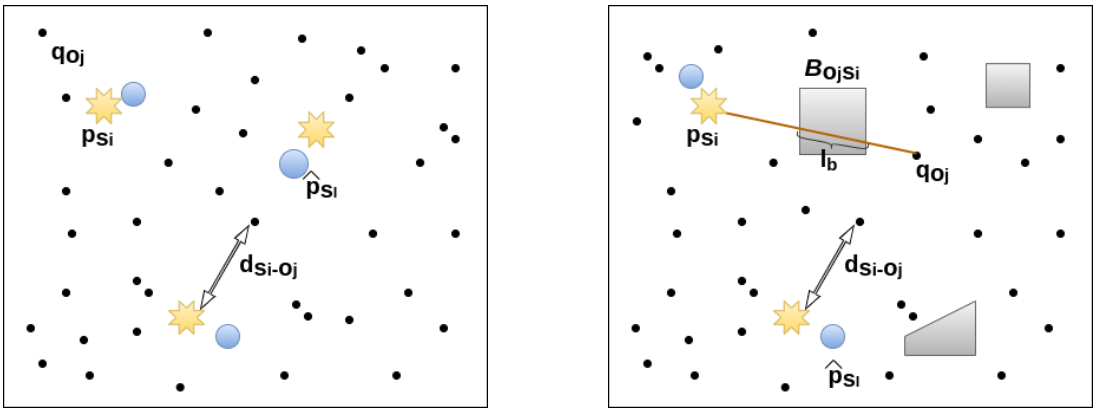
When obstacles are considered, some of the previous expressions must get adapted, since they shield (totally or partially) the radiation rays. Also, the obstacles have different types of materials and shape, and can be located anywhere in the area of interest. Although these characteristics are not implemented in the algorithm, it is known that the material determines the material effectiveness in absorbing the radiation, where materials of higher atomic number, as well as higher density, are capable of absorbing gamma radiation in a more effective way. Besides that, the higher thickness of the material, the higher amount of gamma rays absorbed in its shielding. Then, assuming that each obstacle is

made of homogeneous material, and the radiation source is on one side of the obstacle, the intensity that comes from the i^{th} source, a_{s_i}, becomes $a_{s_i}e^{-\mu l}$, where l is the thickness and μ the attenuation coefficient for the material of the obstacle. Finally, equation (2.1) becomes

$$c_{o_j} = c_b + \sum_{i=1}^{N} \frac{a_{s_i} \exp\left(-\alpha d_{o_j,s_i} - \sum_{b \in \mathcal{B}_{o_j,s_i}} \mu_b l_b\right)}{d^2_{o_j,s_i}} \tag{2.3}$$

where $\mathcal{B}_{o_j,s_i}$ denotes the set of obstacles that intersect with the straight line between $\boldsymbol{q}_{o_j}$ and $\boldsymbol{p}_{s_i}$. Consequently, regarding each obstacle b, l_b is the total thickness along that intersection, and μ_b is the corresponding attenuation coefficient. It is important to note that the information about the presence of obstacles and their properties are not known *a priori*, in such a way that there is no scenario with obstacles included in the training data set.

Thus, the localization problem is to use the vector of measurements, $\mathcal{O}$, in order to make predictions regarding the number of sources, $\hat{N}$, and respective parameters vector, $\hat{\mathcal{S}} = [\hat{\boldsymbol{s}}_1^T \dots \hat{\boldsymbol{s}}_{\hat{N}}^T]^T$, with $\hat{\boldsymbol{s}}_k = [\hat{\boldsymbol{p}}_{\hat{s}_k} \hat{a}_{\hat{s}_k}]^T \in \mathrm{R}^2 \times \mathrm{R}^+$, for $k = 1, ..., \hat{N}$.



Scenario without obstacles.

Scenario with obstacles (in gray color) included.

Figure 2.1: Illustration of Problem Statement.

2.2 Selected Approach

The proposed approach to detect radioactive hot-spots is the use of Machine Learning techniques, namely Artificial Neural Networks (ANNs), and related architectures such as Convolutional Neural Networks (CNNs), essentially due to its ability in dealing with radiological data, which is non-linear, noisy and inconsistent. Besides, the training of such deep architectures has become even more efficient with improvements and breakthroughs in techniques related to training, optimization and regularization [61]. This was possible due to recent improvements in hardware, such as the development of Graphics Process Units (GPUs) [62], and software, with the expansion of open source frameworks, like Tensorflow [63, 64], Theano [65] and Pytorch [66].

2.2.1 Artificial Neural Networks

Artificial Neural Networks, also called just Neural Networks, are inspired by the information processing and distributed communication nodes in biological systems that enables the machine to learn by itself, by adapting the algorithm according to the observational data through the training process. They work by introducing data (which can be real or simulated) to the network via the 'input layer'. Then, the information goes through one or more 'hidden layers', where it is processed via a system of weighted connections. And finally, at the end, there is the 'output layer', which provides the estimations given by the network system.

The most popular ANN architecture is Multilayer Perceptron (MLP), represented in Fig. 2.2, where the layers are composed by units, called artificial neurons, and each one calculates a weighted sum over its N inputs plus a bias, giving the respective output

$$z = \phi\left(\sum_{i=1}^{\mathrm{N}} w_i.x_i + w_0\right) \tag{2.4}$$

being x_i the input values, w_i the weights (and w_0 the bias). Then, ϕ represents the activation function [67], which is a non-linear function and so helps the network to learn complex data and its non-linearities. Otherwise, if a linear function was used instead, consecutive neurons would apply consecutive linear transformations and, in the end, the network would be equivalent to a single linear transformation. When every layer's neuron is connected to all the neurons of the next layer, we have the so-called fully connected (or dense) layers.

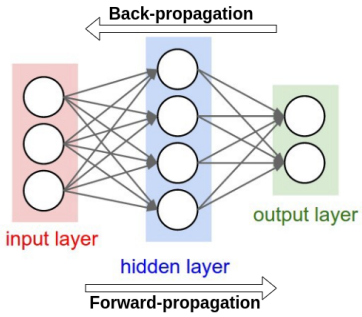


Figure 2.2: Representation of an MLP with one hidden layer. Image from [2].

The weights and the bias are free parameters that, during the training process, the ANNs try to find the best values to optimize the model's performance. The most common optimizer used is called gradient descent algorithm, that computes the function's gradient, ∇F, in a given position and updates that position in the direction of the negative gradient of that function:

$$x^{n+1} = x^n - \eta \nabla F(x^n) \tag{2.5}$$

where x and and x^{n+1} are the previous and the updated position, respectively, while η is the learning rate parameter that controls the size of the steps over each iteration (also called epoch), n. This is a very important parameter that must be as optimal as possible, so the algorithm does not converge (if η is too small) or diverge (if η is too big) either. For ANNs training, the process used based on the gradient descent algorithm is called back-propagation [68], where the network updates the weights (as represented in Fig. 2.3) after doing the forward propagation to obtain the estimations [67]. So, eq. (2.5) is adapted as follows:

$$w_i^{n+1} = w_i^n + \Delta w_i^{n+1} \tag{2.6}$$

$$\Delta w_i^{n+1} = -\eta \frac{\partial \mathcal{L}}{\partial w_i^n} \tag{2.7}$$

with w_i^n and w_i^{n+1} being the previous and the updated i^{th} weight, respectively, Δw_i^{n+1} is the step between them, and $\frac{\partial \mathcal{L}}{\partial w_i^n}$ represents the loss function's gradient with respect to weight w_i^n.

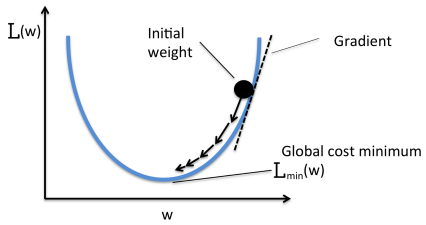


Figure 2.3: Back-propagation process with Mean Square Error loss function used. Image from [3].

The loss (or cost) function, represented in eq. (2.7) by $\mathcal{L}$, is a metric used during the training process that evaluates the error between the current estimations and the training targets. It is by decreasing the loss value that the free parameters are optimized and, therefore, the model is able to learn the data and get better predictions. However, a small loss value may also correspond to a 'overfitting' situation, which means that the model is losing its ability to generalize and then makes worse predictions with new data. This may happen when the algorithm is trying too hard to capture the noise in the training data, where noise refers to the data points that do not represent the true properties of the whole data, but random chance. So, the loss function, $\mathcal{L}$, must be selected according to the desired task, since it commands all the training process and, consequently, allows to obtain predictions closer to reality. In addition to the basic technique of back-propagation for ANNs training, there are other optimizers that have been developed and provide improvements in the quality and speed of the training process, such as the Adam optimizer [69]. The selection of the most appropriate optimizer is also one of the important factors to get better results.

The most common ways to avoid overfitting is through Regularization techniques of Early-stopping, Dropout layers and Lasso (L1) or Ridge (L2) regressions. The first technique means selecting the weights that correspond to the training epoch whose validation error reaches its minimum [4], as repre-

sented in Fig. 2.4. The referred validation error is applied to a data set apart from the training data set - the so-called validation data set, which provides an unbiased evaluation of the model during the training process, and so helps selecting the proper parameters of the model.

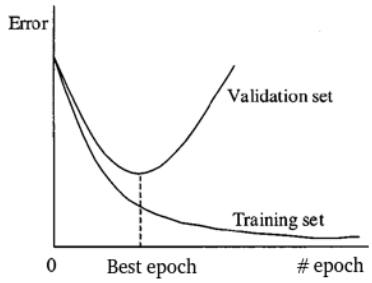


Figure 2.4: Early-stopping regularization technique, where the best epoch corresponds to the early-stopping point. Image from [4].

The use of Dropout layers is a technique that selects a random number of neurons of each layer, with a chosen rate, to be dropped during a training epoch [5]. In Fig. 2.5 it is represented the common use of the Dropout layer technique applied to all the layers, except the output layer. To note that, when running the model on the validation data set, all neurons will be used (i.e. rate becomes 0) since the model provides better performance when all the neurons that were trained are used.

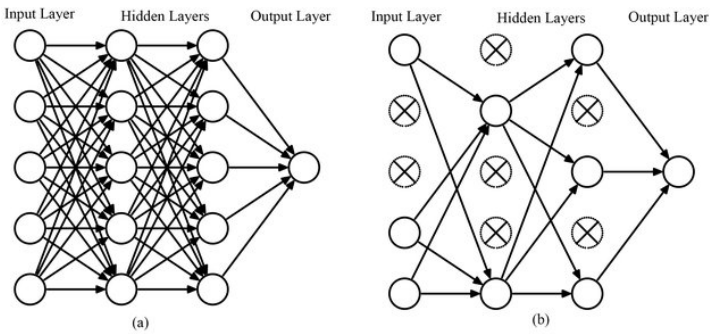


Figure 2.5: Representation of an MLP before (a) and after (b) applying the Dropout layers technique [5].

Lastly, L1 is a technique that adds the absolute value of magnitude of weight as penalty term to the loss function (equation (2.8)), while L2 adds the squared magnitude of weight (equation (2.9)) [70].

$$\mathcal{L}_1 = \mathcal{L} + \lambda \sum_i |w_i| \tag{2.8}$$

$$\mathcal{L}_2 = \mathcal{L} + \lambda \sum_i w_i^2 \tag{2.9}$$

Naturally, if $\lambda = 0$ we get back $\mathcal{L}$. On the other hand, if λ is too large, then we would have underfitting, where the model does not learn enough from the training data, which means that it is important how λ

is chosen.

Thus, we have the so-called hyperparameters, i.e., the variables that determine the network structure (such as the number of hidden layers and neurons, the activation function and the weights initialization) and the variables that determine how the network is trained (such as the learning rate and the number of epochs). These hyperparameters are then external parameters chosen by the model operator, and they have a huge impact on the accuracy of the network, besides that there may be different optimal values for different hyperparameters, which means that it is clearly non-trivial the choice of those values.

2.2.2 Convolutional Neural Networks

As the problems become more demanding and complex, such as dealing with images as inputs, the ANN model also gets deeper, with more layers and neurons, which can cause some trouble related to a huge amount of parameters. One solution is the use of Convolutional Neural Networks (CNNs), a class of ANN that is commonly applied to tasks like image recognition and object detection [6, 71, 72]. Such a architecture receives a matrix of numbers as input that, when the observation is an image, those numbers are simply the pixel values. This way, the machine is able to learn the patterns like vertical and horizontal edges, round shapes and even other patterns undetectable by the human eye.

The CNNs are composed by convolutional layers in which, unlike what occurs in MLPs, the output of each neuron is result of a convolution of a (normally small) subset of neurons along different position from the previous layer, followed by an activation (non-linear) function. It is through the convolution process that the machine learns the patterns, since, as represented in Fig. 2.6, a convolution is a specialized type of linear operation used for feature extraction that, given an input matrix and a weight matrix (called kernel or filter), returns a new matrix as output, labeled as feature map. This process helps reducing the number of free parameters where only the kernel's features are learnt and optimized, which are applied in multiple subsets of the image input.

The features corresponding to each kernel are learnt by the network through the training process. Furthermore, there are several parameters that influence the output dimensions: the stride (number of neurons shifts over the input matrix), the zero-padding (when the filter does not fit perfectly the input and zeros can be added so that it fits), the kernel size (related to the width and height dimensions) and the depth (number of different filters used, which is equivalent to the number of feature maps generated).

Another important property of the CNNs is the parameter sharing. The weight values within kernels are shared across all kernels used across the same input. So, if a given feature of the kernel is relevant for one position in the image input, then the same feature will be relevant for another position. Therefore, a kernel used in one part of the input data keeps the same across other regions of this input, where all the regions have contributed to the learning of the features.

Typically, the output of the last convolutional layer is flattened, i.e., the output matrix is transformed

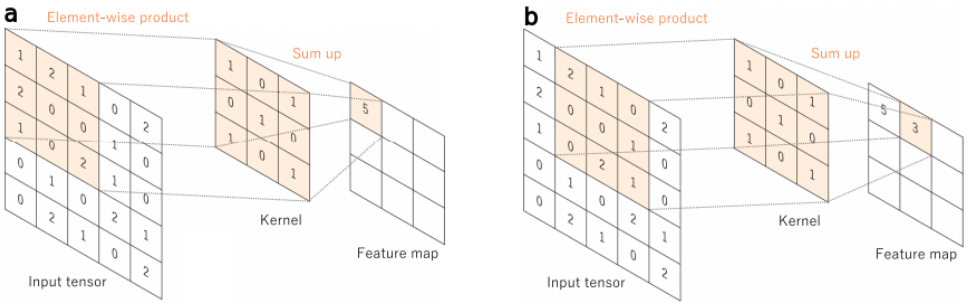


Figure 2.6: An example of convolution operation, from [6], with a kernel size of 3×3, no padding, and a stride of 1.

into a one-dimensional vector, that will be then used as input of a set of fully connected (FC) layers that composes an MLP architecture. Thus, the FC layers take the final outputs of the convolution process and use them for decision-making.

Considering the problem statement described in section 2.1, since there is a set of observations throughout the area of interest, the use of CNNs allows adopting these observations as an image, by transforming them into a matrix of intensities instead of pixels. So, through the CNN kernels, it is possible to get the most important features between neighbouring observations. Also, unlike usual pictures where pixels may not have any direct relation with other pixels, in this case all the intensities from the whole map are related to each other. Then, the use of FC layers allows setting a relationship between all those features obtained from CNNs, and, in the end, better performance of the model can be achieved.

2.2.3 Classification and Regression-based model

The way the neural network is modelled depends on its purpose. The number of hidden layers, the number of neurons in each layer, the parameters initialization and the activation and loss functions are some of the many configurable properties, defined by the operator, that allow the neural network to adapt to different possibilities. Moreover, there exist two main types of problems: classification and regression problems. The first one is related to the problem of predicting a discrete class label output for a given observation. For instance, a given animal that can be classified as 'cat' or 'dog' is a case of a binary classification problem, once there are only two possible output classes: 'cat' and 'dog'. If there was more than two output classes, it would be a multi-class classification problem, and, if the input could be assigned to more than one class, then it was called multi-label classification problem. Furthermore, these models can also predict a continuous value instead, as the probability or confidence of a given observation belonging to each output class. On the other hand, regression problems are associated to the task of predicting a continuous quantity output for a given input. This continuous output variable is a real-value, such as an integer or floating point value. For instance, in a car trip, having the distance and the speed of the trip as inputs, the output could be the amount of money needed for fuel consumption.

In the problem of radioactive hot-spot localization, if only the single-source problem was considered, a regression problem with 3 outputs (the positions x_{s_i} and y_{s_i} and the activity a_{s_i} of the sources $s_i \in \mathcal{S}$ as targets) could be a solution. However, if, for instance, the number of sources is $N = 2$, we would have 2×3 output neurons, and so the ANN would have to learn which one of the outputs is for one source and which one is for the other source. We could say that 'output 1' is for left side of the map and 'output 2' for the right side, but what if both sources are on the same side? The map cannot be simply divided by the number of sources, not only because they can be in any region of the map but also because we do not know how many sources are there.

So, the solution is not to use each set of 3 outputs for each target source, but to divide the map into several well-distributed 'boxes' throughout the map, and to take advantage of the capacity of ANNs to solve classification problems, in such a way that there is one output for each box with the respective probability of having a source in there. Having this, now it is possible to compute the number of sources (assuming for now that each box has at most one source) and the regions (boxes) where they are. Then, to compute the parameters of the sources (activity and location), the regression problem is also applied in the same way it would have been done for the single-source problem. The difference is that this has to be done for each box instead of the entire map and, for that, 3 outputs (a_b, x_b, y_b) are associated for each box b. So, having the map divided into $S \times S$ boxes, this means that we have $S \times S$ outputs for the classification problem and $3 \times S \times S$ outputs for the regression problem.

To join both regression and classification problems in only one model, a solution based on the Yolo (You Only Look Once) state-of-the-art [7] is applied. The Yolo idea is used for multiple objects detection and localization. It starts by dividing the image input into an $S \times S$ grid (or windows), as represented in Fig. 2.7, and then the label for the training data is such that it is implemented both localization and classification algorithm for each grid cell.

Figure 2.7: Division of input image into 7×7 boxes. Image from [7].

If the center of an object is in a given grid cell, then this grid cell is responsible for detecting that object. Each grid cell is able to predict B bounding boxes and confidence scores for those boxes. These confidence scores represent how confident and how accurate the model is that the box contains an object (it should be zero if no object exists in that cell, and one otherwise). Therefore, each bounding

box consists of 5 predictions: x, y, w, h, and confidence. The (x, y) coordinates represent the center of the box relative to the bounds of the grid cell, and the width (w) and height (h) are predictions relative to the whole image. Besides, each grid cell also estimates C conditional class probabilities, represented by $P(\text{Class}_i|\text{Object})$, which represents the probability of having the Class$_i$, knowing that there is an Object in the given grid cell. To note that there is only one set of class probabilities per each grid cell, independently on the number of bounding boxes B. Thus, the predictions can be encoded by an $S \times S \times (5 \times B + C)$ tensor, once there are 5 parameters for each of the B bounding boxes plus C classes, for each of the $S \times S$ grid cells.

So, using the example from Fig. 2.7, the label z for each grid cell can be represented as an eight-dimensional vector:

$$z = \begin{vmatrix} p \\ x_b \\ y_b \\ w_b \\ h_b \\ c_1 \\ c_2 \\ c_3 \end{vmatrix} \tag{2.10}$$

where p defines the probability of an object being present in the grid cell or not, (x_b, y_b, w_b, h_b) the bounding box parameters (if there is an object), and (c_1, c_2, c_3) can be the classes related to dog, bike and car, respectively. In case there is no object (e.g. top-left cell of Fig. 2.7), then we have

$$z = \begin{vmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{vmatrix} \tag{2.11}$$

where '?' means 'Don't care', since it does not matter what $(x_b, y_b, w_b, h_b, c_1, c_2, c_3)$ contains as there is no object in this grid cell. On the other hand, if it is considered the cell where the dog is centered (Yolo algorithm takes only the mid-point of the objects and they are then assigned to the grid cell that contains their mid-point), then the corresponding output is given by

$$z = \begin{vmatrix} 1 \\ x_b \\ y_b \\ w_b \\ h_b \\ 1 \\ 0 \\ 0 \end{vmatrix} \tag{2.12}$$

In Fig. 2.8, it is represented how Yolo algorithm uses the predictions from bounding boxes and confidence with the class probability map to get, in the end, the final detections. Moreover, it is used $S = 7$, $B = 2$ and $C = 20$, although only 5 different classes are shown in the class probability map in this example. These predictions are then encoded as a $7 \times 7 \times (5 \times 2 + 20) = 7 \times 7 \times 30$ tensor. Having $B = 2$ means that the model is able to predict at most two objects (of the same class) in each grid cell.

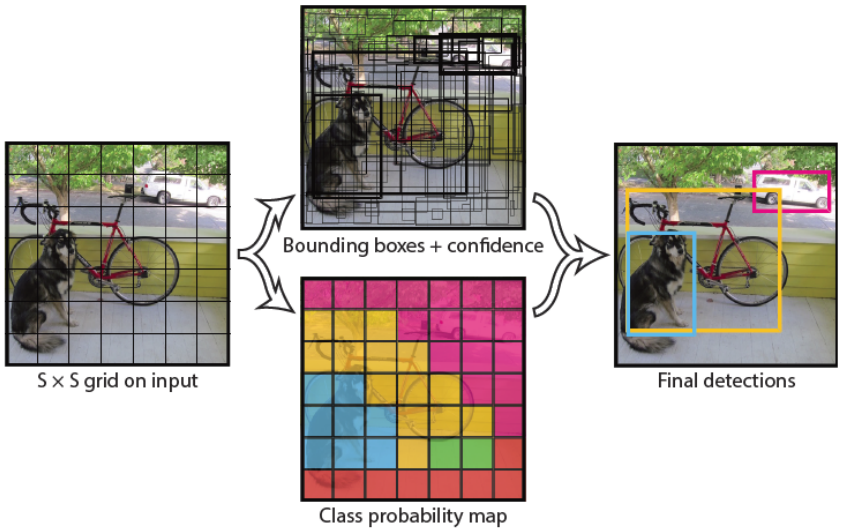


Figure 2.8: Representation of Yolo state-of-the-art [7].

To adapt this Yolo idea on the problem of this thesis, we have to take into account that only one class must be considered: the radioactive source. Therefore, it only matters if there is a source in the grid cell or not. Besides, since it is assumed that the radioactive sources are point-like sources, this means that parameters like width and height make no sense in this context. Thus, the output for each cell is simpler than the Yolo approach and can be represented by:

$$z_b = \begin{vmatrix} p_b \\ x_b \\ y_b \\ a_b \end{vmatrix} \tag{2.13}$$

where p_b is now defined as the probability of having a source in the respective grid cell, b. There are no classes ($C = 0$) because if there is an object, it means automatically that it is a source. If $p_b = 1$, and supposing that it is the i^{th} source the one present in this cell b, then (x_b, y_b) parameters are equivalent to the positions (x_{s_i}, y_{s_i}) and a_b refers to its activity a_{s_i}. Otherwise, if $p_b = 0$, then $(x_b, y_b, a_b) = (?, ?, ?)$ as done in Yolo approach. So, in this case, there are 4 predictions instead of 5 and, dividing the entire scenario into 3×3 cells ($S = 3$), and so $b = 0, ..., 8$, it means that, in total, there are $3 \times 3 = 9$ outputs for classification problem, and $(3 \times 3) \times 3 = 27$ outputs for regression problem. This is, the outputs are encoded by an $S \times S \times (4 \times B + C) = 3 \times 3 \times (4 \times 1 + 0) = 3 \times 3 \times 4$ tensor. Thus, despite a regression

problem is going to be done even in cells without any source, by combining the results of both problems, we can select which outputs of regression problem matter (while the other ones must be neglected) and, in the end, the final estimations are obtained with regressions only on the cells where the classification problem have determined that there is a source, as represented in Fig. 2.9.

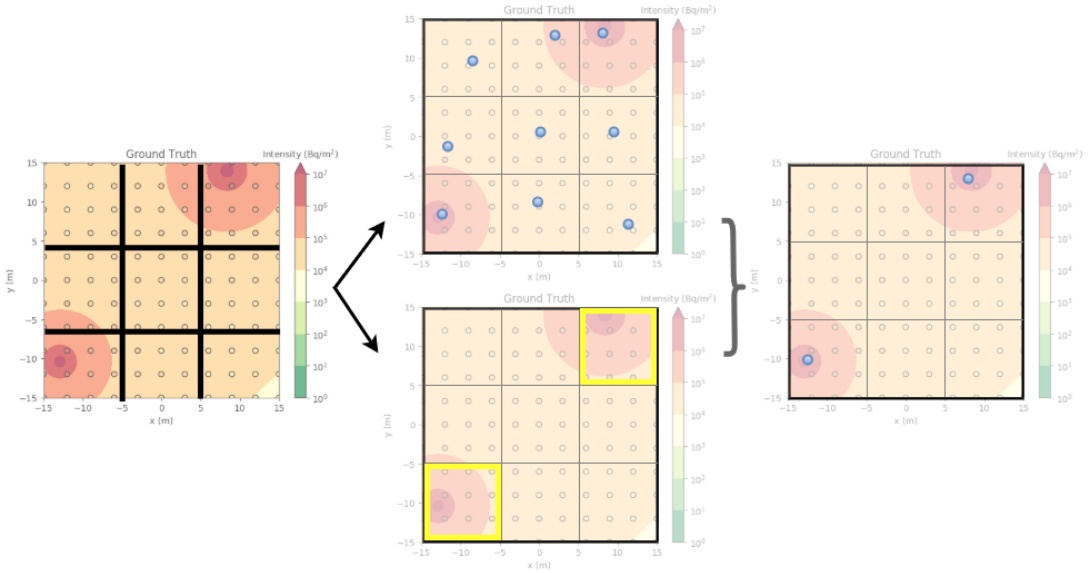


Figure 2.9: Scheme of the classification and regression-based model adopted for radioactive hot-spot detection. On the left, it is represented the division of the map into 3×3 grid cells. In the middle, the regression (top) and the classification (bottom) problems are represented separately. And on the right, a combination of the results from both regression and classification problems is obtained in order to get the final estimations.

Regarding the bounding boxes, we have $B = 1$ to simplify the model, which means that only one source can be detected per each grid cell. This is a computational advantage, where the terms 'grid cell' and (bounding) 'box' have the same meaning in this selected approach, which is represented by the letter b. This is related to the main limitation of Yolo. Its algorithm imposes strong constraints on bounding box predictions, once each grid cell is able to estimate only two boxes and can only have one class, and so the number of nearby objects that can be predicted is very limited. For instance, if an image with flocks of birds is given as input, the model struggles since the objects are small and appear in groups. Besides, since the model learns to predict bounding boxes from data, it also struggles to generalize objects with new or unusual aspect ratios or configurations.

To solve this limitation, in [8] Yolo gets some improvements by implementing the so-called anchor boxes, A. Instead of having bounding boxes and $(5 \times B + C)$ labels for each grid cell, now one has $A \times (5 + C)$ labels. The idea is to make the model capable of detecting distinct (or not) objects in different anchor boxes that belong to the same grid cell, according to its aspect ratio, scale and position, like it is represented in Fig. 2.10.

Although the Yolo approach tries to solve the problem of multiples objects present in the same cell

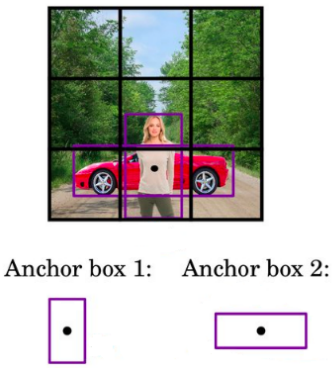


Figure 2.10: Representation of Anchor boxes used in Yolo state-of-the-art [8].

with anchor boxes, this is not a solution when those objects are multiple point-like sources, once the aspect ratio and scale do not make sense in this case, as already explained for height and width parameters. So, some other solution is needed to solve this problem.

2.2.4 Divide And Conquer Algorithm

In order to solve the problem of having more than one source in the same grid cell, a method called Divide and Conquer (DAC) is applied. This is a strategy typically used to solve a problem recursively by applying three steps according to the level of recursion: divide, conquer and combine, as can be seen through the scheme in Fig. 2.11.

In the first step, ”divide”, the problem breaks into smaller subproblems that are similar to the original problem, until the problem is small enough to be solved. In the problem of this thesis, these subproblems are parts/regions of the original map, so the input of the model is just the set of observations that is measured in this region.

Then, the ”conquer” step is when the original problem is already divided into the smallest possible parts and, therefore, the subproblems can now be solved. If these subproblems are small enough, it means that the subproblems are solved as base cases. In the case of radioactive hot-spot detection, the subproblems are considered small enough when we can assume that they can only have one source at most and/or the model has no more capacity to get accurate predictions if the sub-problem dimensions become even smaller.

Finally, in the last step - ”combine” - the solutions of the subproblems are combined in order to solve the whole problem.

Therefore, the goal is to use the pre-trained ANN model through this DAC method. To do so, during the first step of ’divide’, only the outputs related to the classification problem are used, where p_b has a different meaning, referring to the probability of having at least one source ($p_b = 1$) or not having any

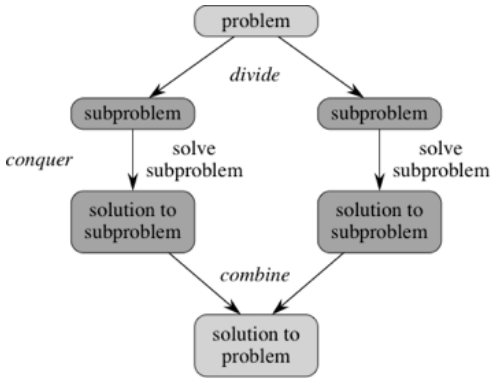


Figure 2.11: Divide and Conquer method scheme. Image adapted from [9].

source ($p_b = 0$) in the corresponding cell b, as follows:

$$p_b = \begin{cases} 0 & \text{if } N_b = 0 \\ 1 & \text{if } N_b \geq 1 \end{cases}$$

with N_b denoting the number of sources regarding cell b.

This way, the subproblems are the parts of the original scenario where the ANN model thinks there is one or more sources. Only on the last division, where it is applied the ”conquer” step, both the classification and regression outputs are used, following the way detailed in the previous subsection 2.2.3. In Fig. 2.12 it is represented one division (called ”Division 1”) where the ”divide” step is applied, whereas the ”conquer” step occurs on the last division (called ”Division 2”).

Finally, the last step, called ”combine”, is when we are able to combine all the predictions of number, location and activity of the found sources, and, in the end, to get the visualization of the results on the whole scenario and compare them with the real values.

For this approach, the number of divisions, i.e. the number of times the ”divide” step is applied, depends on (i) the quantity and quality of the observations that are used as input of the ANN model over the successive divisions; and (ii) the dimensions of the scenario where the ”conquer” step will be applied. The quality of the observations is related to the distribution of their positions, since they must be dense and spatially well-distributed throughout the scenario. For instance, as we can see in Fig. 2.12, where the small points represent the observations, all the $S \times S$ cells ($S = 3$ in this case) have observations with enough quantity and quality to perform two divisions. If the yellow cell of ”Division 2” also had the same number of observations and spatially well-distributed, then it would be possible to make a ”Division 3”. Regarding statement (ii), since the last division is where both classification and regression outputs are used, then the corresponding scenario dimensions must be identical to those of the training data set (this will be explained later in section 3.4). Otherwise, a new training process must be performed with an appropriate training data set.

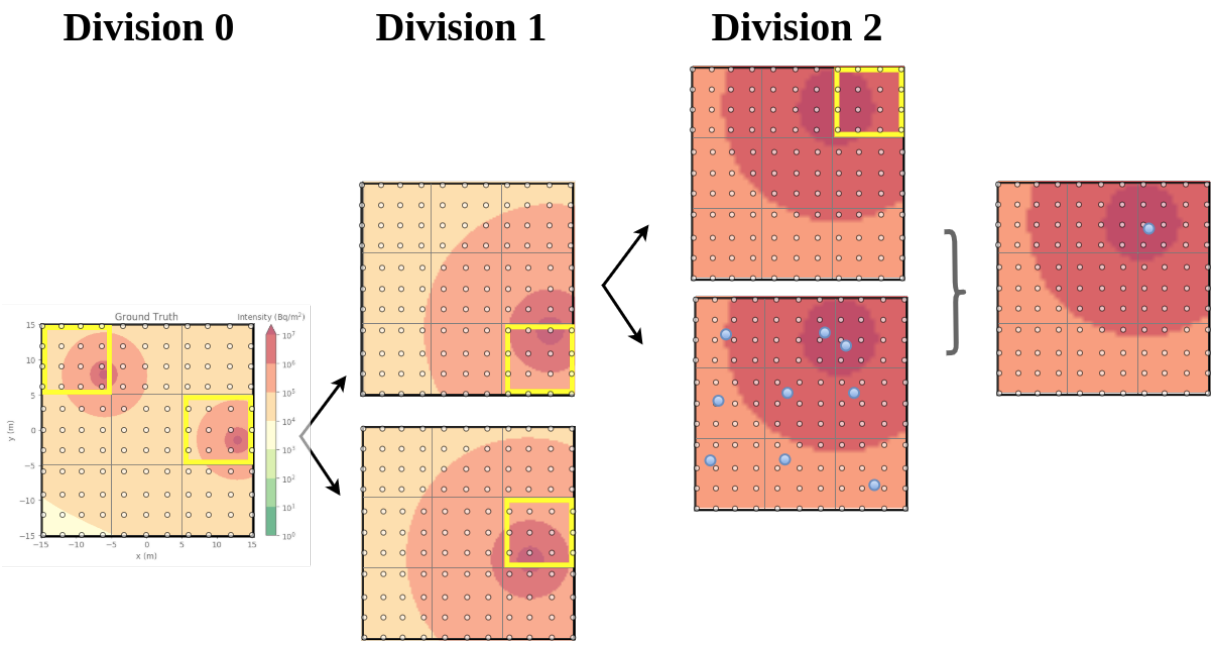


Figure 2.12: Representation of the way the Divide and Conquer method is applied in this thesis, with an example of two divisions. The initial map is called ”Division 0” and, the two boxes with a source are then shown in ”Division 1”, after performing the ”divide” step. Then, in ”Division 2” proceeds the ”conquer” step, where both regression and classification problems are applied to the boxes from ”Division 1” that have a source, and on the right, a combination of both results is depicted. To note that, although not represented, ”Division 2” also occurs on the bottom box from ”Division 1”.

Thus, to overcome the problems and limitations involved in the development of an algorithm based on ANNs, which were mentioned in section 1.2.2, the selected approach of this thesis aims to build an ANN model using a training data set composed only of simulated data, and adapting the simulator parameters according to the scenario conditions. Then, using the DAC algorithm with the pre-trained ANNs, and taking advantage of the capacity of the regression and classification problems (which will be explained afterward in section 3.4), it is possible to apply the model in both large and short-scale scenarios. Only when there are significant variations regarding the dimensions of the last division of the DAC process, the range of activities of the sources, the background intensity, or the way the observations are acquired, new training processes are performed according to the new conditions. Besides, the ANN abilities, and the way they are used in this approach, may solve some limitations of most existing algorithms, such as the existence of a maximum number of sources to get accurate detections (since some models degrade or the runtime explodes as the number of sources increases), the non-scalable applications of the models, and the obstacles information given *a priori*. Lastly, having the ANN model already trained, it is possible to use this approach in real-time (during a real surveillance, for instance), obtaining fast responses within an extremely short time (independently on the number of sources and the presence of obstacles), which can be highly demanded in radiological operations related with the detection of hot-spots.

2.3 Quality Metrics

For each model considered, the neural network is trained using the training set, while its error is evaluated at the same time in the validation set. According to the Early-stopping technique, those parameters corresponding to the best iteration are saved, and are then used with the test data set in order to evaluate the performance of the model.

Therefore, there are some criteria to evaluate and then select the best models, such as the shortest value of $d_{p_{s_i},p_{\hat{s}_i}}$ (distance from estimated source position to target source position), as well as the difference between the considered activity in real or simulated scenarios and the estimated one. To do so, the Mean Squared Error (MSE) is used to measure the average of the squares of the errors, i.e. the average squared difference between the estimated values and the actual values. Therefore, regarding the activity predictions, the following expression is adopted:

$$MSE_a = \frac{1}{n}\sum_{i=1}^{n}(\hat{a}_i - a_i)^2 \tag{2.14}$$

with n being the total number of samples, $\hat{a}$ the activity prediction and a the real value. For the location predictions, a similar expression is used:

$$MSE_{x,y} = \frac{1}{n}\sum_{i=1}^{n}\left[(\hat{x}_i - x_i)^2 + (\hat{y}_i - y_i)^2\right] \tag{2.15}$$

with $(\hat{x}, \hat{y})$ being the estimated values and (x, y) the real values. These two metrics are calculated separately since the activity values may be on the order of $10^2 - 10^8$ whereas the location values are on the order of $10^0 - 10^2$, and so they are on completely different scales.

In addition, an important criterion to take into account to compare the various models, namely to evaluate the classification outputs, is the Precision and Recall concept [73]. The Precision measures how accurate are the predictions by giving the percentage of corrected identifications, while the Recall measures the efficiency of the model to find the proportion of positives correctly identified. Also, there are the terms true positives (*TP*), true negatives (*TN*), false positives (*FP*), and false negatives (*FN*), to compare the results from the model (associated to positive/negative) with the reality (related to true/false). Therefore, the corresponding rates are given by:

$$TPR = \frac{TP}{TP + FN} = Recall \tag{2.16}$$

$$TNR = \frac{TN}{TN + FP} \tag{2.17}$$

$$FPR = \frac{FP}{TP + FN} = 1 - TNR \tag{2.18}$$

$$FNR = \frac{FN}{TP + FN} = 1 - TPR \tag{2.19}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.20}$$

Besides that, it is possible not only to compute these previous metrics, but also other important definitions to evaluate how good the prediction are, such as the Accuracy (eq. (2.21)), which corresponds to the measure of all the correctly identified cases, and the F1-score (eq. (2.22)), which makes a relationship between Recall and Precision. The Accuracy metric is also used during the training process for the classification outputs, once all the classes are equally important. Then, a model is considered as ”accurate” when a high enough value of Accuracy is obtained.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.21}$$

$$F1\text{-}score = 2 \times \frac{Recall \times Precision}{Recall + Precision} \tag{2.22}$$

Regarding these concepts of Precision and Recall, one of the main goals of such a model with classification outputs, is to avoid false alarms, which are related to the false positives, as well as to prevent missed detections, which are associated with false negatives. This way, although this work aims to obtain a high value of Accuracy as already explained, high values of Accuracy may by obtained and still get some false alarms and missed detections. That is why the F1-score metric is also used, which provides a better measure of the incorrectly classified cases. As it is possible to see by Fig. 2.13, even with Precision = 1, if Recall = 0, then the F1-score remains equal to 0, and vice versa. This means that F1-score emphasizes the lowest value, in such a way that if one of the parameters is small, the other one no longer is so relevant.

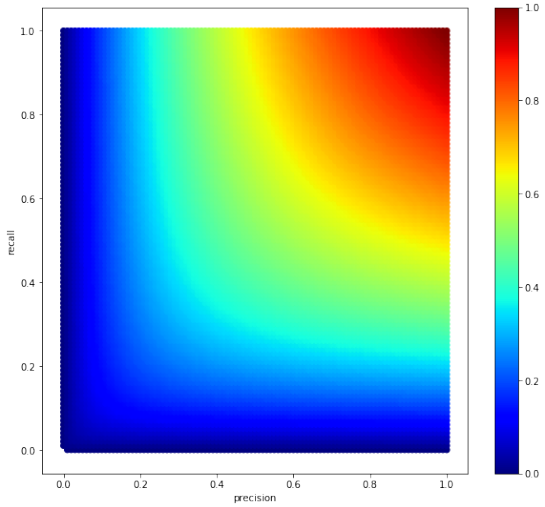


Figure 2.13: Contour plot of F1-score as a function of Precision and Recall, with both varying between 0 and 1.

3

Setup and Data Assumptions

Contents

The assumed setup is depicted in Fig. 3.1. Since the radiation sources are assumed to be stable, static and located on the ground, an UAV is used to get measurements. More precisely, they are acquired by a drone, which carries on a Geiger-Müller counter to measure the intensity, and a GPS receiver to measure the position. Two other sensors, a depth camera and a rangefinder, are also used to measure the distance to the ground, allowing the mission to be planned at a given constant distance to the ground.

This experiment is performed in both simulated and real scenarios. Many scenarios with a different number of sources, as well as different corresponding parameters, are used during data acquisition in order to, after the training process, given observations from another scenario, the ANN model is capable of predicting the parameters of the sources present in this scenario.

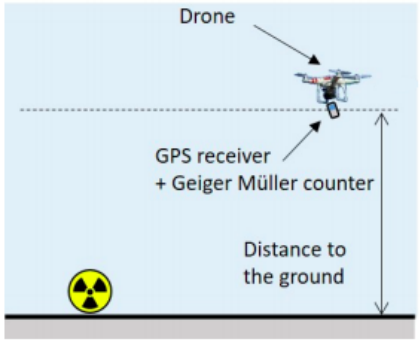


Figure 3.1: Experimental setup with a drone flying over the scenario. Image from [10].

3.1 Simulated Data

In the MSc Thesis of Yoeri Brouwer [74], it was developed in Python a computer algorithm able to provide a radiological heatmap in 2D, given the initial conditions of the radioactive sources: number, activity and spatial distribution, which come from an external file. The simulated scenario, represented in Figures 3.2 and 3.3, consists of a dense grid of 600x600 cells where the intensity of each cell is calculated according to the radioactive activity and to the distance to the source(s), taking into account the model of intensity propagation radiation where the radiation intensity decreases with the inversion of square distance to the sources. The background intensity is also included in every calculation. In addiction, this simulator is also able to provide random observations of the scenario. The corresponding intensities are acquired through random samples that come from a Poisson distribution with the radioactivity mean, being then transformed into intensity values (given by the quotient between the radioactivity and the cross-sectional area of the detector).

Later, a similar simulator was also developed with obstacles included, by using Shapely [75] - a Python library that helps in manipulation and analysis of geometric objects in the Cartesian plane. In Fig. 3.4, it is exemplified a simulation with two obstacles, by using this simulator.

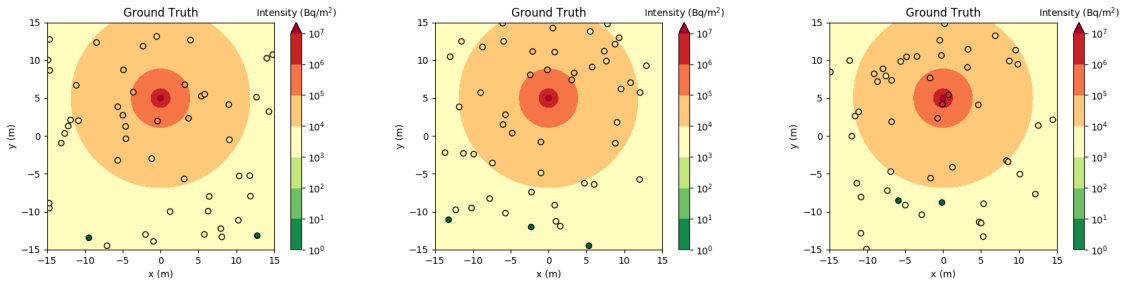


Figure 3.2: Examples of different sets with $M = 20$ measurements and with the same source.

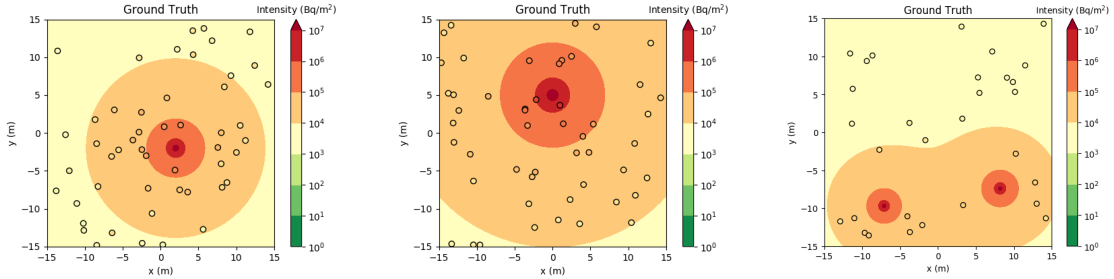


Figure 3.3: Examples of variations about the number, activity and position of the sources.

Thus, taking into account the objectives of this thesis, a simulator based on the previous two simulators is used with some improvements implemented. Firstly, regarding the simulated measurements, the usual uniform random process is not used since it may lead to observations with positions too close to each other, which does not reflect reality. Therefore, it is adopted two different ways to get them:

- Randomly, by taking the ”Poisson-disk sampling” method that uses Robert Bridson’s algorithm [76], and it is demonstrated in Fig. 3.5 (left). This is a popular approach to get non-clustered random sample of points and, therefore, more realistic observations are made. The implemented code is based on a project previously developed by Christian Hill in [77].

- Grid sampling, represented in Fig. 3.5 (right), where a discretization of the space is adopted, in such a way that we have a grid of pre-defined $G \times G$ fixed positions to get the observations from the drone. Since there is always an uncertainty associated to the position of the drone, it is used the normal (Gaussian) distribution from Numpy library [78] to get the positions for each observation. The mean (”center”) of the distribution is given by those fixed positions, while the standard deviation (spread or “width”) is such that must represent the uncertainty of the drone’s position, being assumed a value of 25 cm as default.

Then, these observations and the corresponding simulation plots are saved in files with ’*csv*’ and ’*png*’ formats (as represented in Fig. 3.5), respectively.

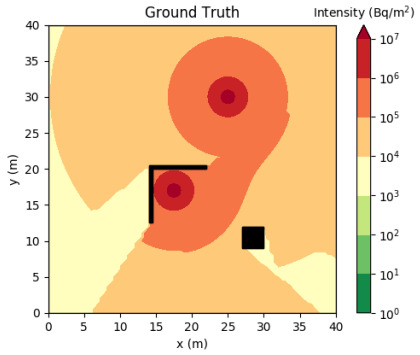


Figure 3.4: Example of a simulation with obstacles (in black color).

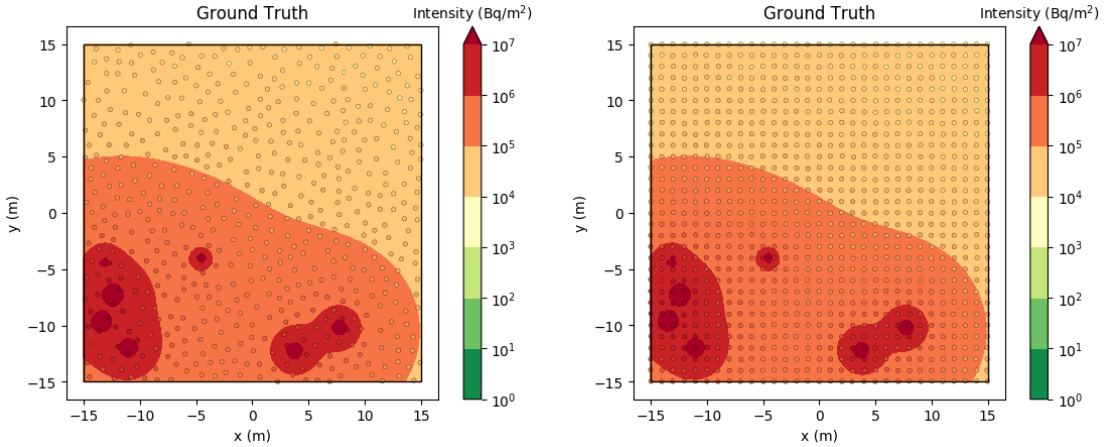


Figure 3.5: Representation of the two ways to get measurements: the Random approach, with a Poisson disc sampling (left), and the Grid approach, with a grid sampling based on Gaussian distribution (right).

To create the external file with information about the number, location and activity of the target sources, it was developed a source generator program called ***sources_generator.py***. This program allows the user to choose a fixed number of sources for all the scenarios, or to choose a 'random' option, imposing a maximum number of sources. Also, regarding their spatial distribution, it is possible to do it in a way completely random, through a uniform random distribution, or by imposing some conditions about the number of cells that divides the scenario, so each cell can have at most one source. Lastly, for the activity of the sources, a range of values is defined so the algorithm computes a random value that belongs to that range. Thus, all these manageable possibilities allow getting a more varied data set so the algorithm can learn and then predict from more generalized observational data.

A non-point-like source approach involves a complex scenario and its simulation provides some difficulties, in relation to how radioactivity behaves from a source with a non-trivial configuration. Therefore, the assumption made in equation (2.1) would have to undergo some changes that are not so obvious. So, in order to study the problem of non-point-like sources, a more complex scenario is simulated by

arranging multiple sources in a certain configuration, as depicted in Fig. 3.6 with an L-shaped configuration.

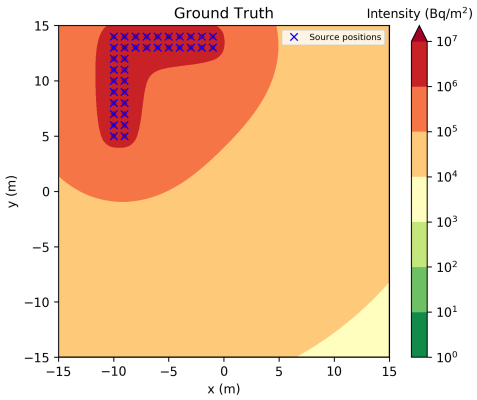


Figure 3.6: Simulation of a non-point-like source, with an L-shaped configuration. It is done by using 36 sources with $5 \times 10^6\ Bq$, separated by $10\ cm$ from each other.

3.2 Real Data

The most fundamental unit (and also the SI unit) for radioactivity is becquerel (Bq), which means the number of nuclear disintegrations per second. Regarding the intensity measurements (number of particles detected within the exposure time interval) the unit is Bq/m^2. However, unlike simulated data where it is possible to control measurement units, real sensors may not provide intensity measurements in the most manageable unit. They count the number of detected particle interactions, without measuring the energy of the particles. In the case of a GMC, this is a device that detects ionizing particles through a Geiger-Müller tube and count their number in a given time interval. Therefore, since a GMC has no way to measure the energy of the particles, the measurements are provided in units of counts per second (CPS), or in counts per minute (CPM). To transform the intensity of the observations, c_{o_j}, from CPM into Bq/m^2 unit, the following expression is adopted:

$$c_{o_j}[Bq/m^2] = \frac{c_{o_j}[\text{CPM}]}{60[s/min] \times A_{det}[m^2] \times eff_{det}} \tag{3.1}$$

where A_{det} and eff_{det} represent the area and the efficiency of the detector, respectively. In order to simulate scenarios where the intensity measurements have CPM units as well, it is assumed $A_{det} = 5\ cm^2$ and $eff_{det} = 0.9$ as default values. If the GMC sensor changes, the GMC verification procedure detailed in [79] helps with the sensor calibration and the way the measurements can be normalized.

Another way to get measurements without using a drone is the one depicted in Fig. 3.7. This alternative experiment is associated to a human operation walking in the scenario, equipped with a mobile phone connected to the GMC, and, as done with the drone, with an approximately constant

distance between the sensor and the ground. Both experimental approaches are done with the aid of a mobile application, denominated MARIA (Mobile Application for Radiation Intensity Assessment), which is being developed at Instituto de Plasmas e Fusão Nuclear (IPFN) and is detailed in the Appendix A.

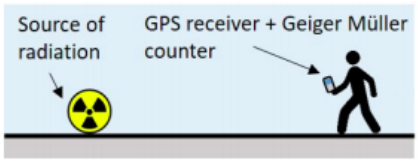


Figure 3.7: Alternative for experimental setup, with a person operation instead of a drone. Image from [10].

In case of small scenarios, where experimental sources have low activities and so only measurements close to the source give relevant information, the mobile phone's GPS may not have enough sensibility to get the position values. To solve this problem, the positions of the observations must be pre-defined, before the data acquisition, so each intensity measurement can be associated to the corresponding position *a posteriori*. In this thesis, such small scenarios are presented in section 5.2.

3.3 Data Acquisition

In the simulator case, it was implemented a program called ***get_data_simulator.py***, where the frameworks Numpy and Pandas [80] are used. It uses the '*csv*' files with the measurement data and the source parameters to build the corresponding arrays that are then used for the training process as input and target, respectively. This way, the information of each simulation is organized as follows:

$$[a_{s_1} x_{s_1} y_{s_1}, ..., a_{s_N} x_{s_N} y_{s_N}] \tag{3.2}$$

for the target sources, while the corresponding measurement data becomes

$$[c_{o_1} x_{o_1} y_{o_1}, ..., c_{o_M} x_{o_M} y_{o_M}] \tag{3.3}$$

with $M >> N$.

Regarding the real data, after the acquisition from the drone, both the measurements of position and intensity are then synchronized and recorded in a light computer installed on the UAV, so they can be processed offline in an external and more powerful computer.

By using MARIA application, a file with 7 columns (date, hour, latitude, longitude, altitude, accuracy and intensity) is obtained, where each line corresponds to an observation. As already explained, the intensity values have CPM units and, the altitude values have meters unit, being supposed that all these values are approximately equivalent. The longitude and latitude values have degrees unit and,

therefore, they have to be converted into (x, y) coordinates, taking a reference position. To do so, the so-called Equirectangular projection [81,82] can be used when the scenarios are too small when compared to the size of the Earth, and so the Earth's curvature may be neglected. Therefore, this projection maps meridians (longitude lines) and parallel circles of latitude to vertical and horizontal straight lines, respectively, of constant spacing. Thus, after transforming the values of longitude and latitude from unit of degrees into unit of radians, the first measurement is defined as the origin of the referential and all other coordinates are relative to that origin.

3.4 Training Data generation

The quality and quantity of the training data is very important. It refers to the initial data that is used to develop the Machine Learning model, from which the model learns and builds its rules. Then, using the simulator described in section 3.1, a set with a huge amount of data can be obtained, with well-diversified configurations regarding the source parameters, namely its number, activity and spatial distribution.

The generated training data has similar dimensions to the last division of the Divide and Conquer method since it is here when both the regression and classification problems are used. Instead, in the previous divisions, only the classification problem is needed, and so the model should be able to get accurate predictions even with larger dimensions compared to the ones used during the training process. This is related to the fact that, for the regression problem, the distance between the observations has a great influence in computing the activity of the sources (as seen in eq. (2.3)). That is, if scenarios with different dimensions were present in the training data set, then the distance among all input neurons and also their relative positions on the map would have to be different and, thus, the model would not be able to learn the relationship between the intensities and the positions of the observations to compute the regression values. To note that there is a greater influence regarding the activity outputs once they are not normalized as the location outputs. On the other hand, for the classification problem it only matters the relative boxes where there is at least one source, and so the model is much more capable of adapting to different dimensions of those used for training. Therefore, although the more distinct are the dimensions of the scenarios used during the training process, the lower is the accuracy obtained regarding the classification outputs, the model is still capable of getting accurate predictions. For this reason, in section 4.3.1, a study of the threshold's influence is performed in order to use the appropriate values or determine the range of possible values according to each division of the DAC process.

It is generated $20,000$ simulations, each one with a different set of sources, and the corresponding configuration was as follows (exemplified in Fig. 3.8) :

- Scenario dimensions of $3.33 \times 3.33\ m$, since the last division of DAC process for the simulation results (in section 5.1) has the same dimensions.

- As already referred, there are no obstacles on simulations during this phase of training, or non-point-like sources either.
- Regarding the generated sources:
  - their activity is set to be between 10^6 and 10^8 Bq;
  - their positions are defined in such a way that the scenario is divided into 3×3 cells (with a size of 1.11×1.11 m each) and each cell can have at most 1 source;
  - and the maximum total number of possible sources for each scenario is 4, which is a relative high number considering the dimensions of the scenario and the range of activities of the sources. In training processes with larger dimensions or weaker sources, this maximum number of sources should be 9 (which corresponds to the total number of boxes), as done for results of section 5.3.
- 11×11 data points (measurements) for the Grid approach, and about 200 for the Random approach. In the next chapter 4, the reason for this difference between the two approaches for data acquisition is explained.

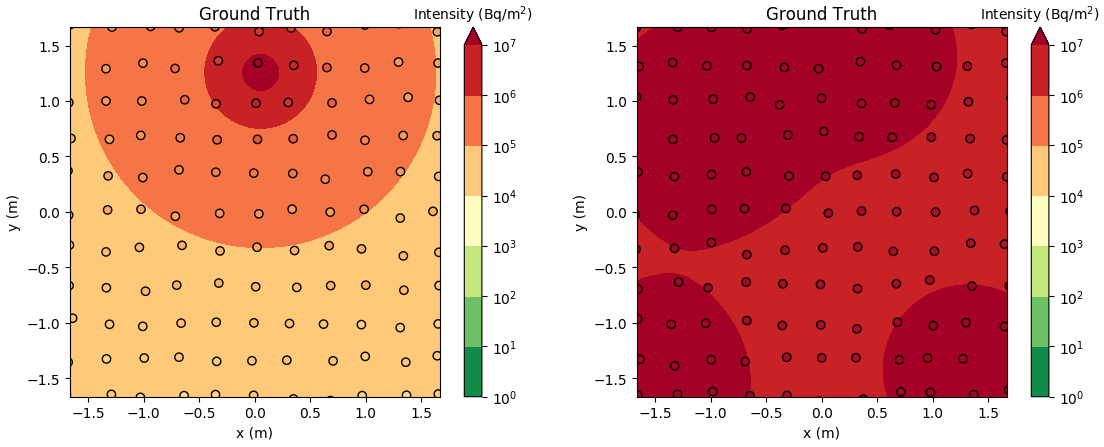


Figure 3.8: Illustration of two simulations using the Grid approach, with the settings used to build the training data set. On the left, there is one source, whereas on the right there are four sources.

The whole data set is then split into two parts: training and test sets. The training set is used during the training process with known outputs (called targets), so the model learns on this data in such a way that can be able to get outputs of other data later on. On the other hand, the test set is used to evaluate the final model's performance and compare it in different tested approaches. Besides that, for the training process, the training data set is still divided in two subgroups: training and validation sets, so the method called Early-stopping can be used.

4

Neural Networks Implementation

Contents

The neural networks algorithm was implemented in Python supported by Jupyter Notebook [83]. It was used the framework Keras [84], running on top of the open-source Machine Learning platform TensorFlow [63, 64], which provides high-level building blocks for developing deep learning models and allows to export the models to run in the browser or on a mobile device. Also, TensorFlow is supported by GPU, that enables multiple matrix computations simultaneously and then provides a faster and better training process.

All the computations associated to the neural networks implementation were performed through the CUDA (and CUDnn) [85] compatible Nvidia GeForce MX150 and an Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz $\times$ 8 processor.

4.1 Data Preprocessing

Data preprocessing is a fundamental step of Machine Learning since the quality of data and the useful information that can be derived from it directly affects the ability of the model to learn. Also, all the acquired data may not be fit enough to be used by Machine Learning algorithms and, therefore, this is extremely important step in order to transform the data before feeding it into the model.

In fact, often more time is spent on the data preparation than on the actual machine learning, as happens in the case of this thesis. NumPy [78], Pandas [80], Sklearn [86] and Keras [84] are the open-source frameworks used in this thesis to help creating the programs for data preprocessing.

Firstly, a program called ***data_pooling.py*** was created. In Deep Learning, data pooling is commonly used in CNN layers in order to reduce the resolution of the feature map (see Fig. 2.6) but retaining the most important features of the map required for getting the output, through translational and rotational invariants. The two most common functions used in the pooling operation (represented in Fig. 4.1) are:

- Maximum (Max) Pooling, which calculates the maximum value for each patch of the feature map;
- Average (Avg) Pooling, which computes the average value for each patch on the feature map.

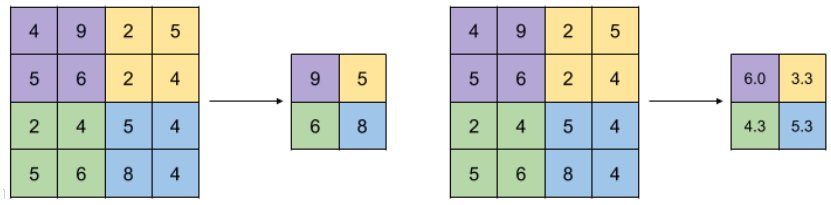


Figure 4.1: Representation of max pooling (left) and average pooling (right), with a 2×2 filter size and a $(2, 2)$ stride. Image from [11].

Unlike usual approaches with CNN layers, in this thesis the data pooling method is only used on data preprocessing, in order to get the correct number of inputs needed to fit the neural network model.

The max pooling is used to extract the observation with the highest intensity value, from the set of observations that belong to a given region of the map, which corresponds to an input of the model. On the other hand, the average pooling is used to make an average of the intensities and positions (x and y) of all the observations that belong to the same region of the map, and then obtain only one input. The reason why the CNN layers do not make pooling operations is related to the fact that, in this work, the number of observations is too much smaller than the usual number of pixels of a ordinary image and, therefore, there is no advantage in further reducing the 'resolution of intensities'. For instance, while an image has normally a resolution with more than 100×100 pixels, in this thesis the observations are used to build an input with 11×11 intensities at most.

This program is used only when data is acquired randomly, whereas when a discretization of the space is made there is no need to use it. Since the input of the model (see Fig. 4.6) is composed by a square matrix, of side s, which can be seen as a set of $s \times s$ grid cells with well-defined boundaries, it means that if the data is acquired in a random way, then the data must be clustered in each input grid cell, according to its positions, so we can perform the pooling method and get only one value for the input neuron that is associated to that grid cell. However, when the data is acquired following a space discretization, each of the $G \times G$ pre-defined positions to get the measurements should have the same correspondence as each of the $s \times s$ matrix entries, i.e., we can say that $G = s$ and so there is no need to perform a pooling method since every matrix entry has already only one measurement associated.

Then, a program denominated ***data_loading.py*** is created in order to, using the simulated data acquired from the program ***get_data_simulator.py***, two vectors are built as follows:

- The input vector, with a format $(N_{\text{sim}}, 3 \times M)$, where N_{sim} represents the number of simulations (or samples) with a set of source(s) and $3 \times M$ means the total number of values $(c_{o_j}, x_{o_j}, y_{o_j})$ of all the measurements, with $j = 1, ..., M$.

- The target vector, with a format $(N_{\text{sim}}, 3 \times N)$, where N_{sim} represents once more the number of simulations with a set of source(s) and $3 \times N$ is the total number of values $(a_{s_i}, x_{s_i}, y_{s_i})$ of all the parameters of the sources, with $i = 1, ..., N$.

Depending on the kind of data that is being acquired, i.e., if the model is learning from random or grid data, this program uses different computations, such as the use of data pooling in the case of random data. Also, this program saves a NumPy dictionary with all these input and target vectors since this step can take too long due to the data size, and so there is no need to repeat it every time the "user" adjusts the model hyperparameters and performs a new training process.

Finally, the program ***data_preprocessing.py*** was created so the previous input and target vectors can be transformed to fit the respective input and output layers of the neural network model. As already explained in section 2.2, in order to make the best use of the CNN properties, the input must be a matrix

of intensity values, whose corresponding measurement positions in the map match with the positions of the matrix entries. So, having $s \times s$ observations, the intensity values from the s bottom observations are on the first line of the matrix, while the intensity values from the upper observations are on the s^{th} (and last) line of the matrix. In Fig. 4.2 it is represented with $s = 11$, which is the default value used in this thesis for the Grid approach. When we are dealing with the Random approach, the default value is $s = 9$.

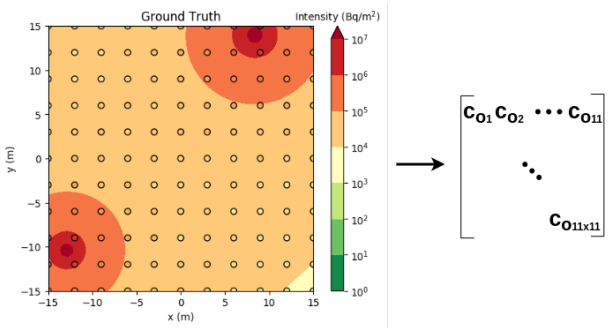


Figure 4.2: Matrix of intensity values, with respective positions matching the positions of the matrix entries.

If the positions of the measurements were fixed, this two-dimensional matrix could be the input of the model. However, since the drone's positions have always uncertainty associated and the measurements may be acquired in a random way, then the positions (x_{o_j}, y_{o_j}) should be part of the input too to get more accurate results. To do so, as represented in Fig. 4.3, a three-dimensional matrix is used as input, where each dimension is composed by the values of intensity c_{o_j}, x-coordinate x_{o_j} and y-coordinate y_{o_j}, respectively, following a similar procedure of the one represented before in Fig. 4.2. Then, we need to reshape the input vector of each sample to become a three-dimensional square matrix, i.e., with a format $(s, s, 3)$.

Still, the input values of the (x_{o_j}, y_{o_j}) positions get normalized within the range $[-1, +1]$ (as drawn in Fig. 4.5), since the goal of classification problem is always to identify which of the 3×3 boxes have at least one source and, therefore, these predictions are not influenced when input values are larger. This is really important taking into account that when DAC is being applied, the initial scenario dimensions can be much larger than the last division. So, to make this normalization, the following transformation is used:

$$x \rightarrow 2 \times \frac{x - x_{min}}{x_{max} - x_{min}} - 1 \tag{4.1}$$

for both x_{o_j} and y_{o_j} values.

On the other hand, the target vector must be divided into three distinct vectors: one for the classification problem and the other two for the regression problem, with one referring to the activity values of the sources (a_{s_i}) and the other to the position values (both x_{s_i} and y_{s_i} parameters). For the classification

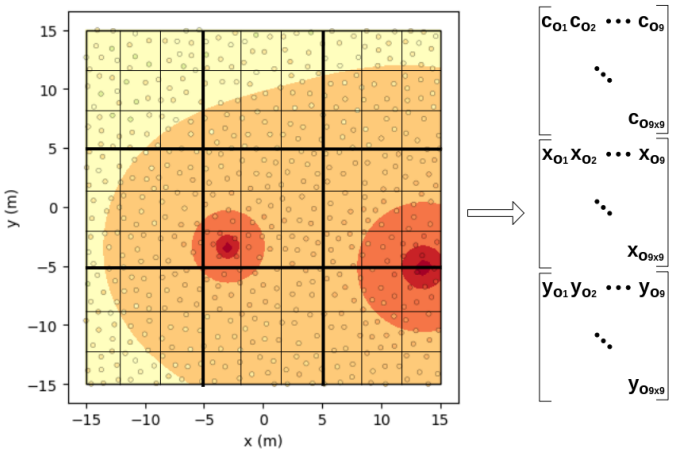


Figure 4.3: Model's input, when data is acquired through the random way. After applying data pooling, only one value is obtained for each entry of the 3D matrix, with $s = 9$. For the Grid approach, the model's input is similar of Fig. 4.2, but with a 3D matrix (with x and y values included) instead of a 2D matrix (with only the intensity values).

target, it is followed the scheme with 3×3 boxes represented in Fig. 4.4 in order to encode the positions of the sources so it is possible to associate each classification output neuron to a specific box of the map. To do so, it is made a correspondence between (x_{s_i}, y_{s_i}) values and the respective box number and, in the end, we get a categorical vector array. In the case of the example retracted in Fig. 4.4, where the boxes 0 and 8 are the ones with a source, the target has the following shape:

$$[p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8] = [1, 0, 0, 0, 0, 0, 0, 0, 1] \tag{4.2}$$

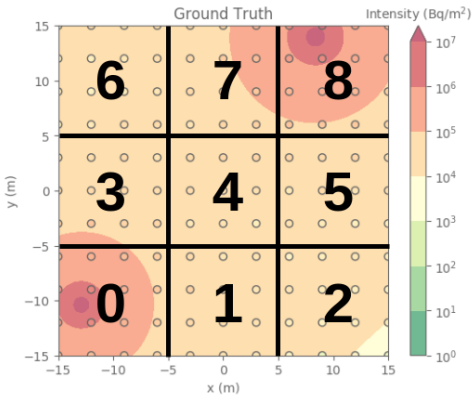


Figure 4.4: Illustration of the map division with 3×3 boxes, where each one corresponds with one of the classification outputs. This is, the i^{th} box (with $i = 0, ..., 8$) is related to the classification output p_i.

Regarding the regression outputs, the position values need to be normalized once more, namely the

(x_{s_i}, y_{s_i}) values. However, unlike previously, this normalization has the difference that, knowing the box where the source s_i is, the corresponding position values are transformed in such a way that they are re-scaled to the $[0, 1]$ range, by dividing by the respective box's width and height, as drawn in Fig. 4.5. Therefore, the regression target vectors, from the example depicted in Fig. 4.4, can be represented as

$$[x_0, nan, nan, nan, nan, nan, nan, nan, x_8, y_0, nan, nan, nan, nan, nan, nan, nan, y_8] \tag{4.3}$$

for the position values, with $x_{0,8}$, $y_{0,8} \in \mathbb{R}$, and for the activity values we have

$$[a_0, nan, nan, nan, nan, nan, nan, nan, a_8] \tag{4.4}$$

with $a_{0,8} \in \mathbb{R}^+$.

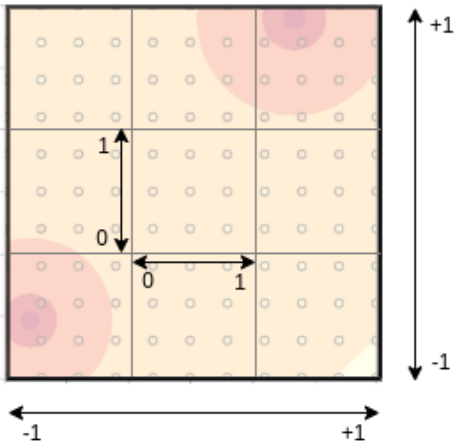


Figure 4.5: Normalization of input positions (within $[-1, +1]$ range) and output positions ($[0, +1]$ range).

The vector entries where the corresponding box has no sources have the 'nan' reference, which represents the expression "Don't care" used by Yolo authors in [7], as explained in subsection 2.2.3. The purpose of using a 'nan' value is related to the TensorFlow framework's ability on working with mask functions, which will be better explained on the next section.

Finally, the data preprocessing ends with the division of whole data set already processed with the previous transformations, by defining the percentage distributions for the training, validation and test sets.

4.2 Neural Networks model

Having the input and output layers already delineated, the hyperparameters of the model have to be defined, i.e., we have to build the hidden layers and the way how they are connected with each other as

well as with the input and outputs layers.

As detailed in section 2.2, the model is implemented based on CNNs, where the initial convolutional layers extract features from the input while the fully connected (or dense) layers, that form an MLP, predict the classification and regression outputs.

Since the input is three-dimensional, 3D convolutional layers are adopted (by using the so-called *Conv3D* from Keras framework). The size of the dense layers, the number of hidden layers, the number of respective neurons, and the number of convolutions (as well as the kernel size, the stride and the depth) have been subject of extensive experiments. In the end, the best hyperparameters found are represented in Fig. 4.6, where it was observed that adopting small differences in these hyperparameters according to the classification, localization and activity outputs, better results are achieved.

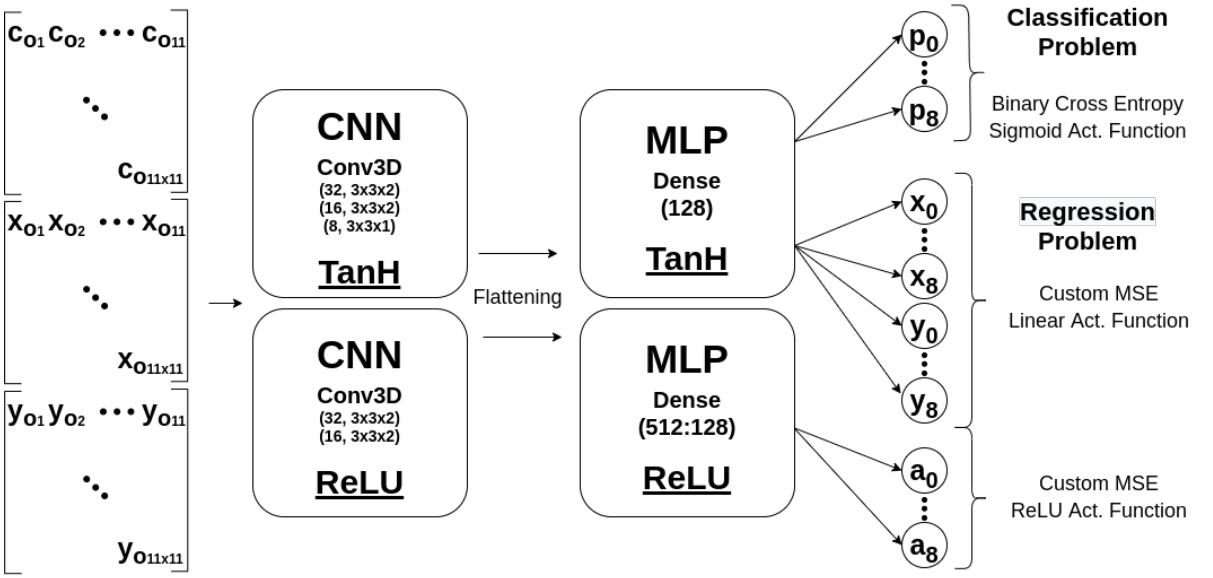


Figure 4.6: Scheme of neural networks model with $3 \times 11 \times 11$ input shape (for the Grid approach, while for the Random approach it would be $3 \times 9 \times 9$). The first CNN layer is composed of 32 kernels with a $(3, 3, 2)$ size, a stride $(1, 1, 1)$ and with no padding. The second CNN layer has 16 kernels with the same size and, for the upper sub-model, the third layer has 8 kernels with a $(3, 3, 1)$ size. Then, after flattening the outputs from the last CNN layer, they become inputs of the MLP layers, which are fully connected (or dense) layers composed by 512 and 128 neurons (and only one layer with 128 neurons for the upper sub-model). The upper sub-model adopts the Hyperbolic Tangent (TanH) activation function, whereas the lower sub-model uses the ReLU. In the end, it is presented the final outputs, associated with the corresponding loss function and activation function.

As we can see, the output of the model can be divided in two ways: the one that is associated to the positions of the sources and the other one to its activities. This way, the model can also be divided into two sub-models, where some hyperparameters are distinct. Besides the ones already specified, the activation function is different too. The sub-model responsible for the positions of the sources, i.e. the one with classification and (regression) position output neurons associated, is built with the so called

Hyperbolic Tangent (TanH) activation function, given by

$$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{4.5}$$

and represented in Fig. 4.7. This function is symmetric in 0 and the range of values is from -1 to $+1$, which can be very useful when we are dealing with localization problems. Therefore, this also explains the input normalization adopted in the previous section.

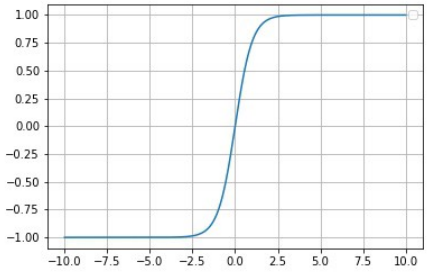


Figure 4.7: Hyperbolic Tangent (TanH) activation function.

Regarding the other sub-model, associated to the regression activity output neurons, the Rectified Linear Unit (ReLU) activation function is adopted, which is represented in Fig. 4.8 and given by

$$\phi(z) = \max(0, z) \tag{4.6}$$

As it can be seen, the ReLU is half rectified, being zero when $z < 0$ and linear, i.e. $\phi(z) = z$, otherwise. So, the range is from 0 to infinity, which is also the range of the activity values of the sources. Besides, this is the activation function must used in CNNs in literature at the moment.

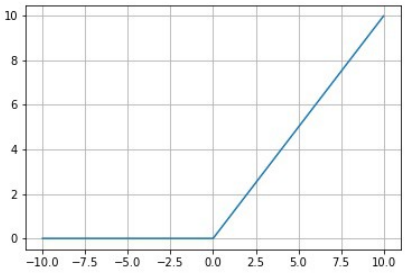


Figure 4.8: Rectified Linear Unit (ReLU) activation function.

Finally, for the output layer, we have to define the loss function and once more an activation function that can fit the shape of the output data. For the classification problem, the output is a categorical vector (represented in expression (4.2)) where more than one entry can be ”1”, and so one input can be

associated with multiple labels. This means that a multi-label classification is being implemented, where the prediction of each output neuron is independent from the other neurons and, therefore, the last layer uses a sigmoid function as activation function, which is given by

$$\phi(z) = \frac{1}{1 + e^{-z}} \tag{4.7}$$

and illustrated in Fig. 4.9. The output of this function, also called logistic function, varies between 0 and 1, which is the desired range for each output neuron of the classification problem, since they represent a probability. Being a multi-label classification problem, the loss function used during the training process

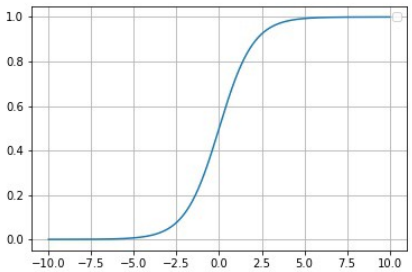


Figure 4.9: Sigmoid activation function.

is the Binary Cross Entropy function, which is given by the following expression:

$$\mathcal{L}_p = \mathcal{L}(p, \hat{p}) = -\frac{1}{N_{\text{sim}}} \sum_{i=1}^{N_{\text{sim}}} \Big[p_i \log(\hat{p}_i) + (1 - p_i) \log(1 - \hat{p}_i) \Big] \tag{4.8}$$

where p_i represents the real value while $\hat{p}_i$ is the prediction. With this loss function, it is possible to penalize bad predictions and get a low loss value for good predictions.

For the regression output, as explained in the previous section, the outputs associated to the activity of the sources remain on the range $[0, +\infty]$ and, for that reason, the ReLU function is used as the activation function in the last layer, as it was done for the hidden layers. On the other hand, although the outputs for the localization prediction are normalized within the $[0, 1]$ range, and so one might consider using the ReLU function for the same reason as before; in this case, the output must not be limited once it may not belong to the $[0, 1]$ range. That is, if we get a value outside this range (which is unlikely but possible), it simply means that the model is predicting that the source is not on the corresponding box, but may be around by. Although this may lead to more than one prediction for the same source on the border of that box, it can also have the advantage of detecting a source that was undetectable on its own box due to a stronger intensity coming from one source of a neighboring box. So, for the localization output, the linear function is used as the activation function. Regarding the loss function, for both outputs, a customized function based on the Mean Squared Error (MSE) loss function is used.

Firstly, we recall the MSE loss functions for activity (a) and location (x, y) predictions (similar to equations (2.14) and (2.15), respectively) that are given, respectively, by

$$\mathcal{L}_a = \mathcal{L}(a, \hat{a}) = \frac{1}{N_{\text{sim}}} \sum_{i=1}^{N_{\text{sim}}} (\hat{a}_i - a_i)^2 \tag{4.9}$$

$$\mathcal{L}_{x,y} = \mathcal{L}(x, \hat{x}, y, \hat{y}) = \frac{1}{N_{\text{sim}}} \sum_{i=1}^{N_{\text{sim}}} \Big[(\hat{x}_i - x_i)^2 + (\hat{y}_i - y_i)^2 \Big] \tag{4.10}$$

where $(\hat{a}_i, \hat{x}_i, \hat{y}_i)$ represent the regression predictions related to the target values (a_i, x_i, y_i). This is the most commonly used regression loss function, which computes the sum of squared distances between the targets and the predictions, allowing that larger mistakes provide more loss error than smaller mistakes, and so the model is punished for larger mistakes. However, as it can be seen from expressions (4.3) and (4.4), the regression targets are composed by 'nan' values. Therefore, the loss function needs to be adapted so these values can be neglected by the regression loss function, and so it does not provide any punishment during the training process. To do so, those 'nan' values are masked by using the *boolean_mask* function from the TensorFlow framework. This way, expressions (4.9) and (4.10) can by adapted, respectively, by

$$\mathcal{L}_a = \frac{1}{N_{\text{sim}}} \sum_{i=1}^{N_{\text{sim}}} p_i (\hat{a}_i - a_i)^2 \tag{4.11}$$

$$\mathcal{L}_{x,y} = \frac{1}{N_{\text{sim}}} \sum_{i=1}^{N_{\text{sim}}} p_i \Big[(\hat{x}_i - x_i)^2 + (\hat{y}_i - y_i)^2 \Big] \tag{4.12}$$

in such a way that, if there is no source in a certain cell, multiplying by p_i ensures that the regression predictions of that cell do not get penalized. Thus, the global loss function can be seen as follows:

$$\mathcal{L}_{p,a,x,y} = \frac{1}{N_{\text{sim}}} \sum_{i=1}^{N_{\text{sim}}} \Big\{ - \Big[p_i \log(\hat{p}_i) + (1 - p_i) \log(1 - \hat{p}_i) \Big] + p_i (\hat{a}_i - a_i)^2 + p_i \Big[(\hat{x}_i - x_i)^2 + (\hat{y}_i - y_i)^2 \Big] \Big\} \tag{4.13}$$

In addition, during the data acquisition step, it may happen that some regions of the scenario do not have any observation, which leads to the problem that some of the $s \times s$ entries of the input matrix do not get any value. In order to simulate this situation, and to make the neural network model able to get good results even when this problem occurs, it is adopted a strategy based on the Dropout layers regularization technique. However, unlike the usual strategy used on neural networks represented in Fig. 2.5, in this thesis, only on the input layer this method is used. This way, although it is being used to avoid overfitting, the main purpose is to allow getting a high performance even if some regions of the map are not fulfilled by any observation, by extrapolating information from those missing observations based only on those available.

4.3 Training Process and Results

For the training process, the program ***cnn_training.py*** was developed. This step is related to one of the main downsides of ANNs: the long time required for training, which depends essentially on the software libraries adopted and on the hardware platforms (such as the Central Processing Unit (CPU), GPU, and RAM memory) used. Still, the training process can be concluded or interrupted, and later resumed with new (or not) training data, by loading the previously trained model, as long as some conditions remain fixed. In this thesis, the parameters that must remain fixed are the background intensity, the range of activity values of the sources, the scenario dimensions (width and length) used during the last division of the DAC process, and the way the observations are acquired (i.e., the use of the Grid approach or the Random approach). Otherwise, if any of these parameters change considerably, then a new training process must be performed, as it happens in sections 5.2 and 5.3.

Firstly, following the data split process detailed in section 3.4, the training, validation and test data sets are divided according to a ratio of 80%-10%-10%, respectively. The optimizer used during training is the Adam algorithm [69], with a learning rate $\eta = 10^{-4}$, and a *L2* (ridge regression) regularization is implemented on every layer, with $\lambda = 0.001$ (see eq. (2.9)). Then, having simulated $20,000$ samples in total, which leads to $\sim 16,000$ data for training, it means that we get approximately 65 batches, using a batch size of 256. The batch size is the number of samples within each batch, and the optimizer algorithm is performed at the end of each batch. Then, 600 epochs are performed, where each epoch represents a loop made over all the batches. Besides, we consider that all the regions of the map, which have a correspondence to the input neurons, are fulfilled with observations, and so the dropout rate is being assumed as zero. Later, it is shown how this parameter may influence the results, when missing observations must be considered.

As auxiliary program for the training process, the program called ***callback_training.py*** was also created in order to save all the information throughout the epochs. This way, it is possible to:

- save the best weights, by using the Early-stopping method;
- reproduce the way the neural network is learning over the epochs, by representing the predictions evolution of the validation data set, regarding both classification and regression outputs.

The behaviour of the loss function, $\mathcal{L}$, is depicted in the following Figures 4.10-4.14, with blue and orange lines denoting the train and the validation error, respectively, as well as the accuracy, and a vertical dashed line representing the Early-stopping point. In Fig. 4.10 and 4.11 we can see the classification output accuracy, which is the measure (in percentage) of how accurate the model's prediction is, compared to the target data, as explained in section 2.3.

On the left plot of the following Figures 4.12, 4.13 and 4.14, although it seems we are getting an error too high, around $\sim 10^{14}\ Bq^2$, it does not mean that the predictions are not good. Since the target

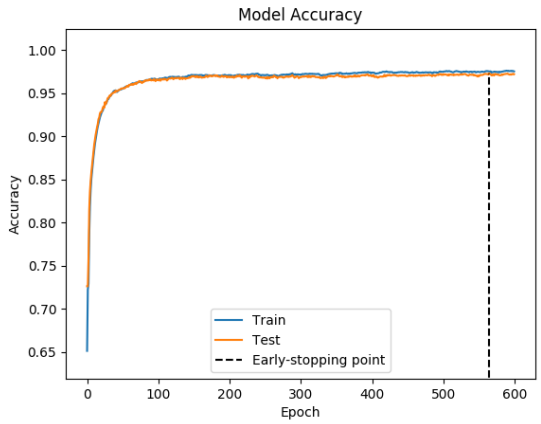


Figure 4.10: Accuracy regarding classification output using the Grid approach.

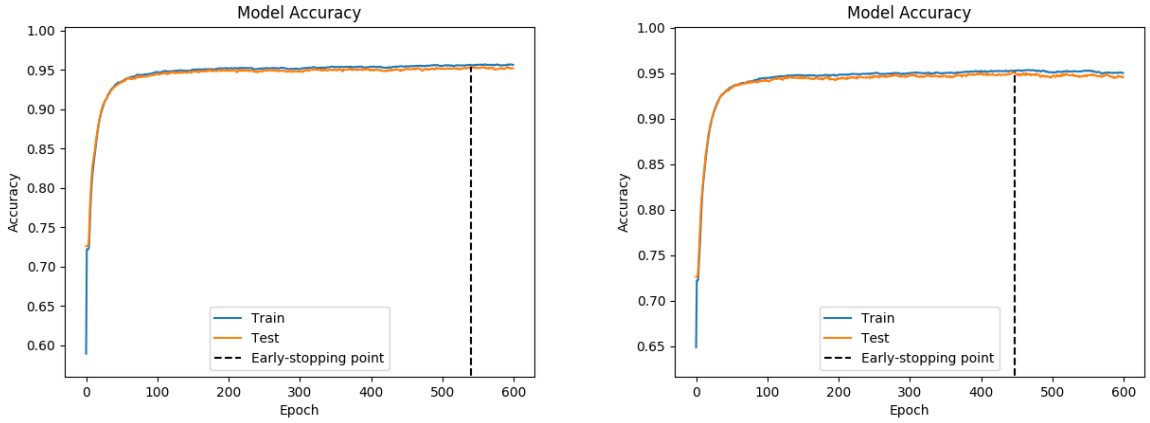


Figure 4.11: Accuracy regarding classification output using the Random approach, where average (left) and max (right) pooling are applied.

activities are within the range $[10^6, 10^8]$ Bq, and $\mathcal{L}_a$ is based on MSE function (see eq. (4.11)), when computing the loss error, all the quadratic errors of entire training data are being summed, leading to an expected huge loss error and also to the appearance of some spikes in the plots.

In Table 4.1 the best values of accuracy and loss error are presented, related to the Early-stopping point (once from this point on the validation error begins slowly increasing), for the three different approaches regarding the way the observations enter the ANN. As we can see, the accuracy is very high, which indicates that the model is predicting the regions with a source almost perfectly, although the Grid approach provides better results as expected, since the observation positions are almost fixed, which makes easier the learning process. This happens too with the regression outputs, where the loss error is clearly lower for the Grid approach. Regarding the position predictions, we can verify that the error value is lower than $0.02\ m^2$. Although this loss error is being computed with normalized values (and so

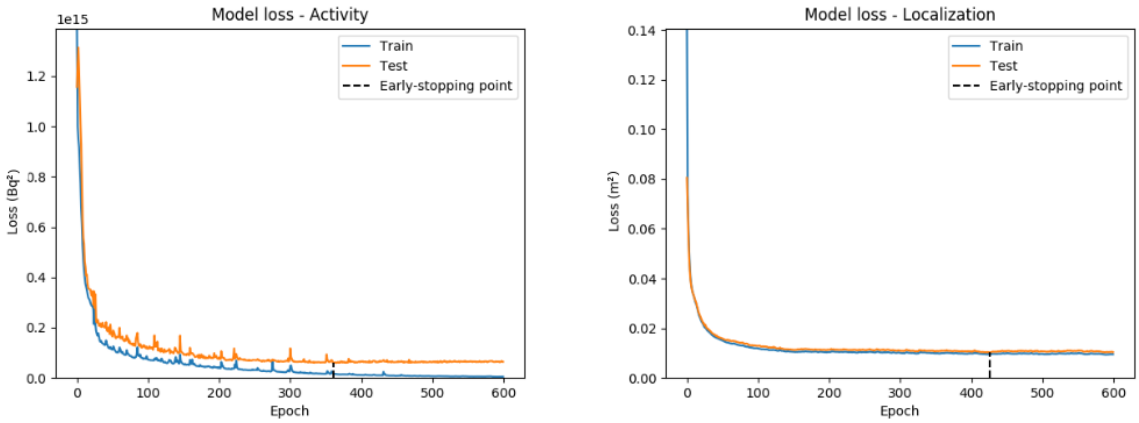


Figure 4.12: Loss error of regression outputs for the Grid approach, namely the activity (left) and the positions (x, y) (right) of the sources.

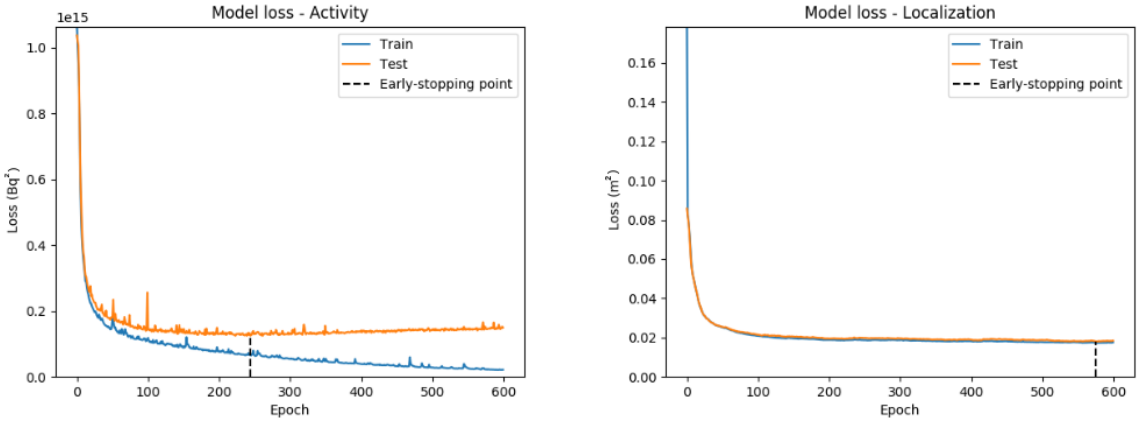


Figure 4.13: Loss error of regression outputs, using average pooling with the Random approach.

the actual MSE value would be higher), we can see that the model is able to get precise results, taking into account that the training data is formed by scenarios with dimensions of $3.33 \times 3.33\ m$, which are much larger when compared to the loss error.

Therefore, we can say that both approaches with random data provide similar results, although the average pooling may be more useful since the loss error of activity prediction may be significantly smaller, depending on the situations. Besides that, if there is a lot of data within each region that is associated to each input neurons, the maximum pooling approach extracts only the observation with the highest intensity, whereas the average pooling approach extracts intensities so smoothly, and it may not be able to extract good features, which may lead to bad results on the source detection problem. Concluding, it depends on the situation we are dealing with, namely the number of observations that can be acquired

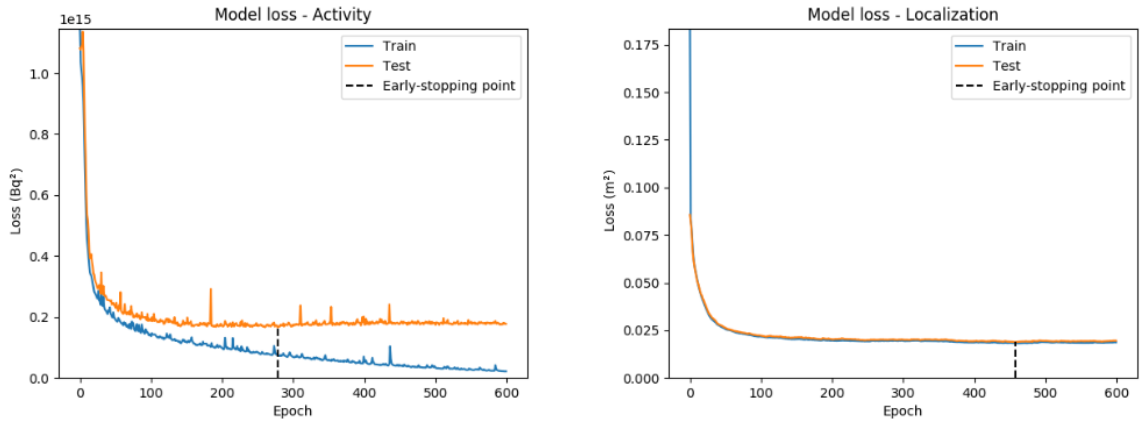


Figure 4.14: Loss error of regression outputs, using max pooling with the Random approach.

from different regions and the fluctuations in intensity verified in the observations.

Approach	Pooling	Acc (%)	$\mathcal{L}_a$ ($\times 10^{14}\ Bq^2$)	$\mathcal{L}_{x,y}$ ($\times 10^{-2}\ m^2$)
Grid	-	97.33	0.594	1.025
Random	Avg	95.38	1.247	1.802
	Max	95.14	1.655	1.879

Table 4.1: Comparison of the Early-stopping points regarding the different approaches during the training process.

To reproduce the way the model is learning throughout the epochs, in Fig. 4.15 it is depicted how predictions evolve during training with the Grid approach, using an example scenario taken from the validation data set. We can see that, firstly, the model is not able to predict exactly which boxes have a source, evolving until the model starts being capable of predicting some correct boxes and, in epoch 150 we already see the boxes selection with 100% of accuracy. Regarding the positions, the predictions of the model rapidly get close to the targets, as well as the activity predictions, although we can see that the latter is getting even closer to the targets when we pass from epoch 150 to epoch 350, as expected once it is around this epoch where the Early-stopping point is, according to Fig. 4.12 (left). To note that the blue points, and corresponding activity predictions (on the right side of the plots), that are on the white boxes must be neglected, as detailed before in Fig. 2.9.

4.3.1 Threshold study for Divide And Conquer Algorithm

As previously introduced in subsection 2.2.4, the DAC method is applied in such a way that only on the last division both classification and regression outputs are used, whereas in the previous divisions only the classification output is used. An important characteristic of the classification problem is the threshold value, i.e the value of probability (classification output) from which it means that there is one or at least

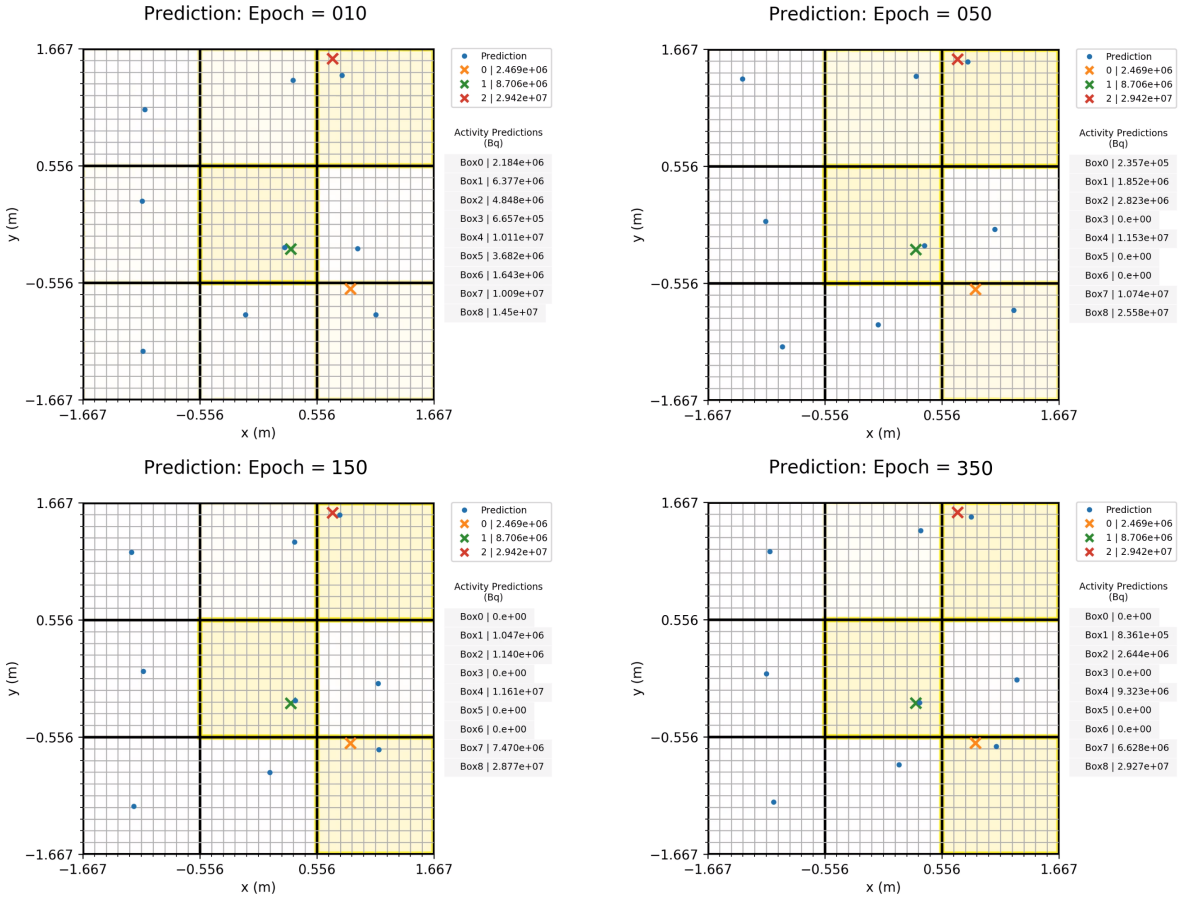


Figure 4.15: Evolution of predictions during the training process, with epochs 10, 50, 150 and 350 used as examples.

one source on the corresponding box. Although it is commonly assumed that the threshold should be 0.5, the choice of a threshold is a problem-dependent. Besides, we may assume different values of threshold depending on the division and the goals we have according to that division. Therefore, in order to analyze the impact of changing the threshold value, the quality metrics that regard the classification problem (detailed in section 2.3) are used with predictions from the test data set, as we can see in Figures 4.16 and 4.17.

In the first divisions, the goal is to have as few false negatives as possible so as not to have lost sources at the beginning of the DAC process, in addition to having a low FPR so that unnecessary boxes are not approached later. This way, a threshold value of 0.1 is a good starting value for ”Division 0”, where the accuracy and F1-score start to be greater than 80%, FNR remains almost zero and FPR goes to zero. On the other hand, in the last division, the goal is to obtain the best combination of accuracy and F1-score (which is also a combination of Recall and Precision) and with FPR practically zero. Therefore, a threshold value of 0.9 is selected for this step, since for higher values, accuracy and F1-score start decreasing considerably, while FNR starts reaching values above 10%. In intermediate

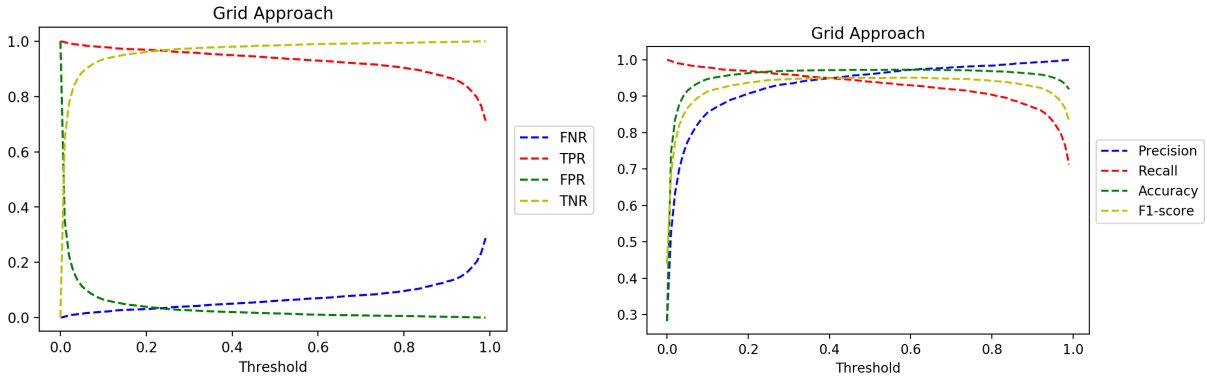


Figure 4.16: Threshold study for the Grid approach.

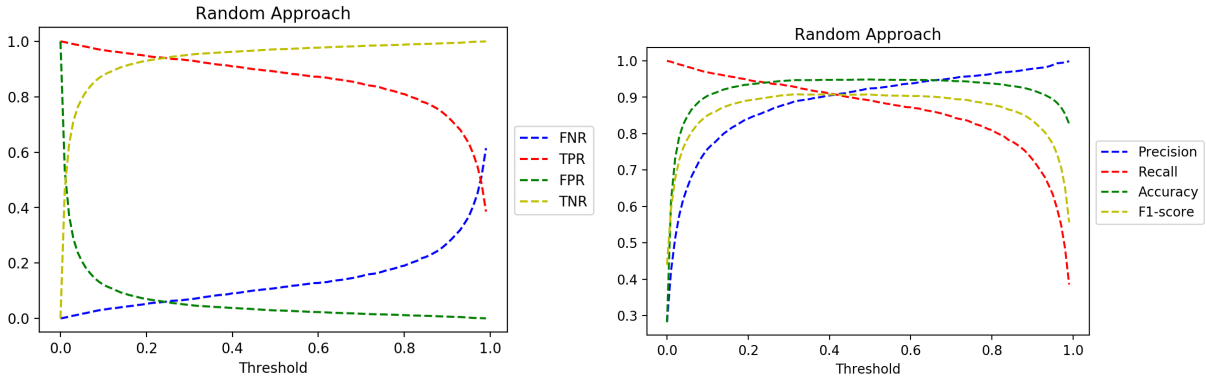


Figure 4.17: Threshold study for the Random approach, using the max pooling. For average pooling, the plots are quite similar.

divisions, the threshold value should increase from 0.1 to 0.9, being the best range the one in which the best results for accuracy and F1-score are obtained, and both FNR and FPR remain less than 10%. So, the $[0.5, 0.8]$ range is the choice for the Grid approach, whereas the $[0.4, 0.7]$ range is the one for the Random approach.

4.3.2 Considering Missing Observations

Use of Dropout Input Layer

When the input layer is considered as a dropout layer, it means that a given rate of random input neurons are being neglected, and so they will not be considered for the second layer. The problem of having regions of the map without any observation is more likely to happen when we collect data according to the Random approach. Therefore, in the next tables it is presented the evaluation metrics for the rates 0, 0.05, 0.10 and 0.15, using the average (Table 4.2) and the maximum pooling (Table 4.3). When higher dropout rates are tested, the results are not better and the performance gets worse as dropout increases.

Dropout Rate	Acc (%)	$\mathcal{L}_a$ ($\times 10^{14}$ Bq^2)	$\mathcal{L}_{x,y}$ ($\times 10^{-2}$ m^2)
0	95.38	1.247	1.802
0.05	94.97	1.333	1.883
0.10	94.64	1.284	2.019
0.15	94.55	1.547	2.047

Table 4.2: Training with dropout input layer and average pooling.

Dropout Rate	Acc (%)	$\mathcal{L}_a$ ($\times 10^{14}$ Bq^2)	$\mathcal{L}_{x,y}$ ($\times 10^{-2}$ m^2)
0	95.14	1.655	1.879
0.05	94.62	1.504	1.887
0.10	94.73	1.608	1.958
0.15	94.17	1.517	2.047

Table 4.3: Training with dropout input layer and max pooling.

Testing in scenarios with missing observations

Analyzing both tables 4.2 and 4.3 we can see that, as expected, the higher the rate the higher the loss error and the lower the accuracy. Even so, these variations may not be as relevant and, as follows, tests are made to assess the performance gain once more using scenarios with missing observations. To do this, from the test data set, a certain number of random input values is masked by imposing them to zero and apply the different models from above. It is important to recall (see section 2.2.1) that when we are running the model on the validation data set, all input neurons are being used (i.e., dropout rate becomes 0). For this reason, the accuracy values shown in previous tables are higher than the following ones. Besides that, now F1-score is also being computed (and accuracy again) by using the threshold value of 0.9, since the dimensions (3.33×3.33 m) are appropriate for the last division of the DAC process.

Then, taking into account that the tested models have $9 \times 9 = 91$ input neurons (once we are considering the Random approach), in table 4.4 it is evaluated the performance of the different models with average pooling (for max pooling, the results are quite similar), considering 0, 10, 15 and 20 random regions of the map (associated with the respective input neurons) without any observation. In order to help the visualization of these results, Fig. 4.18 shows the plots with the metrics used, as a function of the number of missing observations. Therefore, analyzing the accuracy and F1-score metrics, we see that, initially with all the observations, the dropout rate of 0 provides the best results, but when we have 10 missing observations the other dropout rates provide closer results. When we have 15 and 20 missing observations, the dropout rates of 0.05 and 0.10 produce the best results, whereas the dropout rate of 0 becomes the worst in the end. Regarding the activity and localization of the sources, which are evaluated through the MSE metric (with equations (2.14) and (2.15), respectively), we see that only with all observations, the dropout rate of 0 provides similar results compared to the remaining rates, whereas, for 10 or more missing observations, the results improve as the dropout rate increases. To note that, once MSE is computed for localization predictions after restoring these values (i.e., the normalization inverse process), the results are greater than what we have seen previously with $\mathcal{L}_{x,y}$.

# Missing Observations	Dropout	Acc (%)	F1-score (%)	MSE$_a$ ($\times 10^{14}$ Bq^2)	MSE$_{x,y}$ (m^2)
0	0	92.47	84.78	1.479	1.111
	0.05	90.81	80.72	1.610	1.105
	0.10	90.48	79.92	1.637	1.094
	0.15	89.91	78.50	1.643	1.095
10	0	91.22	83.39	2.144	1.152
	0.05	90.68	80.77	1.898	1.138
	0.10	90.18	79.29	1.795	1.125
	0.15	89.41	77.25	1.770	1.121
15	0	88.00	78.98	2.901	1.174
	0.05	90.60	80.92	2.317	1.155
	0.10	90.19	79.42	2.069	1.143
	0.15	89.46	77.39	2.042	1.131
20	0	82.66	72.62	3.534	1.192
	0.05	89.52	79.22	2.704	1.173
	0.10	89.87	78.97	2.361	1.161
	0.15	89.13	76.80	2.429	1.149

Table 4.4: Testing missing observations with average pooling. The results with max pooling are quite similar.

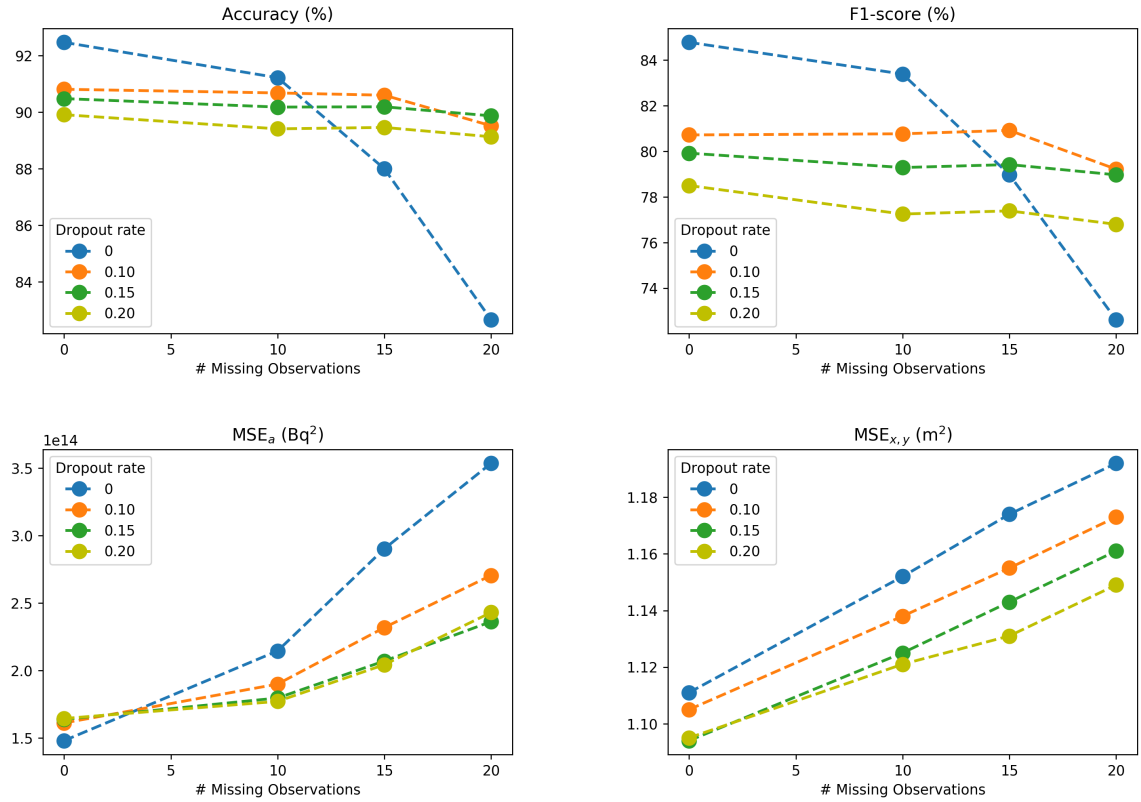


Figure 4.18: Plots to illustrate the influence of the dropout rate regarding the metrics Accuracy, F1-score, MSE$_a$ and MSE$_{x,y}$, as a function of the number of missing observations, using data from table 4.4.

5

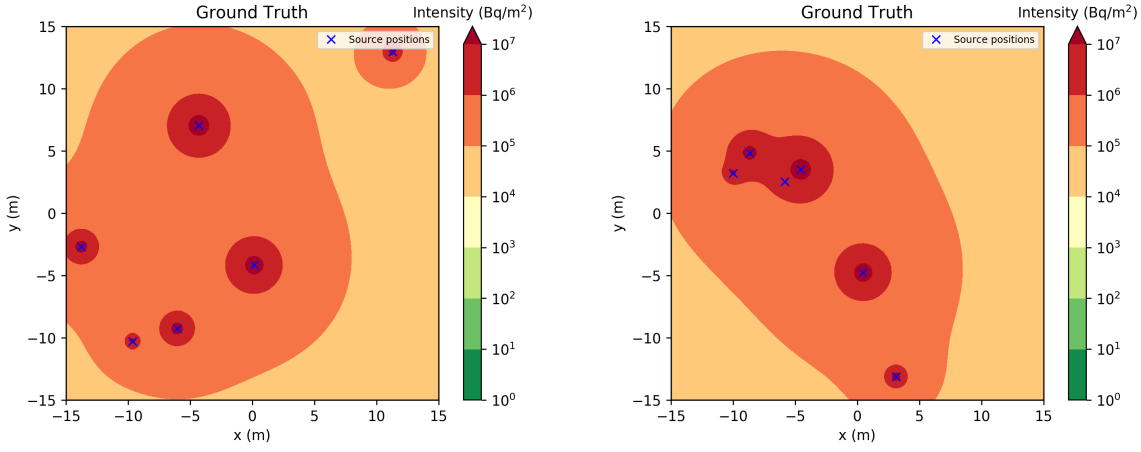
Results

Contents

In order to test the DAC algorithm with the pre-trained models, different simulated and real scenarios are considered. These models are tested with more test data acquired from scenarios created at Instituto Superior Técnico and real scenarios located at old deposits of radioactive ore for uranium production. Still, the simulator is also used as a helper in this section in order to simulate different scenarios with specific conditions that cannot be easily found in the previous real scenarios, regarding the number, activity and spatial distribution of the radioactive sources, as well as the complexity of the obstacles and the presence of non-point-like sources.

5.1 Simulation Results

Starting with the simulation results, two scenarios are approached, both with 6 sources but with different complexities, related to the proximity between the sources (which can influence the results). In Fig. 5.1, on the left there is a scenario (called **1.a)**) with 6 sources distributed throughout the map, and only one of the 3×3 boxes has two sources (on the lower left corner box - called box 0 according to Fig. 4.4). On the right, in the so-called scenario **1.b)**, we have a more complex scenario, since one of the boxes (the box number 3) has 3 sources, and one of which is weak (compared to the others) and very close to another stronger one (on the right box - number 4).

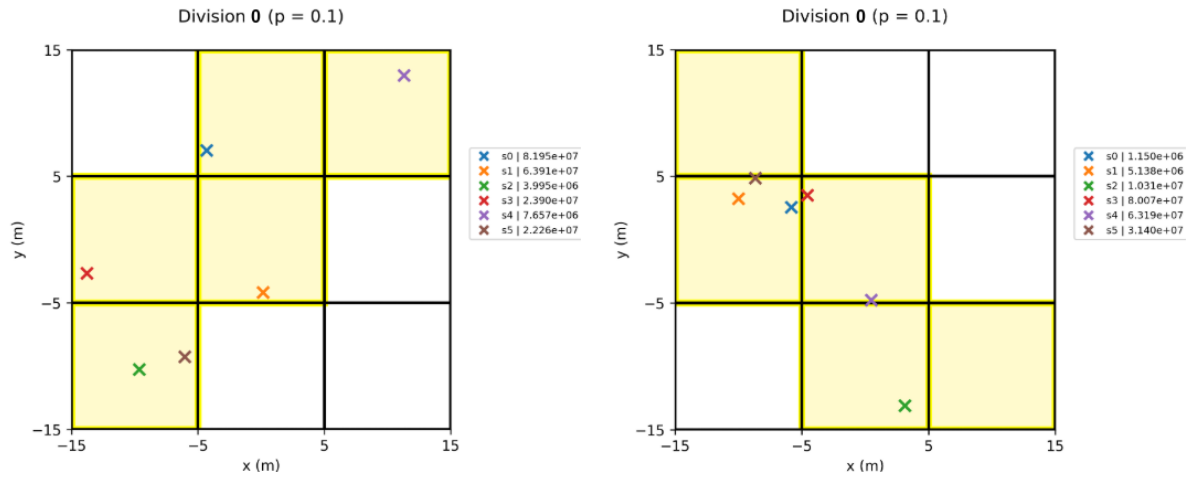


Scenario **1.a)** Scenario **1.b)**

Figure 5.1: Simulated scenarios with 6 sources.

As follows, in Fig. 5.2 it is depicted the "Division 0" of the DAC algorithm, with a threshold value of 0.1. We can see that, in scenario **1.a)**, all the boxes are well predicted (with 100% of accuracy). The same does not happen in scenario **1.b)**, where two false positives arise, that can be explained by the fact that there are two sources very close to these boxes and, besides that, a low threshold value is used in order to not have undetected sources on the beginning of the DAC process, as explained in section 4.3.1.

However, this is not a problem to be worried about, once, on the following divisions, the threshold gets higher and false positives are going to decrease until the last division, where the maximum threshold value is applied in such a way that the most false positives do not appear on the final result.



Scenario **1.a)** Scenario **1.b)**

Figure 5.2: Results of ”Division 0” of DAC on the simulated scenarios with 6 sources.

In these two cases, the DAC algorithm is applied with 2 divisions, since the initial dimensions are $30 \times 30\ m$ and, with this number of divisions, the last division (”Division 2”) has dimensions of $3.33 \times 3.33\ m$, which are the same used to build the training data set. It is important to note that, although this was done on purpose, the application of the pre-trained neural network model is scalable in scenarios with different conditions (as we will see forward), with very good results regarding the classification outputs. Regarding the regressions outputs, namely the estimations of the parameters (activity and location) of the sources, it is also scalable in scenarios with similar dimensions, as explained in section 3.4. However, if the DAC algorithm was applied with a last division with significantly different dimensions, then a new training process should be done with an appropriate training data set, as previously explained in section 4.3.

In relation to the scenario **1.a)**, in Fig. 5.3 the whole process of the DAC algorithm is represented, with a threshold value of 0.7 used in ”Division 1”. As expected, since all sources in these scenarios are spatially well-distributed, although there are false positives appearing in this division, during the last division of the corresponding boxes we have no yellow box, which means that those false positives turn out to be insignificant and will not appear in the ”combine” step of the DAC process, which is represented in Fig. 5.4. Then, we can conclude that all the 6 sources are detected, and also the predictions of the corresponding parameters of activity and position are both very accurate, as it can be verified through the relationship between the ’*o*’ (predictions) and the ’*x*’ (targets) symbols, as well as the similarity between colors (which represent the activity values). An important result that can be seen with this example is

related to the ability of the ANN model to recognize that there is no source in a given scenario, as it happens in 4 boxes of Fig. 5.3.

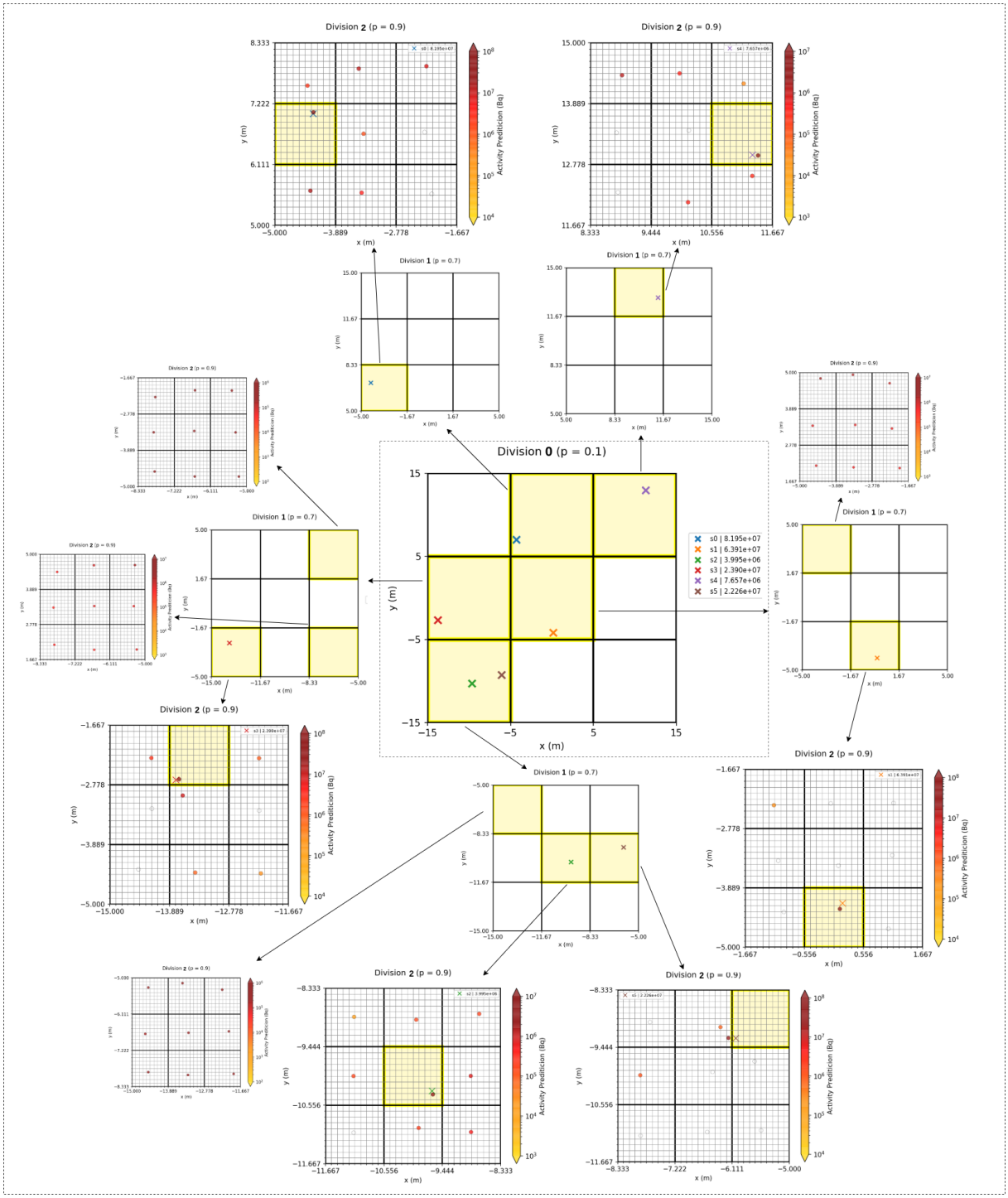


Figure 5.3: Representation of DAC applied on scenario **1.a)**. It was done with 2 divisions (plus the "Division 0"), applying the threshold values of 0.1, 0.7 and 0.9, by this order.

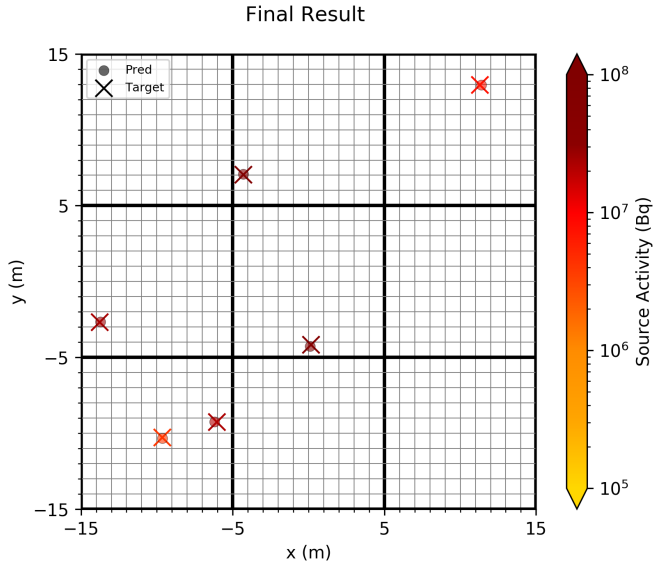


Figure 5.4: Final result after applying DAC on scenario **1.a)**.

Regarding the scenario **1.b)**, the final step of DAC is represented in Fig. 5.5, while the entire process is illustrated in Appendix B. Now, only 5 sources are detected and so, there is one missed detection. This may be related to the fact that this non-detected source is so weak that there is an overlay of the closest source's intensity, in such a way that the model is not able to identify it. Besides that, there are 5 false positives, where 4 of them are associated with target sources. This problem could be solved if we assume that, for a circumference with a given radius (relatively small) and centered in each prediction's position, only the prediction with the highest activity should be considered, while the other predictions that belong to that circumference would be neglected.

On the other hand, in practical terms, the "user" can also identify where to confirm the existence of hot-spots, in terms of location. Besides, the "user" must take into account that the predictions are independent of each other, i.e. the estimated activity is not distributed among the different predicted sources and, therefore, the real value does not decrease.

Testing a complex scenario with a high number of sources

One of the main problems of this thesis is related to the fact that it is not known how many sources the ANNs are able to detect. During the training process, it is used a maximum number of 4 sources, which is already a high number taking into account the short dimensions of the scenario and the activities of the sources. This is, for a scenario with dimensions of $30 \times 30\ m$, having 4 sources per $3.33 \times 3.33\ m$ of area, one could think that it would be possible to detect almost $4 \times (30 \times 30)/(3.33 \times 3.33) \simeq 325$ sources, which is unreal and almost impossible. Therefore, in order to test this ability of the model, follows a scenario , called **1.c)**, with a high number of sources, namely 16, as represented in Fig. 5.6.

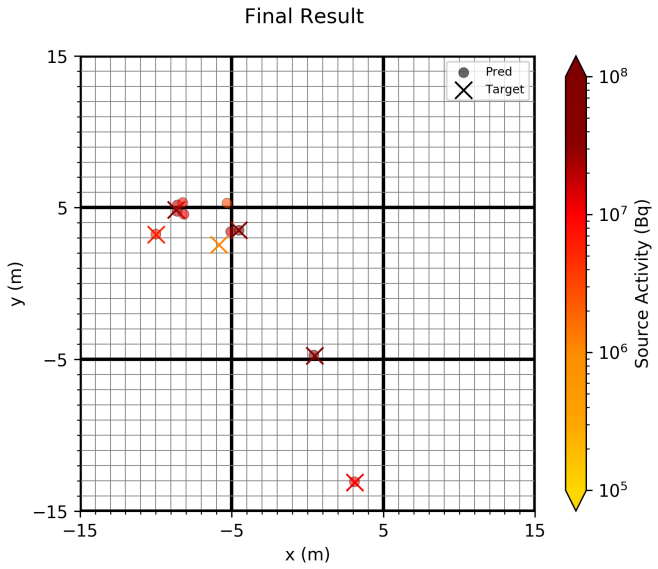


Figure 5.5: Final result after applying DAC on scenario 1.b).

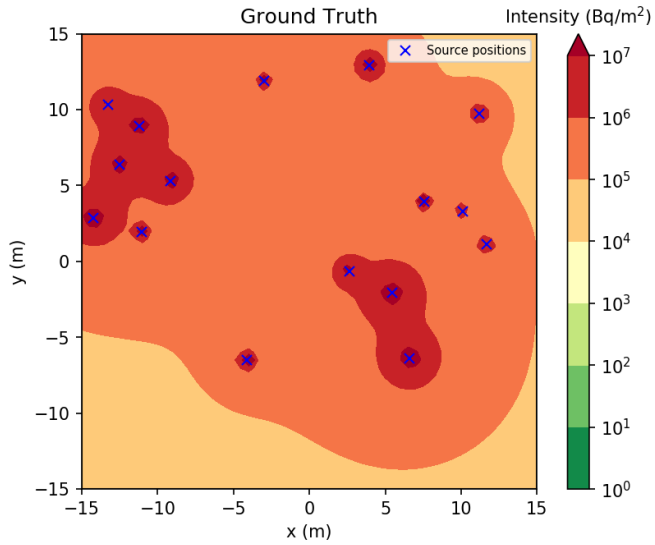


Figure 5.6: Simulated scenario **1.c)** with 16 sources.

Analyzing the final step of DAC, which is illustrated in Fig. 5.7, we can see that all the 16 sources are detected, even with sources so close to each other and some of them much weaker than others. Regarding the parameters of the detected sources, we can verify that the error of the estimated positions is very low (mainly when compared with the scenario dimensions), as well as the estimated activities, where their colors are quite similar to the target colors. The only negative point is the appearance of false positives, which can be easily solved *a posteriori* in practical terms by the ”user”, as previously explained.

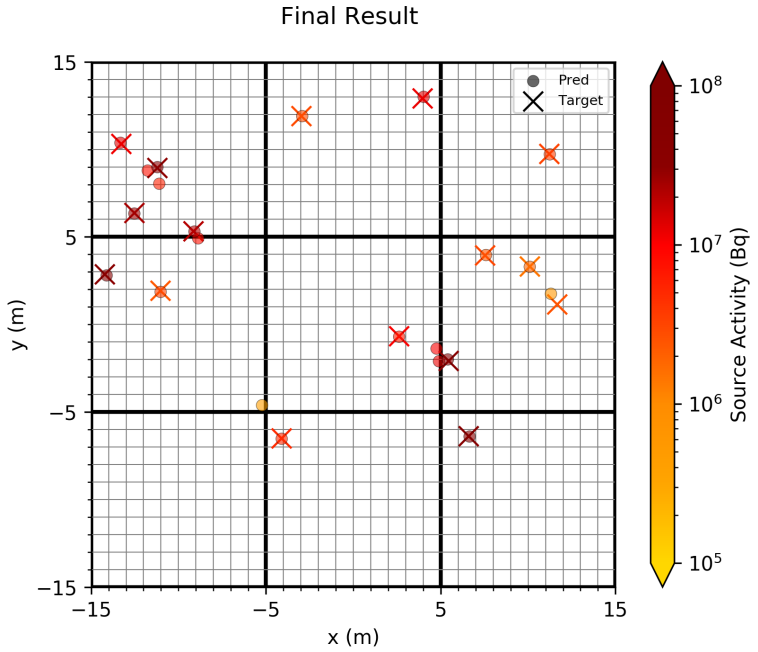


Figure 5.7: Final result after applying DAC on the scenario **1.c)** with 16 sources.

5.1.1 Considering the presence of Obstacles

In order to test the presence of obstacles in a scenario, two obstacles were added to the previous scenario **1.b)** (see Fig. 5.1 (right)). Recall that there was not a new training process, and the weights of the model are still the same of the ones used on the previous tests. The configuration and the position of the obstacles are represented in Fig. 5.8, where the one with an L-shaped configuration was set to be located between the non-detected source (regarding the test already done with the scenario **1.b)**) and the other closest source. These obstacles have an attenuation coefficient (see equation (2.3)) of 18.4 m^{-1}, which is enough to allow part of the radiation from the sources to pass through the obstacles.

In Fig. 5.9 it is represented the final result for the scenario **1.d)** with obstacles (with the entire process in Appendix B). It is possible to see that all the 6 sources are now detected, including the one that was not detected without the obstacles. This also means that, even with a more complex environment that was not part of the training data set at all, the neural network model is capable of detecting the sources, presenting even better results in relation to the model's accuracy. This may be related to the shielding created by the obstacles, which reduces the interference between two or more sources, and leads to better observations for the individual sources. Regarding the disadvantages, we see that, although there are less false positives, new false positives may occur near to the position of the obstacle(s), due to the fact that there may be gaps of intensity (or isolated points with high intensity) that lead to wrong interpretations by the neural network. Still, as we can see, this false positive has a very low activity (less

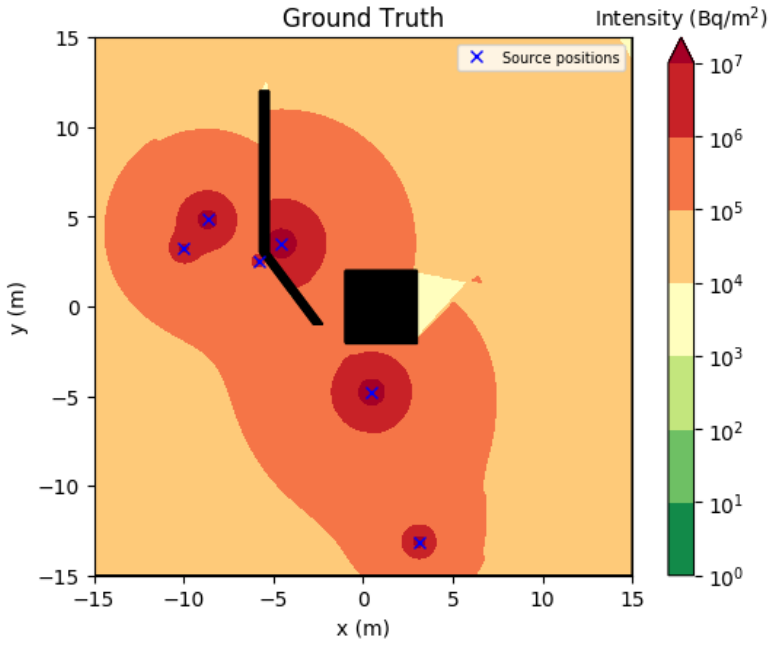


Figure 5.8: Simulated scenario **1.d)** with obstacles (represented in black color).

than $10^5\ Bq$). Besides that, the localization or activity errors may also increase, although it does not represent a significant difference.

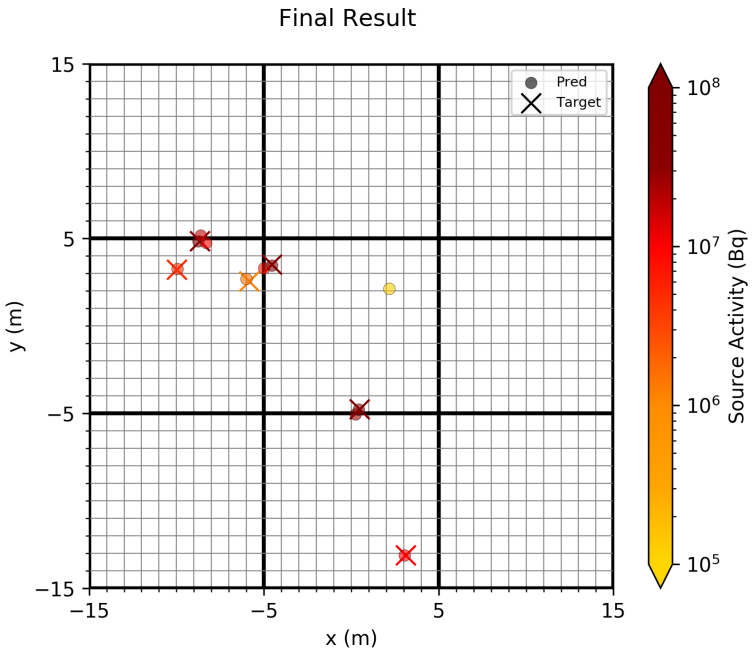


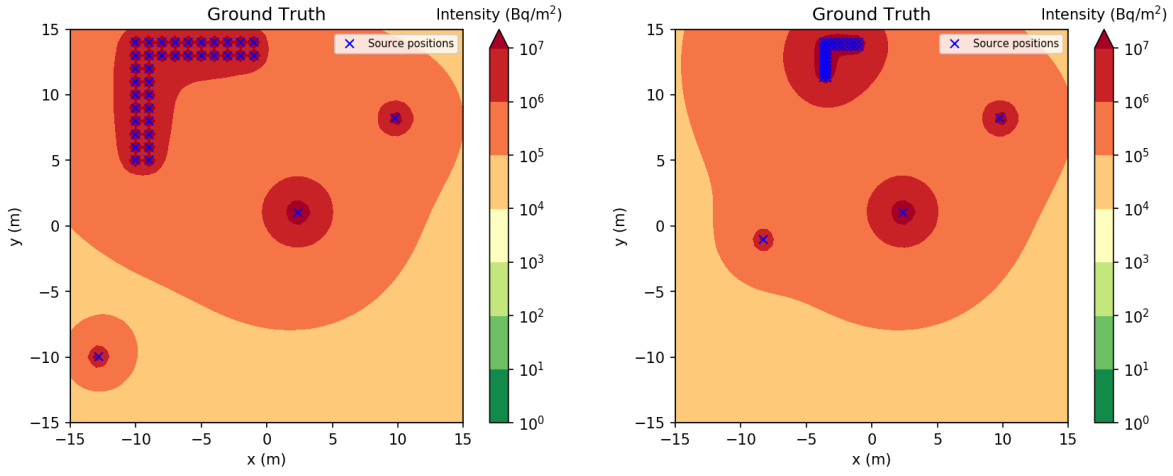
Figure 5.9: Final step of DAC on the scenario **1.d)** with obstacles.

Therefore, it can be concluded that the presence of obstacles does not provide significant negative

effects, and may even provide good effects (namely by reducing false negatives and false positives). Also, unlike most of the state-of-the-art approaches, this algorithm does not require previous information about obstacles. As previously mentioned in section 1.2.1, the work [41] does not require previous knowledge about the obstacles either, being able to provide similar results, namely the decreases of false negatives and false positives when obstacles are present. However, although it is also capable of providing fast results, the algorithm is dependent on a network of detectors to get measurements at multiple places simultaneously. This way, it is unsuitable for experiments with UAV, which can be a considerable limitation for real-time operations, for instance.

5.1.2 Non-point-like sources Approach

As already seen in section 3.1, to test the non-point-like source approach, the simulator is used in such a way that a non-point-like source is actually an arrange of multiple sources in a certain configuration. In Fig. 5.10 two different scenarios are presented. On the left, with scenario **1.e)**, there are 36 sources in a L-shaped configuration with activity values of $5 \times 10^6\ Bq$ and with a distance of $10\ cm$ from each other. On the right, with scenario **1.f)**, we have the same number of sources with the same values of activity, but the distance between each other is just $3\ cm$. In addition, in both scenarios, three other regular (point-like) sources were added to see if the non-point-like source has any impact on their detections. This way, it is possible to see how the neural network behaves with this kind of environments, namely how many sources are predicted, as well as the estimated localization and the total activity.

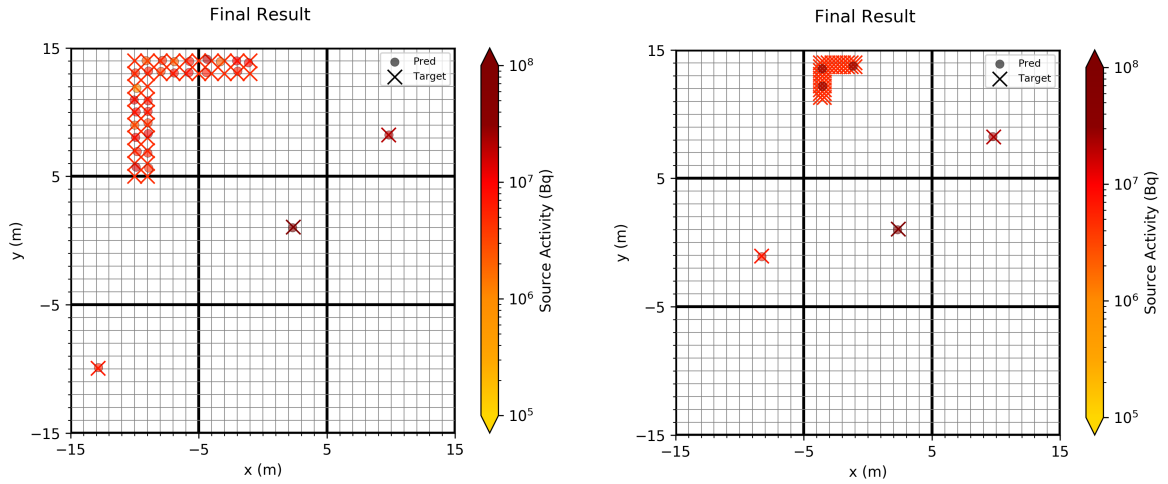


Scenario **1.e)** Scenario **1.f)**

Figure 5.10: Simulated scenarios with 1 non-point-like source and 3 point-like sources.

Then, in Fig. 5.11 it is represented the final results of DAC when applied in scenarios **1.e)** and **1.f)**, with all previous steps presented in Appendix B. In both cases, we see that the point-like sources are

detected with good predictions in relation to the corresponding parameters, which means that there is not any negative impact. Regarding the detection of the non-point-like source, in scenario **1.e)** we can verify that the number of detected sources corresponds almost to the total number of actual sources (28 of 36), where some of the predictions are located in the middle of two targets, and the activity values are close to the actual value ($5 \times 10^6\ Bq$). On the other hand, in scenario **1.f)**, there is only 3 predictions, which is closer to what was desirable (i.e., only one prediction), located in points with almost the same distance between each other, and activity values within the range $[8, 9] \times 10^7\ Bq$. This way, having 3 predictions close to each other, with strong and similar activities, the "user" can connect these 3 predicted point-like sources and thus obtain a non-point-like source with an activity of $\sim 9 \times 10^7\ Bq$.



Scenario **1.e)** Scenario **1.f)**

Figure 5.11: Final result of DAC algorithm, applied on scenarios with non-point-like sources included.

5.2 Lab Results

At Instituto Superior Técnico, some lab tests were carried out. The three available sources allowed to test the algorithm with 1, 2 and 3 real sources. These sources were much weaker than those used before in training and simulation results, in such a way that instead of a $[10^6, 10^8]\ Bq$ range of source activities, now we have $[10^2, 10^3]\ Bq$ approximately. This way, only measurements acquired very close to the sources might be relevant and, thus, a new training process was performed so that this range of activities could be considered within a scenario with dimensions of $1 \times 1\ m$, and using the Grid approach with 11×11 inputs. In Fig. 5.12 it is presented the lab setup, and the representation of a scenario with two sources.

In Fig. 5.13 it is depicted two of the several tests performed, which can represent the good results in

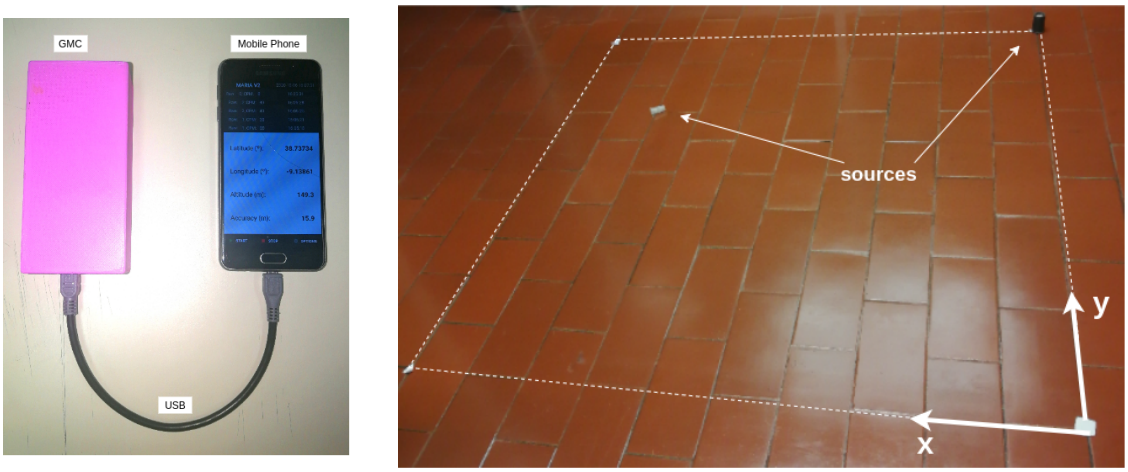
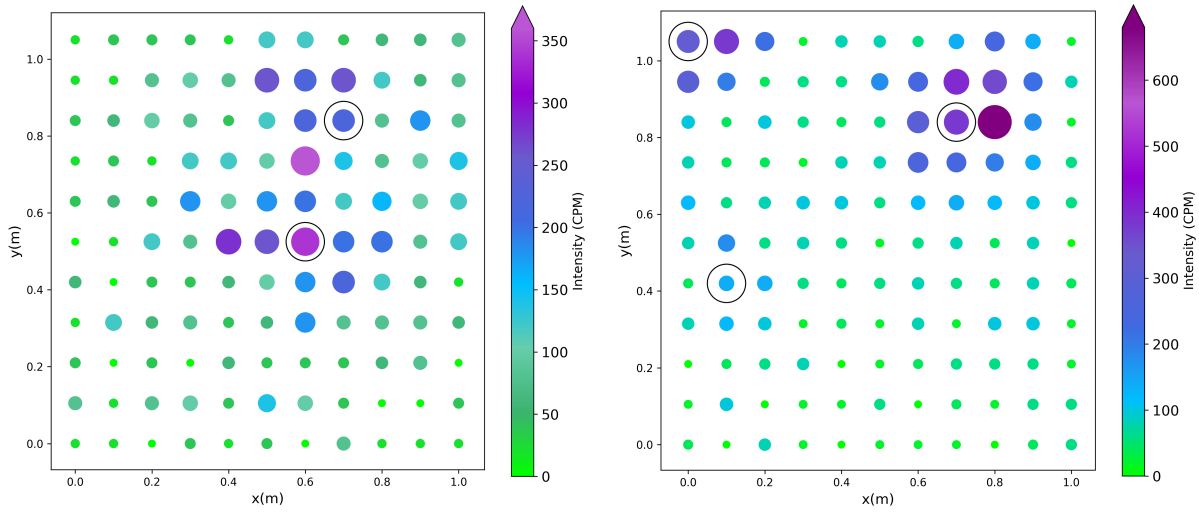


Figure 5.12: Lab setup tested at Instituto Superior Técnico. GMC sensor, connected to a mobile phone running the MARIA application (left). Scenario with dimensions of $1 \times 1.05\ m$, and well-discretized fixed points (with $\Delta x = 0.10\ m$ and $\Delta y = 0.105\ m$) to get the intensity measurements (right).

general, as well as the problems that may arise. At first sight, we can visualize some isolated points with intensity of $\sim 150\ Bq$ that are at a considerable distance from the sources. This is related to the fact that the sources are covered by lead (to protect us from radiation) and, once this protection is not entirely homogeneous, the sources are not as isotropic as desirable and, in the end, this leads to isolated points that do not seem to make sense.

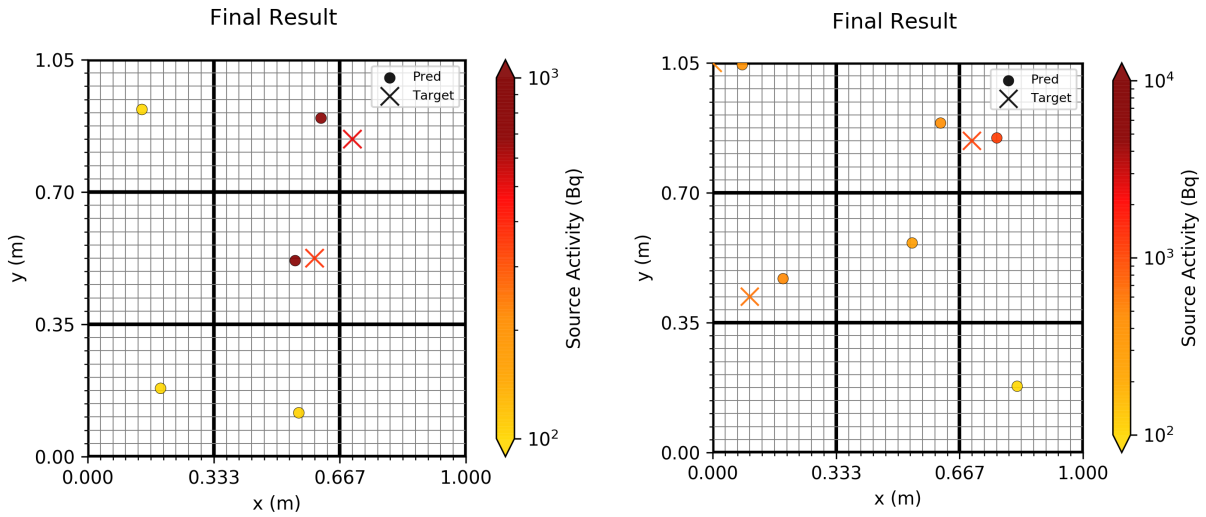


Lab scenario **2.a)** Lab scenario **2.b)**

Figure 5.13: Data from lab scenarios, with 2 and 3 sources on scenarios **2.a)** and **2.b)**, respectively. The positions of the sources are represented with a circle, and each observation has a size proportional to its intensity.

Since these scenario dimensions are already too small, more than 11×11 data Grid points would not bring any advantage. Therefore, the DAC algorithm is applied only in "Division 0" (without further divisions), which means that the ANN model is used only once to get the final predictions, as illustrated

in Fig. 5.14.



Lab scenario **2.a)** Lab scenario **2.b)**

Figure 5.14: Final lab results.

Analyzing both results from Fig. 5.14, we can see the problems associated with the isolated data points already mentioned: false positives may appear, although most of them with very low estimated activity. Regarding the true positives (see Table 5.1), in the scenario **2.a)**, there are predictions very close to the targets but with estimated activities relatively higher than expected. This may be related to the fact that these two sources are separated from each other at a short distance, leading to overlapping intensities, as can be seen in Fig. 5.13 (left) the high values of intensity between them. About the scenario **2.b)**, good results are obtained, since both estimations of the position and activity of the sources are very close to the targets.

Lab scenario	Target			Prediction		
	$a_s\ (Bq)$	$x_s\ (m)$	$y_s\ (m)$	$\hat{a}_s\ (Bq)$	$\hat{x}_s\ (m)$	$\hat{y}_s\ (m)$
2.a)	300-600	0.60	0.53	814	0.55	0.52
	500	0.70	0.84	730	0.62	0.89
2.b)	300-600	0.10	0.42	491	0.19	0.47
	500	0.00	1.05	409	0.08	1.05
	1100	0.70	0.84	1127	0.77	0.85

Table 5.1: Lab results, regarding the true positives. To note that the source with activity of $300 - 600\ Bq$ did not have a stable value and, for that reason, a range of values is considered.

Thus, although these scenarios are much smaller than the previous ones, it is possible to conclude that the application of a neural network model to detect radioactive hot-spots is scalable, in such a way that it can be applied both in scenarios of 1 square meter as well as in scenarios with an area almost 1000 times larger.

5.3 Real Results

Finally, in this section two different real scenarios are approached, with dimensions that are very larger than the ones considered in the previous subsections, and represent a real context where the proposed solution of this thesis is intended to be used. Figures 5.15 and 5.16 show the considered scenarios **3.a)** and **3.b)**, respectively, with the radiological heatmap provided by MARIA application, as well as the plot of all acquired data in the corresponding scenarios.

In both scenarios the ANN model is used with the Random approach and, although there are several regions without any observation, we have already seen (in section 4.3.2) that the ANNs are perfectly capable of getting good predictions even with null inputs. Regarding the application of the DAC algorithm, while the scenario **3.a)** has enough data for two divisions, the scenario **3.b)** allows only one division. With this number of divisions, we get a last division in both cases with dimensions of approximately 20×20 m, and the range of activity values of the sources is about $[10^2, 10^4]$ Bq. Then, a new training process is performed according to the previous conditions, and using the Random approach with max pooling, since there are regions of the map with a low number of observations, and so only the observations with the highest intensity are considered.

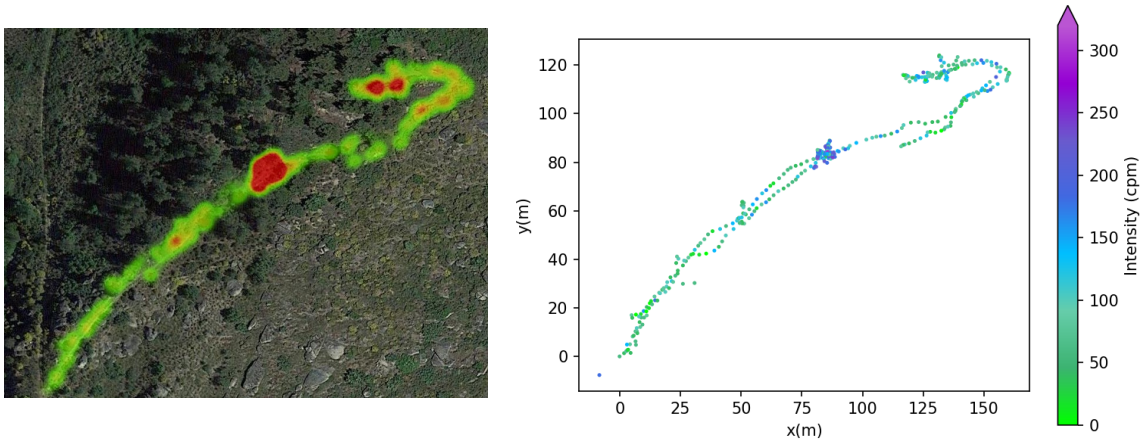


Figure 5.15: Data from real scenario **3.a)**. On the left it is presented the radiological heatmap provided by MARIA application. On the right, a plot with all acquired data is depicted.

Then, after applying the DAC algorithm (with all steps illustrated in Appendix B), Figures 5.17 and 5.18 show the final results of both scenarios. Since the activity and position of the targets are unknown, instead of visualizing these parameters, a plot of all observational data with an intensity higher than 100 CPM is done, so the location of possible hot-spots can be seen. Thus, about the real scenario **3.a)**, we see in Fig. 5.17 that the three main hot-spots seen in Fig. 5.15 (left) are detected and, besides that, there are also a few sources identified in other points of the map. For instance, on the bottom-left corner, there are 4 detected sources, and this is related to the fact that there are some isolated observational points that have enough intensity to make the model interpret that there may be a source there. It is

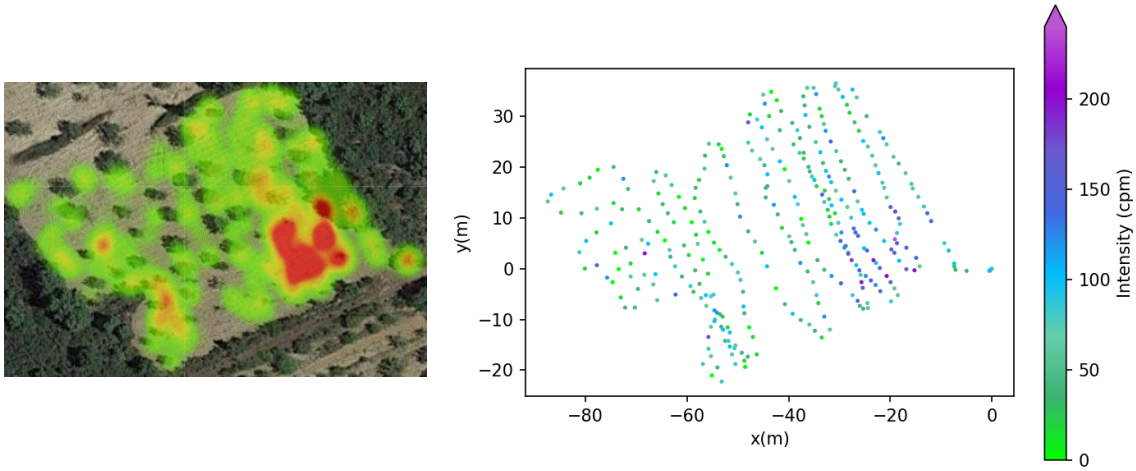


Figure 5.16: Data from real scenario **3.b)**. On the left, the radiological heatmap provided by MARIA application is shown. On the right, a plot with all acquired data is depicted.

important to note that the prediction with the highest activity is the one associated with the strongest hot-spot, which represents a good result.

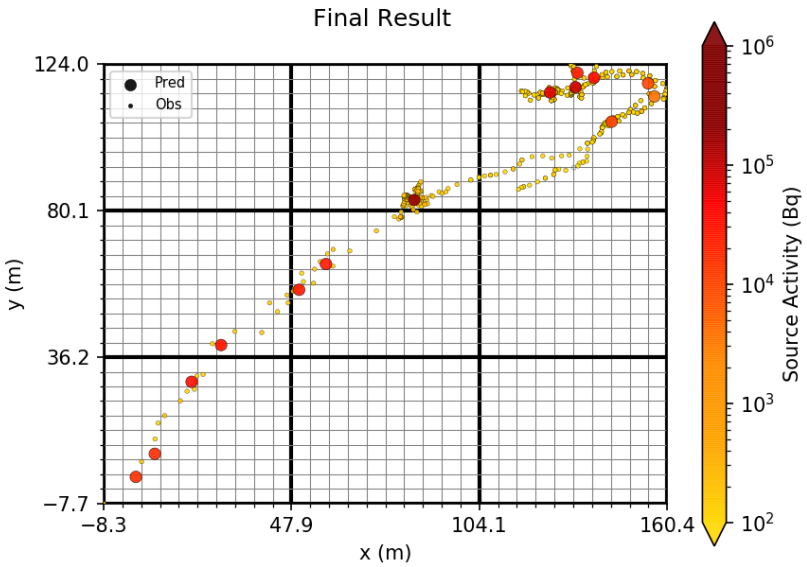


Figure 5.17: Final results after applying DAC on the real scenario **3.a)**. The smallest points represent the observations with intensities higher than 100 CPM.

On the other hand, regarding the results of scenario **3.b)** represented in Fig. 5.18, the strongest hot-spot seen in Fig. 5.16 (left) concerns 5 detected sources and, in addition, there are other predictions throughout the entire map that make sense, taking into account nearby observational data with high

intensities. In fact, it seems that where acquisitions (with intensities higher than 100 CPM) were made, there is, at least, one estimation, which is once more related to the fact that some of them are isolated points.

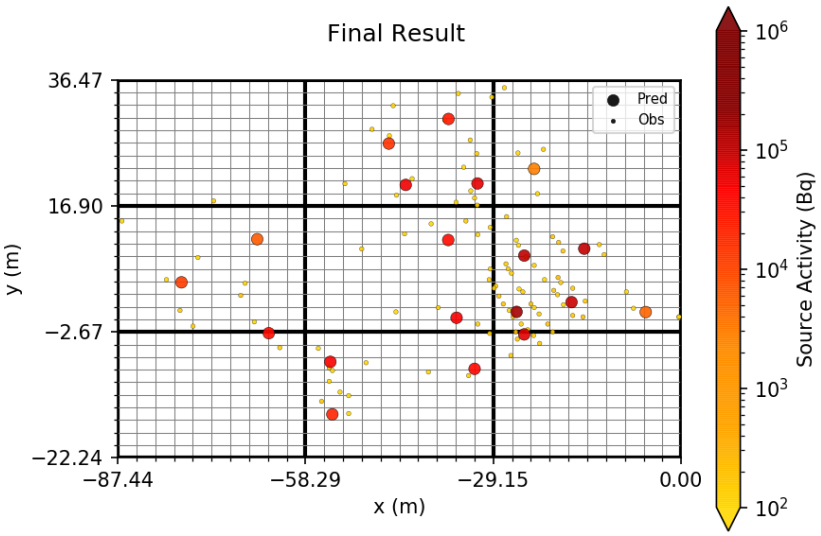


Figure 5.18: Final results after applying DAC on the real scenario **3.b)**. The smallest points represent the observations with intensities higher than 100 CPM.

6

Conclusions

Contents

Radioactivity is a field that can bring a lot of benefits for the human being, namely in science, medicine and industry areas. However, it also involves many issues and dangers, related to the CBRN threats and the NORMs, which can arise from anywhere (although there are potential locations). Therefore, the detection of dirty bombs in public places, the autonomous inspection of nuclear sites and the remote control of places, like the one that followed the Chernobyl disaster in 1986, are examples of applications that have been developed in this field in the past decade. Despite the success of some state-of-the-art algorithms, in detecting a single or a small number of radioactive point sources, most of them start failing as the scenarios get more complex. A high number of sources, the presence of obstacles, and the existence of sources with non-trivial shapes are some of the main limitations. In addition, even in simpler scenarios, when larger or shorter dimensions are considered, or when the data acquisition approach is different from the one considered, most existing works are not capable of keeping providing good results.

For this reason, in this dissertation, a solution based on Artificial Neural Networks (ANNs) was proposed in order to estimate not only the distribution and the positions of the sources, but also the number and the respective activities of each of them. Besides, this proposed solution was tested in complex environments, involving all the previous limitations.

6.1 Work Accomplished

The results of the present work show that the usage of ANNs provides good results in detecting radioactive hot-spots in simple and more complex scenarios, where the most existing algorithms present some limitations. The main downsides are related to the long time required for training, the computational costs (i.e., the software and hardware needed to train and obtain the ANN model), and the use of a training data set composed only of simulated data, which may lead to the possibility of overfitting. Nevertheless, having the ANN model already trained, it can be used through the DAC algorithm in order to obtain fast and accurate predictions, which can be very useful in real-time operations (e.g. for a real surveillance).

In the simulation, lab and real results, the dimensions of the various tested scenarios had variations from $1\ m^2$ to more than $10^4\ m^2$. This was possible due to the usage of the DAC method, where the number of divisions may be adjusted according to the scenario dimensions (width and length) and the quantity and quality of the observations. Therefore, this shows that the model is scalable to very large regions, essentially with regard to the detection of parts where at least one source is present. Regarding the activity and the exact location of the sources, although these predictions also provide good results, the training process needs to be adapted to the conditions of the scenario, namely the range of activities of the sources, the background intensity, and the dimensions of the last division of the DAC algorithm.

Also, about the way the necessary input data is acquired, it was demonstrated that both the grid approach and the random approach provide good results. Besides that, there is no need to have all ANN model inputs filled in with observational data, since it has been shown that it is possible to handle missing observations by extrapolating the information from empty inputs based only on those available, so that accurate results are still obtained.

Regarding the more complex environments, it can be concluded that the proposed solution in this thesis is capable of leading with a high number of sources, without any limitation. The only limitation is related to the proximity between the sources, which is more evident in cases where the difference between these sources is considerably high, and then can lead to missed detections (or false negatives). The presence of obstacles was also tested and, even without any obstacles included in the training data set and without requiring any prior information about them, the algorithm is capable of obtaining good results and even better in some cases, by reducing the number of missed detections. The only disadvantage is related to the fact the new false positives (although with low activity estimated) may arise close to the position of the obstacles. Lastly, scenarios were tested with non-point-like sources, also with none included in the training data set. Since these types of sources were generated by arranging multiple sources in a certain configuration, the performance of the model was dependent on the distance between each of these sources. This is, the shorter the distance, the better the results provided. In this context, a good result means that a low number of detected sources was associated with a non-point-like (target) source, since the training data set is composed only of point-like sources, and so it is not possible that the model can predict the exact shape of such target source.

In addition, through the lab and real results, it was seen that it is possible to use the MARIA application with an ANN model, since observational data acquired by MARIA can be used as input of the ANNs, thus obtaining the final predictions. Besides, regarding the real results, the model is able to detect the hot-spots (targets) that can also be seen through the radiological heatmaps provided by MARIA.

Thus, we can affirm that Machine Learning, namely the Deep Learning sub-field, has indeed conditions to make such a difference on the nuclear problem of radioactive hot-spots detection, taking into account the reproducibility of the presented model in the most varied scenarios, as well as the expected improvements in computing time and in Machine Learning frameworks with developments in both software and hardware.

6.2 Future Work

During the development of this work, several ideas and possibilities were emerging in order to achieve a more complete approach, as well as an even better approximation to reality.

The most pertinent idea for future work is probably to adapt the model to a 3D environment, by

considering the altitude of the observations as an input instead of assuming that it is an approximately constant parameter, once the simulator used is already prepared for radiation propagation in 3D. Then, another opportune idea is to expand the training data set with real data, instead of using only simulated data and using real scenarios just to test the model. Although this was not possible due to time and logistic problems, as well as the implications related to the COVID-19 pandemic, new radiological data sets will be available from future experiments performed in real scenarios under the FRIENDS project [87], and therefore, getting a hybrid training data set (i.e., with both simulated and real data) will be a possibility.

Additionally, another idea is performing a quantitative and qualitative comparison between the usage of ANNs and other algorithms whose the goal is also the detection of radioactive hot-spots. This way, it will be possible to explore and verify if the advantages of the ANNs seen throughout this thesis outweigh their disadvantages (such as the computational costs, the time required for training and the training sets required).

Bibliography

[1] E. Wacholder, E. Elias, Y. Merlis, "Artificial neural networks optimization method for radioactive source localization," *Nuclear Technology*, vol. 110, no. 2, pp. 228–237, 1995. [Online]. Available: https://doi.org/10.13182/NT95-A35120

[2] "Neural networks part 1: Setting up the architecture," https://cs231n.github.io/neural-networks-1/, accessed: 2020-09-15.

[3] "Machine learning faq," https://sebastianraschka.com/faq/docs/closed-form-vs-gd.html, accessed: 2020-08-15.

[4] Y. REN and G. BAI, "New neural network response surface methods for reliability analysis," *Chinese Journal of Aeronautics*, vol. 24, pp. 25–31, 02 2011.

[5] N. Srivastava, G. Hinton *et al.*, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[6] R. Yamashita, M. Nishio, R. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into Imaging*, vol. 9, 06 2018.

[7] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015. [Online]. Available: http://arxiv.org/abs/1506.02640

[8] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *CoRR*, vol. abs/1612.08242, 2016. [Online]. Available: http://arxiv.org/abs/1612.08242

[9] "Divide and conquer algorithms," https://www.khanacademy.org/computing/computer-science/algorithms/merge-sort/a/divide-and-conquer-algorithms, accessed: 2020-09-15.

[10] Y. Brouwer, A. Vale, B. Gonçalves, and H. Fernandes, "Radioactive hot-spot detection using unmanned aerial vehicle surveillance," *EPJ Web of Conferences*, vol. 225, p. 06005, 01 2020. [Online]. Available: https://doi.org/10.1051/epjconf/202022506005

[11] “Student notes: Convolutional neural networks (cnn) introduction,” https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/, accessed: 2020-09-08.

[12] “Maria,” https://www.ipfn.tecnico.ulisboa.pt/MARIA/, accessed: 2019-12-17.

[13] M. Stabin and D. Broga, “Radiation protection and dosimetry: An introduction to health physics,” *Medical Physics*, vol. 35, p. 4766, 10 2008.

[14] N. Zakariya and M.-T. Kahn, “Review article benefits and biological effects of ionizing radiation,” 01 2014.

[15] “Cbrn risk mitigation and security governance programme,” http://www.unicri.it/topics/cbrn/, accessed: 2019-12-10.

[16] A. Ramseger, M. Kalinowski, and L. Weib, “Cbrn threats and the economic analysis of terrorism,” *DIW Berlin, German Institute for Economic Research, Economics of Security Working Paper Series*, 01 2009.

[17] “Reassessing cbrn threats in a changing global environment,” https://www.sipri.org/publications/2019/other-publications/reassessing-cbrn-threats-changing-global-environment, accessed: 2019-12-10.

[18] “Protection against cbrn threats in turkey,” https://www.indracompany.com/en/protection-cbrn-threats-turkey, accessed: 2019-12-11.

[19] “International atomic energy agency. naturally occurring radioactive material,” https://www.iaea.org/topics/radiation-safety-norm, accessed: 2019-11-22.

[20] M. Ojovan and W. Lee, “4 - naturally occurring radionuclides,” in *An Introduction to Nuclear Waste Immobilisation (Second Edition)*, second edition ed., M. Ojovan and W. Lee, Eds. Oxford: Elsevier, 2014, pp. 31 – 39.

[21] J. Ring, “Radiation risks and dirty bombs,” *Health physics*, vol. 86, pp. S42–7, 03 2004.

[22] H. S. Cho and T. H. Woo, “Mechanical analysis of flying robot for nuclear safety and security control by radiological monitoring,” *Annals of Nuclear Energy*, vol. 94, pp. 138 – 143, 2016.

[23] B. Li, Z. Wang, C. Li, Z.-R. Peng, and L. Ge, “Use of multi-rotor unmanned aerial vehicles for radioactive source search,” *Remote Sensing*, vol. 10, 05 2018.

[24] K. Oyama, H. Wakabayashi *et al.*, “Integration of 3d trajectory maps into a local distribution map of radiation dose using unmanned aerial vehicle,” *2012 IEEE 36th Annual Computer Software and Applications Conference Workshops*, p. 607–612, July 2012.

[25] P. Martin, S. Kwong *et al.*, "3d unmanned aerial vehicle radiation mapping for assessing contaminant distribution and mobility," *International Journal of Applied Earth Observation and Geoinformation*, vol. 52, p. 12–19, 2016.

[26] C. Q. Wu, M. L. Berry, K. M. Grieme, S. Sen, N. S. V. Rao, R. R. Brooks, and G. Cordone, "Network detection of radiation sources using localization-based approaches," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2308–2320, 2019.

[27] N. S. V. Rao, M. Shankar, J. Chin, D. K. Y. Yau, C. Y. T. Ma, Y. Yang, J. C. Hou, X. Xu, and Sartaj Sahni, "Localization under random measurements with application to radiation sources," in *2008 11th International Conference on Information Fusion*, 2008, pp. 1–8.

[28] M. Morelande , B. Ristic , A. Gunatilaka, "Detection and parameter estimation of multiple radioactive sources," *10th International Conference on Information Fusion*, pp. 1–7, 9-12 July 2007.

[29] ——, "Radiological source detection and localisation using bayesian techniques," *IEEE Transactions on Signal Processing*, vol. 57, pp. 4220 – 4231, Nov. 2009.

[30] H. E. Baidoo-Williams, "Maximum likelihood localization of radiation sources with unknown source intensity," 2016.

[31] Xiaohong Sheng and Yu-Hen Hu, "Maximum likelihood multiple-source localization using acoustic energy measurements with wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 53, no. 1, pp. 44–53, 2005.

[32] P. Kumar, g. Ssundaram, and R. Thiruvengadathan, "Advances in detection algorithms for radiation monitoring," *Journal of Environmental Radioactivity*, vol. 217, p. 106216, 06 2020.

[33] M. Gomez Fernandez, K. Higley, A. Tokuhiro, K. Welter, W.-K. Wong, and H. Yang, "Status of research and development of learning-based approaches in nuclear science and engineering: A review," *Nuclear Engineering and Design*, vol. 359, 12 2019.

[34] J. Ma and J. Jiang, "Applications of fault detection and diagnosis methods in nuclear power plants: A review," *Progress in Nuclear Energy - PROG NUCL ENERGY*, vol. 53, pp. 255–266, 04 2011.

[35] "How machine learning is already shaping the nuclear world," http://nti.org/7358AP, accessed: 2019-12-11.

[36] M. Chandy, C. Pilotto, R. McLean, "Networked sensing systems for detecting people carrying radioactive material," *5th International Conference on Networked Sensing Systems*, p. 148–155, Kanazawa,Japan, 17–19 June 2008.

[37] D. Macedo, “Radioactive hot-spot detection using unmanned aerial vehicle surveillance.” Master’s thesis, Instituto Superior Técnico, Portugal, 11 2018, supervised by Alberto Vale and Horácio Fernandes.

[38] J. W. Howse, L. O. Ticknor, and K. R. Muske, “Least squares estimation techniques for position tracking of radioactive sources,” *Automatica*, vol. 37, no. 11, pp. 1727 – 1737, 2001.

[39] N. S. V. Rao, M. Shankar, J. Chin, D. K. Y. Yau, S. Srivathsan, S. S. Iyengar, Y. Yang, and J. C. Hou, “Identification of low-level point radiation sources using a sensor network,” in *2008 International Conference on Information Processing in Sensor Networks (ipsn 2008)*, 2008, pp. 493–504.

[40] X. Xu, N. Rao, and S. Sahni, “A computational geometry method for localization using differences of distances,” *TOSN*, vol. 6, 02 2010.

[41] M. Ding and X. Cheng, “Fault tolerant target tracking in sensor networks,” 05 2009, pp. 125–134.

[42] D. Shah and S. Scherer, “Robust localization of an arbitrary distribution of radioactive sources for aerial inspection,” 2017.

[43] H. E. Baidoo-Williams, S. Dasgupta, R. Mudumbai, and E. Bai, “On the gradient descent localization of radioactive sources,” *IEEE Signal Processing Letters*, vol. 20, no. 11, pp. 1046–1049, 2013.

[44] Yang Cheng and Tarunraj Singh, “Source term estimation using convex optimization,” in *2008 11th International Conference on Information Fusion*, 2008, pp. 1–8.

[45] K. Liu, Y. Cheng, P. Li, and T. Singh, “Source localization on two-dimensional grid,” in *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, 2011, pp. 1–5.

[46] Y. Zou and Krishnendu Chakrabarty, “Sensor deployment and target localization based on virtual forces,” in *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, vol. 2, 2003, pp. 1293–1303 vol.2.

[47] J. Chin, D. K. Y. Yau, and N. S. V. Rao, “Efficient and robust localization of multiple radiation sources in complex environments,” in *2011 31st International Conference on Distributed Computing Systems*, 2011, pp. 780–789.

[48] A. Pozdnoukhov, M. Kanevski, “Multi-scale support vector algorithms for hot spot detection and modelling,” *Stochastic Environmental Research and Risk Assessment*, vol. 22, p. 647–660, August 2007.

[49] U. Abeyratne, Y. Kinouchi, H. Oki, J. Okada, F. Shichijo, and K. Matsumoto, “Artificial neural networks for source localization in the human brain,” *Brain topography*, vol. 4, pp. 3–21, 02 1991.

[50] S. Jun and B. Pearlmutter, “Fast robust meg source localization using mlps,” 06 2002.

[51] ——, “Subject-independent magnetoencephalographic source localization by a multilayer perceptron,” *Advances in Neural Information Processing Systems*, 03 2004.

[52] V. Borisov, J. Scheible, “Lithography hotspots detection using deep learning,” 07 2018, pp. 145–148.

[53] M. Saha, C. Chakraborty, I. Arun, “An advanced deep learning approach for ki-67 stained hotspot detection and proliferation rate scoring for prognostic evaluation of breast cancer,” *Scientific Reports*, vol. 7, p. 3213, 2017.

[54] “Tendencies, prospects, and recent developments of machine learning,” https://yalantis.com/blog/trends-in-machine-learning-in-past-years/, accessed: 2019-12-11.

[55] World Institute for Nuclear Security, “Data analytics for nuclear security,” https://www.wins.org/files/30.01.2015_data_analytics_for_nuclear_security_rev_1.0_en_final_1.pdf, Jan 2015.

[56] M. van Otterlo and M. Wiering, *Reinforcement Learning and Markov Decision Processes*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 3–42.

[57] Z. Liu and S. Abbaszadeh, “Double q-learning for radiation source detection,” *Sensors (Basel, Switzerland)*, vol. 19, no. 4, 2019.

[58] C. Cortes and V. Vapnik, “Support-vector networks,” *Mach. Learn.*, vol. 20, no. 3, p. 273–297, 09 1995.

[59] Nicholas Tsoulfanidis, Sheldon Landsberger, *Measurement and Detection of Radiation.*, 4th ed. Boca Raton: CRC Press, 2015.

[60] *Evaluation of guidelines for exposures to technologically enhanced naturally occurring radioactive materials*. Washington, DC: National Academy Press, 1999.

[61] M. R. Minar and J. Naher, “Recent advances in deep learning: An overview,” 2018.

[62] K.-S. Oh and K. Jung, “Gpu implementation of neural networks,” *Pattern Recognition*, vol. 37, pp. 1311–1314, 06 2004.

[63] M. Abadi, A. Agarwal *et al.*, “Tensorflow: Large-scale machine learning on heterogeneous distributed systems,” 2016.

[64] M. Abadi, P. Barham *et al.*, “Tensorflow: A system for large-scale machine learning,” in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. Savannah, GA: USENIX Association, 11 2016, pp. 265–283.

[65] T. T. D. Team, R. Al-Rfou *et al.*, “Theano: A python framework for fast computation of mathematical expressions,” 2016.

[66] A. Paszke, S. Gross *et al.*, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8026–8037.

[67] I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*. MIT, 2017.

[68] M. Buscema, “Back propagation neural networks,” *Substance use misuse*, vol. 33, pp. 233–70, 02 1998.

[69] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014.

[70] C. Cortes, M. Mohri, and A. Rostamizadeh, “L2 regularization for learning kernels,” 2012.

[71] W. Zhiqiang and L. Jun, “A review of object detection based on convolutional neural network,” in *2017 36th Chinese Control Conference (CCC)*, 2017, pp. 11 104–11 109.

[72] S. Anwar, M. Majid, A. Qayyum, M. Awais, M. Alnowami, and K. Khan, “Medical image analysis using convolutional neural networks: A review,” *Journal of Medical Systems*, vol. 42, p. 226, 10 2018.

[73] K. M. Ting, *Precision and Recall*. Boston, MA: Springer US, 2010, p. 781.

[74] Y. Brouwer, “Radiological monitor for mobile robots operating in scenarios with nuclear threats,” Master’s thesis, Instituto Superior Técnico, Portugal, 11 2019, supervised by Alberto Vale and Rodrigo Ventura.

[75] S. Gillies, “The shapely user manual,” https://pypi.org/project/Shapely/, 2013.

[76] R. Bridson, “Fast poisson disk sampling in arbitrary dimensions,” *ACM SIGGRAPH*, 08 2007.

[77] C. Hill, “Poisson disc sampled noise,” *GitHub repository*, 2018. [Online]. Available: https://github.com/scipython/scipython-maths/tree/master/poisson_disc_sampled_noise

[78] T. E. Oliphant, *A guide to NumPy*. Trelgol Publishing USA, 2006, vol. 1.

[79] J. Borbinha, A. Vale, B. Gonçalves *et al.*, “Performance analysis of geiger–müller and cadmium zinc telluride sensors envisaging airborne radiological monitoring in norm sites,” *Sensors*, vol. 20, no. 5, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/5/1538

[80] The pandas development team, “pandas-dev/pandas: Pandas,” 02 2020.

[81] “Equirectangular projection,” https://en.wikipedia.org/w/index.php?title=Equirectangular_projection&oldid=992364051, 2020, accessed: 2020-10-10.

[82] “Geographic coordinate conversion,” https://en.wikipedia.org/w/index.php?title=Geographic_coordinate_conversion&oldid=993537992, 2020, accessed: 2020-10-10.

[83] T. Kluyver, B. Ragan-Kelley *et al.*, “Jupyter notebooks – a publishing format for reproducible computational workflows,” in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, F. Loizides and B. Schmidt, Eds. IOS Press, 2016, pp. 87 – 90.

[84] F. Chollet *et al.*, “Keras,” https://keras.io, 2015.

[85] NVIDIA, P. Vingelmann, and F. H. Fitzek, “Cuda, release: 10.2.89,” 2020.

[86] L. Buitinck, G. Louppe *et al.*, “API design for machine learning software: experiences from the scikit-learn project,” in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[87] “Friends - fleet of drones for radiological inspection, communication and rescue,” https://www.ipfn.tecnico.ulisboa.pt/FRIENDS/index.html, accessed: 2020-12-08.

A

MARIA mobile application

Mobile Application for Radiation Intensity Assessment, known by MARIA [12], is an Android application for mobile devices (like smartphones), able to detect hazardous radiation and identify possible threats, by constructing a radiological intensity map of the surrounding area. This application was created in 2017 at IPFN with the initial objective of acquiring and recording georeferenced radiological intensity data. Currently, MARIA is in version 3 and is capable of combining location data from a GPS receiver (and telecommunications antenna) with radiological intensity readings from a common radiation sensor such as a GMC. By connecting the latter to the mobile device, through a Universal Serial Bus (USB) port, the user is instantly ready to start taking measurements. Then, this acquired data is then stored on a remote server where advanced analytics can be performed.

In Fig. A.1 (left) it is shown the MARIA splash-screen with information about how the application works, the version number, the IPFN and IST logos (which also work as a direct link to their home-pages), and two buttons to begin acquiring measurements: ”Login” for IPFN members and ”Guest” for the remaining users. Then, the user has two ways to proceed: use a connected radiation sensor or use the simulator (for demonstration), which draws random samples from a Poisson distribution (identical to what is done in the simulator used for this thesis, as explained in section 3.1). During data acquisition,

the user sees an image like the one depicted in Fig. A.1 (right), where the ”Sensor Mode” is displayed, used for reading raw sensor data. This is an interface with only three buttons: ”Pause/Start”, ”Stop” and ”Options”. The latter command allows the user to configure the reading interval, switch between ”Map Mode” and ”Sensor Mode”, and use other useful tools.

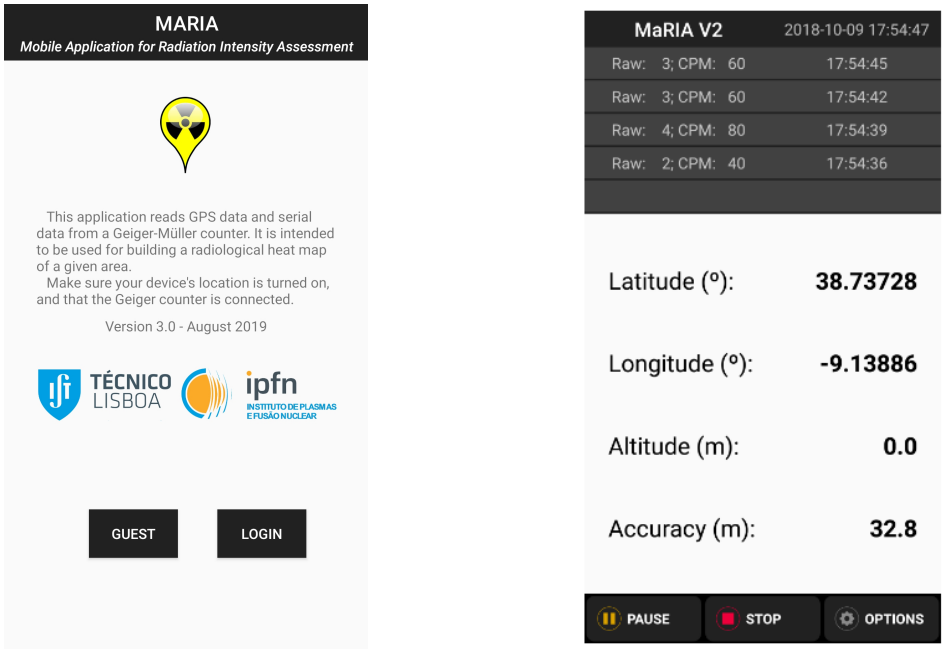


Figure A.1: The MARIA application splash-screen, in version 3 (left). Screenshot of MARIA application during data acquisition, in sensor mode, displaying the past 4 measurements and the current positional data (right). Images from [12].

The other possible mode to see during data acquisition, the ”Map Mode”, is used for visualizing the data representation through a heatmap. This is done by applying an algorithm in the combination of the acquired data, and a map view displays the intensity measurements at the location where they were made over Google Maps™, allowing the user to monitor the radiation levels in space and time. In Fig. A.2 we have the heatmap of two different scenarios. The circles represent the measurements, which have a color according to their intensity values. This color is in a gradient ranging from green (low intensity) to red (high intensity). The small blue circle seen in the left image represents the user's current location.

Thus, since MARIA can currently measure, visualize, and record radiological data, it will be a possibility to incorporate the pre-trained neural network model, obtained in the end of this thesis, in version 4 of MARIA, by implementing it in such a way that it can be run in real-time.

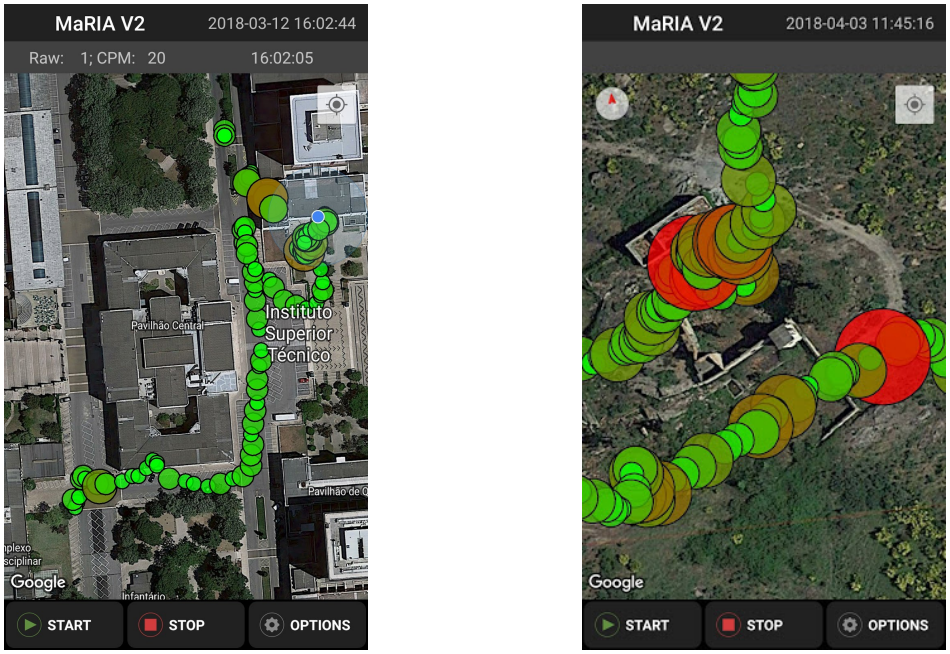


Figure A.2: Screenshot of MARIA application during data acquisition, in map mode, in scenarios located at IST (left) and a random place with two radioactive hot-spots (right). Images from [12].

B

Representation of Divide And Conquer algorithm applied on tested scenarios

The present appendix displays the entire scheme of the DAC algorithm applied on the tested scenarios, where at least one division was made. The corresponding input/observational data, as well as the final results, were previously presented in chapter 5. To note that the scenario **1.c)** with a high number of sources (16 in total), represented in Fig. 5.6, is not present here once there are too many boxes from ”Division 1” and ”Division 2”.

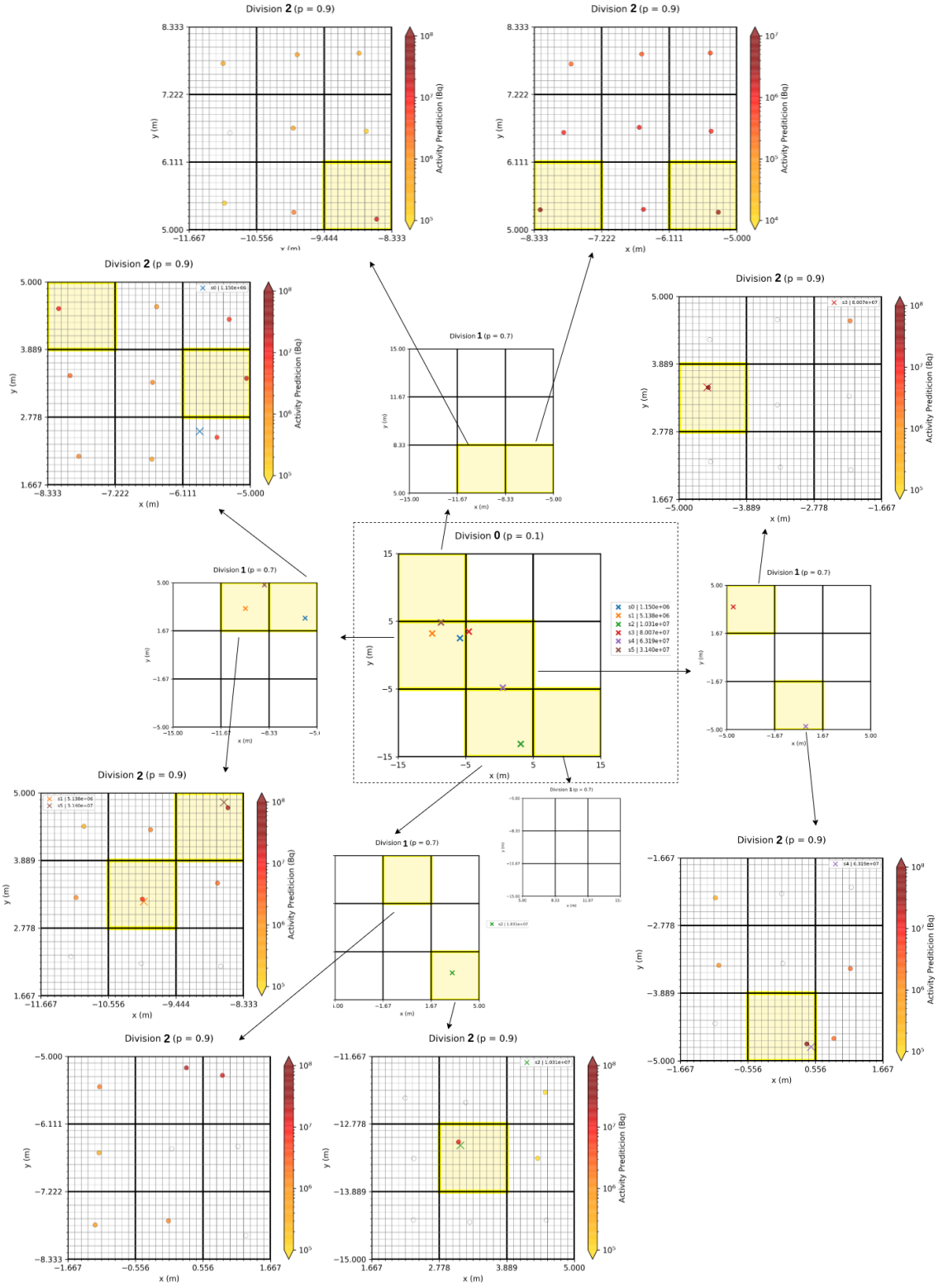


Figure B.1: Simulation result of scenario **1.b)**. In ”Division 1”, there is one scenario without yellow boxes, as no source was detected. The same happens in ”Division 2”.

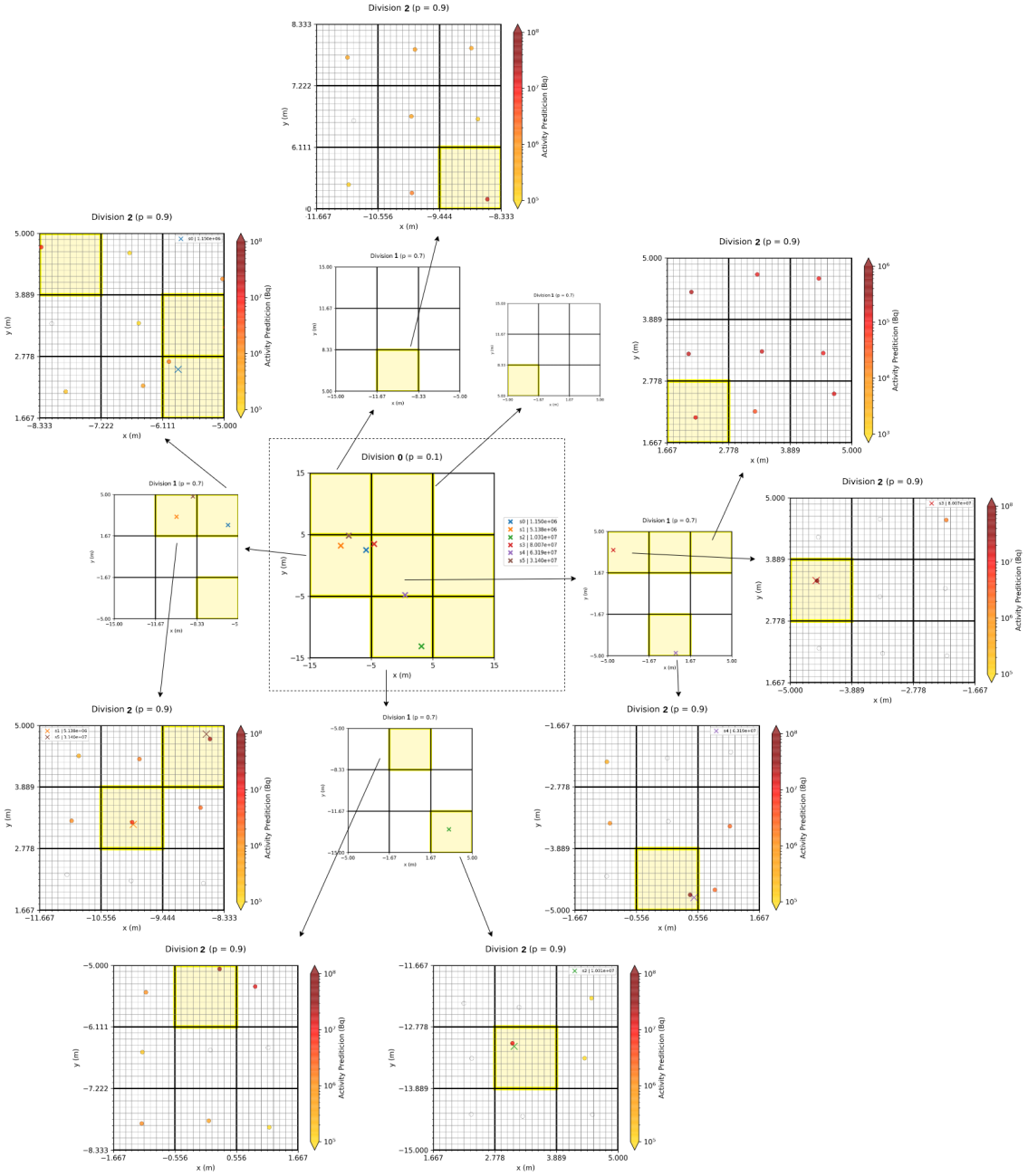


Figure B.2: Simulation result of scenario **1.d)**, which is the scenario **1.b)** with obstacles present. The divided cells with zero yellow boxes (i.e., without any source detected) are not represented to facilitate the visualization of the process.

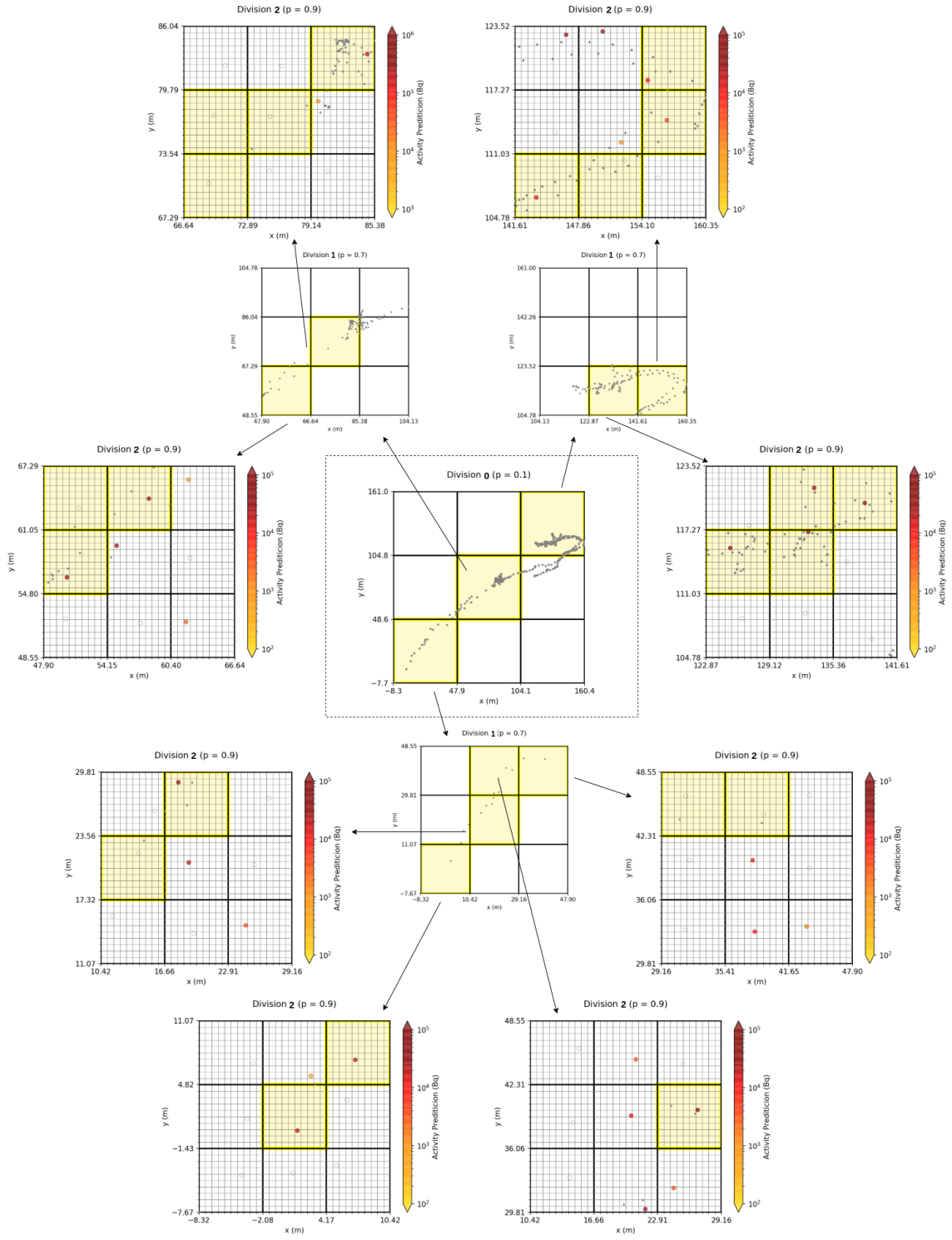


Figure B.3: Result of real scenario **3.a)**. The gray points represent the observations with intensity values higher than 100 CPM, which may be related to possible hot-spots.

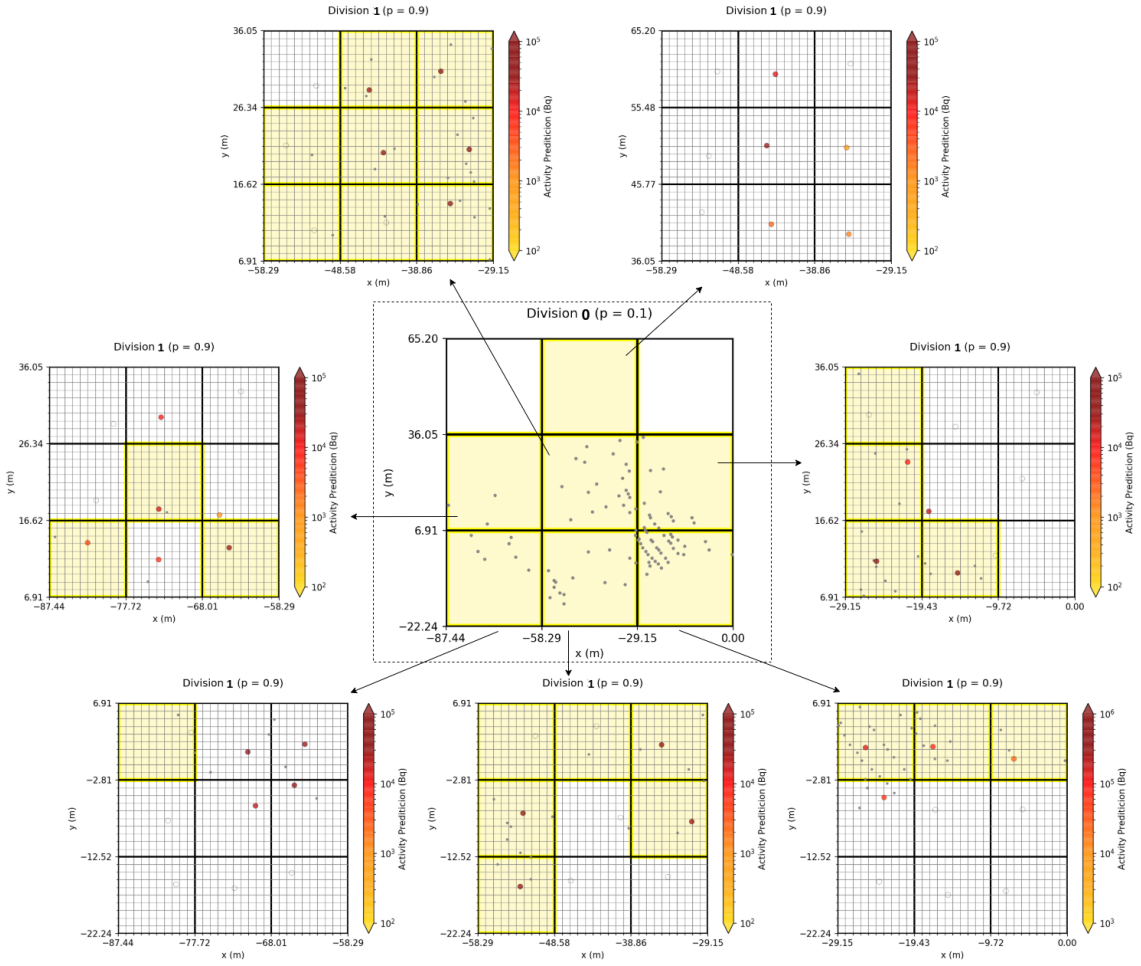


Figure B.4: Result of real scenario **3.b)**. The gray points represent the observations with intensity values higher than 100 CPM, which may be related to possible hot-spots. In this case, the DAC process was applied only until "Division 1", due to the available number of observations. If more divisions were made, the available data was not enough, and so a great rate of the ANN model inputs would be empty.