POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master's Degree Thesis

Semi-Supervised Techniques for Solar Panel Segmentation in Aerial Images

Supervisors

Prof. Paolo GARZA

Dr. Edoardo ARNAUDO

Candidate

Antonino MONTI

December 2022

Abstract

There has been an ever-growing awareness in recent years about climate change, a phenomenon which is caused primarily by the massive use of fossil fuels by humans. One of the main solutions to this issue is to switch to low-carbon and renewable forms of energy, which include solar energy.

To promote the use of photovoltaic panels, it is useful to have an extensive and updated database of installed panels and plants, which could help with performance evaluations, but the availability of such maps is scarce. A solution to this problem is to employ artificial intelligence systems that automatically detect panels in aerial images, specifically by detecting, segmenting and classifying every panel in a given image. This is the objective of Instance Segmentation, one of the most frequently-tackled tasks in the field of Computer Vision.

However, the effective training of a Deep Learning model is at times impeded by the scarcity of data: it is expensive and time-consuming to gather enough annotated data to form a reasonably large dataset. This is the case for the dataset used for this thesis, which does not have enough annotations to obtain satisfying results. It therefore becomes necessary to employ methods to artificially increase the size of the dataset, such as semi-supervised learning and Unsupervised Domain Adaptation. The former allows an algorithm to generate annotations by itself on a set of unlabeled data after training on a small portion of labeled data, while the latter consists in taking an algorithm that has satisfying performances on a given domain and apply it to a different but related domain where data is scarce.

The goal of this thesis is therefore to evaluate the efficacy of semi-supervised methods for a model trained with the purpose of detecting and segmenting solar panels in aerial images. To this end, two such methods were explored and adapted for the task at hand. The first is Noisy Boundaries, a framework for semi-supervised instance segmentation which provides additional components to resist and exploit the noise inherent in the boundaries of artificially-generated annotations. The second is the union of DACS, a framework that performs Unsupervised Domain Adaptation, with Instance Mixing techniques, with the end goal of increasing the size of the dataset by copying and pasting groups of instances from labeled images to unlabeled images, along with the automatic generation of labels on the latter. Said techniques are evaluated and compared, highlighting their differences and demonstrating their effectiveness in contexts with scarce annotations.

Acknowledgements

I would like to thank prof. Paolo Garza for the opportunity and his availability as supervisor for this thesis. I would also like to thank Edoardo Arnaudo from the LINKS Foundation, for the patience, support and feedback offered for both the work performed in these months and the drafting of the thesis itself.

I would like to thank my family, for the unconditional love shown to me in these 23 years, for the financial support and the sacrifices that allowed me to pursue my master's degree at the Polytechnic of Turin, but most importantly for their emotional support, for supporting me and standing by my side through thick and thin and for bearing with me when I was particularly nervous. Despite what my sometimes grumpy attitude might suggest, I feel extremely lucky and grateful to have you in my life.

Last, but most certainly not least, I would like to thank all the friends I made during these five years of university, especially, in no particular order: Mattia, Gabriele, Francesco D.S., Giovanni, Francesco S., Riccardo, Emanuele, Francesco D.G., Giuseppe, Salvatore, Alan, Fausto, Fabio and Gianluca. This adventure wouldn't have been as meaningful as it has been for me without all of you, and you've helped me grow a lot as a person. I don't know what the future holds in store for us, but I know that I will cherish the memories of all the laughs, the conversations and the experiences we shared along the way, and I hope we'll make as many more as our life paths will allow in the years to come.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In recent years, there has been an ever-growing awareness about the phenomenon of climate change, which has an impact on Earth's climate system. The increase in temperatures has been much faster than in previous changes in the planet's climate, and the main cause of this, along with other factors such as deforestation, is the burning of fossil fuels by humans, which causes an increase in greenhouse gases. Therefore, one of the most effective ways to combat this is to cut such emissions by drastically reducing the use of fossil fuels and transition to using electricity generated by low-carbon and renewable energy sources, such as solar energy, which is exploited by means of photovoltaic panels.

In order to better plan out the installation of solar panels and evaluate their efficiency, especially in the case of large-scale industrial plants, having a comprehensive and up-to-date database of existing plants is crucial. However, such maps are not always available, and one way to build them is to employ automatic detection systems on aerial images, which is possible by using artificial intelligence and, specifically, Machine Learning and Deep Learning systems.

One of the tasks in the field of Computer Vision that can be employed to carry this detection out is that of Instance Segmentation, which consists in locating single instances of objects within a given image and segmenting each of them by producing a segmentation mask representing the shape of the instance. In the case of aerial images and solar panels, this consists in locating instances of panels within the images, computing the shape and extension of each by means of a segmentation mask and, additionally, detecting the type of each panel, allowing to estimate the extension and, consequentially, the power and efficiency of each panel.

Manually gathering enough aerial images and especially annotations to effectively train a Deep Learning model is very expensive, both in terms of time and effort: this is the case, for instance, for the dataset used specifically for this thesis, which contains aerial images from the regions of Alessandria and Asti in the Piedmont region in Italy, but which has few manually-generated annotations of solar panels

at its disposal. It therefore becomes necessary to employ methods in order to artificially expand the size of datasets. To this end, two training paradigms in particular come to the aid: one is semi-supervised training, in which an algorithm is fed both annotated and non-annotated content in order to learn from the former how to generate its own labels on the latter, and the other is Unsupervised Domain Adaptation, which consists in taking an algorithm that already performs well in a given domain with plenty of training data and employing it to solve a problem in a different, but related, domain with less data available.

Numerous works have tackled both of these paradigms, and in this thesis two in particular are explored, adapted for the task at hand and finally compared. The first is Noisy Boundaries, a framework which employs semi-supervised training to generate pseudo-labels plus additional components to resist and exploit the noise inherent in the boundaries of segmentation masks. The second is the union of an existing framework called DACS, which focuses on Unsupervised Domain Adaptation, with Instance Mixing techniques, with the aim of extending the available data by copying annotations from labeled images and pasting them on unlabeled images.

The thesis is structured as follows:

- Chapter 2, *State of the Art*, introduces the reader to background information necessary to understand the problems and corresponding solutions presented in this thesis and gives an overview of existing literature, with a focus on works that deal with aerial instance segmentation and working with low resources;
- Chapter 3, *Data Sources*, gives an overview of the dataset used for the purpose of this work, detailing both its properties and the procedure used to generate the datasets;
- Chapter 4, *Methodology*, presents the problem statement for the thesis and describes the process used to obtain baselines and the semi-supervised methods employed;
- Chapter 5, *Experiments*, presents both the implementation details for the experiments and a comparison of the results obtained;
- Chapter 6, *Conclusions*, explores future developments for the work performed and presents final remarks on the results obtained and the thesis as a whole.

Chapter 2

State of the art

This chapter gives the reader background information necessary to understand the problems and corresponding solutions described in this thesis. It first talks about fundamental concepts in the field of Deep Learning and Computer Vision (Section 2.1), giving particular attention to Convolutional Neural Networks and image segmentation tasks (Section 2.2). Then, it follows with an overview of existing literature (Section 2.3) dealing with image segmentation and training models with limited resources available, with a focus on works that tackle the problems of aerial instance segmentation and, in particular, of detection and mapping of solar panels within aerial images, which is the main objective of this thesis.

2.1 Deep Learning and Computer Vision

This section provides a general overview about the main concepts behind Deep Learning and Computer Vision. In regards to the second part, it describes Convolutional Neural Networks along with a brief history of their development and applications, and then focuses on the task of instance segmentation, which is at the heart of this thesis.

2.1.1 Deep Learning background

Machine Learning (ML) is a field of study in Computer Science and a subset of Artificial Intelligence which is focused on the use of data and algorithms to allow computer programs to learn autonomously, imitating the way in which humans learn. A Machine Learning algorithm can make use of Artificial Neural Networks (ANN), computing systems whose structure is directly inspired by the structure of an organic brain: Neural Networks are in fact made of interconnected nodes (or neurons), organized in multiple layers.

Deep Learning (DL) is a subset of ML in which algorithms use Deep Neural Networks (DNNs), which are made up of several layers of neurons.

In ML, datasets are split into three separate subsets, each with its own purpose: the *training set*, as the name implies, is a subset used during training of the algorithm; the *validation set* is a subset used during training to ensure the algorithm is learning properly and the hyperparameters used are adequate; finally, the *test set* is used after training to evaluate the algorithm's performance.

A ML algorithm learns by comparing its predictions with real data, and by subsequently modifying its parameters in order to obtain better results for the next batch of predictions. In practice, learning occurs by using a *loss function*, which returns the error between real and predicted data: the lower the returned value, the better the algorithm's performance. Thus, the algorithm's goal is to minimize the loss function's value as much as possible.

Algorithms can be classified depending on the way they learn from input data. There are three main categories under which algorithms fall: Supervised, Unsupervised and Semi-supervised.

Supervised Learning is a paradigm in which the algorithm uses exclusively labeled data in order to learn to classify data or predict outcomes. By making use of labeled data, the algorithm can improve its accuracy over time.

Unsupervised Learning is the opposite of Supervised Learning, meaning that the algorithm makes use of only unlabeled data during training. Algorithms that follow this paradigm are often used to discover hidden patterns in data without any external intervention.

In **Semi-Supervised Learning**, the algorithm uses both a small portion of labeled data and a large portion of unlabeled data during training. Usually, at first the algorithm uses only labeled data, and afterwards uses what it learned to generate labels of its own (called *pseudo-labels*) on unlabeled data, which it then combines with manually-labeled data for further training, this time with the use of a larger dataset. This paradigm can be particularly useful since manually creating labels for training data is expensive and time-consuming, while gathering unlabeled data does not require the same amount of work. Allowing an algorithm to generate training labels on its own can thus be of great practical value.

This is also true in the case of **Unsupervised Domain Adaptation** tasks, in which the aim is to take an algorithm that performs well in one or more domains and adapt it to work on a different, but related, field. Again, this is useful when the target domain does not provide labels of its own.

As mentioned earlier, Neural Networks (NNs) are computing systems inspired by the structure of the human brain and which are made up of nodes (or neurons) organized in interconnected layers.

These networks are comprised of an input layer, one or more hidden layers and

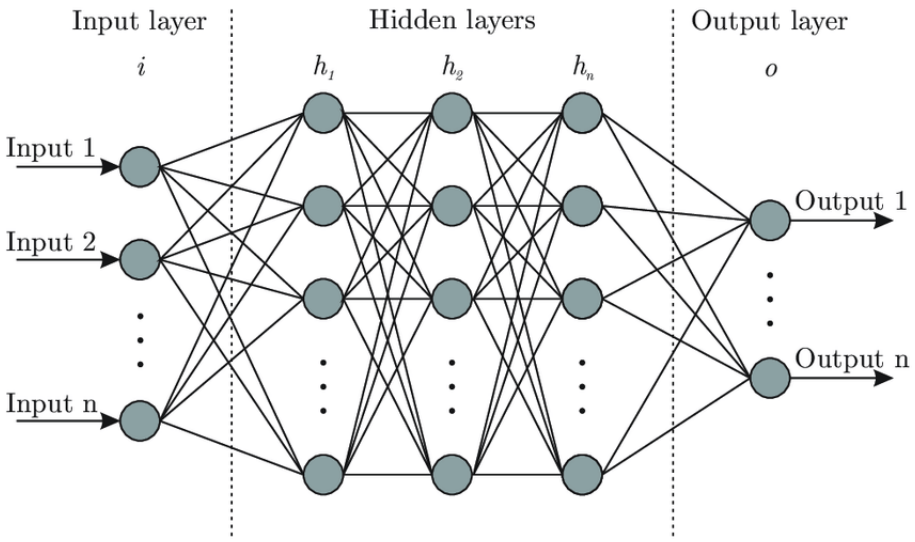

Figure 2.1: Example of the structure of a Neural Network.
[**Credit**: *Medium*]

an output layer. Each of the neurons is connected to at least another one in the network and has a weight and a threshold associated with it, plus an activation function, which is used by the node to process its input.

The number of neurons in the input and output layers depends on what the network receives in input and what it is expected to return in its output. However, as the size of the input begins to grow, so does the number of parameters the network has to handle and fine-tune at each iteration, requiring an ever-growing expense of time and resources for training the model. This is the case, for instance, of networks that have to analyze images, which are usually comprised of at least three channels (RGB, i.e. Red, Green and Blue), thus greatly multiplying the number of input features.

It is therefore necessary to find a way to compress the input and reduce how many features are fed to the network.

2.1.2 Convolutional Neural Networks and CV

Convolutional Neural Networks (CNNs) exist to tackle the problem of feature overgrowth, along with other issues. As the name suggests, the main difference between a regular neural network and a CNN is the use of an operation called *convolution*. The use of convolution allows to compress great numbers of low-level features into fewer, higher-level features for the next layer in the network; it also helps reduce overfitting, since the number of input weights is reduced.

Convolution is implemented in *convolutional layers* inside CNNs, and makes use

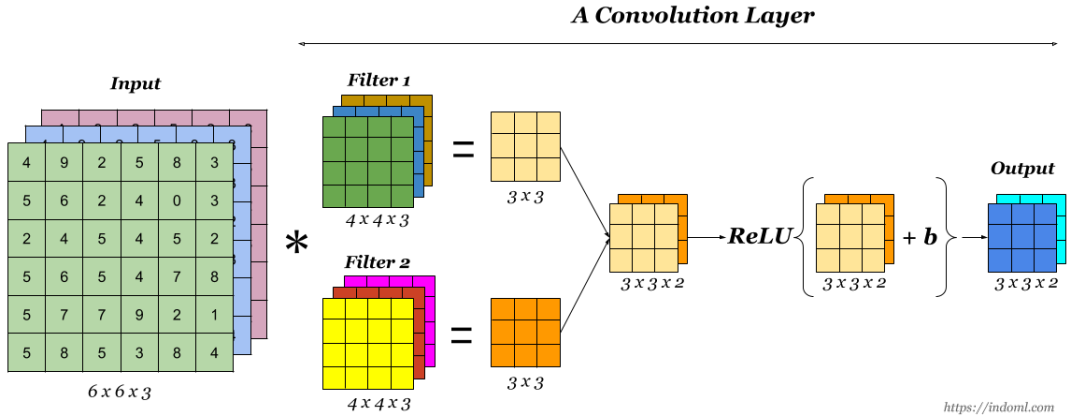

Figure 2.2: Structure of a convolutional layer.
[**Credit**: *towardsdatascience*]

of one or more *kernels*, fixed-size windows of learnable weights that slide across the image, computing the output features as a weighted sum of the pixels inside the area delimited by the kernel itself. A convolution is defined by three parameters: the size of the kernel(s), stride (i.e. by how many cells the kernel moves inside the input matrix) and padding, the combination of which governs the size of the output.

After applying a convolution to the input features, convolutional layers finally apply a non-linear activation function in order to introduce non-linear behavior in an otherwise linear operation.

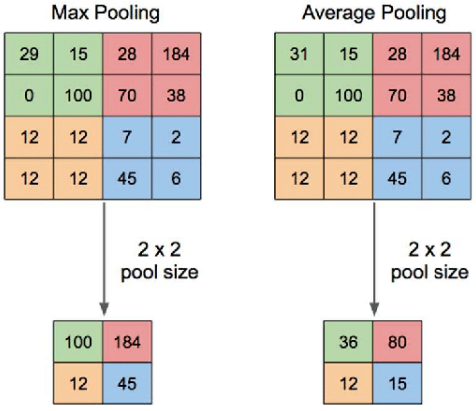

Figure 2.3: Two examples of pooling: max pooling and average pooling.
[**Credit**: *researchgate*]

Another type of layers commonly found in CNNs are *pooling layers*, which, intuitively, implement pooling operations. Pooling is used to further reduce the

number of features by dividing the input into regions and applying a function to each of them. There are three main types of pooling: *max pooling*, which selects the maximum value from a region, *min pooling*, which operates in the opposite way, and *average pooling*, which computes the average value of a selected group of features. A pooling operation is defined by the size of the regions and a stride.

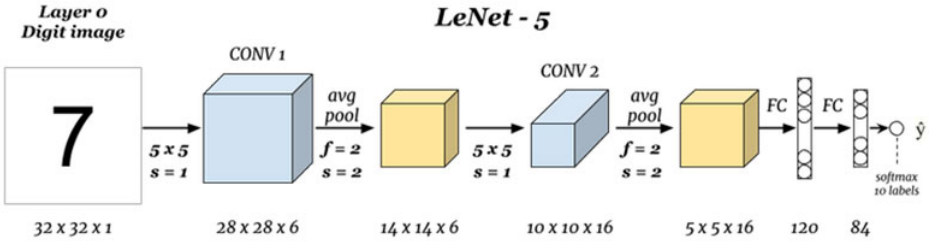

Figure 2.4: Structure of LeNet-5.
[**Credit**: *topbots*]

One of the first CNNs to ever be designed was **LeNet-5**, presented by LeCun et al. [1]. Its structure was very simple and was not as deep as modern CNNs. LeNet-5 operated on grayscale images and was designed to solve tasks of digit recognition.

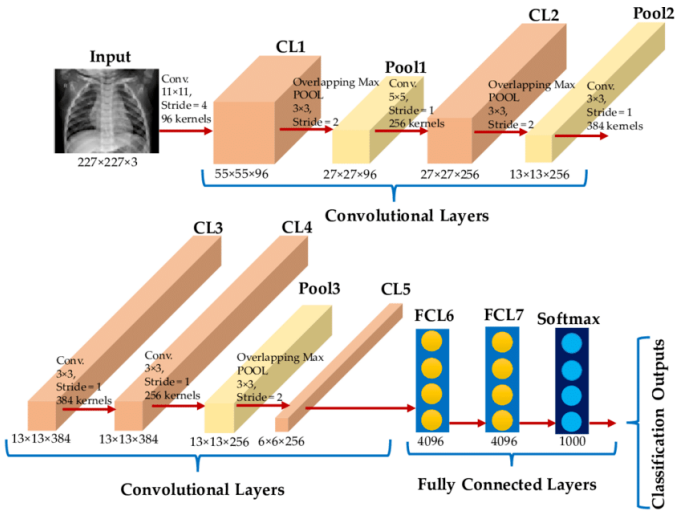

Figure 2.5: Structure of AlexNet.
[**Credit**: *researchgate*]

One of the first large-scale applications of CNNs to perform well with RGB images, and to use the ImageNet dataset [2], was **AlexNet**, proposed by Krizhevsky et al. [3]. As shown in figure 2.5, AlexNet has a greated depth than LeNet-5, and

this characteristic, according to the original paper, was paramount to obtaining high performances, despite the greater computational costs. The network had a large number of parameters (about 60 million), increasing its complexity: this was possible first and foremost thanks to the advent of GPUs as means of parallel computing, which allowed to cut both computing times and costs by several orders of magnitude.

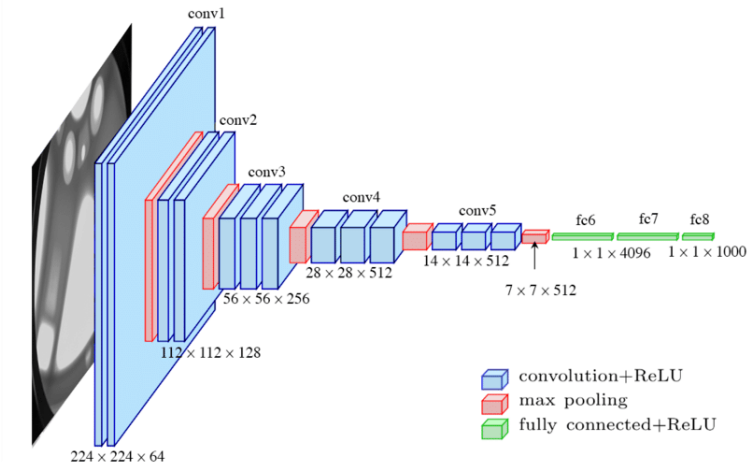

Figure 2.6: Structure of VGG-16.
[**Credit**: *researchgate*]

VGG-16, presented by Simonyan et al. [4], was an attempt to reduce AlexNet's complexity by using a simpler and more uniform network architecture: each level uses the same parameters for convolution and max pooling, and uses more than one convolutional layer. Despite being deeper than AlexNet, the structure of VGG-16 is thus more uniform since levels are very similar to each other.

CNNs are nowadays widely used in **Computer Vision**, a field with the main goal to give computer systems the ability to derive high-level information from images, videos and other visual media as a human would. Some of the most frequent tasks in this field include classification and/or detection tasks such as image classification or object detection, image segmentation tasks such as semantic segmentation and instance segmentation, and generative tasks such as Super Resolution and Neural Style Transfer.

In order to achieve satisfying performances with Deep Learning models, datasets need to be large and varied enough to provide the algorithm with sufficient data to learn how to make correct predictions. However, gathering data is often costly,

both in terms of time and resources, and some datasets can therefore be too small or too uniform in order to make training efficient or even possible.

A classic solution adopted in literature for this issue is the use of *data augmentation* techniques, the purpose of which, as the name implies, is to artificially increase the quantity and variety of data within the dataset by applying subtle alterations to existing data.

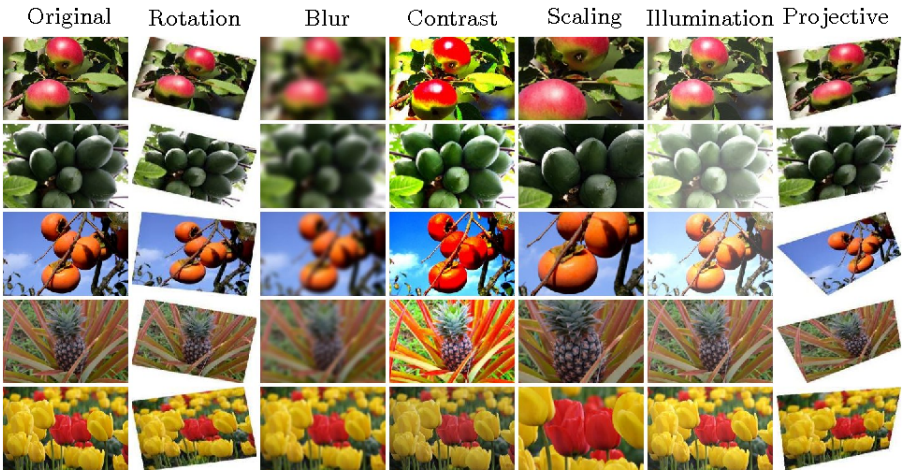

Figure 2.7: Showcase of some data augmentation techniques applied to images. [**Credit**: *Medium*]

In the field of Computer Vision, which deals with images, there are mainly two types of data augmentation techniques: *geometric augmentations*, which include transformations such as rotation, scaling and translation and *photometric augmentations*, which operate on the values of each pixel such as contrast, brightness and color.

2.2 Instance Segmentation

As mentioned earlier, instance segmentation is among the most frequently tackled tasks in the field of Computer Vision, and the one this thesis revolves around. It can be thought of as the union of two other tasks: object detection and semantic segmentation.

Object detection consists in detecting and classifying objects within an image. The goal here is twofold: detecting each object's position inside the image and then assigning each a class. This is achieved through the use of *bounding boxes*, which are essentially rectangles surrounding the objects, defined by their position inside the image and their width and height.

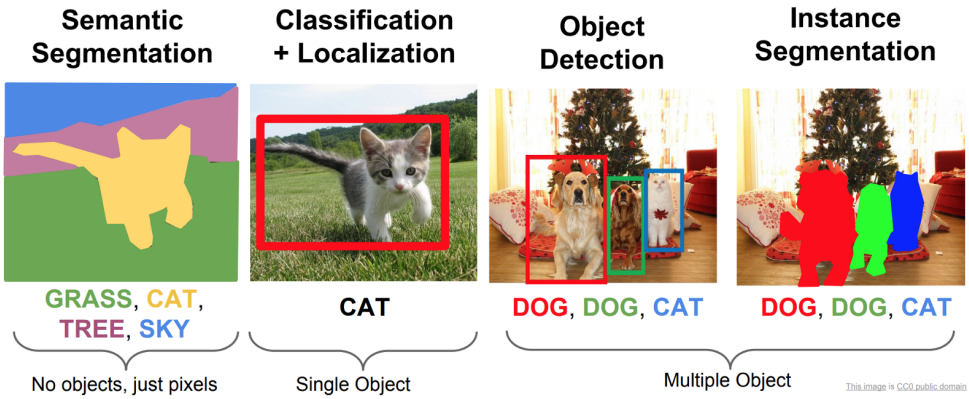

Figure 2.8: Difference between Object Detection, Semantic Segmentation and Instance Segmentation.

[**Credit**: *towardsdatascience*]

The goal of **semantic segmentation**, on the other hand, is to classify every single pixel inside the image. This means that all the pixels associated to a class are considered as a single entity, even if they are not close to each other.

The aim of **instance segmentation** is to identify each instance of each object inside an image. This is achieved by first predicting a bounding box for each object (like in object detection) and then applying a segmentation process to the area inside the box, thus detecting the pixels that represent the instance. The main difference between this process and the one carried out by semantic segmentation is that the latter does not distinguish between single instances of an object, while the purpose of instance segmentation is exactly that.

2.3 Related works

2.3.1 Instance segmentation

Given that part of the process of instance segmentation essentially follows that of object detection, this paragraph will explore some of the approaches historically employed to solve the latter problem and will then present solutions devised for the former.

Unlike tasks such as classification, object detection requires localization of the (likely many) instances inside the input images. Some of the first solutions proposed regression as a potential approach; however, later works [5, 6] show that such a method could prove to be suboptimal in practice. Other works [7, 8, 9] propose the use of a sliding-window approach, in which a window of fixed size is defined and moved inside the image, defining regions that are then processed by a convolutional

network. Sliding-window detectors were initially CNNs with few layers; however, with the passing of time and the increase in depth of CNNs such an approach proved to be challenging to implement and possibly inefficient.

Thus, another method employed to perform object detection is region proposals, consisting in defining *regions of interest* (*RoIs*) within the image. These regions are assumed to have useful information and are fed to a ConvNet, which proceeds to classify each of them.

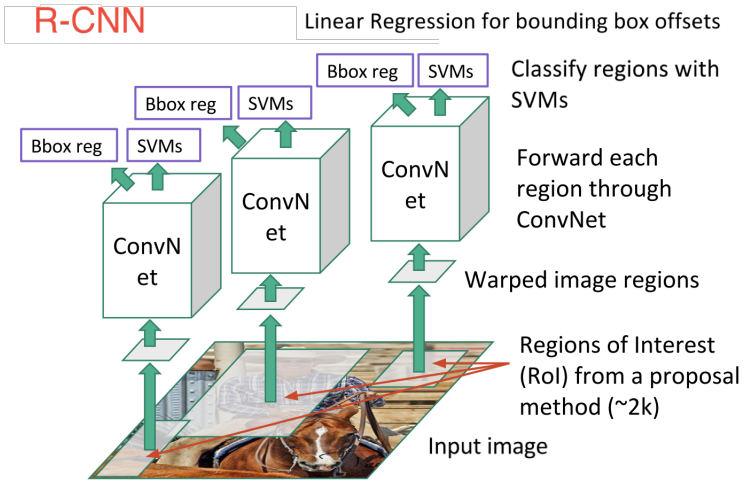

Figure 2.9: Structure of an R-CNN.
[**Credit**: *stackoverflow*]

Region-based CNNs (R-CNNs) are CNNs that use region proposal, and were first introduced in the work of Girshick et al. [6]. Such an approach allowed to manage a reasonable number of RoIs and to evaluate performances independently on each region. [10, 11, 12, 3] The process of R-CNNs was extended and improved in order to include RoI Pooling with Fast R-CNN, proposed by Girshick et al. [13]. This, as the model's name implies, allowed for faster processing and more accurate detection. Faster R-CNN, later proposed by Ren et al. [14], further built upon this idea by adding a new stage to detection with a Region Proposal Network (RPN), a CNN which shared its backbone with Fast R-CNNs.

Given the success R-CNNs had in the field of object detection, they were adapted and employed to perform instance segmentation as well. Mimicking region proposal, many approaches used a method called segment proposal [6, 15, 16]. Some works performed segment proposals before region proposals [17, 18, 19], while others performed it afterwards [20].

A real breakthrough, both in terms of performance and efficiency, was obtained with Mask R-CNN. The framework, proposed by He et al. [21], performs segment

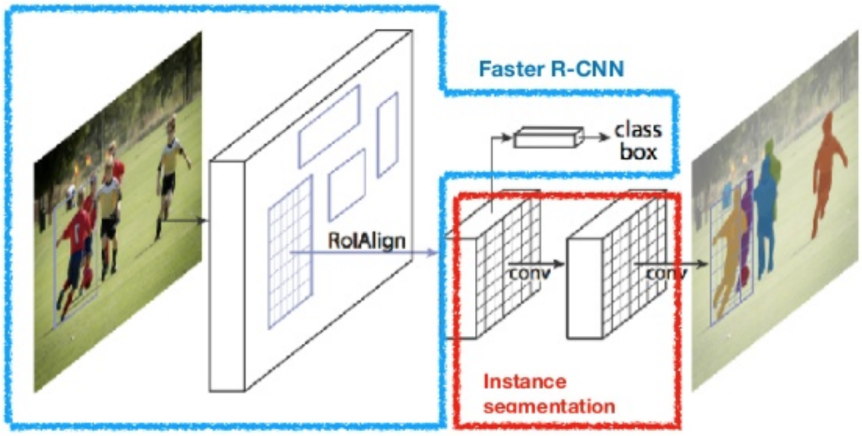

Figure 2.10: Structure of Mask R-CNN.
[**Credit**: *pythonawesome*]

proposal in parallel with region proposal, by adding a branch for predicting object masks in parallel with the existing branch for bounding box detection and classification.

The applications of instance segmentation are numerous. An example is the development of self-driving cars, which are required to have a detailed view of their surroundings and to recognize the objects in their field of view. A process of instance segmentation can help by detecting all objects (including pedestrians) and thus giving the car the information it needs. Another example can be found in the medical field, where detecting and segmenting anomalies in medical scans can be of great use.

Another field of application for instance segmentation is the detection of objects within aerial images. The field has received much attention with the release of datasets such as DOTA [22], iSAID [23] and Visdrone [24], and many existing works have designed frameworks and methods to obtain satisfying results for both object detection [25, 26, 27] and instance segmentation [28, 29, 30].

A specific use case of aerial image segmentation, and the main objective of this work, is the **detection and mapping of solar panels** in aerial images. Having a comprehensive and up-to-date database of the location of solar panels is essential for policy making and financial assessment, but such maps are not always available, especially for large areas, making automatic detection a valid choice for building them. In literature, a few works [31, 32, 33] have addressed the task, with the prevalent method of choice being semantic segmentation. Few methods employed instance segmentation, and part of the purpose of the work of this thesis is to propose a method that tackles this task specifically.

2.3.2 Learning with low resources

As mentioned earlier, training an algorithm with small datasets can prove challenging or even impossible, since the model is not provided with enough information to make reasonable predictions on real data; furthermore, the labeling process is often time- and resource-consuming, meaning there is often a scarcity of labeled data. Thus, using non-fully supervised methods such as semi-supervised training or domain adaptation can be of great help in training models with few resources available.

In the field of semantic segmentation, several methods [34, 35, 36, 37] have obtained competitive results on challenging benchmarks while requiring much less annotations effort than fully supervised methods. Others, like the frameworks DACS and DAFormer, proposed by Tranheden et al. [38] and Hoyer et al. [39] respectively, address the issue of data scarcity by making use of synthetic data, which is much easier to obtain and label, to train algorithms. In particular, the main contribution of DACS is the mixing of synthetic and real data, along with the corresponding labels and pseudo-labels. DAFormer further improves on this idea by employing Transformers [40, 41] and particular training strategies.

The problem of limited data in instance segmentation is aggravated by the need of bounding box annotations in addition to pixel-wise mask annotations. However, while masks are rather expensive to annotate, bounding boxes prove to be cheaper to produce. The ShapeProp framework, proposed by Zhou et al. [42], exploits the abundance of box annotations by using a model that learns to extract salient regions from the object detection output and mixes them with the few mask annotations to extract shape representation. The work of Tian et al. [43] is also based on this and proposes a framework which supervises mask training without relying on any mask annotations. This is achieved by redesigning the loss used for mask learning, without any modifications to the segmentation network.

Other methods employ the use of pseudo-labels generated by the model itself on unlabeled images, mixing them with existing labeled data. However, the boundaries of the generated masks and boxes often contain a lot of noise, but the work of Wang et al. [44] points out that such noisy boundaries are double-edged, and thus aims to exploit the positive aspects while mitigating the negatives.

Finally, the field of aerial instance segmentation presents some additional challenges: objects in aerial images are often too small for traditional instance segmentation methods to perform well, and there are few datasets with mask annotations, most of them being designed for object detection and therefore having only box annotations. The work of de Carvalho et al. [45] addresses these two problems by proposing, among other methodologies, a bounding box-free instance segmentation method which exploits object interiors and contours to isolate them and generate separate instances. On the other hand, the work of Li et al. [46], similarly to [43],

exploits solely box annotations by using a box-supervised method, and tackles the issues of aerial image segmentation by proposing a level set method which is able to accurately recover object boundaries and distinguish each instance from the background.

The main limitation of the above works is that only bounding box annotations are used, without exploiting any segmentation masks. The two approaches outlined in this thesis both aim to use both types of annotations present in the dataset, with the first being the Noisy Boundaries framework described above, and the other employing a semi-supervised method that generates pseudo-labels and makes use of instance mixing inspired by the techniques used in Unsupervised Domain Adaptation.

Chapter 3

Data Sources

This section provides an overview of existing state-of-the-art datasets (Section 3.1) for the purpose of instance segmentation and, specifically, aerial instance segmentation, together with the datasets used specifically for the purpose of this thesis (Section 3.2). In particular, it also describes the processes of data preparation (Section 3.3) employed for the supervised and the semi-supervised settings to generate the datasets.

3.1 Existing datasets

3.1.1 COCO

The COCO (Common Objects in COntext) dataset, presented in the work of Lin et al. [47], was created to help advance the state-of-the-art in object recognition tasks, including image classification, object detection, semantic segmentation and instance segmentation. The dataset contains 2.5 million images depicting everyday scenes containing common objects in their natural context.

All images are labeled using per-instance segmentations, making them particularly useful for precise object localization and also making them useful for all the aforementioned tasks. The dataset contains natural photos of 91 different object types, namely people, animals and common objects of varying sizes and shapes.

3.1.2 DOTA and iSAID

DOTA (Dataset for Object deTection in Aerial images), presented by Xia et al. [22], is a dataset with the specific purpose of advancing research in Earth Vision applications. As the name suggests, DOTA is a collection of about 2.800 aerial images, gathered from several sources and platforms, and contains about 188.000

Figure 3.1: A collection of annotated images from the COCO dataset. [**Credit**: *researchgate*]

instances, each annotated with an oriented bounding box and a corresponding class.

Figure 3.2: A collection of annotated images from the DOTA dataset. [**Credit**: *captain-whu.github.io*]

iSAID (Instance Segmentation in Aerial Images Dataset), presented by Waqas et al. [23], builds on DOTA to create a dataset with essentially the same purpose, but for instance segmentation. The dataset contains the same 2.800 images from DOTA, but was annotated from scratch for two reasons: first, DOTA only contains bounding box annotations, making it unsuitable for instance segmentation; second, DOTA also suffers from problems such as missing instance annotations or inaccurate bounding boxes. The result is that iSAID contains more than 688.000 annotated instances, more than double the original number of instances.

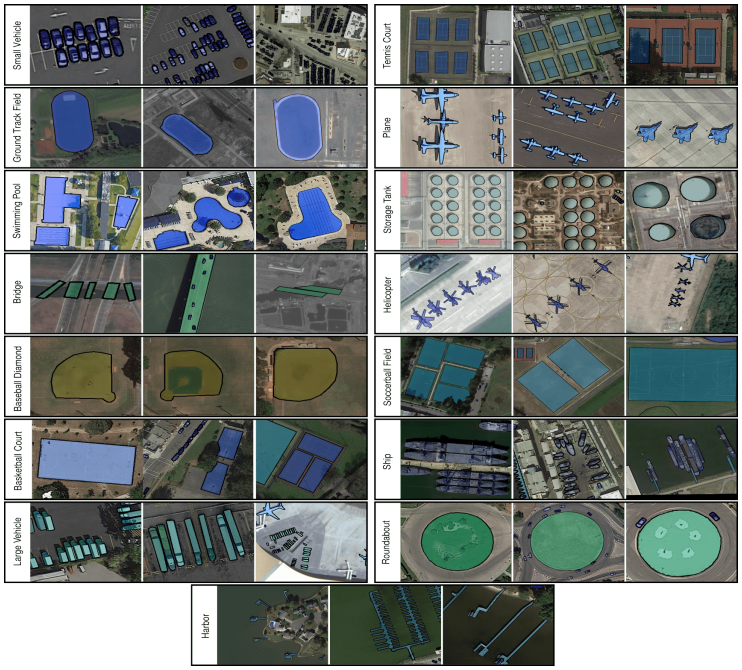

Figure 3.3: A collection of annotated images from the iSAID dataset. [**Credit**: *captain-whu.github.io*]

The dataset has been tested with Mask R-CNN and PANet [21, 48], but the results proved to be suboptimal, indicating that specialized solutions are necessary for instance segmentation in aerial images.

3.2 Solar panels dataset

The dataset used for this work also comprises high-resolution aerial images. Specifically, it is an orthophoto dataset comprising two large areas in the Piedmont region in Italy, including the provinces of Asti and Alessandria. These areas are further split into 105 different georeferenced tiles in GeoTIFF format: each tile covers a wide area of about 6×4.8 Km with a resolution of 30cm/pixel, equivalent to

an average size of 20.000 × 16.000 pixels, and provides four bands, namely Red, Green, Blue and Infrared (RGB-IR). As for annotations, the dataset was manually annotated with more than 9.000 panels provided as a shapefile, along with useful metadata such as panel category, its extent and the plant to which it belongs, as shown in Table 3.1.

	ID	Plant ID	Region	Province	Type	Category	Orientation	Power	Geometry
0	16346	777777777	Piemonte	Asti	Industriale	Poli cristallino	Sud-Est-Ovest	947.60	POLYGON ((8.11165 45.03408, ...
1	16348	777777777	Piemonte	Asti	Industriale	Poli cristallino	Sud-Est-Ovest	947.60	POLYGON ((8.11142 45.03477, ...
2	6908	1648	Piemonte	Asti	Industriale	Mono cristallino	Sud	100.91	POLYGON ((8.12238 44.86116, ...
3	12013	1727	Piemonte	Alessandria	Industriale	Mono cristallino	Ovest	105.69	POLYGON ((8.44282 44.95028, ...
4	14625	1781	Piemonte	Asti	Industriale	Poli cristallino	Sud	575.77	POLYGON ((8.32086 44.87613, ...

Table 3.1: Example of metadata associated with the panels.

In particular, each individual panel (i.e. each instance) is classified either as monocrystalline or polycrystalline, which are the most common categories of photovoltaic (PV) panels. The main difference between them is the configuration of silicon: in monocrystalline panels, each PV cell is made of a single silicon crystal, while in polycrystalline each cell is made up of multiple silicon crystals that are melded together during manufacturing. This classification is crucial for the purpose of delineation, given that the silicon structure affects their performance and appearance.

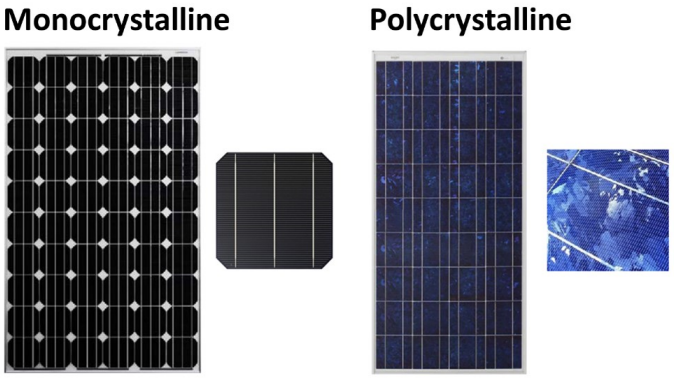

Figure 3.4: Comparison between polycrystalline and monocrystalline solar panels. [**Credit**: *prostarsolar.net*]

Monocrystalline panels are characterized by black PV cells with rounded edges and have a higher conversion efficiency. They are also more expensive because the manufacturing process is more complex.

Polycrystalline panels, on the other hand, have blue PV cells with straight edges and have a lower efficiency than monocrystalline panels, meaning that more panels have to be employed to achieve the same power; furthermore, they are also more

affected by high temperatures, reducing their effectiveness on hot days. However, they are also cheaper because the manufacturing process is simpler.

Within the dataset, monocrystalline panels are much more scarce than polycrystalline panels, with the latter making up roughly 95% of available annotations, making it difficult for the model to accurately detect and classify the former.

Another challenge presented by the dataset is precisely the total number of available annotations, which is too limited to properly train a model on the dataset.

3.3 Data preparation

The generation of the train/test/validation split for training followed two slightly different processes for supervised training and semi-supervised training, which will be explained in this section.

In both cases, two different versions of the dataset were produced: one with RGB-only images and one with RGB-IR images. The number of images produced is the same, with the only difference being the number of bands. This means that, ultimately, four different datasets were used: two (RGB and RGB-IR) for fully-supervised training and two (RGB and RGB-IR) for semi-supervised training.

3.3.1 Supervised setting

Figure 3.5: Example images from the training set of the supervised dataset, showing annotations for monocrystalline (in red) and polycrystalline (in blue) panels.

In regards to the supervised setting, each of the 105 large tiles is split into patches of 512×512 pixels: this size was chosen as a compromise between including the maximum panels possible in each patch and not slowing down training too much. Patches are assigned to a set depending on the province they belong to: tiles comprising parts of the province of Asti are used exclusively for the test set, while

tiles comprising parts of Alessandria are used for both the training and validation sets. The reason behind this decision is that the number of panel instances in Asti is much smaller than the number of panels in Alessandria, thus providing the model with a large number of images for training and validation; however, the distribution of mono and polycrystalline panels remains comparable.

To generate patches, the procedure analyses each tile and checks whether it intersects any panels. If no intersection is found, the tile is discarded. Otherwise, the script gathers information from the shapefile metadata about every plant included in the tile, and then iterates them with a loop. For each plant, the procedure centers the window on the plant, outlining its boundaries and operating solely within them; then, it computes the number of patches based on the width and height of the plant boundaries and finally computes an overlap by dividing the remainder by the number of patches. This decision was made in order to maximize the number of panels contained in each image, thus improving the quality of the sets.

During this windowing process, the procedure checks whether the current patch intersects any panel from the shapefile; if there is no intersection, the patch is skipped. Otherwise, the patch is saved as a raster file in TIFF format.

	Train	Val	Test
Number of images	519	95	169

Table 3.2: Number of images for each set for the supervised setting.

Patches generated from the province of Asti were all assigned to a test set, while patches generated from the province of Alessandria were split into a training and a validation set with percentages 70/30 respectively, with the end result detailed in Table 3.2.

As for annotations, panel metadata from the shapefile was converted into a COCO-compliant format compatible with the adopted framework, described in Section 5.1.1.

3.3.2 Semi-supervised setting

As mentioned earlier, the pre-processing method for the semi-supervised setting was organized slightly differently because it was necessary to also generate unlabeled images.

In the case of the training set, the procedure has to save both labeled and unlabeled images to be used for semi-supervised training. While labeled images are generated in the same manner as described for the supervised setting, the procedure for generating unlabeled images cannot be the same employed for the other sets,

since centering on plants would make obtaining images without any panels very unlikely. Thus, another windowing process was adopted: this procedure takes tiles from Alessandria and generates patches with a fixed overlap of 256 (half the size of the sides of a single patch) without centering on plants. This means that it has to iterate through the entire tile, greatly increasing the number of images produced. For each patch the procedure checks whether it intersects any panels at all; if it intersects at least one panel, then the image is skipped; otherwise, if no intersection is found it is saved as an unlabeled image, with no associated annotations. The end result is detailed in Table 3.3.

	Train - labeled	Train - unlabeled	Val	Test
Number of images	519	55647	95	169

Table 3.3: Number of images for each set for the semi-supervised setting.

Chapter 4

Methodology

This chapter presents the problem statement for this work (Section 4.1) and describes the process followed to obtain baselines in a fully supervised setting (Section 4.2), together with the semi-supervised methods employed to obtain better results (Section 4.3).

4.1 Problem statement

This work addresses the problem of semi-supervised domain adaptation in an instance segmentation setting by employing semi-supervised techniques for a dataset of aerial images containing solar panels. Specifically, the goal within the task of instance segmentation is to first locate instances of panels within the image, and then assign a class to and segment each instance by producing a segmentation mask which delineates its shape. The end result is the training of a model which, given an aerial image external to the training dataset as input, can accurately predict the location, class and precise shape of each panel instance, if any, contained in the image, producing a set of bounding boxes, class predictions and segmentation masks as a result.

Let S_l be the set of labeled images, containing N_l samples, and S_u be the set of unlabeled images, containing N_u samples. A batch of b images and ground truth labels, X_l and Y_l respectively, are sampled from S_l, while a batch of b images X_u are sampled from S_u.

Let n_l be the number of instances associated with the image $x_l \in X_l$. The labels $y_l \in Y_l$ associated with x_l are defined as $y_l = \{(b_i, m_i, c_i) | i \in [1, n_l]\}$, where b_i, m_i and c_i represent the bounding box, mask and class annotation for the i-th instance, respectively.

The images X_u are fed to the network f_θ to generate a batch of pseudo-labels $\hat{Y}_u$. The algorithm then follows the standard procedure of a supervised learning

approach, by computing predictions, performing backpropagation and a step of gradient descent, executing all of these operations with both labeled images and ground truth labels X_l and Y_l and unlabeled images and pseudo-labels X_u and $\hat{Y}_u$. This process is repeated for a set number of iterations N.

4.2 Baselines

Baselines for the supervised setting were obtained by training Mask R-CNN [21] on the datasets described in Section 3.3.1. Both RGB and RGB-IR datasets were used, with slightly different results.

As briefly described in Section 2.3.1, the process and architecture of Mask R-CNN follow those of Faster R-CNN, with the addition of a mask head to generate instance mask predictions for instances contained in each RoI.

Specifically, it consists of two stages. The first stage is the Region Proposal Network (RPN), which is a CNN that takes an image as input and outputs a set of candidate object bounding boxes.

The second stage is made up by two components that work in parallel with each other: the first is essentially the architecture of Fast R-CNN, which extracts features using RoIPool from each candidate box to predict classes and bounding boxes for each; the second is a component that outputs a binary mask for each RoI. While useful for classification and bounding box predictions, the RoIPool operation has a negative impact on the prediction of pixel-accurate masks. Mask R-CNN therefore introduces *RoIAlign* to align the extracted features with the input to improve the quality of mask predictions.

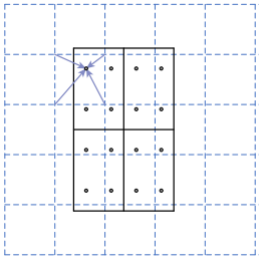

Figure 4.1: Graphical representation of the RoIAlign operation in Mask R-CNN. The dashed grid represents a feature map, the solid lines an RoI and the dots the four sampling points in each bin.

In particular, *RoIAlign* exists to address the problem of misalignments between the RoI and the extracted features due to the quantizations introduced by the RoIPool operation. This is achieved by avoiding any quantizations of the RoI boundaries or bins and by using bilinear interpolation to compute the exact values

of the input features at four regularly sampled locations in each RoI bin. The results of this operation are then aggregated by using max or average.

Since the base version of Mask R-CNN works with RGB images, its input layer has to be adapted for it to accept RGB-IR images, which have 4 channels total. This is carried out by simply copying the weights of the layer associated with the red channel of the input to a fourth channel, as is common practice in similar computer vision tasks [49].

4.3 Semi-supervised methods

This section describes the two semi-supervised methods employed to address the issue of data scarcity in the proposed solar panels dataset. In particular, it first outlines the methodology employed by the Noisy Boundaries framework and then describes an approach designed for this work, referred to as IDACS (Instance-DACS).

4.3.1 Noisy Boundaries

One of the semi-supervised approaches employed was designed by adapting the framework of Noisy Boundaries [44] to the solar panels dataset.

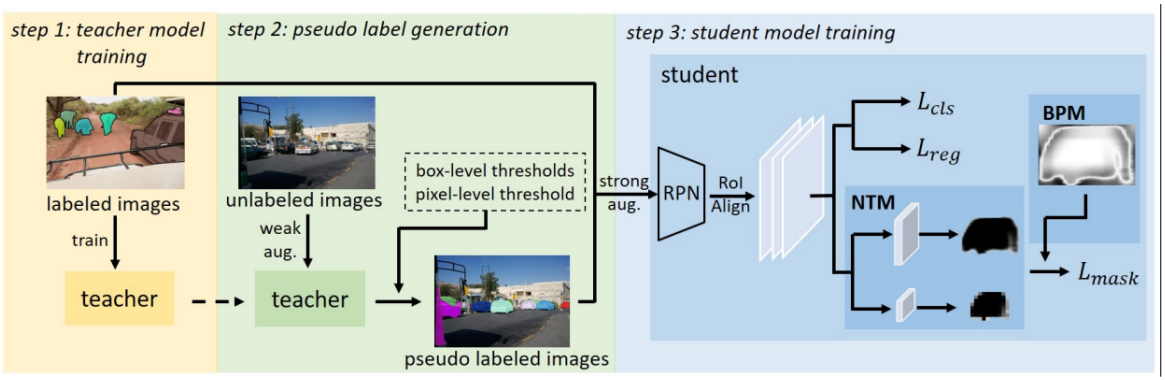

Figure 4.2: Framework for semi-supervised instance segmentation as detailed in the Noisy Boundaries paper.

The Noisy Boundaries framework is composed of three main steps. First, a teacher model is trained on the set of labeled images by also employing Mask R-CNN as the model; said teacher is used to generate pseudo-labels on images on which a student model is later trained.

Second, the pre-trained teacher model is used to make inference on unlabeled images to generate pseudo-labels. This step makes use of data augmentation such

as scaling or horizontal flipping (referred to as "weak augmentations") to improve mask quality and reduce miscalibration of neural networks. To acquire generated pseudo-labels, the raw inference mask is processed by comparing scores with two kinds of thresholds: a box-level threshold and a pixel-level threshold. However, these thresholds are not fixed and are computed to match the distribution between labeled and unlabeled images: this is done to prevent bias towards dominant classes in the detection branch and to prevent the imbalance between foreground and background pixels from affecting predictions.

In the third and final step, the student model is trained on labeled images and pseudo-labeled images, with both sets being subjected to strong augmentations, in contrast to the weak augmentations applied to unlabeled data in the previous step.

The contribution of the Noisy Boundaries framework also comprises two additional components, both employed within the student model's pipeline: a *Noise-Tolerant Mask head* (NTM), which helps the model better resist the noise inherently existent in pseudo-labels, and a *Boundary-Preserving Map* (BPM), which improves the model's ability to learn from boundary-related regions.

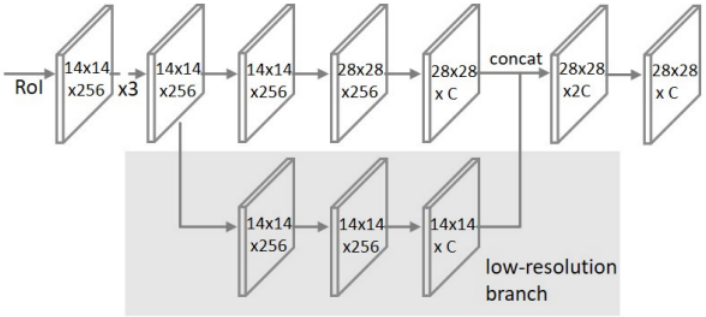

Figure 4.3: Structure of the Noise-Tolerant Mask head as shown in [44].

Specifically, the main idea behind the NTM is that noise is more prevalent in boundary-related regions, the details and features of which are visible only when the mask resolution is high enough. If the size and the resolution of the RoI are small enough, image details, where noise mainly lies, can be implicit. For this reason, the NTM adds a branch for low-resolution mask prediction: with a smaller size and lower resolution, its features are cleaner and more noise-resistant, meaning it is able to use more accurate information. However, the predicted segmentations are coarse and usually do not preserve details, so the original high-resolution mask head, which aims to learn fine-grained information, is still retained. The features from the low-resolution branch are fused into the high-resolution branch.

On the other hand, the BPM was designed to preserve information derived from instance boundaries, which are essential to the quality of predicted masks. To promote boundary learning, the model should focus more on pixels close to the

boundaries, but pixels that are most likely to be noisy are the ones extremely close to the boundaries. Therefore, the goal of the BPM is to re-weigh mask loss for pixels, by assigning to each pixel a value that is inversely proportional to their distance to boundaries, while at the same time assigning a low value to pixels in extreme proximity to boundaries to suppress noise.

For the purpose of this work, the framework was slightly adapted to fit the requirements for the task. First, it was modified to handle both RGB and RGB-IR images, in the same way Mask R-CNN was adapted for the baselines; second, the second step of the framework used the solar panel dataset's unlabeled image set to generate pseudo-labels.

Algorithm 1 Noisy Boundaries algorithm

Require: Labeled set S_l, unlabeled set S_u, instance segmentation networks t_θ, s_γ.
Initialize network weights θ, γ with pre-existing values θ', γ'.
for $i = 1$ to N_t **do**
  $X_l, Y_l \sim S_l$
  Train teacher network t_θ on labeled batch X_l, Y_l
end for
$\hat{S}_u \leftarrow \emptyset$
for $j = 1$ to N_u **do**
  $x_u \sim S_u$
  $\hat{y}_u \leftarrow t_\theta(x_u)$ ▷ Generate pseudo-labels on unlabeled image
  $\hat{y}'_u \leftarrow$ Pseudo-labels filtered on box-level thresholds T_b and pixel-level thresholds T_p
  Add unlabeled image and pseudo-label pair $(x_u, \hat{y}'_u)$ to $\hat{S}_u$
end for
for $k = 1$ to N_s **do**
  $X_l, Y_l \sim S_l$
  $X_u, Y_u \sim \hat{S}_u$
  Train student network s_γ on labeled and pseudo-labeled batches X_l, Y_l, X_u, Y_u
end for
return s_γ

The algorithm for the Noisy Boundaries framework is detailed in pseudocode in Algorithm 1.

4.3.2 IDACS

The approach designed specifically for this work is a semi-supervised method that combines the framework DACS and instance mixing techniques, and was thus

named *IDACS (Instance-DACS)*.

As mentioned in Section 2.3.2, *Domain Adaptation via Cross-domain mixed Sampling (DACS)* [38] is a framework for semantic segmentation tasks that addresses the problem of Unsupervised Domain Adaptation (UDA). In particular, DACS attempts to address the issue of low-quality pseudo-labels arising from the domain shift by mixing images from the source and target domain along with the corresponding labels and pseudo-labels. The resulting samples are then trained on, along with the existing labeled data.

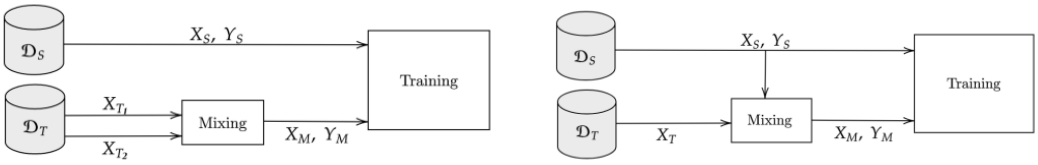

Figure 4.4: Diagrams comparing the process of "Naive Mixing" (left) and DACS (right).

Works prior to DACS that dealt with UDA applied an approach referred to as "Naive Mixing", where target-domain samples are mixed to generate augmented images and corresponding pseudo-labels. However, this solution performed poorly in practice, causing a problem referred to as class conflation, in which classes with fewer occurrences are confused with more frequent and similar classes.

Figure 4.5: Example of mixing in DACS, with X_S, X_T and X_M being source, target and mixed images, respectively.

Thus, the main idea behind DACS is to mix images across domains: target-domain samples are first fed to the network to produce pseudo-labels; then, augmented images are generated by mixing a set of pixels from a source domain sample and another set of pixels from a target domain sample, along with any ground truth labels or pseudo-labels associated with each set. These images are then used to train the network.

In the context of this work, the source domain and the target domain were made up of labeled and unlabeled images, respectively, from the same dataset and domain. The mixing process had to be adapted for instance segmentation by means of instance mixing.

The work of Ghiasi et al. [50], referred to as *Copy-Paste*, best summarizes the approach of **instance mixing** adopted for this work. The method can be thought of as a simple copy-paste operation of instances from one image to another. While in DACS mixing is performed by pasting pixels belonging to a specific semantic class on the target image, which can thus contain multiple instances of the same type, in instance mixing this is done by cutting out instances separately from one another and pasting them on the target image.

Figure 4.6: Example of the Copy-Paste procedure.

The process proposed by Copy-Paste is very simple. Two images are randomly selected and augmentations such as random scale jittering and random horizontal flipping are applied to each of them. Then, a random subset of instances from one images is selected and pasted on the other. Finally, ground-truth annotations are adjusted accordingly, by removing fully-occluded objects and updating their mask and bounding box annotations.

Testing for Copy-Paste was done on COCO [47], which, as explained earlier in Section 3.1, is a dataset containing natural photos depicting common objects in their everyday context. This means that geometric augmentations, such as rotations, are not suited for this dataset, since certain transformations could mislead the model during its training by generating potentially unnatural scenes. However, in the setting of aerial images and, specifically, of solar panel detection and segmentation, such problems do not exist, since panels are always viewed from above, so for the purpose of this work augmentations such as rotation, shifting and vertical flipping were applied to the source images before performing instance mixing. Furthermore,

Figure 4.7: Example of mixing in the solar panels dataset, showing, from left to right, source, target and mixed images.

only a percentage of panel instances were selected before mixing to add more variety. An example of mixing in the solar panels dataset is shown in Figure 4.7.

Algorithm 2 IDACS algorithm

Require: Labeled set S_l, unlabeled set S_u, instance segmentation network f_θ. Initialize network weights θ with pre-existing values θ'.

1: **for** $i = 1$ to N **do**
2: $X_l, Y_l \sim S_l$
3: $X_u \sim S_u$
4: $\hat{Y}_u \leftarrow f_\theta(X_u)$ ▷ Generate pseudo-labels on unlabeled batch
5: $\hat{Y}'_u \leftarrow$ Selection of pseudo-labels from $\hat{Y}_u$ where confidence $\geq T$
6: $Y'_l \leftarrow$ Selection of n'_l out of $n_l (0 < n'_l \leq n_l)$ labels for each image $y_l \in Y_l$
7: $X_m, Y_m \leftarrow$ Result of mixing and augmentation of X_l, Y'_l with $X_u, \hat{Y}'_u$
8: Train f_θ on labeled and mixed batches X_l, Y_l, X_m, Y_m
9: **end for**
return f_θ

The implementation of IDACS is detailed as pseudocode in Algorithm 2, where X_m and Y_m represent the batch of images and labels generated as the result of instance mixing, Y'_l is the subset of labels selected from each labeled image and T is the threshold value for pseudo-label filtering.

Chapter 5

Experiments

This chapter presents the implementation details for the experiments ran for the purpose of this work (Section 5.1), together with an explanation of the metrics used to do evaluation (Section 5.2). Finally, the results of the experiments are presented in detail (Section 5.3), along with a qualitative comparison.

5.1 Implementation details

This section first gives an overview of the frameworks and tools used to run the experiments described in this chapter, together with a detailed description of the hyperparameters and the procedures used during the experiments, starting with the setup for the supervised setting and then moving on to the setup for experiments made with the Noisy Boundaries and IDACS frameworks.

5.1.1 Frameworks and tools

The following is a list of frameworks and tools used in the context of this thesis:

- *Python* is a high-level programming language, supporting multiple programming paradigms, including structured, object-oriented and functional programming. It is one of the most popular programming languages overall, commonly used in several computer science fields including artificial intelligence and machine learning.
- *QGis* is an open-source Geographic Information System (GIS) which allows the user to analyse, edit, compose and export spatial information through a graphical user interface. It displays information through different layers and it was useful to map out the distribution of panels in the areas of Alessandria and Asti.

- *pandas* and *GeoPandas* are open-source tools, based on Python, used to analyse and manipulate data, with the latter being an extension of the former with the purpose of facilitating the analysis and processing of geospatial data. They were mainly used to process the panel annotations contained in the shapefiles provided.

- *raster.io* is a library offering APIs based on n-dimensional arrays to read and write geospatial raster data organized in formats such as GeoTIFF. It was used, along with *pandas* and *GeoPandas*, during the pre-processing phase to build the solar panels datasets used for this work.

- *OpenCV* is an open-source computer vision library, written in C++ and offering APIs for languages such as Python, Java and MATLAB.

- *PyTorch* is a machine learning framework based on the Torch library, frequently used to develop deep learning software. One of the main features it offers is computing through tensors, a custom class similar to NumPy arrays which can be operated on with an NVIDIA GPU supporting CUDA.

- *MMDetection* is an open-source object detection and instance segmentation library based on PyTorch, and it is part of the OpenMMLab project. It supports most popular and contemporary object detection frameworks such as Faster R-CNN and Mask R-CNN and organizes its content modularly, simplifying the construction of a custom object detection framework by combining, creating and editing different modules. It was used as the basis to implement frameworks and run experiments.

- *MMCV* is a library developed for computer vision research as part of the OpenMMLab project, offering functionalities such as IO APIs, image and video processing and visualization of images with relative annotations. It is used extensively in the context of MMDetection to implement and support a variety of features.

- *Tensorboard* is a toolkit, developed as part of the TensorFlow platform, used in the field of machine learning and it offers visualization tools such as tracking of metrics (e.g. loss and accuracy), visualization of model graphs and more. In the context of this work, it was used to track the evolution of losses and average precision during training in order to evaluate model performances.

Every experiment was performed on a machine with an Intel Xeon Silver 4126 CPU and one NVIDIA RTX 2080Ti GPU.

5.1.2 Experimental setup

All experiments described in this chapter were done on the solar panels dataset described in Section 3.3, which comprises images of size 512×512 pixels. Unless specified otherwise, batch size was set at 10 as a compromise between shortening training times and GPU memory capabilities.

Within the training pipeline, images were augmented both geometric and photometric augmentations. Horizontal and vertical flip, random 90-degree rotation and transposition were chosen as geometric augmentations, while gaussian noise and blur (only one applied at a time) were used as photometric augmentations, all with a probability of 50% to be applied to each image. Geometric augmentations were set to be applied to all the channels of a single image, while photometric augmentations were applied solely to RGB channels, leaving the IR channel (if present) unchanged. Additional photometric distortions (which included randomly changing brightness, saturation, hue and contrast values) were also applied after these augmentations.

Finally, all models had their weights initialized with a checkpoint from a pre-trained ResNet model on DOTA [22], which does not provide RPN and RoI head weights.

Supervised setting

For the supervised setting and for both RGB and RGB-IR datasets, training was set to last 300 epochs, with an SGD optimizer and a learning rate set to $3e-3$, with a warm-up of 30 epochs and a warm-up ratio of $1e-4$. The learning rate was also set to decay from epoch 250 to epoch 295, while momentum was set at 0.9 and weight decay at $1e-4$.

Noisy Boundaries

As mentioned in Section 4.3.1, the Noisy Boundaries framework is made up of three steps. The hyperparameters used for the first and third steps were exactly the same as the ones detailed in the supervised setting, given that the former is essentially a supervised training of Mask R-CNN and the latter is again a supervised training with pseudo-labeled images and additional components.

IDACS

For IDACS, training was set to last 40.000 iterations instead of 300 epochs. This choice was made because the size of the dataset was computed by accounting for all possible combinations of labeled and unlabeled images, meaning a single epoch would have to iterate through about 22 million images, making training unfeasible

in a reasonable amount of time. Considering that 300 epochs for the previous datasets were equivalent to about 20.000 iterations, this number of iterations felt appropriate. The learning rate was kept at $3e-3$, while its warm-up was set to end at 500 iterations and its decay started at iteration 32000 and ended at iteration 35000. Batch size was reduced to 6 due to limited GPU memory capabilities.

Before mixing, 75% of instance annotations from labeled images were selected, and they were augmented with additional augmentations in order to increase variety in the resulting mixed images. Namely, the set of geometric augmentations described earlier was used, along with additional affine augmentations (with a 50% probability of being applied), which included translation, free rotation, scaling and shearing; however, transformations such as translation bear the risk of producing images without any annotations, so in cases where this happened only the result of geometric augmentation was kept.

As for pseudo-labels, the confidence threshold for generated predictions was set to 0.968, as done in [38], in order to keep only the model's best predictions. Furthermore, two kinds of experiments were run: one kind where pseudo-labels are generated starting from the very first iteration, and another where they are generated after 5.000 iterations. This choice was made in order to see whether it is better for the model to start right away with generating pseudo-labels or to delay generation in order for the network to first learn only with ground truth and mixed annotations and possibly generate pseudo-labels with a higher quality. This means that four experiments in total were run with this setting: two with the RGB dataset and two with the RGB-IR dataset.

5.2 Evaluation metrics

The performance of models performing object detection and instance segmentation is often evaluated by using *Mean Average Precision (mAP)*. The mAP formula is based on a few sub-metrics, namely Intersection over Union (IoU), the confusion matrix, precision and recall.

For each instance of each object in a given image, a region called *ground truth* must be defined, representing the actual location of the object inside the image. In the case of instance segmentation tasks, ground truths are defined both for bounding boxes and for segmentation masks.

Intersection over Union (IoU) is a metric that evaluates the precision of each prediction based on the overlap between the predicted region and the corresponding ground truth. It can be formally defined as follows:

$$IoU = \frac{A_I}{A_U} \tag{5.1}$$

where:

- A_I is the area of the intersection between the predicted region and the ground truth;
- A_U is the area of the union between the two regions.

A prediction is considered correct if the value of IoU is equal to or greater than a set threshold t_{IoU}, which typically has a value of 0.5.

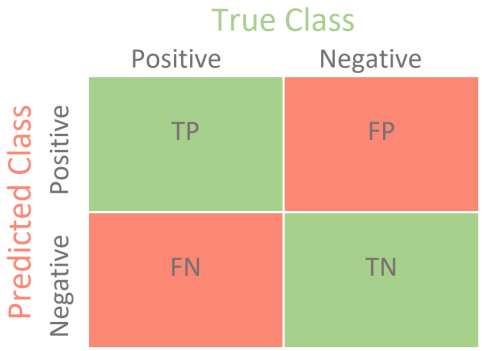

Figure 5.1: Example of a simple confusion matrix.
[**Credit**: *towardsdatascience*]

A *confusion matrix* is an $N \times N$ table (with N being the number of classes) used to evaluate the number of correct and incorrect predictions made by a model. In the case of object detection and instance segmentation, predictions are evaluated by using a confusion matrix in conjunction with IoU; specifically:

- *True Positive (TP)*: correct class prediction and $IoU \geq t_{IoU}$;
- *False Positive (FP)*: wrong class prediction or $IoU < t_{IoU}$;
- *False Negative (FN)*: the object is part of the ground truth, but no prediction was made;
- *True Negative (TN)*: the object is not part of the ground truth and no prediction was made.

The definitions of **precision** and **recall** are based on the number of predictions that fall into each of the above categories. Formally:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5.2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5.3}$$

In short, precision measures how many predictions out of the ones that predicted a positive value are correct, while recall measures how many predictions correctly predicted a positive value.

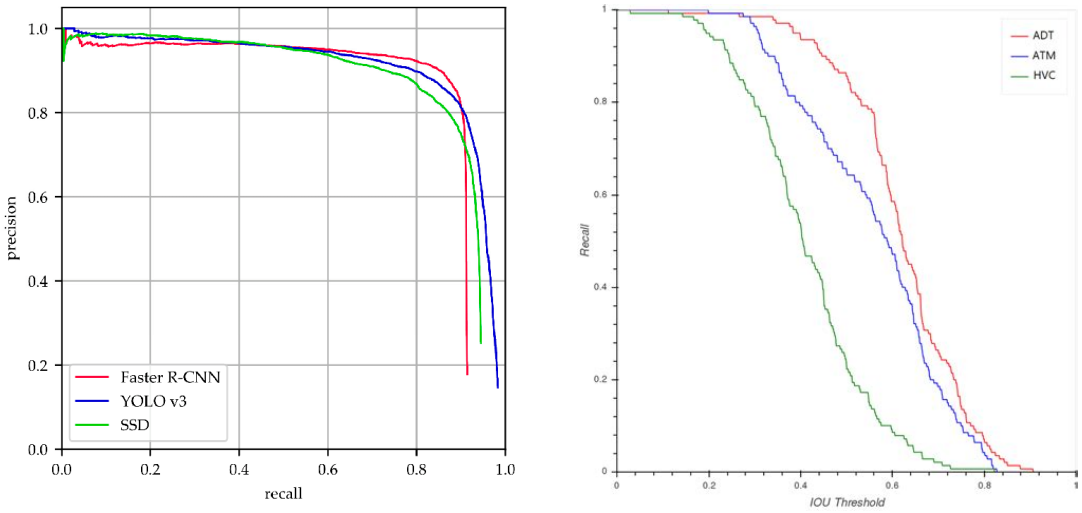

Figure 5.2: Examples of precision-recall curves (left) and recall-IoU curves (right). [**Credit**: *researchgate*]

The evolution of precision and recall can be represented graphically, generating *precision-recall curves*. The area underneath the curve is called *Average Precision (AP)*, and is a measure of the model's performance for a specific class. Similarly, the evolution of recall over multiple IoU values can be represented by means of *recall-IoU curves*, with IoU values usually ranging between 0.5 and 1; the area underneath this curve is called *Average Recall (AR)*.

Mean Average Precision (mAP), as the name implies, is defined as the mean of all APs for all classes:

$$mAP = \frac{\sum_{c \in C} AP_c}{|C|} \tag{5.4}$$

mAP is often denoted as mAP^{IoU}, where IoU is the threshold value used to identify True Positives and False Positives: the higher the threshold, the better the model will have to be at making predictions.

Mirroring mAP, **Mean Average Recall (mAR)** is defined as the mean of all ARs for all classes:

$$mAR = \frac{\sum_{c \in C} AR_c}{|C|} \tag{5.5}$$

```
Average Precision (AP):
  AP                  % AP at IoU=.50:.05:.95 (primary challenge metric)
  AP^IoU=.50          % AP at IoU=.50 (PASCAL VOC metric)
  AP^IoU=.75          % AP at IoU=.75 (strict metric)
AP Across Scales:
  AP^small            % AP for small objects: area < 32^2
  AP^medium           % AP for medium objects: 32^2 < area < 96^2
  AP^large            % AP for large objects: area > 96^2
Average Recall (AR):
  AR^max=1            % AR given 1 detection per image
  AR^max=10           % AR given 10 detections per image
  AR^max=100          % AR given 100 detections per image
AR Across Scales:
  AR^small            % AR for small objects: area < 32^2
  AR^medium           % AR for medium objects: 32^2 < area < 96^2
  AR^large            % AR for large objects: area > 96^2
```

Figure 5.3: COCO evaluation metrics for object detection.
[**Credit**: *cocodataset.org*]

In recent years, the COCO dataset has become the standard to evaluate object detection algorithms, and for this reason a specific evaluation metric has been proposed by its creators, as shown in Figure 5.3. While COCO uses mAP as the main evaluation metric, it is simply referred to as AP, and likewise mAR is referred to as AR. Unless specified otherwise, all metrics are computed by considering a maximum of 100 possible detections on a single image (this is specified in the *maxDets* column in the tables of Section 5.3).

The first three AP metrics are computed over multiple IoU values. Specifically, the first AP is computed over values of IoU ranging from 0.5 to 0.95, with incremental steps of 0.05, while the second and third are computed with IoU values above 0.5 and 0.75, respectively.

The fourth, fifth and sixth metrics, on the other hand, are computed across multiple object scales, by considering only small, medium and large objects respectively, with small objects being defined as having an area smaller than 32 square pixels, medium objects having an area included between 32 and 56 square pixels and large objects having an area larger than 56 square pixels. All three are computed by considering IoU values ranging from 0.5 to 0.95.

The first three AR metrics are computed by considering only 1, 10 or 100 detections per image respectively, and IoU values ranging from 0.5 to 0.95 for the first and IoU values above 0.5 and 0.75 for the second and third. Finally, the last

three AR metrics are computed across different scales, following the same criteria described for AP across scales.

5.3 Results

For the sake of simplicity, all comparisons between results are made by mainly taking into account the first and second metric in the list of COCO metrics, namely AP between IoU values of 0.5 and 0.95 (referred to as $AP_{0.5:0.95}$) and AP with IoU above 0.5 (referred to as $AP_{0.5}$). Furthermore, for comparison purposes a "significant increase" in metrics is defined as an increase of at least 0.015.

Supervised setting

Results for the supervised setting baselines are detailed in Table 5.1 for both the RGB and the RGB-IR datasets. As can be easily inferred, the RGB-IR dataset tends to produce better results than its RGB counterpart, with an increase over the latter of 0.02 for $AP_{0.5:0.95}$ for both masks and bounding boxes and about 0.05 and 0.04 for $AP_{0.5}$ for bounding boxes and masks, respectively. This is likely thanks to the extra information provided by the IR channel, and represents a trend that is repeated for the other settings as well.

Metric	IoU	Area	maxDets	BBox - RGB	Mask - RGB	BBox - RGB-IR	Mask - RGB-IR
Average Precision (AP)	0.50:0.95	all	100	0.258	0.224	0.278	0.244
Average Precision (AP)	0.50	all	100	0.403	0.369	0.450	0.406
Average Precision (AP)	0.75	all	100	0.302	0.255	0.302	0.270
Average Precision (AP)	0.50:0.95	small	100	0.256	0.218	0.275	0.243
Average Precision (AP)	0.50:0.95	medium	100	0.275	0.241	0.293	0.246
Average Precision (AP)	0.50:0.95	large	100	0.374	0.353	0.294	0.291
Average Recall (AR)	0.50:0.95	all	1	0.028	0.026	0.031	0.029
Average Recall (AR)	0.50	all	10	0.171	0.152	0.178	0.160
Average Recall (AR)	0.75	all	100	0.328	0.292	0.328	0.295
Average Recall (AR)	0.50:0.95	small	100	0.301	0.280	0.303	0.289
Average Recall (AR)	0.50:0.95	medium	100	0.377	0.323	0.386	0.305
Average Recall (AR)	0.50:0.95	large	100	0.575	0.430	0.605	0.375

Table 5.1: Results on the RGB and RGB-IR datasets for the supervised setting.

Noisy Boundaries

Results for the Noisy Boundaries setting are detailed in Table 5.2 for both the RGB and RGB-IR datasets. While there is a definite improvement with the RGB-IR dataset compared to its baseline counterpart, with an increase of 0.012 for $AP_{0.5}$ for bounding boxes and 0.052 and 0.043 for $AP_{0.5:0.95}$ and $AP_{0.5}$ for masks, results for the RGB dataset actually worsened in some cases, with a decrease in $AP_{0.5:0.95}$ and $AP_{0.5}$ for both bounding boxes and masks. A possible explanation for this

phenomenon is that since the teacher model has less accurate performances with the RGB dataset so are the pseudo-labels it generates, making it difficult for the student model to properly learn from what it considers ground truth annotations.

Metric	IoU	Area	maxDets	BBox - RGB	Mask - RGB	BBox - RGB-IR	Mask - RGB-IR
Average Precision (AP)	0.50:0.95	all	100	0.255	**0.245**	0.286	**0.296**
Average Precision (AP)	0.50	all	100	0.374	0.364	**0.462**	**0.449**
Average Precision (AP)	0.75	all	100	0.312	**0.301**	**0.325**	**0.370**
Average Precision (AP)	0.50:0.95	small	100	**0.239**	0.230	0.278	**0.285**
Average Precision (AP)	0.50:0.95	medium	100	**0.297**	**0.290**	0.296	**0.319**
Average Precision (AP)	0.50:0.95	large	100	**0.393**	**0.471**	0.239	0.284
Average Recall (AR)	0.50:0.95	all	1	0.032	0.030	0.029	0.028
Average Recall (AR)	0.50	all	10	0.172	0.164	0.172	0.171
Average Recall (AR)	0.75	all	100	0.306	0.296	**0.366**	**0.400**
Average Recall (AR)	0.50:0.95	small	100	0.273	0.278	**0.340**	**0.387**
Average Recall (AR)	0.50:0.95	medium	100	0.371	0.337	0.390	**0.389**
Average Recall (AR)	0.50:0.95	large	100	**0.590**	**0.530**	0.530	**0.475**

Table 5.2: Results on the RGB and RGB-IR datasets for the semi-supervised Noisy Boundaries setting. Metrics in bold represent a significant improvement over baselines.

IDACS

Results for the IDACS semi-supervised settings are detailed in Table 5.3 and Table 5.4 for the IDACS 0 and IDACS 5.000 settings, respectively, and for both RGB and RGB-IR datasets.

The IDACS 0 setting (i.e., the one with pseudo-label generation starting at the first iteration) performs slightly better than baselines, with an increase of 0.021 for $AP_{0.5:0.95}$ for masks for the RGB dataset and slight increments for other cases.

Generally speaking, however, delaying pseudo-label generation seems to improve performances, both for the RGB and RGB-IR datasets, with the IDACS 5.000 experiments presenting an increase of 0.028 (RGB) and 0.009 (RGB-IR) for $AP_{0.5:0.95}$ for bounding boxes and 0.036 (RGB) and 0.015 (RGB-IR) for $AP_{0.5:0.95}$ for masks. There was no significant increase for $AP_{0.5}$ except for masks for the RGB dataset, where there was an increase of 0.016. Despite this, the model generally performed better than baselines, with an overall increase in both AP and AR metrics all over the board, especially in the cases of AP and AR across scales, with a significant improvement in multiple metrics.

This increase in results for the model with delayed generation suggests that letting the model initially train solely on labeled data allows it to learn how to generate higher-quality predictions, while generating them from the start can result in the model misleading itself with its own inaccurate predictions. Interestingly enough, the model with delayed pseudo-label generation seems to perform slightly better on mask predictions with the RGB dataset rather than with the RGB-IR

dataset, while bounding box scores remain comparable overall between datasets.

Metric	IoU	Area	maxDets	BBox - RGB	Mask - RGB	BBox - RGB-IR	Mask - RGB-IR
Average Precision (AP)	0.50:0.95	all	100	0.269	**0.245**	0.272	0.245
Average Precision (AP)	0.50	all	100	0.372	0.358	0.375	0.358
Average Precision (AP)	0.75	all	100	**0.327**	**0.293**	**0.329**	**0.296**
Average Precision (AP)	0.50:0.95	small	100	0.249	0.229	0.257	0.233
Average Precision (AP)	0.50:0.95	medium	100	**0.324**	**0.298**	**0.328**	**0.293**
Average Precision (AP)	0.50:0.95	large	100	0.303	**0.436**	**0.324**	**0.397**
Average Recall (AR)	0.50:0.95	all	1	0.029	0.027	0.032	0.029
Average Recall (AR)	0.50	all	10	0.167	0.155	0.169	0.154
Average Recall (AR)	0.75	all	100	0.300	0.284	0.299	0.281
Average Recall (AR)	0.50:0.95	small	100	0.267	0.258	0.271	0.263
Average Recall (AR)	0.50:0.95	medium	100	0.384	0.361	0.373	**0.338**
Average Recall (AR)	0.50:0.95	large	100	**0.710**	**0.500**	0.605	**0.450**

Table 5.3: Results on the RGB and RGB-IR datasets for the semi-supervised IDACS setting, with pseudo-label generation at iteration 0. Metrics in bold represent a significant improvement over baselines.

Metric	IoU	Area	maxDets	BBox - RGB	Mask - RGB	BBox - RGB-IR	Mask - RGB-IR
Average Precision (AP)	0.50:0.95	all	100	**0.286**	**0.260**	0.287	**0.259**
Average Precision (AP)	0.50	all	100	0.395	**0.385**	0.396	0.378
Average Precision (AP)	0.75	all	100	**0.341**	**0.311**	**0.342**	**0.307**
Average Precision (AP)	0.50:0.95	small	100	**0.274**	**0.246**	0.260	0.238
Average Precision (AP)	0.50:0.95	medium	100	**0.332**	**0.303**	**0.370**	**0.328**
Average Precision (AP)	0.50:0.95	large	100	**0.392**	**0.439**	0.305	**0.381**
Average Recall (AR)	0.50:0.95	all	1	0.029	0.028	0.033	0.030
Average Recall (AR)	0.50	all	10	0.168	0.154	0.176	0.159
Average Recall (AR)	0.75	all	100	0.315	0.294	0.315	0.295
Average Recall (AR)	0.50:0.95	small	100	0.289	0.274	**0.278**	**0.269**
Average Recall (AR)	0.50:0.95	medium	100	0.379	**0.354**	**0.410**	**0.367**
Average Recall (AR)	0.50:0.95	large	100	**0.615**	**0.465**	**0.645**	**0.440**

Table 5.4: Results on the RGB and RGB-IR datasets for the semi-supervised IDACS setting, with pseudo-label generation at iteration 5.000. Metrics in bold represent a significant improvement over baselines.

5.3.1 Comparison between experiments

A comparison of the results obtained is reported in Table 5.5, put together by considering the metrics mentioned earlier for both bounding boxes and masks.

By taking $AP_{0.5}$ as the comparison metric, the best results are obtained with the RGB-IR Noisy Boundaries setting, for both masks and bounding boxes, while IDACS settings unfortunately do not outperform neither Noisy Boundaries nor baselines.

On the other hand, by taking $AP_{0.5:0.95}$ as reference, all semi-supervised methods outperform baselines (with the exception of the RGB Noisy Boundaries setting, presumably for reasons stated earlier). In particular, for bounding boxes the best

Method	$AP_{0.5:0.95}$ - BBox	$AP_{0.5}$ - BBox	$AP_{0.5:0.95}$ - Mask	$AP_{0.5}$ - Mask
RGB - baseline	0.258	0.403	0.224	0.369
RGB-IR - baseline	0.278	0.450	0.244	0.406
RGB - IDACS 0	0.269	0.372	0.245	0.358
RGB-IR - IDACS 0	0.272	0.375	0.245	0.358
RGB - IDACS 5.000	**0.286**	0.395	0.260	0.385
RGB-IR - IDACS 5.000	**0.287**	0.396	0.259	0.378
RGB - Noisy	0.255	0.374	0.245	0.364
RGB-IR - Noisy	**0.286**	**0.462**	**0.296**	**0.449**

Table 5.5: Comparison between performed experiments by using $AP_{0.5:0.95}$ and $AP_{0.5}$ as reference metrics.

results were obtained by the RGB-IR Noisy Boundaries setting and both IDACS 5.000 settings, with all three obtaining almost the same score and a definite increase from the baselines. This discrepancy in results for bounding boxes obtained with IDACS in different IoU values could suggest that it makes fewer predictions than Noisy Boundaries, but that these few tend to be of a higher quality, meaning it is more capable of correctly locating instances within images. This assumption is corroborated by the fact that if a stricter metric like $AP_{0.75}$ (i.e. AP computed by setting the IoU threshold to 0.75, which is the third AP metric) is taken as reference, results with IDACS tend to be better.

For masks, the best results with $AP_{0.5:0.95}$ were obtained with the RGB-IR Noisy Boundaries setting, likely because of the NTM and BPM components, described in Section 4.3.1, employed by the framework, although all other semi-supervised settings still outperform baselines.

In conclusion, while the best overall results were obtained with the RGB-IR Noisy Boundaries semi-supervised setting, the IDACS settings, especially IDACS 5.000, do perform rather well with both datasets and hold a lot of potential. In fact, by taking metrics other than $AP_{0.5:0.95}$ and $AP_{0.5}$ into account, it can be noted that there is a notable difference between the Noisy Boundaries and IDACS settings: the latter tends to obtain better results in both AP and AR metrics across scales, with significant increases in metrics over baselines in both bounding boxes and masks across all datasets. For instance, by taking AP over medium objects as reference metric, we can see that IDACS 5.000 performs better than Noisy Boundaries, with a difference of 0.035 (RGB) and 0.013 (RGB-IR) for bounding boxes and 0.074 (RGB) and 0.009 (RGB-IR) for masks. Furthermore, overall IDACS 5.000 was able to make improvements on the RGB dataset as well, on which Noisy Boundaries was not able to produce satisfying results as stated earlier.

5.3.2 Qualitative comparisons and observations

Qualitative comparisons of the results obtained on the test set are shown in figures from 5.4 to 5.9. Specifically, Figures 5.4, 5.6 and 5.8 depict ground truth annotations for industrial, rural and urban panels, respectively, while Figures 5.5, 5.7 and 5.9 depict inference result on those same images. These three settings were chosen to show how the models perform on different panel configurations.

An interesting phenomenon is depicted in Figure 5.10. As mentioned, the solar panel dataset suffers from a scarcity of annotations, meaning test set images might have some areas where panels are not annotated despite being plausibly present. In these cases, the models can still predict the presence of such panels, though they are penalized because no ground truth annotations were provided for them.

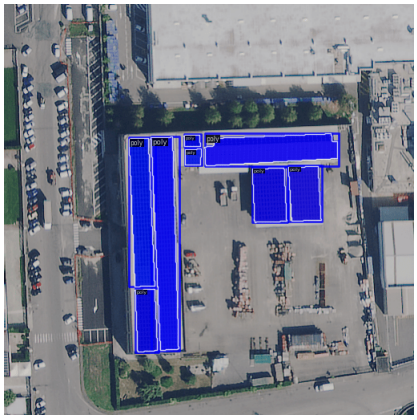

Figure 5.4: Ground truth annotations for a sample image from the test set depicting an industrial panel plant.

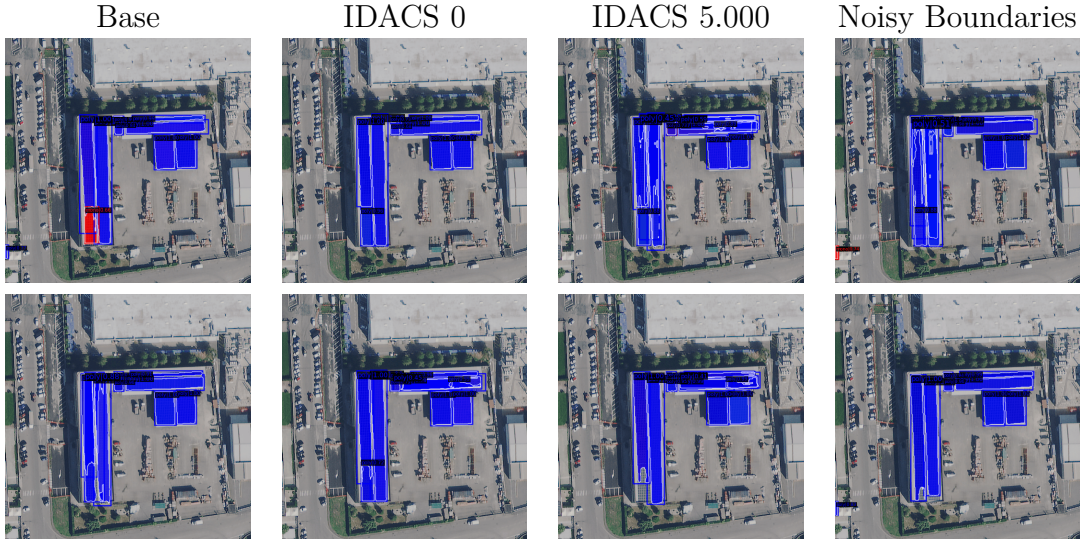

Figure 5.5: Comparison of inference results between methods on a sample industrial panel plant. The first row depicts results for methods ran on the RGB dataset, while the second depicts result for the RGB-IR dataset.

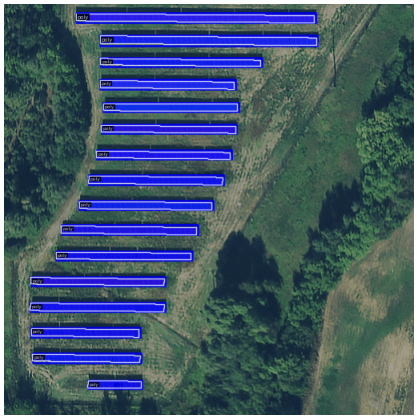

Figure 5.6: Ground truth annotations for a sample image from the test set depicting a rural panel plant.

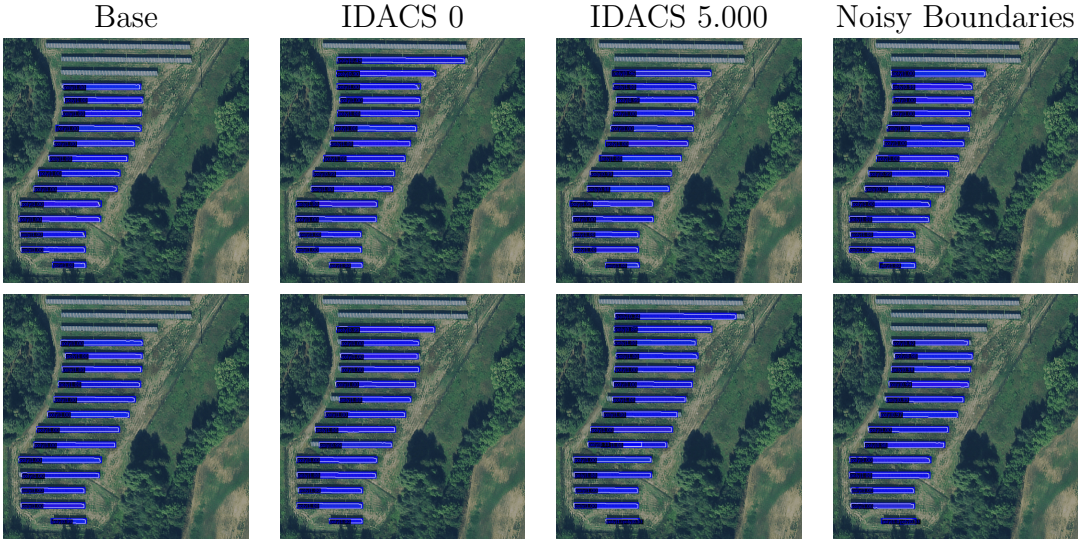

Figure 5.7: Comparison of inference results between methods on a sample rural panel plant. The first row depicts results for methods ran on the RGB dataset, while the second depicts result for the RGB-IR dataset.

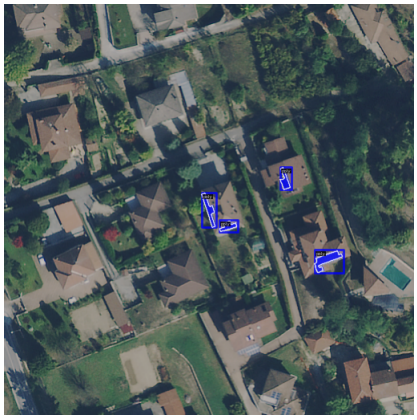

Figure 5.8: Ground truth annotations for a sample image from the test set depicting panels in an urban setting.

Figure 5.9: Comparison of inference results between methods on a sample image depicting panels in an urban setting. The first row depicts results for methods ran on the RGB dataset, while the second depicts result for the RGB-IR dataset.

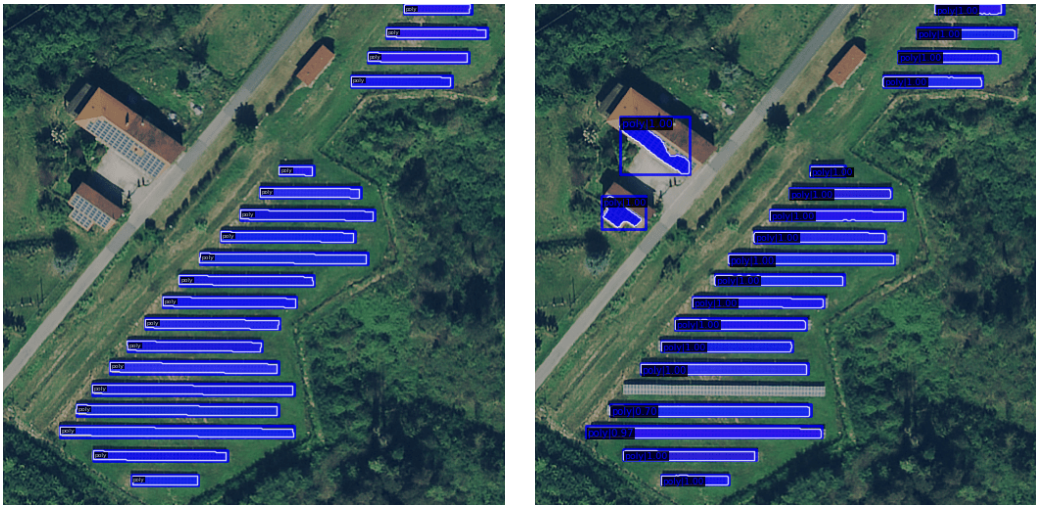

Figure 5.10: Sample images from the test set showing how panels not present in ground truth annotations (left) can be detected by the models (right, Noisy Boundaries RGB-IR as an example).

Chapter 6

Conclusions

This chapter presents known limitations for both the methods described and the dataset used for this thesis and possible future extensions for the work (Section 6.0.1) and finally some closing remarks (Section 6.0.2).

6.0.1 Limitations and future works

The methods analysed present some limitations. The architecture of Noisy Boundaries is composed of multiple stages, meaning its efficiency is limited and that it requires more resources. Meanwhile, the proper functioning of IDACS heavily depends on the images sampled for mixing, since using the same labeled images over and over is bound to cause overfitting and performance issues in the model itself despite augmentations. All methods are also limited by the discrepancy between classes in the dataset, due to the fact that polycrystalline are much more frequent than monocrystalline, making proper classification of the latter more difficult.

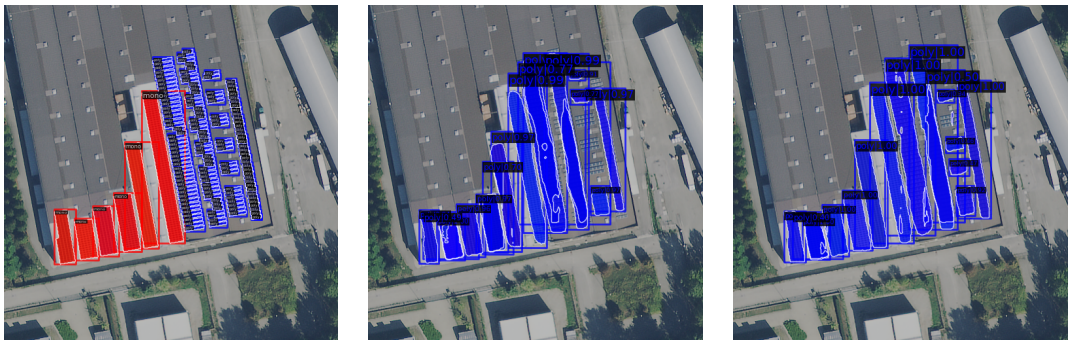

Figure 6.1: Inference results showing the models' difficulty in both classifying panels and correctly segmenting clusters of separate panels. From left to right: ground truth, Noisy RGB-IR and IDACS 5.000 RGB-IR.

Furthermore, all models present difficulties in correctly segmenting clusters of multiple panels grouped together, as shown both in Figure 6.1 and Figure 6.2. The latter, in particular, shows how this is exacerbated when panels are oriented diagonally.

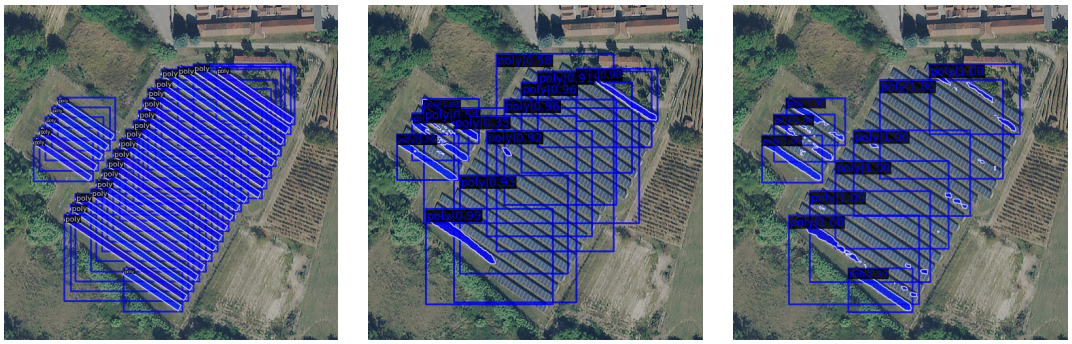

Figure 6.2: Inference results showing the models' difficulty in correctly segmenting diagonally-oriented panels, especially when grouped in clusters. From left to right: ground truth, Noisy RGB-IR and IDACS 5.000 RGB-IR.

A few possible solutions and improvements can be examined for potential future works. Perhaps the most immediate among them is to turn Noisy Boundaries into a single-stage framework, simplifying its architecture and potentially making it more efficient. As for IDACS, it is important to have as much variety as possible in the combination of labeled and unlabeled images to obtain satisfying results, and randomly sampling a relatively small subset of the 22 million possible combinations does not guarantee the generation of diverse mixed images. Thus, it could greatly benefit from a proper sampling method for the choice of images to mix together.

The problem of class discrepancy is already partly tackled by Noisy Boundaries itself, which employs box-level and pixel-level thresholds to prevent bias towards dominant classes in generated pseudo-labels. Implementing this filtering procedure in IDACS could improve the model's performances during classification. Furthermore, the sampling method mentioned above could be designed in order to create a balance in the frequency of classes, by evening out as much as possible the number of monocrystalline and polycrystalline panels to be pasted on unlabeled images. Another possibility is to integrate the Noise-Tolerant Mask head and the Boundary-Preserving Map used in Noisy Boundaries in the IDACS pipeline: these two components could help the model better learn from its own pseudo-labels.

Finally, the issue of diagonally-oriented panels could be solved by employing Oriented Bounding Boxes for object detection, which, as the name implies, are not axis-aligned but are rotated in order to better fit the instance's shape. This would be particularly useful also considering panels have a rectangular shape most of the time.

6.0.2 Final remarks

In summary, the purpose of this work was to evaluate and compare two semi-supervised methods with the goal of addressing the issue of data scarcity in a setting where aerial images are used. Experiments produced interesting results, highlighting the differences and demonstrating the overall effectiveness of the presented techniques.

In conclusion, this thesis showed how semi-supervised methods are a valid choice for situations in which data is scarce and where there is therefore a need to artificially increase the size of datasets, specifically in the case of aerial images segmentation. The results showed how both of the methods presented are viable approaches for this problem, and the above considerations suggest there is much room for improvement.

Bibliography

[1] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. «Gradient-based learning applied to document recognition». In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324 (cit. on p. 7).

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. «Imagenet: A large-scale hierarchical image database». In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255 (cit. on p. 7).

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. «Imagenet classification with deep convolutional neural networks». In: *Communications of the ACM* 60.6 (2017), pp. 84–90 (cit. on pp. 7, 11).

[4] Karen Simonyan and Andrew Zisserman. «Very deep convolutional networks for large-scale image recognition». In: *arXiv preprint arXiv:1409.1556* (2014) (cit. on p. 8).

[5] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. «Deep neural networks for object detection». In: *Advances in neural information processing systems* 26 (2013) (cit. on p. 10).

[6] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. «Rich feature hierarchies for accurate object detection and semantic segmentation». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587 (cit. on pp. 10, 11).

[7] Pierre Sermanet, Koray Kavukcuoglu, Soumith Chintala, and Yann LeCun. «Pedestrian detection with unsupervised multi-stage feature learning». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2013, pp. 3626–3633 (cit. on p. 10).

[8] Henry A Rowley, Shumeet Baluja, and Takeo Kanade. «Neural network-based face detection». In: *IEEE Transactions on pattern analysis and machine intelligence* 20.1 (1998), pp. 23–38 (cit. on p. 10).

[9] Régis Vaillant, Christophe Monrocq, and Yann Le Cun. «Original approach for the localisation of objects in images». In: *IEE Proceedings-Vision, Image and Signal Processing* 141.4 (1994), pp. 245–250 (cit. on p. 10).

[10] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. «Selective search for object recognition». In: *International journal of computer vision* 104.2 (2013), pp. 154–171 (cit. on p. 11).

[11] Jan Hosang, Rodrigo Benenson, Piotr Dollár, and Bernt Schiele. «What makes for effective detection proposals?» In: *IEEE transactions on pattern analysis and machine intelligence* 38.4 (2015), pp. 814–830 (cit. on p. 11).

[12] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. «Backpropagation applied to handwritten zip code recognition». In: *Neural computation* 1.4 (1989), pp. 541–551 (cit. on p. 11).

[13] Ross Girshick. «Fast r-cnn». In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448 (cit. on p. 11).

[14] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. «Faster r-cnn: Towards real-time object detection with region proposal networks». In: *Advances in neural information processing systems* 28 (2015) (cit. on p. 11).

[15] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. «Simultaneous detection and segmentation». In: *European conference on computer vision*. Springer. 2014, pp. 297–312 (cit. on p. 11).

[16] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. «Hypercolumns for object segmentation and fine-grained localization». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 447–456 (cit. on p. 11).

[17] Jifeng Dai, Kaiming He, Yi Li, Shaoqing Ren, and Jian Sun. «Instance-sensitive fully convolutional networks». In: *European conference on computer vision*. Springer. 2016, pp. 534–549 (cit. on p. 11).

[18] Pedro O O Pinheiro, Ronan Collobert, and Piotr Dollár. «Learning to segment object candidates». In: *Advances in neural information processing systems* 28 (2015) (cit. on p. 11).

[19] Pedro O Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. «Learning to refine object segments». In: *European conference on computer vision*. Springer. 2016, pp. 75–91 (cit. on p. 11).

[20] Jifeng Dai, Kaiming He, and Jian Sun. «Instance-aware semantic segmentation via multi-task network cascades». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3150–3158 (cit. on p. 11).

[21] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. «Mask r-cnn». In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969 (cit. on pp. 11, 17, 23).

[22] Gui-Song Xia, Xiang Bai, Jian Ding, Zhen Zhu, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. «DOTA: A large-scale dataset for object detection in aerial images». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 3974–3983 (cit. on pp. 12, 15, 32).

[23] Syed Waqas Zamir et al. «isaid: A large-scale dataset for instance segmentation in aerial images». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 28–37 (cit. on pp. 12, 17).

[24] Pengfei Zhu et al. «VisDrone-VDT2018: The vision meets drone video detection and tracking challenge results». In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 2018, pp. 0–0 (cit. on p. 12).

[25] Murari Mandal, Manal Shah, Prashant Meena, Sanhita Devi, and Santosh Kumar Vipparthi. «AVDNet: A small-sized vehicle detection network for aerial visual data». In: *IEEE Geoscience and Remote Sensing Letters* 17.3 (2019), pp. 494–498 (cit. on p. 12).

[26] Murari Mandal, Manal Shah, Prashant Meena, and Santosh Kumar Vipparthi. «SSSDET: Simple short and shallow network for resource efficient vehicle detection in aerial scenes». In: *2019 IEEE international conference on image processing (ICIP)*. IEEE. 2019, pp. 3098–3102 (cit. on p. 12).

[27] Bin Zhou, Xuemei Duan, Dongjun Ye, Wei Wei, Marcin Woźniak, Dawid Połap, and Robertas Damaševičius. «Multi-level features extraction for discontinuous target tracking in remote sensing image monitoring». In: *Sensors* 19.22 (2019), p. 4855 (cit. on p. 12).

[28] Lichao Mou and Xiao Xiang Zhu. «Vehicle instance segmentation from aerial image and video using a multitask learning residual fully convolutional network». In: *IEEE Transactions on Geoscience and Remote Sensing* 56.11 (2018), pp. 6699–6711 (cit. on p. 12).

[29] Xiangfeng Zeng, Shunjun Wei, Jinshan Wei, Zichen Zhou, Jun Shi, Xiaoling Zhang, and Fan Fan. «CPISNet: delving into consistent proposals of instance segmentation network for high-resolution aerial images». In: *Remote Sensing* 13.14 (2021), p. 2788 (cit. on p. 12).

[30] Prateek Garg, Anirudh Srinivasan Chakravarthy, Murari Mandal, Pratik Narang, Vinay Chamola, and Mohsen Guizani. «Isdnet: Ai-enabled instance segmentation of aerial scenes for smart cities». In: *ACM Transactions on Internet Technology (TOIT)* 21.3 (2021), pp. 1–18 (cit. on p. 12).

[31] Jiangye Yuan, Hsiu-Han Lexie Yang, Olufemi A Omitaomu, and Budhendra L Bhaduri. «Large-scale solar panel mapping from aerial images using deep convolutional networks». In: *2016 IEEE International Conference on Big Data (Big Data)*. IEEE. 2016, pp. 2703–2708 (cit. on p. 12).

[32] Poonam Parhar, Ryan Sawasaki, Alberto Todeschini, Hossein Vahabi, Nathan Nusaputra, and Felipe Vergara. «HyperionSolarNet: Solar Panel Detection from Aerial Images». In: *arXiv preprint arXiv:2201.02107* (2022) (cit. on p. 12).

[33] Roberto Castello, Simon Roquette, Martin Esguerra, Adrian Guerra, and Jean-Louis Scartezzini. «Deep learning in the built environment: Automatic detection of rooftop solar panels using Convolutional Neural Networks». In: *Journal of Physics: Conference Series*. Vol. 1343. 1. IOP Publishing. 2019, p. 012034 (cit. on p. 12).

[34] Qizhu Li, Anurag Arnab, and Philip HS Torr. «Weakly-and semi-supervised panoptic segmentation». In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 102–118 (cit. on p. 13).

[35] George Papandreou, Liang-Chieh Chen, Kevin P Murphy, and Alan L Yuille. «Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation». In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1742–1750 (cit. on p. 13).

[36] Yunchao Wei, Huaxin Xiao, Honghui Shi, Zequn Jie, Jiashi Feng, and Thomas S Huang. «Revisiting dilated convolution: A simple approach for weakly-and semi-supervised semantic segmentation». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7268–7277 (cit. on p. 13).

[37] Nasim Souly, Concetto Spampinato, and Mubarak Shah. «Semi supervised semantic segmentation using generative adversarial network». In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 5688–5696 (cit. on p. 13).

[38] Wilhelm Tranheden, Viktor Olsson, Juliano Pinto, and Lennart Svensson. «Dacs: Domain adaptation via cross-domain mixed sampling». In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2021, pp. 1379–1389 (cit. on pp. 13, 27, 33).

[39] Lukas Hoyer, Dengxin Dai, and Luc Van Gool. «Daformer: Improving network architectures and training strategies for domain-adaptive semantic segmentation». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 9924–9935 (cit. on p. 13).

[40] Alexey Dosovitskiy et al. «An image is worth 16x16 words: Transformers for image recognition at scale». In: *arXiv preprint arXiv:2010.11929* (2020) (cit. on p. 13).

[41] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. «SegFormer: Simple and efficient design for semantic segmentation with transformers». In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 12077–12090 (cit. on p. 13).

[42] Yanzhao Zhou, Xin Wang, Jianbin Jiao, Trevor Darrell, and Fisher Yu. «Learning saliency propagation for semi-supervised instance segmentation». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 10307–10316 (cit. on p. 13).

[43] Zhi Tian, Chunhua Shen, Xinlong Wang, and Hao Chen. «Boxinst: High-performance instance segmentation with box annotations». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 5443–5452 (cit. on p. 13).

[44] Zhenyu Wang, Yali Li, and Shengjin Wang. «Noisy Boundaries: Lemon or Lemonade for Semi-supervised Instance Segmentation?» In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 16826–16835 (cit. on pp. 13, 24, 25).

[45] Osmar Luiz Ferreira de Carvalho, Osmar Abılio de Carvalho Júnior, Anesmar Olino de Albuquerque, Nickolas Castro Santana, Renato Fontes Guimarães, Roberto Arnaldo Trancoso Gomes, and Dıbio Leandro Borges. «Bounding Box-Free Instance Segmentation Using Semi-Supervised Iterative Learning for Vehicle Detection». In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 15 (2022), pp. 3403–3420 (cit. on p. 13).

[46] Wentong Li, Yijie Chen, Wenyu Liu, and Jianke Zhu. «Deep Level Set for Box-supervised Instance Segmentation in Aerial Images». In: *arXiv preprint arXiv:2112.03451* (2021) (cit. on p. 13).

[47] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. «Microsoft coco: Common objects in context». In: *European conference on computer vision*. Springer. 2014, pp. 740–755 (cit. on pp. 15, 28).

[48] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. «Path aggregation network for instance segmentation». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8759–8768 (cit. on p. 17).

[49] Edoardo Arnaudo, Fabio Cermelli, Antonio Tavera, Claudio Rossi, and Barbara Caputo. «A contrastive distillation approach for incremental semantic segmentation in aerial images». In: *International Conference on Image Analysis and Processing*. Springer. 2022, pp. 742–754 (cit. on p. 24).

[50] Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D Cubuk, Quoc V Le, and Barret Zoph. «Simple copy-paste is a strong data augmentation method for instance segmentation». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 2918–2928 (cit. on p. 28).