



Escola d'Enginyeria de Telecomunicació i
Aeroespacial de Castellet

UNIVERSITAT POLITÈCNICA DE CATALUNYA

MASTER THESIS

TITLE: Crowd counting using Density Maps

MASTER DEGREE: Master's degree in Applied Telecommunications and Engineering Management (MASTEAM)

AUTHOR: Juan Manuel Trujillo Gómez

ADVISOR: Cristina Barrado

DATE: June 26th, 2022

Title: Crowd Counting using Density Maps

Author: Juan Manuel Trujillo Gómez

Advisor: Cristina Barrado

Date: June 26th, 2022

Abstract

Law enforcement agents have to care about the number of people in public areas to ensure security. The problem they have is that they do not have tools to measure the number of people in a fast and precise way. This need has been especially important since 2020 COVID pandemic arrived to our society and the control of people is relevant to avoid spread of COVID.

This Master Thesis is complementing other previous Master Thesis presented in 2021 where via an Android app connected to a drone the system was able to count people from the images captured in real time. This solution was only able to count individual people, as crowds of people are complex to measure following standard object detection algorithms as YOLO technology.

In our Master Thesis we are adding a new functionality by being able not only to count individuals but also counting crowds of people. With this new functionality the app could provide to the police a more accurate tool to be able to count people in different scenarios as prides, sports events, demonstrations, concerts... where crowd is a normal situation.

As main technology driver we are working with CNN (Convolutional Neural Networks). First, we have been implementing a CNN density map using the CSRNet technology that is able to count people by measuring the concentration of people. Therefore, an important part of this Master Thesis is to create a process to split the input images in 2 (segmentation process), one for YOLO (individual persons) and other for CSRNET (crowds of people). This process has been implemented using a second CNN called Region-based CNN (R-CNN), that we found it was the most suitable tool to train a model to detect a crowd.

The solution has been developed in Google Colab platform and using Python as programming language.

We have been working with images taken from drones from Castelldefels Police and UPC but also public datasets.

The final solution has been able to detect crowds and calculate the number of people in that crowd with a maximum error of 20% considering Mean Average Percentage Error (MAPE) and 89 considering Mean Absolute Error (MAE).

ACKNOWLEDGMENT

First of all, I would like to thank the ICARUS research group of the EETAC school of the UPC and especially Professor Cristina Barrado for giving me the opportunity to work on a real research project with the police. Thank you very much Cristina for being accessible, flexible and constant during all this period. Also, I would like to thank Ender Çetin for his technical support in the first weeks until I became more autonomous with the technical part.

Also thank the Castelldefels Police for giving us the opportunity to work with them. First, by providing real drone images with crowd. Secondly, by being able to stay with them in some events, which allowed us to have first-hand information regarding their needs, constraints and opportunities in their daily job working with drones.

On the other hand, I want to acknowledge my colleagues Valeria Ferrer, Fozhong Chen and Javier Palomares for their support during the Master, especially at the beginning, and who were vital, in encouragement and knowledge, to get to the point where I am today. Working as a team was wonderful with them and I learned a lot from them. Also thank the support and guidance of the director of the Master David Rincón, who always had a solution to the doubts raised.

Finally to my dear wife Eva and my adorable children Martí and Clàudia to whom I dedicate this Master Thesis. Without Eva it would have been impossible to study and I am very grateful for her dedication to our family so that I could focus to work and study. I hope I can compensate in the coming months and recover some of the time for the family. And to my two little kids that I hope you feel very proud of the effort and investment that your father has made and that I hope in some way I have been an example for you. And remember, in the end every effort has its reward.

ACRONYMS

MAPE	Mean Average Percentage Error
MAE	Mean Absolute Error
CNN	Convolutional Neural Networks
DNN	Deep Neural Networks
RoI	Region of Interest
R-CNN	Region-based CNN
RPN	Region Proposal Network
FCN	Fully Convolutional Network
FPN	Feature Pyramid Network
ML	Machine Learning
AI	Artificial Intelligence
NLP	Natural Language Processing
IoU	Intersection over Union

CONTENTS

ACKNOWLEDGMENT	4
ACRONYMS	5
INTRODUCTION	9
CHAPTER 1. THEORETICAL CONCEPTS OF CONVOLUTIONAL NEURAL NETWORKS USED IN THE MASTER THESIS	11
1.1 Introduction to Machine Learning.....	11
1.2 Introduction to CNN.....	13
1.3 Mask R-CNN for crowd segmentation	16
1.3.1 Introduction	16
1.3.2 Mask R-CNN solution.....	17
1.3.3 Faster R-CNN network configuration	18
1.4 CSRNet CNN for crowd counting.....	19
1.4.1 Introduction	19
1.4.2 CSRNet solution	20
1.4.3 Dilated convolution.....	21
1.4.4 Network configuration.....	22
CHAPTER 2. IMPLEMENTATION OF CROWD COUNTING	24
2.1 Development environment	24
2.2 Main Neural Network projects used.....	26
2.3 Labelling tools	26
2.4 Converting video to pictures and pictures to video.....	28
2.5 Programming steps	29
2.5.1 Crowd segmentation	30
2.5.2 Crowd counting	34
2.5.3 Data analytics	38
CHAPTER 3. RESULTS	39
3.1 Crowd segmentation results.....	39
3.1.1 Validation	39
3.1.2 Segmentation with false positive.....	41
3.2 Crowd counting results.....	44
3.2.1 Crowd counting with segmented images.....	45
3.2.2 Manual count vs Density Map.....	47
3.3 Further studies	48
3.3.1 R-CNN counting crowd.....	48
3.3.2 Cropping images	48

3.3.3	Without crowd segmentation	51
3.3.4	With different resolutions	52
3.3.5	With false positives	52
3.3.6	Comparison of results using the same image	53
3.4	Sensitivity evaluation	54
3.5	Processing time.....	60
CHAPTER 4. CONCLUSIONS AND FUTURE WORK		62
REFERENCES		65

INTRODUCTION

The main target of this Master Thesis is to provide some tools to the ICARUS research group of the UPC to be able to improve the quality of people crowd counting in images taken from drones in public areas by police forces. This Master Thesis is complementing a previous Master Thesis called “Drones against COVID-19 propagation by controlling capacity in public spaces” [25] defended in 2021 and where the main target was to count individual persons using YOLO technology via an Android application connected to a drone.

Crow count could be used in many scenarios and has particular importance to control crowds specially during COVID pandemic. The same technology could be used to count people but also it could be evolved for other type of object management as cars or animals. We just need to train the systems with new images and configuration.

There were 3 main categories to classify the different methods to count crowd before using CNN [9]. First one is detection-based methods. This method detects people by extracting low-level features of the human body. However, the performance is low as in very congested images it is difficult to detect human features. To improve this problem new methods are in place to detect only some parts of the human body but even that the results are not accurate for very congested scenes. Second category is regression-based methods. As previous methods do not work properly for high congested images, this regression method crops the original image in smaller parts and extracts the low-level features, as foreground and texture, of each cropped image. Then analyses the relation with the other cropped images and calculates the number of people. And the last category is density estimation-based methods. In this method it is generated a linear mapping between features in the local region and its object density maps.

CNN has been used as an evolution method to these 3 methods based on the success in classification and object detection. It has been demonstrated by different researchers that CNN solutions are much more accurate: [10], [11], [12]. So, we can conclude that CNN are the leading solution nowadays to count people. In this Master Thesis we are going to use 2 CNN's, the first one to segment crowds using Mask R-CNN and the second one for crowd counting using CSRNet.

As we did not find any pre-trained model with crowd segmentation we needed to generate our own model using all the images available for the training process. To train we used Shanghai public dataset [17] as main source but also we added some UPC and Police pictures taken from drones in Castelldefels. To test false positives in segmentation process we used Unsplash data set [31]. For crowd counting we used the pre-trained model based on Shanghai data set. Finally, to analyse the different results we used an additional external data set from University of Central Florida (UCF) [6] with high resolution images from public Internet sources.

Thanks to automatize the full process for crowd counting and the result metrics, where we only needed to place in a folder the input images, we were able to test many scenarios and parameters.

The main conclusion is that crowd segmentation is not improving significantly the metrics (less than 1% considering MAPE), at least considering the type of images used in the test, very congested images with an average of 60% of pixels with crowd. Anyway, segmentation is needed because CSRNet is not able to count individual people and also other backgrounds as trees, buildings or sky are adding noise to the images.

We have also demonstrated that CSRNet gets better results with images with high density, high volume of people and congested scenes.

The structure of this Master Thesis is described here:

- Chapter 1: explains a brief introduction to Machine Learning (ML), Convolutional Neural Networks (CNN) and some of the theoretical concepts used in the Master Thesis. Also, a more detailed explanation about Mask R-CNN for crowd segmentation and CSRNet for crowd counting.
- Chapter 2: explains the different tools used to do the research as the development environment and all the programming steps for crowd segmentation and counting.
- Chapter 3: explains the main testing and results performed.
- Chapter 4: explains the conclusions of the research and future work.

CHAPTER 1. Theoretical concepts of Convolutional Neural Networks used in the Master Thesis

1.1 Introduction to Machine Learning

This thesis is using technologies included in what is called Machine Learning (ML). ML is about making machines get better at some tasks by learning from data, instead of having to explicitly code rules. Next, we are going to explain the main theoretical concepts of ML based on reference [26].

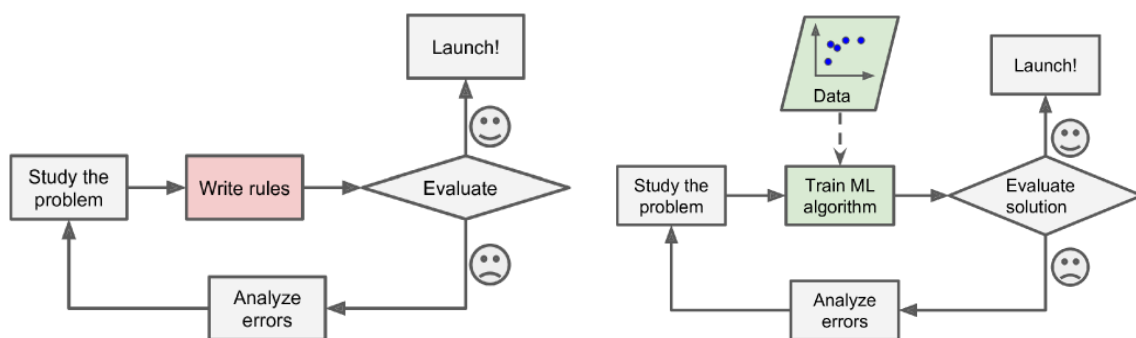


Figure 1.1. On the left traditional approach and on the right ML approach [26]

In figure 1.1 we can see on the left the traditional approach of programming based on rules and on the right the ML approach where the system learns from the data. ML approach is good when we found a problem where the code requires a long list of rules (for example email spam). In that way ML can even simplify the code compared with the traditional approach. Also, ML could find a solution to complex problems where traditional approach is not able to find it or the solution is not good enough. On the other hand, ML is able to adapt to changing environments just processing new data and learning from it, so it is not needed to code again new rules or new functions. Finally, another advantage of ML is that allows to get additional information about data and complex problems so we can understand for example how data is evolving to help improve the monitoring and analysis.

In order to understand better ML we will list next some of the main techniques with an example:

1. Image classification: Analysing images of products on a production line to automatically classify them.
2. Object detection: Detecting tumors in brain scans or as we do in this Thesis detect a crowd in drone picture.

3. Regression: Forecasting crowd counting using Density Maps in a picture as we do in this Thesis.

These 3 ML examples use CNN as a main technical driver. So, we can see as the CNN used in this thesis is one of the technologies included in ML. In section 1.2 we can go deeper on CNN concepts.

Other examples of ML techniques are natural language processing, speech recognition, anomaly detection, data visualization, recommender system, reinforcement learning ... so the list of uses in real life is huge and accessible thanks to the high level of computational power existing today.

ML has 3 types of techniques that can be categorized in:

- Supervised or not: Supervised, unsupervised, semi supervised, and reinforcement learning. Whether they are trained with human supervision or not.
- Online or batch learning: Whether they can learn incrementally on the fly.
- Instance-based or model-based learning: Whether they work by simply comparing new data points to known data points, or instead by detecting patterns in the training data and building a predictive model, much like scientists do.

The typical steps to manage a ML project could be summarized in the next steps: study the data, select a model, train it on the training data and finally apply the model to new data to make predictions (inference).

The way to know how well a model will generalize to new cases is to try it out on new cases. For that reason, the data set needs to be split in training (to train the model), test (to test the model) and validation (to evaluate candidate models). This value tells you how well your model will perform on instances it has never seen before.

The main challenges for ML could be summarized in:

- Insufficient quantity of training data: Even for very simple problems you typically need thousands of examples. Data matters more than algorithms for complex problems.
- Nonrepresentative training data (sampling bias): Training data has to be representative of the new cases you want to generalize. By using a nonrepresentative training set, we trained a model that is unlikely to make accurate predictions.
- Poor-quality data: If training data is full of errors it will make it harder for the system to detect the patterns.
- Data mismatch: Large amount of data for training but this data is not representative of the data that will be used in production.
- Irrelevant features: A critical part of the success of a ML project is coming up with a good set of features to train on.
- Overfitting the training data: the model performs well on the training data, but it does not generalize well.

- Underfitting the training data: it occurs when the model is too simple to learn the underlying structure of the data.

1.2 Introduction to CNN

This section explains the main theoretical concepts of Convolutional neural networks (CNN) based on reference [26]. CNNs emerged from the study of the brain's visual cortex. CNNs have managed to achieve superhuman performance on some complex visual tasks. Neurons in the visual cortex have a small local receptive field, meaning they react only to visual stimuli located in a limited region of the visual field. Some neurons react only to images of horizontal lines, while others react only to lines with different orientations. They also noticed that some neurons have larger receptive fields, and they react to more complex patterns that are combinations of the lower-level patterns. These observations led to the idea that the higher-level neurons are based on the outputs of neighbouring lower-level neurons (in figure 1.2, notice that each neuron is connected only to a few neurons from the previous layer). This powerful architecture can detect all sorts of complex patterns in any area of the visual field. Next, we will explain the main concepts to understand CNN.

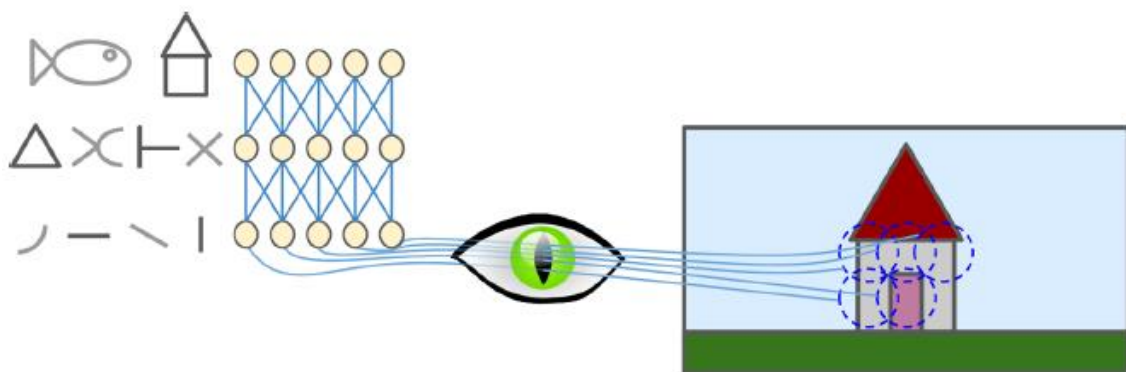


Figure 1.2. Biological neurons in the visual cortex [26]

Convolutional Layers: The most important building block of a CNN is the convolutional layer. Neurons in the first convolutional layer are not connected to every single pixel in the input image, but only to pixels in their receptive fields (see Figure 1.3). In turn, each neuron in the second convolutional layer is connected only to neurons located within a small rectangle in the first layer. This architecture allows the network to concentrate on small low-level features in the first hidden layer, then assemble them into larger higher-level features in the next hidden layer, and so on.

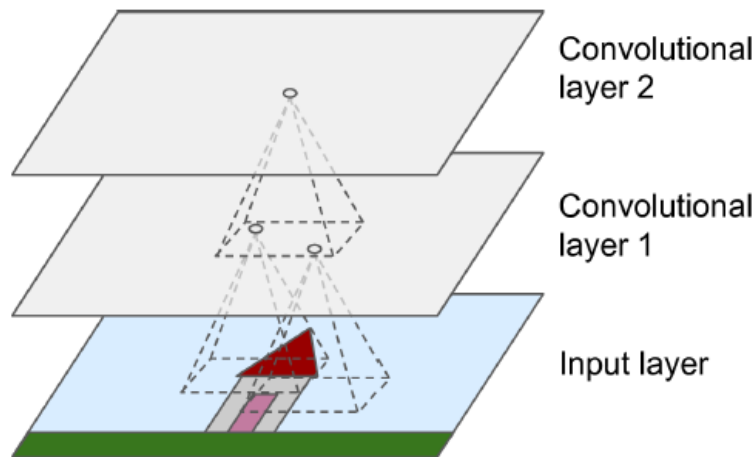


Figure 1.3. CNN layers with rectangular local receptive fields [26]

Filters or convolution kernels: A neuron's weights can be represented as a small image the size of the receptive field. A convolutional layer simultaneously applies multiple trainable filters to its inputs, making it capable of detecting multiple features anywhere in its inputs. See figure 1.4.

Feature map: is a layer full of neurons using the same filter which highlights the areas in an image that activate the filter the most. See figure 1.4.

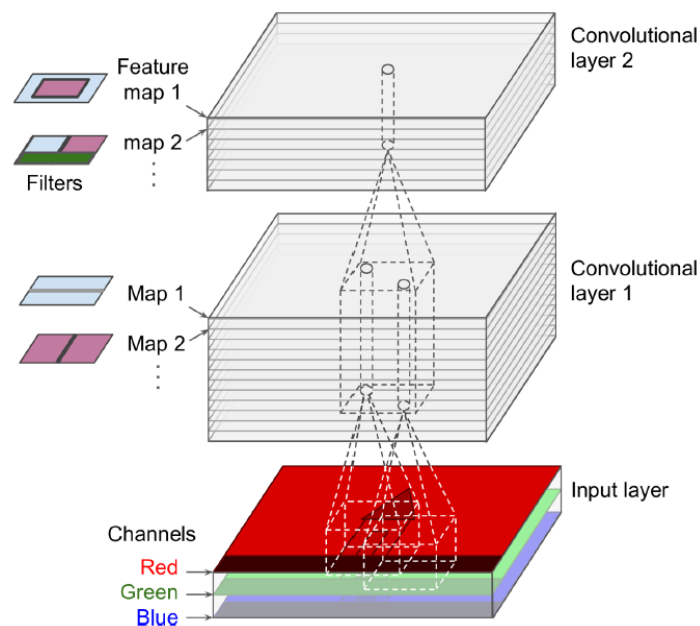


Figure 1.4. Convolutional layers with multiple feature maps, and images with three color channels [26]

Pooling Layers: Their goal is to subsample (i.e., shrink) the input image in order to reduce the computational load, the memory usage, and the number of parameters (thereby limiting the risk of overfitting). A pooling layer has no weights; all it does is aggregate the inputs using an aggregation function such as the max or mean.

Data Augmentation: Data augmentation artificially increases the size of the training set by generating many realistic variants of each training instance.

Transfer Learning: If you want to build an image classifier but you do not have enough training data, then it is often a good idea to reuse the lower layers of a pretrained model.

Main types of techniques working with CNN and images:

1. **Localization:** Localizing an object in a picture can be expressed as a regression task. To predict a bounding box (rectangle that surrounds an object, that specifies its position, class and confidence) around the object, a common approach is to predict the horizontal and vertical coordinates of the object's center, as well as its height and width. You then need to create a dataset whose items will be pre-processed images along with their class labels and their bounding boxes. For regression tasks we can use MAPE and MAE as main metric for quality indicators.
2. **Image classification:** The task to be able to identify an image based on different predefined class labels as animals, flowers, cars,... For example, we can process an image and the computer could classify it as a flower and even the computer can predict the type of flower (rose, gardenia, geranium...). As main metric to measure the quality, we can use the accuracy. Accuracy is the percentage of good classification results versus the total number of cases tested.
3. **Object detection:** The task of classifying and localizing multiple objects in an image is called object detection (see figure 1.5). As main metric to measure the quality, we can use Intersection over Union (IoU). IoU is the area of overlap between the predicted bounding box and the target bounding box (see figure 1.6).

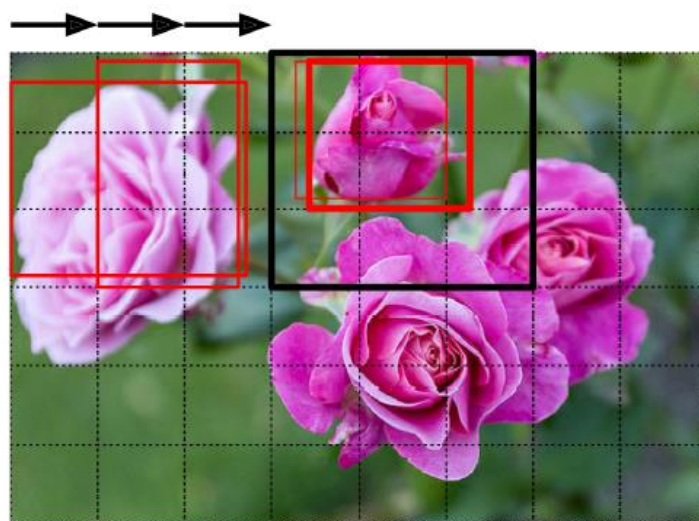


Figure 1.5. Detecting multiple objects [26]



Figure 1.6. Intersection over Union metric example

4. Semantic segmentation: Each pixel is classified according to the class of the object it belongs to. Different objects of the same class are not distinguished (see figure 1.7). IoU also could be used as main metric to control the quality of the results.



Figure 1.7. Semantic segmentation [26]

1.3 Mask R-CNN for crowd segmentation

This section explains the method used to segment crowd called “Mask R-CNN” [1].

1.3.1 Introduction

Mask Region-based CNN (Mask R-CNN) is a framework for object instance segmentation evolved from Faster R-CNN [18]. It can detect an object and also define the segmentation mask of this object. We will use it in this Master Thesis

to detect and segment a crowd although it could be used also to detect other objects already pre-trained as cars, animals,...

Mask R-CNN is adding a branch for predicting segmentation masks on each Region of Interest (RoI), in parallel with the existing branch for classification and bounding box regression. The mask branch is a small Fully Convolutional Network (FCN) [19] applied to each RoI, predicting a segmentation mask in a pixel-to-pixel manner. And only adds a small computational overhead.

Mask R-CNN adds a simple quantization-free layer, called RoIAlign, that preserves exact spatial locations for more accurate segmentation. Also predicts a binary mask for each class independently, without competition among classes, and relies on the network's RoI classification branch to predict the category.

1.3.2 Mask R-CNN solution

Mask R-CNN is conceptually simple: Faster R-CNN has two outputs for each candidate object, a class label and a bounding-box offset; to this we add a third branch that outputs the object mask. Next, we introduce the key elements of Mask R-CNN, including pixel-to-pixel alignment, which is the main missing piece of Faster R-CNN.

In order to understand better the evolution of Mask R-CNN from Faster R-CNN we will explain first Faster R-CNN. Faster R-CNN consists of two stages. The first stage, called a Region Proposal Network (RPN), proposes candidate object bounding boxes. The second stage extracts features using RoIPool (standard operation for extracting a small feature map, e.g., 7x7, from each RoI) from each candidate box and performs classification and bounding-box regression. Mask R-CNN adopts the same two-stage procedure, with an identical first stage (which is RPN). In the second stage, in parallel to predicting the class and box offset, Mask R-CNN also outputs a binary mask for each RoI. This stage allows the network to generate masks for every class without competition among classes. This means that object mask overlapping is allowed. In figure 1.8 we can see an example using Mask R-CNN, in the top of the image we can see some clear examples of object masks overlapping.



Figure 1.8. Example of object segmentation using Mask R-CNN. Source: [30]

1.3.3 Faster R-CNN network configuration

Authors split the architecture in 2 parts:

- Convolutional backbone: used for feature extraction over an entire image. Feature Pyramid Network (FPN) uses a top-down architecture with lateral connections to build an in-network feature pyramid from a single-scale input. Faster R-CNN with an FPN backbone extracts RoI features from different levels of the feature pyramid according to their scale, but otherwise the rest of the approach is similar to ResNet [15]. Using a ResNet-FPN backbone for feature extraction with Mask R-CNN gives excellent gains in both accuracy and speed.
- Network head: for bounding-box recognition (classification and regression) and mask prediction that is applied separately to each RoI. Authors extend the Faster R-CNN box heads from the ResNet [15] and FPN [16] papers. See figure 1.9. as a reference for more details. The head on the ResNet-C4 backbone includes the 5-th stage of ResNet (namely, the 9-layer 'res5' [20]), which is compute intensive. For FPN, the backbone already includes res5 and thus allows for a more efficient head that uses fewer filters.

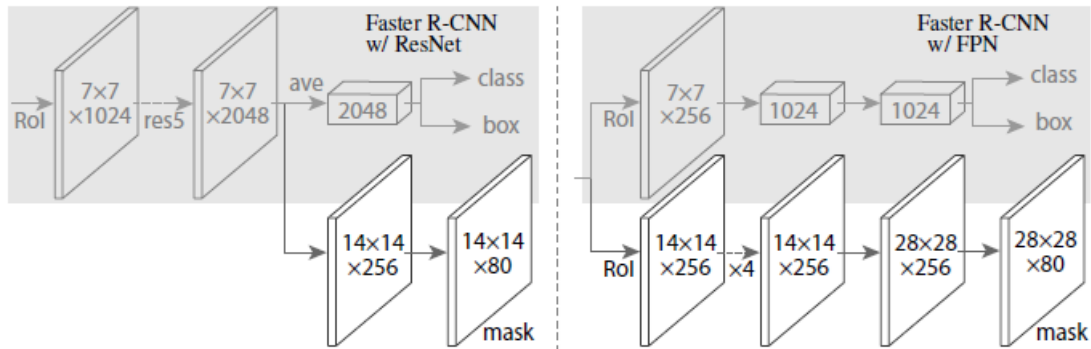


Figure 1.9. Head architecture of Mask R-CNN. Source: [1]

1.4 CSRNet CNN for crowd counting

This section explains the method used to count people called “CSRNet: Dilated Convolutional Neural Networks for Understanding the Highly Congested Scenes” [2].

1.4.1 Introduction

CSRNet is a deep learning method that from an input image can give a count estimation figure and represents a density map of the input image where it is shown the crowd distribution.

Figure 1.10 shows three examples from Shanghai data set. The top images are the original pictures and at the bottom we can see the Density Map for each one. Density Map uses different colours to show the density level. For example, red and yellow colours mean more congested scene and blue dots represent individual people.

Density map plays a key role to understand the results as we can see in a graphical way the congested area in the original image and we can use to check the quality of counting.

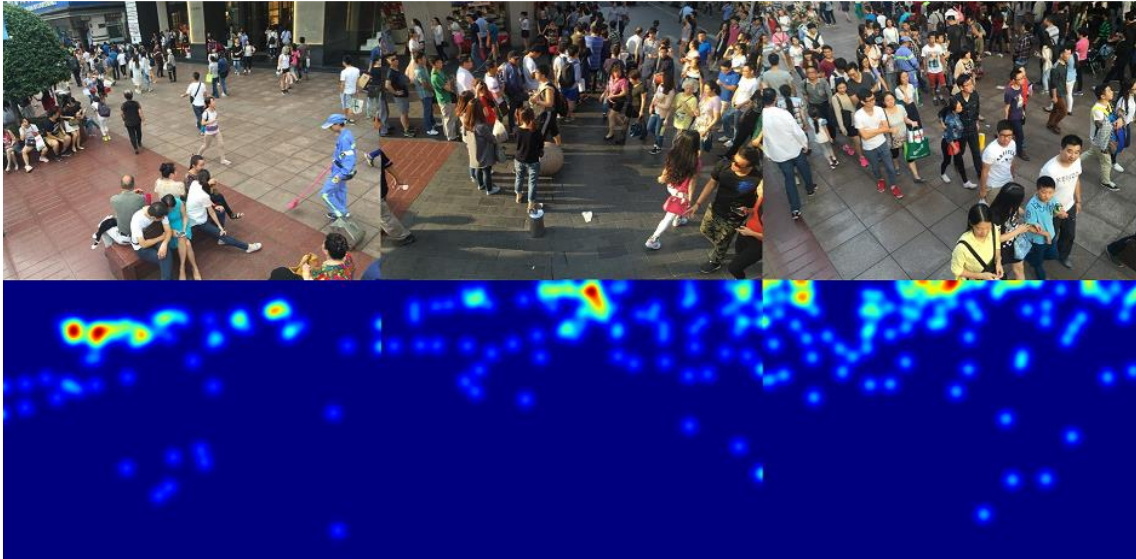


Figure 1.10. Example of density map for 3 images from Shanghai data set.
Source [2]

CSRNet has two main components:

- **Front-end:** Convolutional Neural Network for 2D feature extraction.
- **Back-end:** Dilated Neural Network which uses dilated kernels to deliver larger reception fields and to replace pooling operations.

CSRNet model was trained with 4 crowd data sets: ShanghaiTech [17], UCF CC 50 [22], WorldEXPO'10 [21] and UCSD [23]. Where ShanghaiTech data set has been the main one.

As we are going to see in chapter 3, images with crowd have different perspectives and density of people. Also crowd images have irregular distribution that makes the count difficult. The only relevant method to get high accuracy is by using deep neural networks (DNN).

Small size kernels of convolution filters (like 3 x 3) are used in all layers. The first 10 layers are deployed from VGG-16 [8] as the front-end and dilated convolution layers as the back-end to enlarge receptive fields and extract deeper features without losing resolutions (pooling layers are not used).

1.4.2 CSRNet solution

The CSRNet solution is to deploy a deeper CNN for capturing high-level features with larger receptive fields and generating high-quality density maps without expanding network complexity.

Authors choose VGG-16 [8] as the front-end of CSRNet because of its strong transfer learning ability and its flexible architecture for easily concatenating the back-end for density map generation. Authors first remove the classification part

of VGG-16 (fully-connected layers) and build the proposed CSRNet with convolutional layers in VGG-16. The output size of this front-end network is $1/8$ of the original input size. Authors try to deploy dilated convolutional layers as the back-end for extracting deeper information of saliency as well as maintaining the output resolution.

1.4.3 Dilated convolution

Dilated convolution is a better choice, which uses sparse kernels (as shown in figure 1.11) to alternate the pooling and convolutional layer. This character enlarges the receptive field without increasing the number of parameters or the amount of computation (e.g., adding more convolutional layers can make larger receptive fields but introduce more operations). In dilated convolution, a small-size kernel with $k \times k$ filter is enlarged to $k + (k - 1)(r - 1)$ with dilated stride r . Thus, it allows flexible aggregation of the multi-scale contextual information while keeping the same resolution.

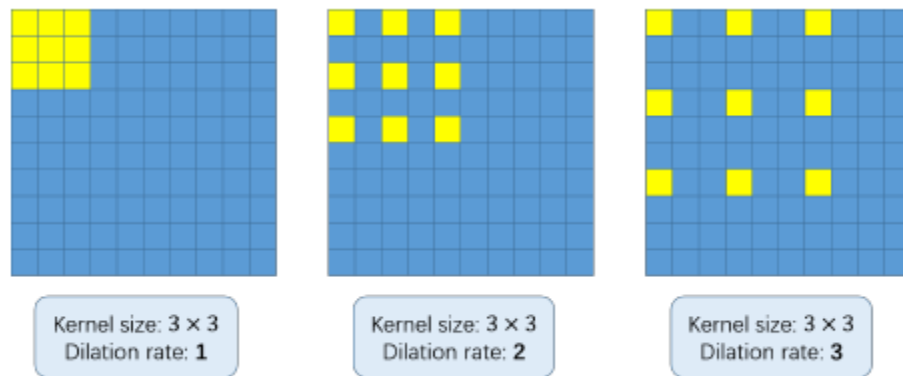


Figure 1.11. 3×3 convolution kernels with different dilation rate as 1, 2, and 3.
Source: [2]

In figure 1.12 we can see an example of dilate convolution. For maintaining the resolution of feature map, the dilated convolution shows distinct advantages compared to the scheme of using convolution + pooling + deconvolution. The input is an image of crowds, and it is processed by two approaches separately for generating output with the same size:

- **First approach:** input is down sampled by a max pooling layer with factor 2, and then it is passed to a convolutional layer with a 3×3 Sobel kernel (see figure 1.12). Since the generated feature map is only $1/2$ of the original input, it needs to be upsampled by the deconvolutional layer (bilinear interpolation).
- **Second approach:** the authors tried dilated convolution and adapt the same 3×3 Sobel kernel to a dilated kernel with a factor = 2 stride (see figure 1.12). The output is shared the same dimension as the input (meaning pooling and deconvolutional layers are not required). Most

importantly, the output from dilated convolution contains more detailed information.

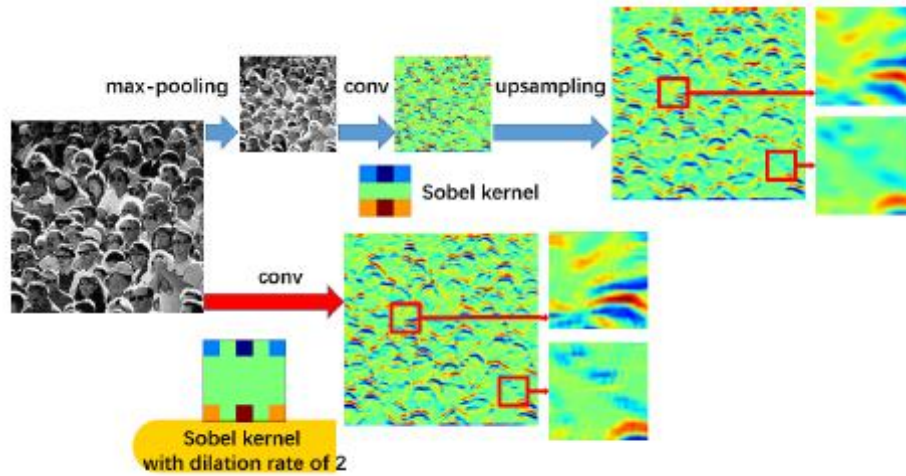


Figure 1.12. Comparison between dilated convolution and maxpooling, convolution, upsampling. The 3 x 3 Sobel kernel is used in both operations while the dilation rate is 2. Source: [2]

1.4.4 Network configuration

The authors tested four network configurations of CSRNet in Table 1.1 which have the same front-end structure but different dilation rate in the back-end. Regarding the front-end, authors adapt a VGG-16 network [24] (except fully-connected layers) and only use 3 x 3 kernels. Experiment shows a best tradeoff can be achieved when keeping the first ten layers of VGG-16 [24] with only three pooling layers. Since the output (density maps) of CSRNet is smaller (1/8 of input size), authors choose bilinear interpolation with the factor of 8 for scaling and make sure the output shares the same resolution as the input image.

Configurations of CSRNet			
A	B	C	D
input(unfixed-resolution color image)			
front-end (fine-tuned from VGG-16)			
conv3-64-1 conv3-64-1			
max-pooling			
conv3-128-1 conv3-128-1			
max-pooling			
conv3-256-1 conv3-256-1 conv3-256-1			
max-pooling			
conv3-512-1 conv3-512-1 conv3-512-1			
back-end (four different configurations)			
conv3-512-1	conv3-512-2	conv3-512-2	conv3-512-4
conv3-512-1	conv3-512-2	conv3-512-2	conv3-512-4
conv3-512-1	conv3-512-2	conv3-512-2	conv3-512-4
conv3-256-1	conv3-256-2	conv3-256-4	conv3-256-4
conv3-128-1	conv3-128-2	conv3-128-4	conv3-128-4
conv3-64-1	conv3-64-2	conv3-64-4	conv3-64-4
conv 1-1-1			

Table 1.1. Configuration of CSRNet. All convolutional layers use padding to maintain the previous size. The convolutional layers' parameters are denoted as "conv-(kernel size)-(number of filters)-(dilation rate)", max-pooling layers are conducted over a 2 x 2 pixel window with stride 2. Source: [2]

CHAPTER 2. IMPLEMENTATION OF CROWD COUNTING

In the previous chapter we have been introducing the theoretical concepts related to the technology implemented. Now we are going to explain the implementation steps used in the Master Thesis. In figure 2.1 shows the summary of the main implementation steps that we will explain in detail in this chapter.

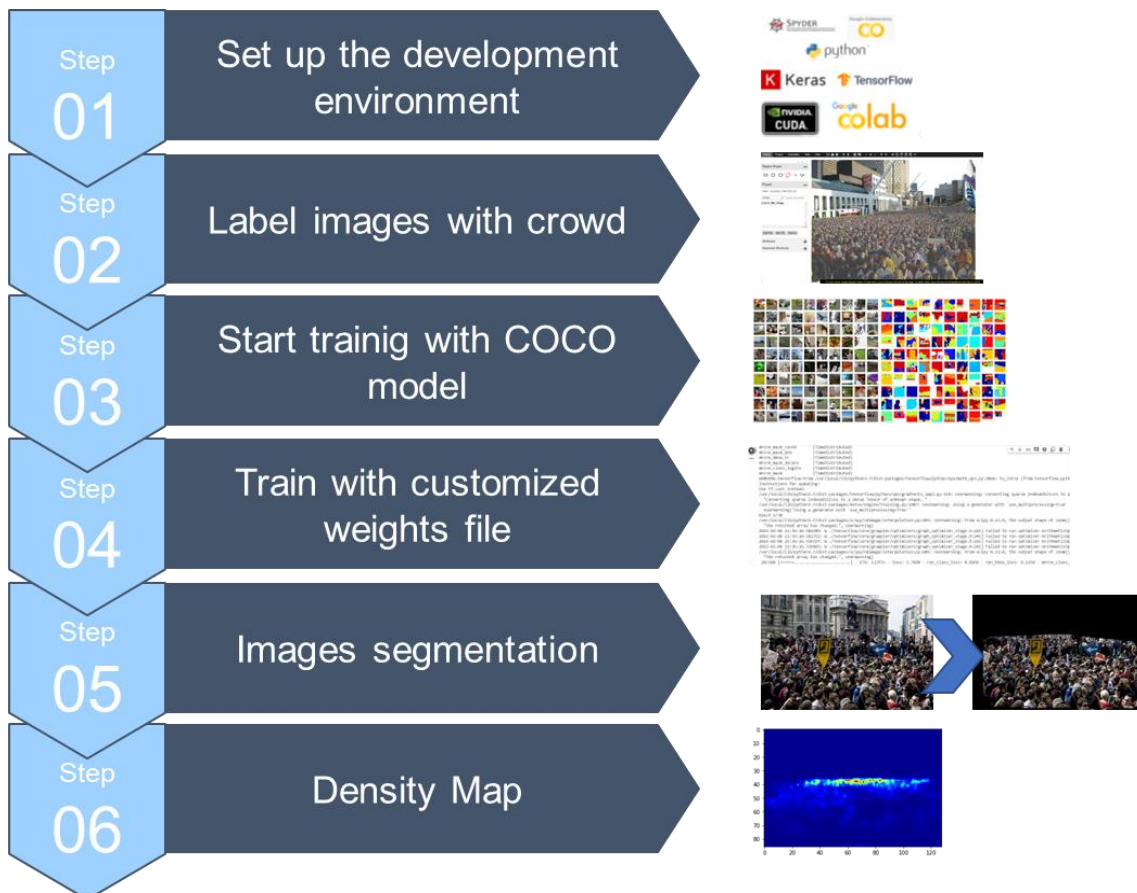


Figure. 2.1 Main implementation steps used in the Master Thesis

2.1 Development environment

Machine learning techniques require a lot of image processing that consumes huge number of computer resources. This is the main reason why we are using Google Colab platform in the cloud where we have available GPUs that we can run in parallel using also CUDA¹ (parallel computing platform and programming model that speed up applications by taking advantage of the power of GPU accelerators).

¹ <https://blogs.nvidia.com/blog/2012/09/10/what-is-cuda-2/>

On the other hand, Google Colab offers free access for students. It has some limitations as the time available for running jobs, number of active sessions or GPU power of processing. Anyway, neither of these limitations have been a problem to execute the Master Thesis. Also, it is an easy way to share the code with other researchers, it does not depend on your local hardware constraints and the code is always secure with a back-up.

As a disadvantage, we need to keep in mind that you need to install always in each new session the full environment of libraries and connectivity. Secondly, you do not control what is installed at each moment in the machine, so you can get different results based on hardware configuration. On the other hand, Google Colab gives only 12 hours of continuous execution time. After that, the whole virtual machine is reset and it is needed to start from zero. Anyway in our Master Thesis we never needed more than the limit time.

To be able to work with convolutional neural networks we use Keras and Tensorflow libraries from Google [4].

Finally, as programming language Python was selected. Python is the most popular and powerful language for machine learning. As Integrated Development Environment (IDE) Spyder is used to program the main python functions locally in the PC. Spyder is an open-source and free platform. Finally, Google Colaboratory has been used to execute the python programs in the cloud and get the results.

Figure 2.2 presents a visualization of all these tools.



Figure. 2.2 Main development tools used in the Master Thesis

2.2 Main Neural Network projects used

For object segmentation and training the main project used is based on paper “Mask R-CNN” [1]. This project provides a framework that can segment objects in an image and identify a segmentation mask of the same object. It is an extension of “Faster R-CNN” framework and it uses Convolutional Neural Networks (CNN). More information was given in section 1.3.

For density map the main project used is based on paper “CSRNet: Dilated Convolutional Neural Networks for Understanding the Highly Congested Scenes” [2]. This project has 2 main functions, it is able to represent graphically a density map of congested scenes and also it is able to count the number of objects in the scene. As the previous project it also uses CNN. More details were also given in section 1.4.

2.3 Labelling tools

During the Master Thesis we have been using 2 main labelling tools. These tools are needed to label the images used for training so the system can understand what an object crowd is. The tools are very similar and basically offer the chance to edit each image and add an area of interest. Then, once you have all the images labelled, it can generate a file with all the images and the coordinates of the introduced polygons. This file and the images are the main inputs for the training process.

Labelbox²:

Labelbox is an online tool very popular and with many features that were used in the previous Master Thesis. It is free but with some constraints that are not a problem for the development of the Master Thesis. It was used to segment manually the images before having the automatic segmentation using Mask R-CNN. Figure 2.3 shows an example of an image labelled for a crowd of people.

² URL: <https://labelbox.com/>

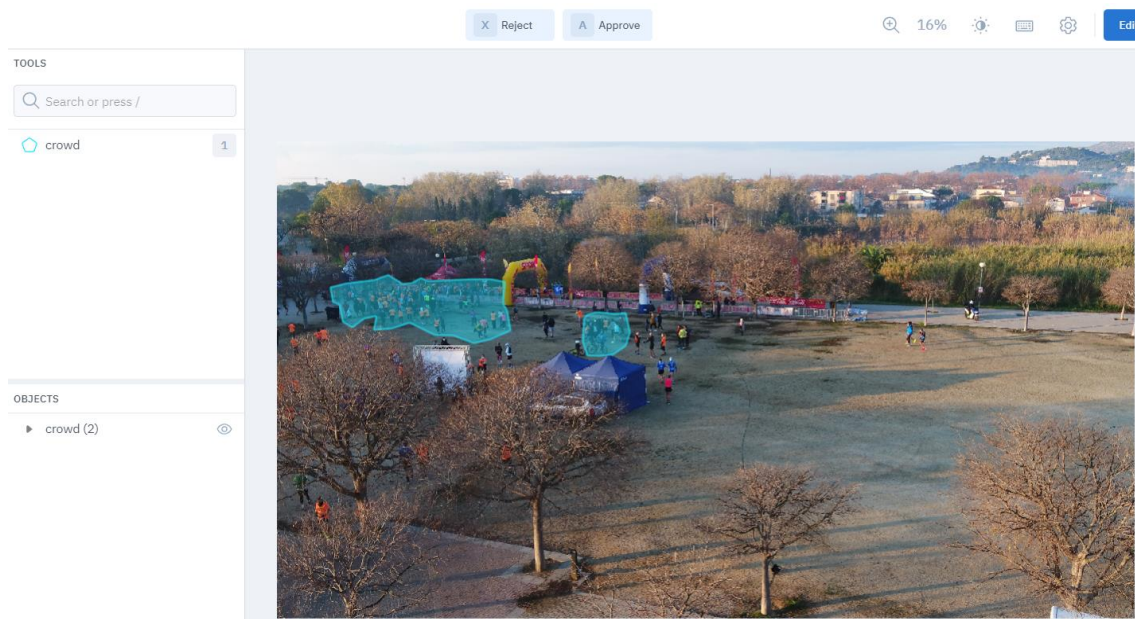


Figure 2.3 Image captured from Labelbox with a real example. Source: [29]

VIA (VGG Image Annotator)³:

VIA is a local web tool that runs in your PC without any installation needed, just the browser. You do not need to upload pictures to label to the cloud, but the number of features is limited although enough for our purpose. As it was used for R-CNN project we used it to facilitate the integration but also because of confidentiality. Being a local tool that does not need to upload pictures to the cloud, allows us to keep better confidentiality when processing the images from the police.

All the pictures for training were labelled using this tool. Figure 2.4 shows an example of an image labelled for a crowd of people.

³ URL: <https://www.robots.ox.ac.uk/~vgg/software/via/>



Figure 2.4 Image captured from VIA with a real example. Source: [17]

2.4 Converting video to pictures and pictures to video

One of the functionalities required was the option to input a video. To be able to integrate the video in the CNN process what we did was to convert a video in a sequence of images, one for each frame, and then process the images as part of the CNN process. Also, once we get the images processed, we wanted the possibility to generate a new video with the result.

For that purpose, the main tool used was Ezgif.com⁴. In figure 2.5. we can see a screenshot of one example used.

⁴ URL: <https://ezgif.com/video-to-jpg>

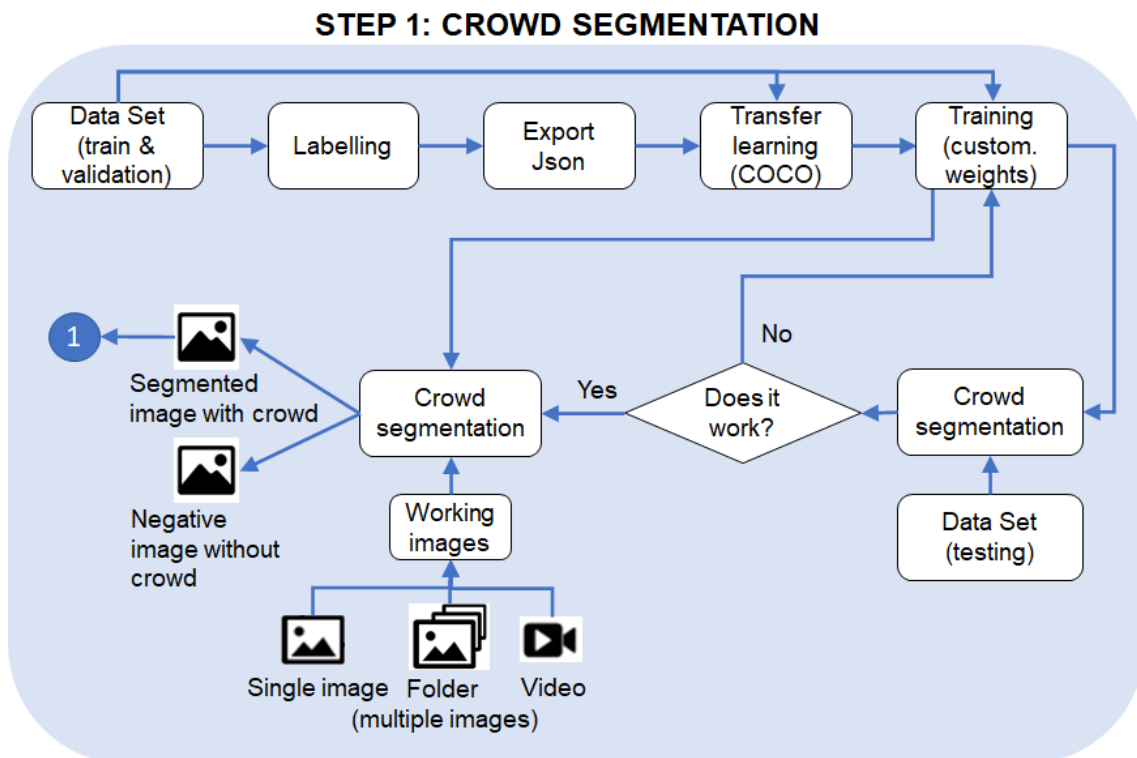
Video to JPG converter



Figure 2.5 Screenshot of Ezgif tool to convert video to images. Source: [28]

2.5 Programming steps

As it is shown in figure 2.6, in our Master Thesis we split the process in 2 main steps. One for crowd segmentation and the second for crowd counting.



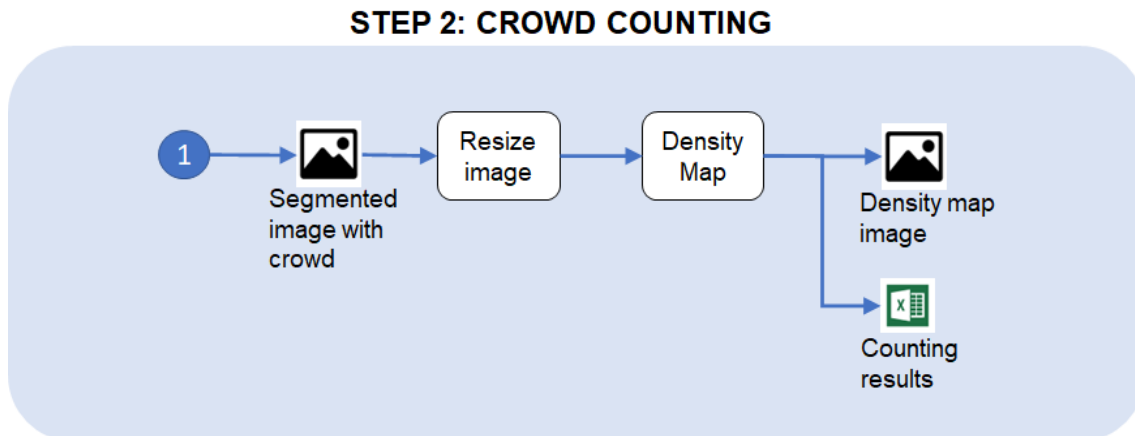


Figure 2.6 Flow diagram of the main programming steps

2.5.1 Crowd segmentation

The first step in our process is to split the original image in 2. The first one with the crowd segmented and the second one with the background, this is, the other area different to the crowd. The reason is simple, we need to process 2 different algorithms to count people, one for crowd and the other for individual people. It means that we need to be able to identify a crowd from the image and segment it from the picture.

To be able to identify crowds we use R-CNN (Region-Based Convolutional Neural Networks) [1] that was the most suitable architecture, as shown in [1], for the type of images we need to process. In following section 2.5.1.1 we explain the reason why we use R-CNN and not POLY-YOLO.

All the process for this section has been centralized in one program file called crowd.py.

The main functionalities step by step are described below.

2.5.1.1 Why R-CNN and not Yolo

Currently there are several methods to detect objects and it is important to introduce some background before moving forward. In the abstract of this document we explained that this work was complementing another previous Master Thesis where individual object detection was the target using Yolo as the main method to detect persons [25]. So, the first option in the research of this Master Thesis was to use Poly-Yolo that is an evolution of Yolo but with the advantage that is able to detect objects using polygons instead of boxes. As crowds are not uniform, having a polygon was a great solution.

But after many research using Poly-Yolo we did not succeed and we needed to pivot using R-CNN instead. The reasons are described in more detail in the

article “R-CNN, Fast R-CNN, Faster R-CNN, YOLO - Object Detection Algorithms” [4] but we can summarize in this main reason:

Yolo algorithm, when there are small objects in the image, is not able to detect them properly due to the method used. Yolo classifies the image in different boxes and assigns the probability of the box to be an object. When the object is small more than one object could share the same box and gives bad results. The classification is done only once box by box. In our Master Thesis images are taken from high altitude and crowd is made of many small persons so it means that our objects are very small and this is the main reason why Yolo was not suitable.

2.5.1.2 Transfer learning from COCO model

The first time we want to detect a new object we can speed up the process by using an existing model already trained to detect other objects. In that sense we use COCO model that was previously trained using public COCO dataset [3]. COCO data set provides more than 80 categories and more than 200.000 labelled images.

Call script:

```
!python crowd.py train --dataset=crowd_dataset/crowd_mix/ --weights=coco
```

Main parameters of the program are:

Main call: Train. Indicates that the process needs to train so it will have as main output a new weights file customized for the objects trained.

Dataset: Indicates the location of the input images used for training. We need to include in the same folder the annotation file called via_region_data.json generated by VIA labelling tool that provides the coordinates of each labelled object.

Weights: Default weights file to start training, in that case COCO.

Output of the program:

The only output of this process is a new weights file with extension .h5 that will include the information needed to segment the test images. The location of the file is:

```
.../Mask_RCNN-2.1/logs/crowd20220221T2219/mask_rcnn_crowd_0009.h55
```

As the training process can take several hours and we could be in risk to lose the job done, the program saves a version of the weights file every “epoch”

⁵ Master Thesis root path: /content/drive/MyDrive/MasterThesis/Master-Code-Crowd/Mask_RCNN-2.1/

(when the full dataset is processed forward and backward through the neural network only once). The program creates a new folder each time with a time stamp as we can see in the previous path (*crowd20220221T2219 – crowdyyyyymmddThh:mi*).

2.5.1.3 Training with customized weights file

Once we have trained the first time a complete execution of the process with COCO weights file, we can run the training with our customized crowd weights file. As it has been done considering our images and new crowd object the results will be better each time.

Call script:

```
!python crowd.py train --dataset=crowd_dataset/crowd_mix/ --weights=last
```

The only difference compared with the previous call is that we use `weights=last` in the call to indicate that it needs to find the most recent weights file generated. As the folder is named using the creation date, it is easy to find the last one.

The output of the program is the same than the previous execution but in this case, we get the final weights file that contains all the information extracted from the training with our new object called crowd. We can iterate as much time is needed this process until we consider that the quality of the result is good enough. Keep in mind that Colab sometimes stops the process for no reason, remember it is a free shared service, and it is needed to restart it again.

For the training we have used the full images available from all data sets (more than 600) and the training process lasted around 10 hours using GPUs from Colab. Also we need to consider the manual labelling process than in our case took more than 8 hours to edit the full image data set.

Table 2.1 shows the distribution percentage used for the full training process between the training, the validation and the testing from each data set. Training and validation it is used by the model to do first the training and after the checking to measure the quality of the results. So, both are part of the training process. The testing images are only used to measure the quality of the results using the weights file generated by the training process. Testing images are extracted from the same data set non-seen images, so it is a good way to control the results of the training with the same type of images. Anyway, other additional images are going to be used to test the quality of the results in different scenarios.

Data Set	Total 100%	Training 65%	Validation 25%	Testing 10%
Shanghai*	482	313	121	48
UPC and Police	167	109	42	17
Internet	4	2	1	1
Total	653	423	164	66

Table 2.1. Distribution of images from data sets used for training

*Shanghai data set represents the 74% of the total images used in the training and each photo represent a different scenario and environment.

2.5.1.4 Crowd segmentation

Once we get the weights file trained for crowd from the previous step, it is time to start the segmentation of a crowd over the best images. Three different options are described below based on the input.

Call script:

1. To process many images at the same time:

```
!python crowd.py splash --weights=last --imageFolder=.../Mask_RCNN-2.1/imagesTest12/
```
2. To process one single image:

```
!python crowd.py splash --weights=last --image=.../Mask_RCNN-2.1/imageTest12/ST_IMG_120.jpg
```
3. To process a video:

```
!python crowd.py splash --weights=last --video=.../Mask_RCNN-2.1/videoTest2/ManchesterCrowdShort.mp4
```

Main parameters of the program are:

Main call: Splash. Indicates inference, that is, the process has to detect the object crowd.

Then we have 3 different input options to detect a crowd:

1. ImageFolder: Indicates the location of the folder with the images. All images need to be in JPG format. There is no limit in the number of images in the folder.
2. Image: Indicates the location of a single image in JPG format.
3. Video: Indicates the location of a video in MP4 format.

Weights: Default weights to be used. In that case indicating "last" we always take the last one.

Output of the program:

For each input image the program generates 2 new images: one renamed as “segmented” and the other as “negative”. Both keeping the original image name at the beginning to trace it. The “segmented” image is the image containing only the crowd and the “negative” is the opposite, the remaining part of the image that is not a crowd. The negative image is what will need to be processed to detect individual persons.

The “negative” folder is located in the root folder of the original images, `.../Mask_RCNN-2.1/imagesTest5/` for example, and it is called “Results-negative”. The “segmented” folder is located also in the root folder and it is called “Results-segmented”.

Also a CSV file is generated with the summary of all the images processed in the segmentation (called “*Results-segmentation.csv*”).

2.5.2 Crowd counting

For crowd counting the Density Map model is used and all the process could run in one call once the environment is defined.

All the process for this section has been centralized in one program file called *Final-CrowdCountingCSRNet.ipynb*.

Considering Google Colab, the main steps are listed below:

1. Set up environmnet: First it is executed “*nvidia-smi*” to get information about GPU assigned status and general info. Then it is installed open-source GCC (GNU Compiler Collection) to be able to compile and run the code from our project in Linux.

```
Invidia-smi #Optional. Just optimize the processing time

import os
os.environ['PATH'] += ':/usr/local/cuda/bin'

!apt install gcc-5 g++-5 -y
!ln -s /usr/bin/gcc-5 /usr/local/cuda/bin/gcc
!ln -s /usr/bin/g++-5 /usr/local/cuda/bin/g++
```

GPU information is:

```

+-----+
| NVIDIA-SMI 460.32.03   Driver Version: 460.32.03   CUDA Version: 11.2   |
+-----+-----+-----+
| GPU  Name          Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|                                           MIG M.         |
+-----+-----+-----+
|   0   Tesla K80          Off      | 00000000:00:04:0  Off  |           0         |
| N/A   38C    P8      26W / 149W |      0MiB / 11441MiB |      0%      Default |
|                                           N/A           |
+-----+-----+-----+

```

As we can see the GPU assigned for our project is Tesla K80 that is a very powerfull server with 2 GPUs and 12GB of memory for each one.

2. Download project code: The main code used for Density Map it is downloaded from Github in the current location:

```
!git clone https://github.com/vivek-bombatkar/CSRNet-pytorch.git
```

The code will be located in our Google account at: `/content/CSRNet-pytorch`

3. Import libraries: Main libraries are imported including the code from CSRNet project from the previous step.

```

#importing libraries
import h5py
import scipy.io as io
import PIL.Image as Image
import numpy as np
import os
import glob
from matplotlib import pyplot as plt
from scipy.ndimage.filters import gaussian_filter
import scipy
import json
import torchvision.transforms.functional as F
from matplotlib import cm as CM
from image import *
from model import CSRNet
import torch
%matplotlib inline

```

4. Initializatin the model: Once we have the main libraries imported it is just needed to initialize the new model defined in CSRNet project and also load the trained weights file.

```

model = CSRNet()
#defining the model
model = model.cuda()

#Loading the trained weights
!pwd
#checkpoint = torch.load('0model_best.pth.tar')
checkpoint = torch.load('/content/gdrive/MyDrive/MasterThesis/Master-Code-Crowd/0model_best.pth.tar')

```

5. **Resize images:** From the previous point, everything is ready to start working with Density Map function. But an additional step is still needed to prepare the images. To avoid the problem of out of memory of the GPU and also reduce processing time is needed to resize the input images. In our case it was decided that the maximum width allowed in pixels was 1024.

```
if image.size[1] > 1024:
    new_image_width = 1024
    new_image_height = int(image.size[0] / image.size[1] * new_image_width)
    image = image.resize((new_image_height, new_image_width), Image.ANTIALIAS)
```

6. **Input type:** There are 2 ways developed to process the images, one for single file and another for all the images in a folder.

When working with a folder the input folder is defined in “*.../Mask_RCNN-2.1/imagesTest12*”, where “*imagesTest12*” could be renamed to any other name to keep traceability of all the trials, then in the same root folder the resize image process generates the folder “*Results-segmented-smaller*”. Finally, the output folder is located in root folder and caller “*Results-density-map*”.

7. **Call Density Map:** The variable “*output*” is defined that will get the result of our model where the only input is the image to measure.

```
output = model(img.unsqueeze(0))
print("Predicted Count : ",int(output.detach().cpu().sum().numpy()))
temp = np.asarray(output.detach().cpu().reshape(output.detach().cpu().shape[2],output.detach().cpu().shape[3]))
plt.imshow(temp,cmap = c.jet)
fig.savefig('predict.png', dpi=200)
plt.show()
```

The full process for 1 image is shown in figure 2.7.

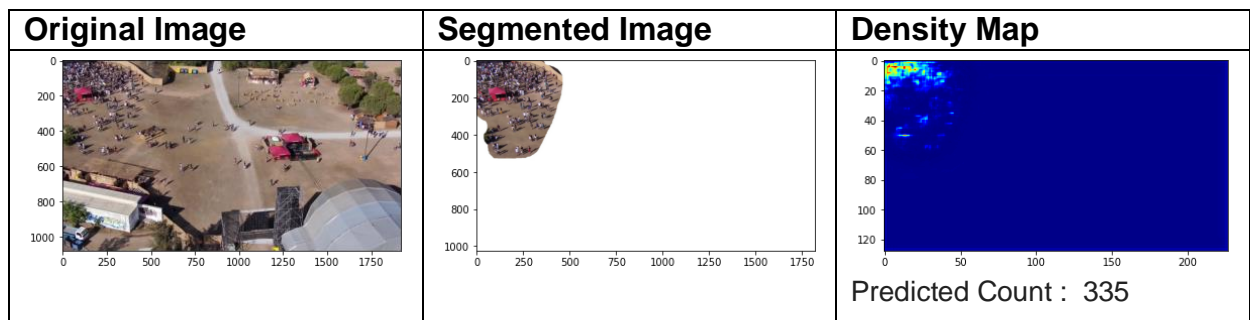


Figure 2.7 Density Map process in 3 steps. Source: [28]

In order to help to analyze the results, it is added an additional step to generate a CSV file (called “*Results-DensityMap.csv*”) with the results of each image in the input folder.

Finally, the count for each image is included in the name of the final file (DJI_0005_DensityMap_count164.JPG).

2.5.2.1 Cropping and masking images tools

As part of the testing to check if we can get better results with Density Map, we tried to crop images in smaller portions to check if the quality of the count was increasing. As it is shown in section 3.3.2, after some trials the results of the cropped images compared with the full original images, were not improving so finally we discard this process. Anyway, as it was developed to automatize the process and also it was used in the study, we are going to explain the functionalities just in case could be useful for other researchers. Also, the tool includes the option to generate segmented images with a crowd from manual labelling that was used to compare the results between the manual crowd segmentation and the CNN.

The name of the program is *ImagesPreparationV6-CROPPINGWITHJSON.py*.

The main steps are listed below:

1. Export Json file: It is used Labelbox to label manually all the images with crowd. Then a json file is exported with all the image paths and coordinates of the crowd polygons.
2. Initialize variables: In the root folder where the main program runs it is needed to create the folders for the input: json file and original images. Also, the output folders for masked files and cropped files. Finally, the size of the cropping needs to be configured in boxes multiple of 32.

```
jsonPath = "json/"
imgPath = "img/"
sourcePath = imgPath
destinationPathMask = imgPath+"maskedFiles/"
destinationPathCrop = imgPath+"croppedFiles/"
classesTypes = "crowd"
imageExtension = ".JPG"
multipleOf32 = 4 # To create 416x416 mini images when cropping value needs to be 13.
maxBackgroundColourPercentage = 98 #maximum percentage of pixels equal to background colour.
```

3. Execute program: 3 main functions:
 - a. jsonFound: Reads the json and generate new segmented images only with the crowd.
 - b. Crop: Crop the segmented images generated in the previous steps.
 - c. deleteEmptyImages: Delete all images with no info or less than 98% of pixels different to the background.

```
# Main
for file in os.listdir(jsonPath):
    if file.endswith(".json"):
        jsonFound(jsonPath, imageExtension)
        crop(destinationPathMask, destinationPathCrop, imageExtension, sourcePath, multipleOf32,
            trainingPercentage, forwardOverlap, sideOverlap)
        deleteEmptyImages(destinationPathCrop, imageExtension)
        print("End of execution")
    else:
        print("Json file doesn't exist")
```

Additionally, there is another program based on this called *ImagesPreparation-ONLYCROPPING.py*. In that case we only need images in a folder and the program generates the cropped images.

2.5.3 Data analytics

To get all the metrics analysed in section 3.4, a specific program was developed with Google Colab called *CrowdCountingCSRNet_showGroundTruth-v3.ipynb*. With this new program we could get the main metrics from Shanghai and UCF data sets and also we can calculate the main features per image as forecasted count, resolution, percentage of the background segmented, density, MAPE, MAE,... for a set of images defined in a folder. All the information is automatically exported in a csv file (called "*results-groundTruth-vs-DensityMap-UCFDataSet.csv*").







CHAPTER 3. RESULTS

The previous chapter explains the steps to execute the Master Thesis. This chapter explains the different results obtained to test the code and what are the main conclusions for each scenario of images.

3.1 Crowd segmentation results

3.1.1 Validation

From the same data set used for crowd segmentation training a few images were selected just for testing. Figure 3.1 shows some representative examples used for testing with the original image on the left and the segmented image on the right. We also indicate in the field “Result” whether we consider the segmentation was OK or not (gray color means no segmented). Also, table 3.1 represents the final segmentation results for each type of testing data set.

Original Image	Segmented image (Gray scale = not segmented)
 <p data-bbox="288 1258 730 1283">File name: IMG_168.jpg. Source: [17]</p>	 <p data-bbox="919 1258 1252 1283">Result: OK. Crowd detected</p>
 <p data-bbox="288 1509 730 1534">File name: IMG_139.jpg. Source: [17]</p>	 <p data-bbox="919 1509 1252 1534">Result: OK. Crowd detected</p>
 <p data-bbox="288 1834 730 1859">File name: IMG_153.jpg. Source: [17]</p>	 <p data-bbox="919 1848 1252 1872">Result: OK. Crowd detected</p>








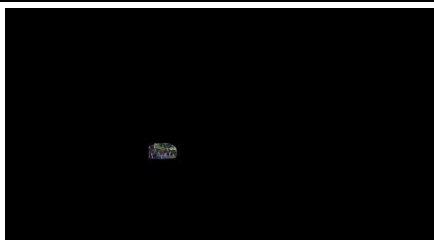

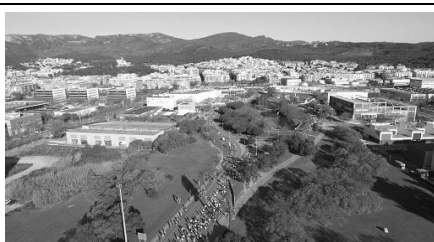


 <p>File name: IMG_176.jpg. Source: [17]</p>	 <p>Result: OK. Crowd detected</p>
 <p>File name: INT_IMG_004.jpg. Source: Internet</p>	 <p>Result: OK. Crowd detected</p>
 <p>File name: DJI_0011.jpg. Source: [29]</p>	 <p>Result: OK. No crowd detected</p>
 <p>File name: DJI_0025.jpg. Source: [29]</p>	 <p>Result: OK. Crowd detected</p>
 <p>File name: DJI_0023.jpg. Source: [27]</p>	 <p>Result: No good. No crowd detected</p>
 <p>File name: DJI_0018.jpg. Source: [27]</p>	 <p>Result: OK. Crowd detected</p>

Figure 3.1. Example of some images used to validate segmentation

Data set	Total images	Segmented OK ^(*)	Segmented No good	Percentage OK	Percentage No good
Shanghai data set	39	39	0	100%	0%
UPC and Police	17	11	6	65%	35%
Total	56	50	6	89%	11%

Table 3.1. Results for segmentation validation images.









Metrics for segmentation (*1): As seen in chapter 1, object detection uses IoU as main indicator to measure the quality of the result. To be able to get this metric we need the theoretical polygon area for each image (ground truth) to compare with the resulting area. As this information is not provided by any of the data sets used in this Thesis, we can not provide an automatic and accurate IoU. For that reason, when in table 3.1 we refer to “Segmented OK” or in figure 3.1 we refer to “Result: OK”, we consider that IoU is higher than 75% after manual inspection image by image. From now on, this consideration applies to all segmentation results.

Main Conclusions: Initially the data set was defined with 66 images but some Shanghai pictures were removed because they were in gray scale and the format was not compatible with R-CNN. So, we tested the result with 56 images from Shanghai and UPC data set. As table 3.1 shows, the conclusions are quite clear as 100% of the pictures from Shanghai data set have been segmented correctly versus the 65% of UPC images. The main reason is that the percentage of images from Shanghai data set represents 74% of the total images used for training. So as any machine learning process, more data means better results. We can consider this result as a data mismatch, where data used for training is not representative of the data that will be used in production.

3.1.2 Segmentation with false positive

To get a good indicator of the quality of the crowd segmentation, it is important to use images where no crowd exists to check if there is any false positive. Three image categories have been used: cities (21 images), animals (20 images) and forest (20 images). The reason to use these categories is because it is the most common use case for drone pictures available on the Internet. The data set used is from Unsplash web site [31] where we can get free high-resolution images from many categories. Unsplash data set has been exclusively used for the purpose of this section.

See figure 3.2 for some examples for each category. The image on the left is the original and on the right is the segmented one (in color and gray). When it appears the same original image in gray scale on the right is because has not been segmented.

Original Image	Segmented image (Gray scale = not segmented)
 <p data-bbox="306 546 743 577">Author: Abigail Keenan. Source: [31]</p>	 <p data-bbox="912 560 1279 591">Result: OK. No crowd detected</p>
 <p data-bbox="300 909 750 940">Author: Denys Nevozhai. Source: [31]</p>	 <p data-bbox="852 909 1343 967">Result: OK. No crowd detected even with high altitude</p>
 <p data-bbox="325 1370 721 1402">Author: Andre Benz. Source: [31]</p>	 <p data-bbox="932 1370 1267 1402">Result: OK. Crowd detected</p>
 <p data-bbox="303 1702 743 1733">Author: Geran De Klerk. Source: [31]</p>	 <p data-bbox="912 1720 1279 1751">Result: OK. No crowd detected</p>







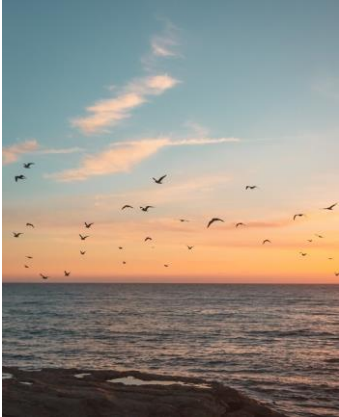
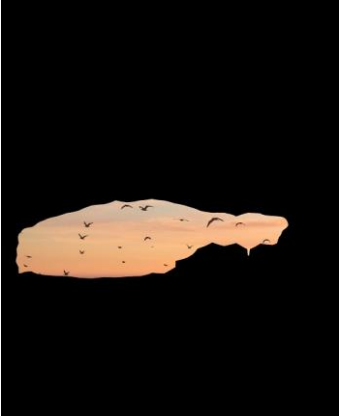
 <p>Author: Josefina Di Battista. Source: [31]</p>	 <p>Result: No good. Penguins detected as crowd. Good segmentation but no good classification</p>
 <p>Author: Matt Reed. Source: [31]</p>	 <p>Result: OK. No crowd detected</p>
 <p>Author: Gary Bending. Source: [31]</p>	 <p>Result: OK. No crowd detected</p>
 <p>Author: Florian Hahn. Source: [31]</p>	 <p>Result: No good. Birds detected as crowd. Good segmentation but no good classification</p>

Figure 3.2. Example of some images used to validate segmentation with false positive

Main Conclusions: As table 3.2 shows, from the 61 images tested the summary is that only 7 were segmented incorrectly. From that 7, 6 corresponds to the animals category, only 1 to the city category and 0 to forest. So, we can conclude that with cities and forest scenarios the results are pretty good as are over 95% of good segmentation and classification. The main problem is with animals where the correct classification percentage is of 70%. The main reason could be the similarity of the human body (eyes, head, body,...) with other animals where CNN could confuse them. On the other hand and in our benefit this category is the less probable in our drones urban scenarios. So, we can conclude that for our Master Thesis target, the results are quite good considering that the images tested are completely external from the training scenario and with some common points for future use.

Category	Total images	Segmented OK	Segmented No good	Percentage OK	Percentage No good
Animals	20	14	6	70%	30%
Cities	21	20	1	95%	5%
Forest	20	20	0	100%	0%
Total	61	54	7	89%	11%

Table 3.2. Results for segmentation validation images with false positive.

3.2 Crowd counting results

Once we have the crowd segmented the next step is to use density map model to count the people.

Thanks to the Python programming developed during the research we have been able to automatize the full process to count crowds that allow us to process huge number of images in just some minutes.

MAPE and MAE are going to be used as main quality indicators. Below you can find the formula of each one [7]:

$$MAPE = \frac{100\%}{n} \sum_{t=1}^n \left| \frac{GroundTruth_t - Forecast_t}{GroundTruth_t} \right|$$

$$MAE = \frac{\sum_{t=1}^n |Forecast_t - GroundTruth_t|}{n}$$

Where $GroundTruth_t$ is the crowd count real value provided by each data set, $Forecast_t$ is the forecasted value calculated from Density Map and n the number of tested images.

Table 3.3 shows the results from Shanghai data set where Ground Truth is available (UPC and Police data set can not be used for that reason). Shanghai

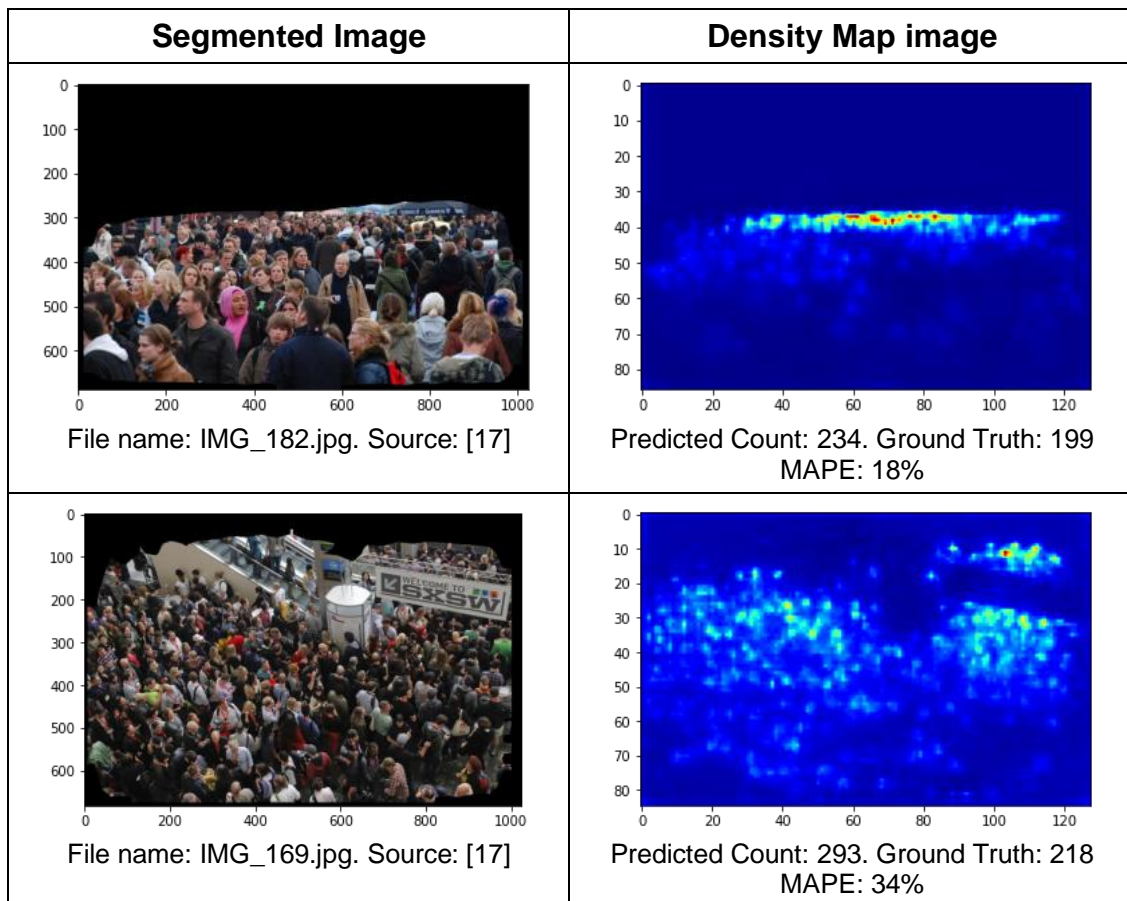
data set was used to train Density Map weights file so it is normal that Training results are much better than Test ones as we can see in table 3.3.

Data set	MAPE	MAE	Number of Images
Shanghai Training	12,89%	55,39	300
Shanghai Test	22,21%	74,86	182

Table 3.3. Comparison of MAPE from Shanghai data set

3.2.1 Crowd counting with segmented images

Figure 3.3 shows the results from Density Map (right column) taken as an input the segmented image processed by R-CNN (left column). Images are taken from the two main data sets (Shanghai and UPC).



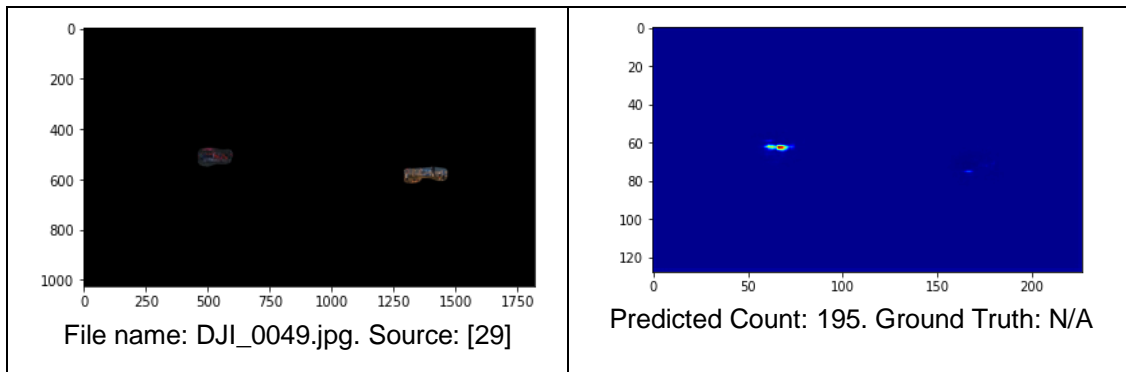


Figure 3.3. Example of some images processed with Density Map

In table 3.4 we can see the main indicators from Shanghai data set. As we can see segmented images only get 0,45% reduction of MAPE considering 62,64% of image segmented (very congested scenes in average).

Main indicators	Shanghai data set
MAPE (not segmented)	20,69%
MAE (not segmented)	81,20
MAPE (segmented)	20,24%
MAE (segmented)	89,22
Number of Images	82
Resolution average	618KB
Count of crowd average	449
Density (resolution/crowd)	2074
Percentage of image segmented	62,64%

Table 3.4. Comparison of the results of the segmentation using CNN

In order to assess how the quality of the automatic segmentation process could impact the quality of the method, we have tested with a sample of 12 images segmented manually. And as we can see in table 3.5 MAPE are very similar than the same result we get using CNN (less than 1% gap).

	With manual segmentation			Using CNN for segmentation		
	Original full image	Segmented manually	GAP	Original full image	Segmented with CNN	GAP
MAPE	26,54%	26,67%	-0,13%	20,69%	20,24%	0,45%
MAE	148,00	153,25	-5,25	81,20	89,22	-8,02

Table 3.5. Comparison of MAPE with manual and CNN segmentation method.

Main Conclusions:

- Segmented images are getting very similar results compared with the original image considering MAPE.
- The quality of the automatic segmentation using CNN is very accurate as we are getting similar results than with the manual segmentation considering MAPE.

3.2.2 Manual count vs Density Map

Figure 3.4 shows a comparative between the results from Density Map (right column) and manual count (left column) where each person has been identified with a dot and an identifier. As all these images have been labelled manually to get the real count (what is called ground truth) we consider that a reasonable error considering MAPE could be lower than 10%.


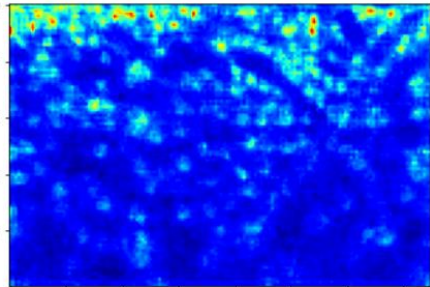

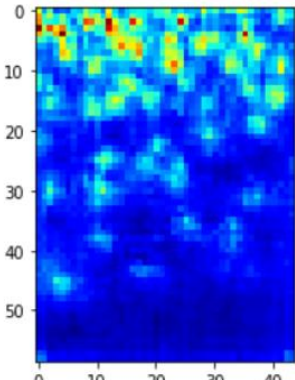

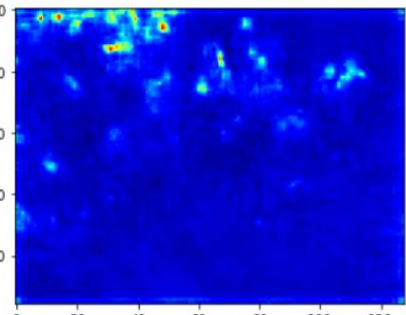
Original Image with manual count	Density Map image
 <p data-bbox="225 824 780 884">File name: INT_IMG_001.jpg. Source: Internet Manual Count: 221</p>	 <p data-bbox="943 853 1209 947">Predicted Count : 172 MAPE: 23% Result: No good</p>
 <p data-bbox="277 1350 727 1411">File name: IMG_272.jpg. Source: [17] Manual Count: 49</p>	 <p data-bbox="951 1366 1201 1460">Predicted Count : 52 MAPE: 6% Result: OK</p>
 <p data-bbox="284 1805 719 1865">File name: IMG_59.jpg. Source: [17] Manual Count: 45</p>	 <p data-bbox="951 1821 1201 1915">Predicted Count : 67 MAPE: 49% Result: No good</p>

Figure 3.4. Example of some images counted manually versus Density Map

Main Conclusions: Shanghai data set was used to train Density Map weights file so it is normal that we can get better results for images from this data set. Even that, as we can see in figure 3.4, results sometimes are not very close to what we have defined as a reasonable error (MAPE lower than 10%).

3.3 Further studies

3.3.1 R-CNN counting crowd

Figure 3.5 shows a testing using R-CNN to count people instead of Density Map. As we can see in the right image from figure 3.5, R-CNN technology is very accurate with people detection when a person could be easily identified (front image). Once we have a crowded scenario (back image) R-CNN could not detect any person as R-CNN could not identify clearly human features as eyes, head, arms, legs,...

From the total number of pixels, just 70% were correctly identified as people by R-CNN. From the remaining 30%, 10% was not crowd and the other 20% was crowd but not detected.

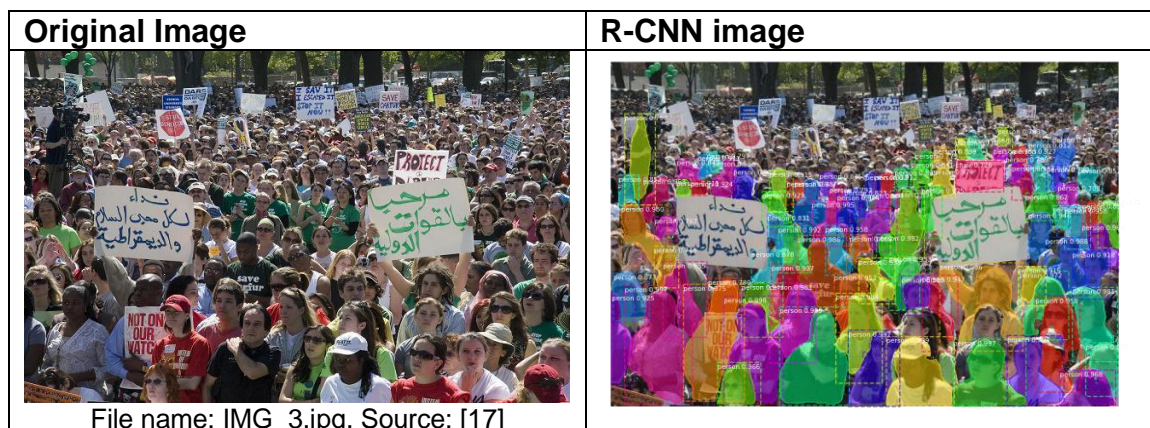


Figure 3.5. Example of one image counted just using R-CNN.

Main Conclusions: This testing justifies the need of using Density Map for crowd people counting.

3.3.2 Cropping images

As explained in section 2.5.2.1 cropping is a new process developed to crop the original images in smaller images trying to see if the quality of the results was better than processing the full image at once. Figure 3.6 shows the results of Density Map working with cropped images that allows to process images with higher resolution but in different steps.


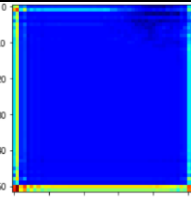
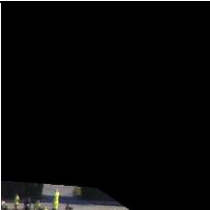
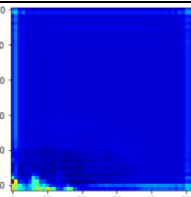
Original cropped image	Density Map image
 <p data-bbox="260 459 746 521">File name: Image1_3_3.jpg. Source: [27] Ground Truth: 1</p>	 <p data-bbox="954 472 1201 535">Predicted Count : 18 MAPE: 1700%</p>
 <p data-bbox="260 741 746 804">File name: Image1_1_7.jpg. Source: [27] Ground Truth: 8</p>	 <p data-bbox="954 741 1201 804">Predicted Count : 18 MAPE: 125%</p>

Figure 3.6. Example of some cropped images using Density Map.

As we can see in figure 3.6 when the cropped image is almost empty or with low number of pixels with crowd, the results are not accurate and always is counting extra. Two types of new functionalities were developed to minimize this effect:

- Remove empty cropped images: As the input image is a segmented image, there are many areas with only black background that are not adding any information. So, when we detected an image where all pixels are black, the image is deleted and therefore not processed by Density Map.
- Remove almost empty cropped images: As we can see in the first example of figure 3.6, images with a big percentage of black background also are providing bad results. The solution developed calculates the percentage of black pixels and deletes the image if this percentage is higher than 95% (a parameter that could be customized depending on the resolution and size of images). For our study, losing this 5% of information of a cropped images was better than keeping the original image.

After applying the previous functionalities to the same image, in figure 3.7 we can see the new results. In this example the total count provided by Density Map is 645, compared with the manual count 729 gives us a 11,52% of deviation.

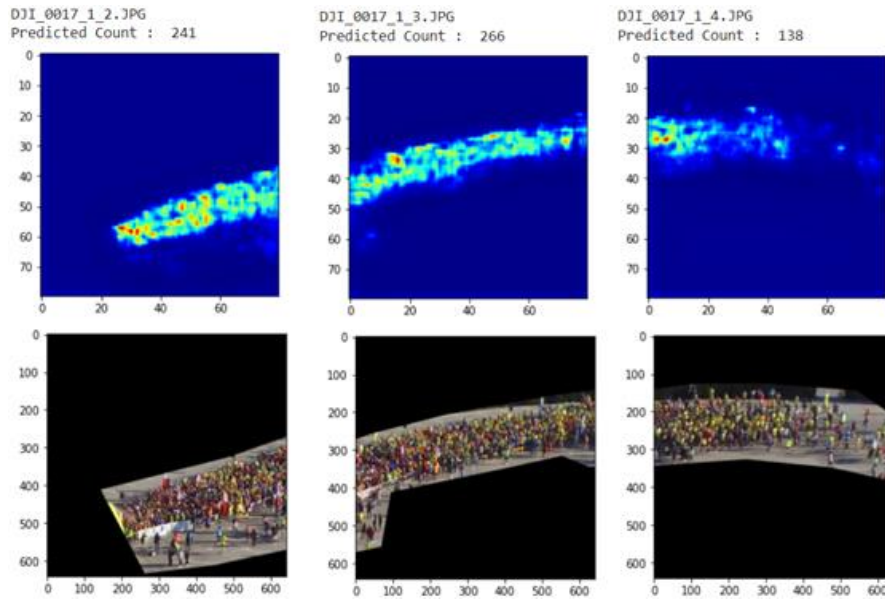


Figure 3.7. Example of final result of a cropped images using Density Map.
Source: [27]

Now we are going to compare the results with the same image but without cropping. As we can see in figure 3.8 the results are very close to the manual count, 730 from Density Map versus 729 with the manual count. What means only 0,14% deviation.


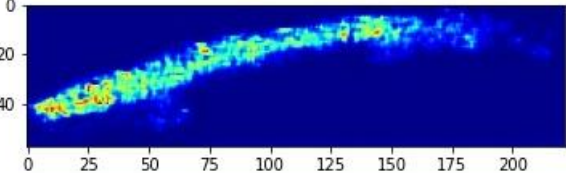
Original Image	Density Map image
 <p data-bbox="331 1355 719 1384">Manual Count: 729. Source: [27]</p>	 <p data-bbox="1018 1384 1283 1413">Predicted Count : 730</p>

Figure 3.8. Example of final result of a cropped images using Density Map.

In order to have a more accurate analysis of the cropping impact in the process, we have compared different cropping scenarios with 11 samples from Shanghai data set that can be seen in table 3.6. The 2 first columns show the result of cropping the original imagen (without segmentation) and we can see that MAPE is the same in both cases. In the next column we can see the impact with different cropping size with the segmented images. In any case the MAPE is improved.

	Original full image	Original full image	Segm. image	Segm. image	Segm. image	Segm. image
Cropping	NO	YES	NO	YES	YES	YES
Segmented	NO	NO	YES	YES	YES	YES
Cropping size	N/A	128x128	N/A	128x128	160x160	192x192
Resolution	1024x768	1024x768	1024x768	1024x768	1024x768	1024x768
MAPE	28%	28%	26%	29%	34%	32%

Table 3.6. Comparison of different examples using cropped images

Main Conclusions: After research in the process of cropping images, we have observed that cropping images is not improving the results with Density Map. Main reasons are:

- Almost empty images, even with crowd, are counting extra with the samples used in the research.
- In average using external data set, cropped images are giving worst results than the full segmented image (considering MAPE as the quality indicator).

3.3.3 Without crowd segmentation

Figure 3.9 shows an image processed by Density Map without segmentation.

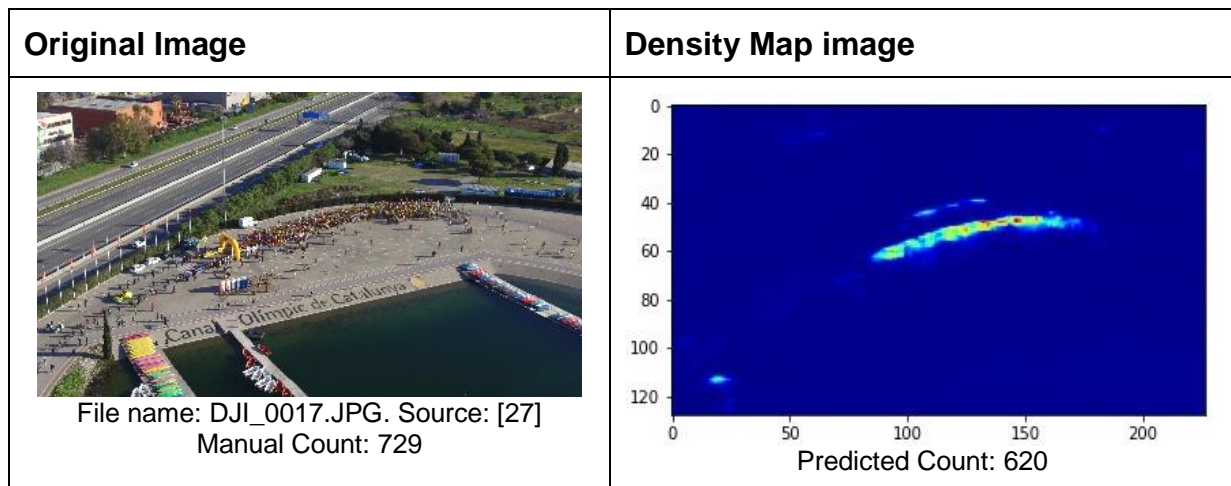


Figure 3.9. Example of an image without crowd segmentation using Density Map. Source: [27]

Main Conclusions: As we can see in figure 3.9, using the original image without segmentation with this example is giving 14,95% of deviation compared with the manual count. The same image with segmentation gives only 0,14% deviation. It demonstrates the importance for Density Map to process an image only with crowd.

3.3.4 With different resolutions

Figure 3.10 shows 2 examples of the same image but with different resolutions and what are the results using Density Map.


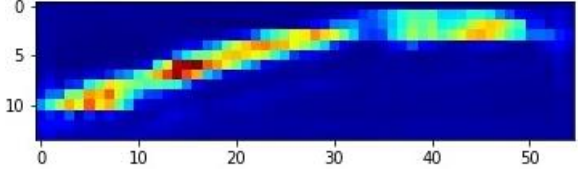

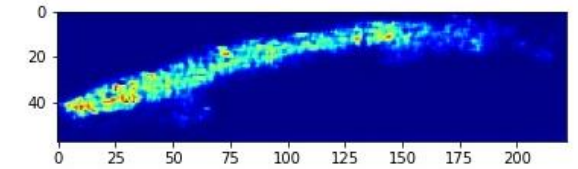

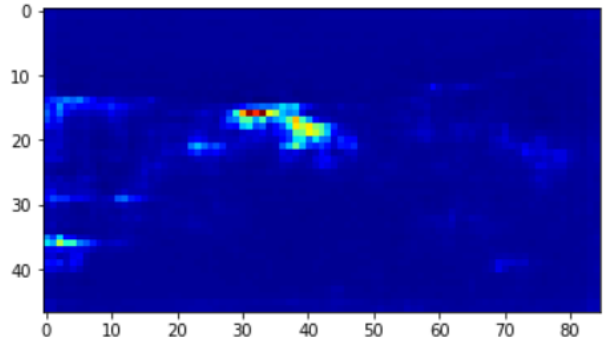
Original Image	Density Map image
 <p data-bbox="379 613 667 674">Low resolution: 32,7 KB Manual count: 729</p>	 <p data-bbox="1018 640 1262 674">Predicted count: 163</p>
 <p data-bbox="379 831 667 891">Low resolution: 211 KB Manual count: 729</p>	 <p data-bbox="1018 875 1262 909">Predicted count: 730</p>

Figure 3.10. Example of an image with different resolutions using Density Map.
Source: [27]

Main Conclusions: As we can see in figure 3.10, reducing the resolution has a very negative impact in the counting results. Specifically, 77,64% of deviation compared with the manual count.

3.3.5 With false positives

Figure 3.11 shows some examples of false positive images using Density Map.

Original Image	Density Map image
 <p data-bbox="312 1805 715 1839">Manual count: 0. Source: Internet</p>	 <p data-bbox="1002 1839 1251 1899">Predicted Count : 68 Result: No good.</p>

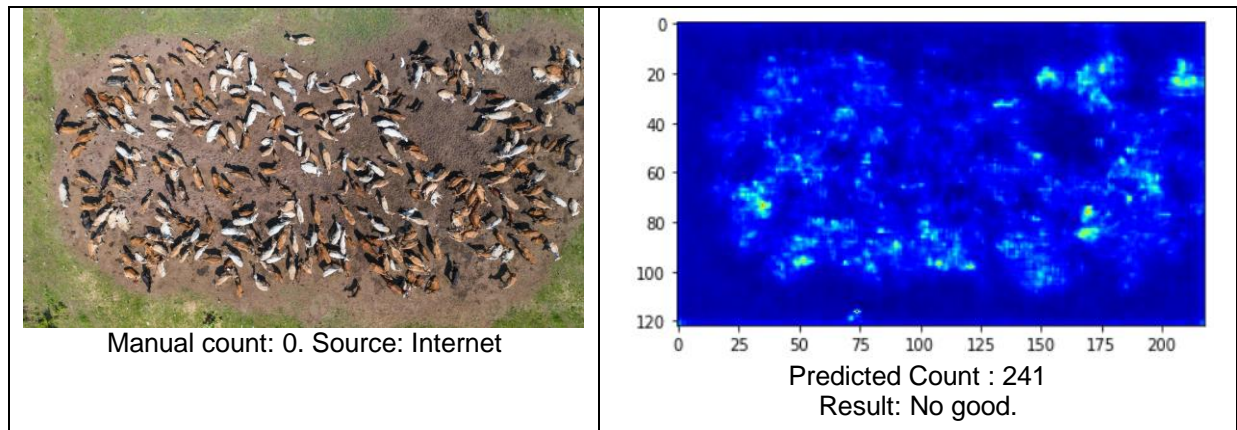


Figure 3.11. Example of images with no crowd and processed by Density Map

Main Conclusions: As we can see in figure 3.11 the input images for Density Map need to be segmented correctly before processing. In the first example we can see a landscape and how Density Map is counting the cliff and trees as crowd. In the second picture is the same problem but in this case is counting cows.

3.3.6 Comparison of results using the same image

After going through different testing scenarios in this chapter, we summarize the results in table 3.7 using the same image (DJI_0017.JPG taken in Canal Olimpico by Castelldefels Police [27]) to help understand the final conclusions.

Rank	Resol.	Cropping	Segm. image	Density Map Count	Manual Count	MAPE
1 st	High	No	Yes	730	729	0,14%
2 nd *(1)	High	Yes (416x416)	Yes	666	729	8,64%
3 rd *(2)	High	Yes (640x640)	Yes	645	729	11,52%
4 th *(3)	High	Yes (416x416)	Yes	629	729	13,72%
5 th	High	No	No	620	729	14,95%
6 th	Low	No	No	485	729	33,47%
7 th	Low	No	Yes	163	729	77,64%

Table 3.7. Results using different methods with the same image

*(1): Including empty cropped images.

*(2): Removing empty and almost empty images.

*(3): Removing empty images.

As we can see in table 3.7 the best method is using a whole segmented image only with crowd and with high resolution. It is clear also that segmentation plays a key role in the quality of the counting as the 4 main methods in the ranking are using segmentation. Also notice that low resolution has a very negative impact as the 2 last methods are the ones with lower resolution. After analysing this data, we discard the cropping process and we focus on high resolution crowd segmented images as the implementation method. Also notice that method in rank 2 is counting empty images so it's not really a valid method.

3.4 Sensitivity evaluation

In order to get objective metrics about the research, UCF-QNRF data set [6] has been used. This is a public large data set of crowd images (1535 in total) with high resolution (average of 5,8MB) and from many different scenarios taken around the world. It has been used a sample of 100 images from the UCF test data set.

Metrics 1: Segmentation results comparison

Table 3.8 shows the results of the segmentation results for the three main data sets including UCF.

Data set	Total Images	Segmented OK	Segmented No good	Percentage OK	Percentage No good
Shanghai data set	39	39	0	100%	0%
UPC and Police	17	11	6	65%	35%
UCF	100	100	0	100%	0%

Table 3.8. Segmentation results comparison



Figure 3.12. Comparison of 2 images from Shanghai and UCF data sets

Main Conclusions: UCF, even not been used at all for segmentation training, gets the same results that Shanghai data set with 100% of good segmentation.

We think the excellent result could be because the type of images is very similar between both data sets (very congested scenes with enough resolution). In figure 3.12 we can see 1 image from each data set to compare visual features, both data set are using images taken from short distances (less than 20 meters). So, we can demonstrate how the segmentation training process is good enough if we use similar type of images for training and inference avoiding data mismatch.

Metrics 2: Crow counting results comparison

Table 3.9 shows the results of the Density Map crowd counting results for the two main data sets including UCF.

Data set	MAPE	MAE	Number of Images
Shanghai Test	22,21%	74,86	182
Shanghai Training	12,89%	55,39	300
UCF	34,50%	135,54	100

Table 3.9. Crow counting result comparison

Main Conclusions: As we can see in table 3.9, UCF results are quite far from the result of the Shanghai data set (34% versus 22% considering MAPE). In the next metrics of this section, we will understand it better by analysing different indicators as the density.

Metrics 3: Comparison between original and segmented images

Table 3.10 shows a comparison between Shanghai and UCF data sets with a summary of the main indicators as MAPE between the original image and the segmented one. We can see as UCF has higher resolution and density but also worst MAPE and MAE than Shanghai. Average MAPE from the full UCF sample is 33,18%. This indicator is worse than the Shanghai data set used for our testing (20,24%).

Main indicators	Shanghai data set	UCF data set
MAPE (not segmented)	20,69%	34,50%
MAE (not segmented)	81,20	135,54
MAPE (segmented)	20,24%	33,18%
MAE (segmented)	89,22	201,99
Number of Images	82	100
Resolution average	618KB	1.379KB
Count of crowd average	449	606
Density (resolution/crowd)	2074	4891
Percentage of image segmented	62,64%	60,02%

Table 3.10. Comparison of UCF and Shanghai data set

Table 3.11 shows a summary of how segmentation and resolution impact in MAPE. The first row represents when not segmented images get better MAPE

(lower value) and the second row the opposite. As we can see in table 3.11, the results are very similar as only 51% of the images get better MAPE when are not segmented but there is a clear relationship between resolution and segmentation.

Type	Best option considering MAPE	Percentage of image segmented	Resolution average
Not segmented images	51%	55,76%	1,28 MB
Segmented images	49%	64,46%	1,47 MB

Table 3.11. Main indicators comparing image segmented and original

Main Conclusions: As it is shown in table 3.10, MAPE indicators between segmented and not segmented are very close, as there is only 1,32% different between an image segmented or not considering UCF. This is strange as we have seen in chapter 3.3.2 image areas where there is no crowd is adding noise to the process. So, it seems that segmenting images is not improving the results. But we need to consider the other 2 indicators in table 3.10 as well. As we can see the images for the testing are very congested as 60,02% (UCF) of the segmented images contain crowd. Also, we can see a relation between resolution and segmentation in table 3.11, as higher both parameters more chance to get better results with the segmentation.

Metrics 4: Evolution of MAPE based on number of people

We can see in figure 3.13 the MAPE evolution based on the number of people for each image (in orange segmented image MAPE and in blue original image MAPE).

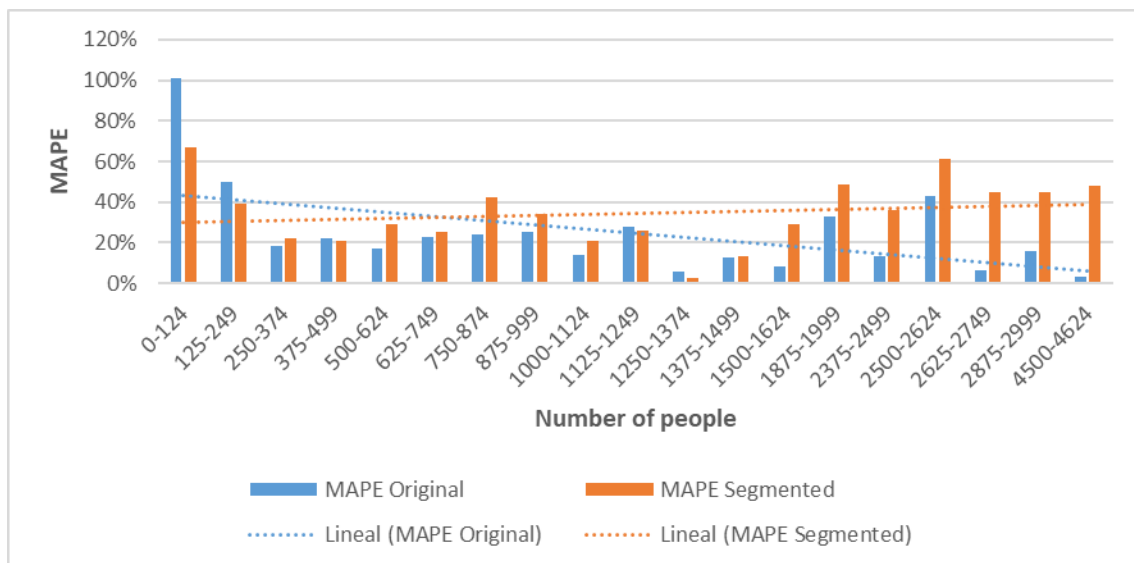


Figure 3.13. Evolution of MAPE based on number of people counted (ground truth). Bar chart representation.

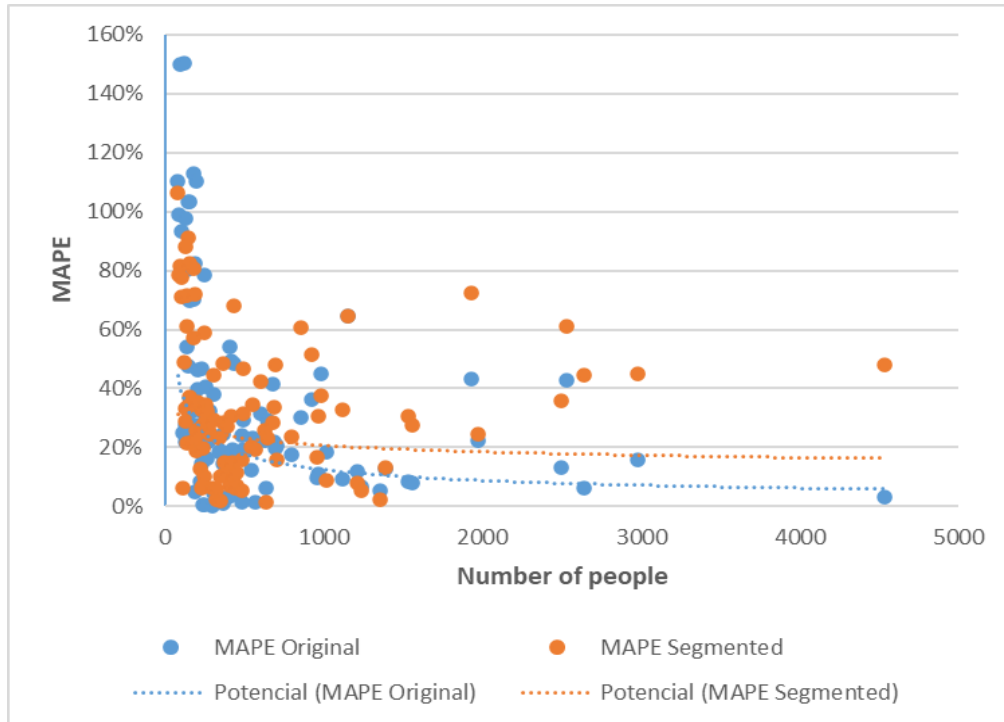


Figure 3.14. Evolution of MAPE based on number of people counted (ground truth). Dot plot representation

Main Conclusions: As we can see in figure 3.13 there is a clear relation between the number of people and the quality of the count. As much congested the scene lower the error. We can see that with less than 250 people we get the worse results. In our UCF sample 30% of the images where below 250 count and the average count was 473 so this could be one of the reason to get MAPE of 33%. Also, as we can see in figure 3.14, where each image used represents a dot, 84% of the samples are below 1.000 people counts so the metrics after this limit are not very representative for the analysis.

Metrics 5: Evolution of MAPE based on image resolution

We can see in figure 3.15 the evolution of MAPE based on resolution.

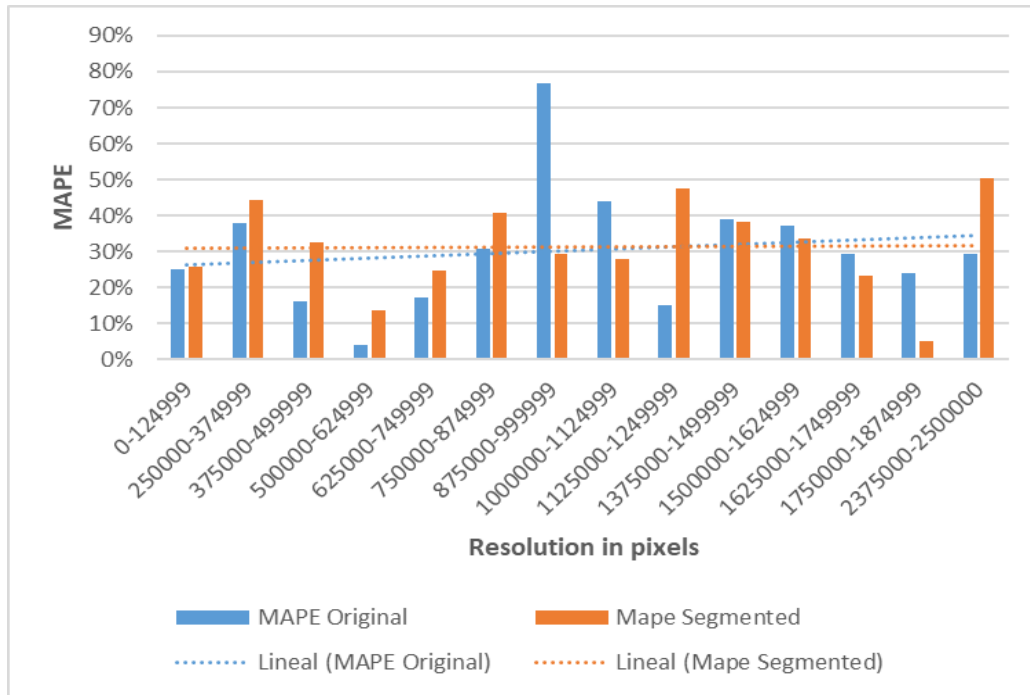


Figure 3.15. Evolution of MAPE based on image resolution

Main Conclusions: As we can see in figure 3.15 there is a relationship between the resolution and the improvement of MAPE. As much resolution better the MAPE.

Metrics 6: Evolution of MAPE based on segmentation

We can see in figure 3.16 the evolution of MAPE based on the segmentation percentage.

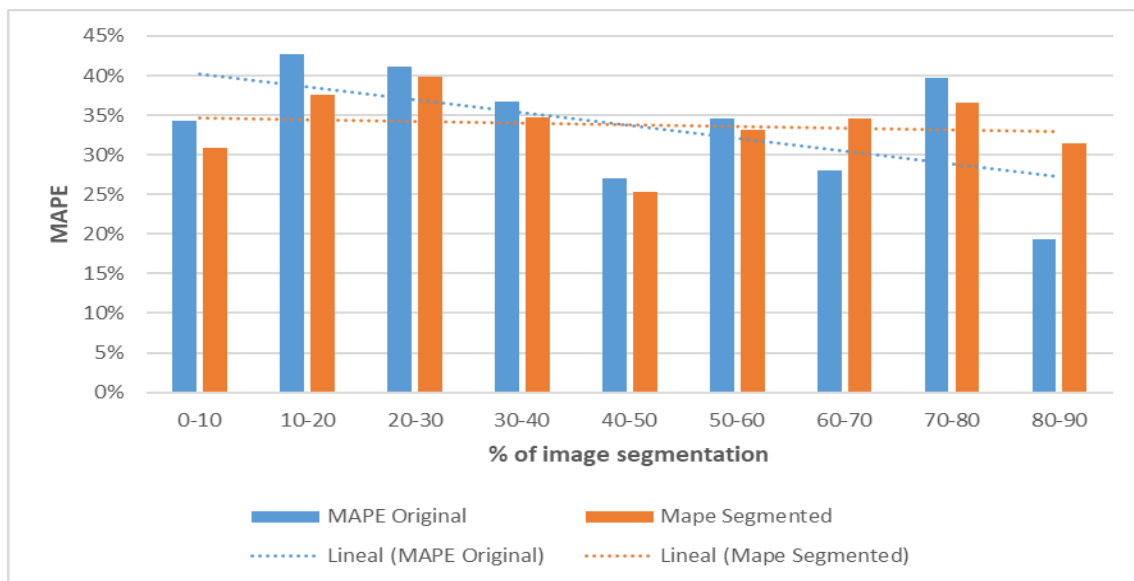


Figure 3.16. Evolution of MAPE based on percentage of background pixels after segmentation.

Main Conclusions: As we can see in figure 3.16 there is an indirect relationship between the percentage of segmentation and the improving of MAPE original. On the x-axis it is represented the percentage of the image where there is segmentation, for example the first slot (0-10%) means that the image has hardly been segmented and therefore the majority of the photo is crowd. We can see how in left part of the graph (low segmentation) MAPE of the original image is better than the one of the segmented image. The opposite happen on the left part of the graph where the level of segmentation is high.

Metrics 7: Evolution of MAPE based on density

We can see in figure 3.17 the evolution of MAPE based on the density (resolution/count of people).

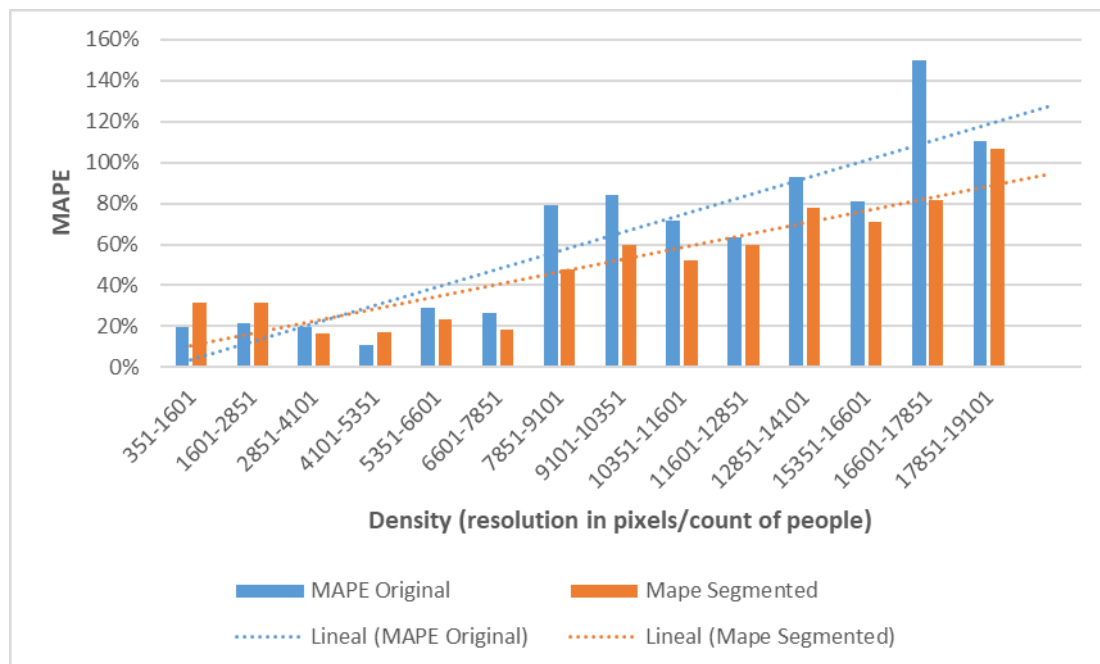


Figure 3.17. Evolution of MAPE based on density

Main Conclusions: As we can see in figure 3.17 there is a clear relationship between the increasing of density and the deterioration of MAPE. This is one of the most important indicators of the analysis of this chapter. We can demonstrate how Density Map works better with very congested images.

Metrics 8: Comparison of the different count methods

We can see in figure 3.18 a comparison between the 3 main count methods used. In blue the ground truth count, in green the segmented image count and in orange the density map count of the original image. The testing sample is of 100 images and is sorted ascending. On the left axis we have the number of people and on the right the average count. It is also included for each method a trend line.

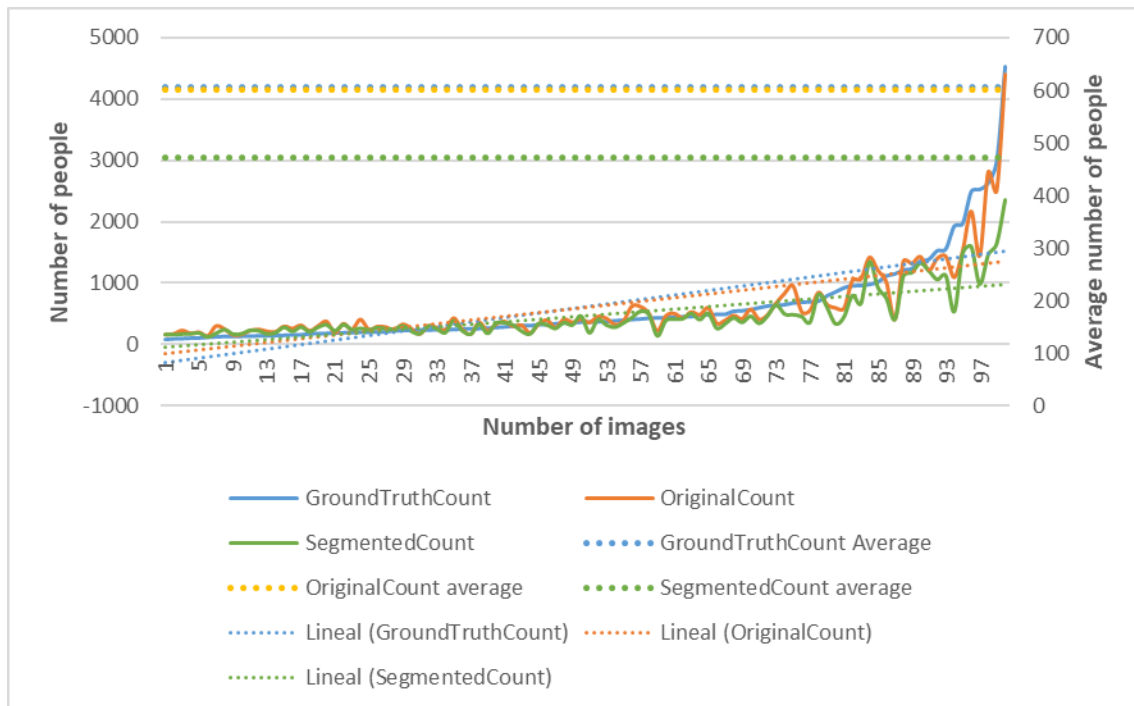


Figure 3.18. Comparison of the different count methods

Main Conclusions: The average count (on the top of the graph and left y-axis) shows that original method and ground truth are very similar. But on the other hand the average of the segmented image is much lower. Although when we compare the trend line we can see as the segmented count is counting more with low values and the opposite when we move to right side of the graph with higher count.

3.5 Processing time

The total elapsed time for the full counting process is an important factor to consider in the future when we need to implement this project in a real scenario with drones. Specially if we want a real time response.

Table 3.12 shows some rows of the file generated after each segmentation execution. The layout of the file is considering for each image: file name, processing time, size of the image and either the images was segmented or not.

Filename	Time (in seconds)	Size	Segmented Flag
IMG_26.jpg	0:00:23.679195	37980	True
IMG_22.jpg	0:00:21.904515	83664	True
IMG_99.jpg	0:00:21.635487	56354	True

Table 3.12. Example of CSV file generated after the segmentation

Table 3.13 shows some rows of the file generated after each Density Map execution. The layout of the file is considering for each image: file name, processing time, size of the image and final crowd count.

Image name	Process time	Image size	Crowd Count
DJI_0005.JPG	0:00:05.173636	107032	164
DJI_0185.JPG	0:00:05.258347	129416	161
DJI_0041.JPG	0:00:05.267386	637818	523

Table 3.13. Example of the results from Density Map image processing.

Table 3.14 shows the average time used for crowd segmentation and Density Map for each of the two main data sets. For UCF data set, average size is lower in Density Map than in segmentation because images were resized before due to Density Map size constraint. Remember that UCF data set has high resolution images.

Data set	Average time	Number of images	Average size
Shanghai: segmentation	21,87"	82	144KB
Shanghai: Density Map	1,29"	82	144KB
Shanghai: Total	23,16"	82	144KB
UCF: segmentation	27,45"	100	3,926KB
UCF: Density Map	3,52"	100	299KB
UCF: Total	30,97"	100	299KB-3,926KB

Table 3.14. Time processing results for crowd segmentation and Density Map

Main Conclusions: The total elapsed time in the worst scenario (UCF data set with high resolution) is around 30" per image. As the image size increases also time does although not proportionally. Segmentation has a big penalty in the full process, but it is needed if the image not only contains crowd. We can conclude that the processing time could allow to final users use the solution in real time without problem, as 30" per image is quite affordable. On the other hand, if time is a big constraint, Density Map graph could be removed from the process and just the final count could be provided to reduce the elapsed time a little bit. Also, higher computational power could reduce the elapsed time but there is an impact in the cost. Lower resolution images could also reduce the elapsed time but then there is an impact in the quality of the count as seen in section 3.3.4. A trade-off between time, resolution, cost and functionality needs to be agreed with the final customer.

CHAPTER 4. CONCLUSIONS AND FUTURE WORK

Conclusions of the thesis:

- **Importance of a large data set:** In any ML system the data is a key factor and in this Master Thesis it has been demonstrated. With large number of images, the system finally was able to detect a crowd with enough accuracy. Although it needed more than 600 images with big variety of scenarios to be able to do a proper training. Even that we had data mismatch with UPC and Police pictures because big part of the data set was created with external public images.
- **Segmentation algorithm for each type of images:** Also it has been demonstrated that each type of images could require an specific segmentation algorithm. In our case for example because we process small objects R-CNN was the best choice.
- **Density map only for crowd:** Density Map is a technology specifically trained for crowd so when you process images that are not a crowd the system gives bad results. Also cropping the images is not a good solution and it has been demonstrated that processing the full crowd image segmented is the best option. So, this is the main reason why it is a key point to do a good segmentation of the crowd as first step.
- **Density map performance:** it has been demonstrated how some image parameters impact the error of the counting at different levels. Some main parameters measured where the resolution, the volume of people, the background and the density (resolution/count of people). Density was the one with higher impact in the error. As much congested the scene (lower number of pixels per person) lower the error considering MAPE.
- **Time & resources:** The fact of adding an additional step for crowd segmentation means that we can get more accurate results by using different count methods, but time is penalized between 21,87 seconds (image size average of 144KB) and 27,45 seconds (image size average of 3,926KB). Crowd count is less time consuming with figures between 1,29 and 3,52 seconds. Of course, this time could be reduced increasing the GPU power processing, reducing resolution (that has an impact in accuracy) and/or losing functionalities as not generating density map or segmentation.

Future lines of development and research:

- **Improve data set:** Images taken from a drone are not easy to process. The main reasons are because of high altitude, brightness due to the sun, very low pixels per person rate in some cases as local police data set and visual obstacles like trees. Also, images taken from the police are difficult to share because of confidentiality. Therefore, building a big

data set of real drone images with people should be a next target to be able to get crowd counting from different scenarios, especially for high altitude.

- **Different scenarios:** Some kind of user interaction of the drone operator could help to define the type of scenario and then the best method to use in the calculation to get better accuracy. Input parameters could be the altitude and the scenario chosen (demonstration, beach, marathon...). For example, it is better to have a specific weights file trained with specific images from demonstrations and another with beaches. So, the operator just needs to select the best scenario and the system will take the weights file that suits better. Of course, in a second phase also the selection of the scenario could be automatized with AI.
- **Improve speed:** The total process duration takes around 30 seconds (25 seconds for crowd segmentation and 5 seconds for counting). A new line of development could be to try to reduce this time to be able to show in almost real time the image processed to the drone operator.

Sustainability considerations:

- **Drones:** In terms of sustainability the drones used by the police are reducing the need for example to fly with helicopters or small planes to get an overview of an area for security supervision. So, it has an important reduction of energy consumption and noise pollution.
- **Energy consumption:** In terms of energy consumption when the final system is running it just takes a few minutes to process the images captured by the drone. So, we can say that sustainability impact is very low. The only moderate energy consumption impact is when the system is training new images that could take many hours running. This process needs to be done just at the beginning of a new project or when a new scenario is introduced, so the impact is very low. Also, it could be scheduled to run at night where the impact is lower. Using computers with high number of GPUs could also reduce the training time but not necessarily the power consumption.

Ethical considerations:

- **COVID:** This Master Thesis has an important ethical consideration as this is mainly funded with the target to support the control of people in open spaces to reduce the risk of pandemic propagation as COVID. Therefore, it will have an important role to prevent the spread of a virus in scenarios of crowd in public areas.
- **Personal privacy data:** All pictures used in the Master Thesis either have been shared voluntarily by the Police department of Castelldefels or have been downloaded in public free web sites or have been taken in EETAC UPC campus outdoor area by UPC staff. Also, in the cases where the author is known a reference to the author has been included in

this document. Highlight also that the police and UPC images do not compromise any personal privacy or confidentiality as they are taken from high altitude in public areas where is not possible to identify a specific person. Therefore, all photos have been used in accordance with current Spanish laws⁶.

⁶ “Ley Orgánica 1/1982” <https://www.boe.es/buscar/act.php?id=BOE-A-1982-11196>

REFERENCES

- [1] Kaiming He, Georgia Gkioxari, Piotr Dollar, Ross Girshick. "Mask R-CNN". Facebook AI Research (FAIR)
https://github.com/matterport/Mask_RCNN
- [2] Yuhong Li , Xiaofan Zhang , Deming Chen. University of Illinois at Urbana-Champaign. Beijing University of Posts and Telecommunications "CSRNet: Dilated Convolutional Neural Networks for Understanding the Highly Congested Scenes"
<https://github.com/vivek-bombatkar/CSRNet-pytorch>
- [3] Images data set: COCO data set for object detection
<https://cocodataset.org/>
- [4] Géron, Aurélien. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. " O'Reilly Media, Inc.", 2019.
- [5] Rohith Gandhi. R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms
<https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e>
- [6] Images data set: University of Central Florida – Center for research in computer vision
UCF-QNRF - Large Crowd Counting Data Set
<https://www.crcv.ucf.edu/data/ucf-qnrf/>
- [7] Wikipedia. Mean absolute percentage error and mean absolute error.
https://en.wikipedia.org/wiki/Mean_absolute_percentage_error
https://en.wikipedia.org/wiki/Mean_absolute_error
- [8] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. Preprint arXiv:1409.1556, 2014.
- [9] Chen Change Loy, Ke Chen, Shaogang Gong, and Tao Xiang. Crowd counting and profiling: Methodology and evaluation. In Modeling, Simulation and Visual Analysis of Crowds, pages 347–382. Springer, 2013.
- [10] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In Proceedings of the IEEE. Conference on Computer Vision and Pattern Recognition, pages 589–597, 2016.
- [11] Lokesh Boominathan, Srinivas SS Kruthiventi, and R Venkatesh Babu. Crowdnet: a deep convolutional network for dense crowd counting. In

- Proceedings of the 2016 ACM on Multimedia Conference, pages 640–644. ACM, 2016.
- [12] Mark Marsden, Kevin McGuinness, Suzanne Little, and Noel E O’Connor. Fully convolutional crowd counting on highly congested scenes. Preprint arXiv:1612.00220, 2016.
- [13] Deepak Babu Sam, Shiv Surya, and R Venkatesh Babu. Switching convolutional neural network for crowd counting. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, volume 1, page 6, 2017.
- [14] Vishwanath A Sindagi and Vishal M Patel. Generating high quality crowd density maps using contextual pyramid CNNs. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1861–1870, 2017.
- [15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In CVPR, 2016.
- [16] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In CVPR, 2017.
- [17] Images data set: Shanghai
<https://paperswithcode.com/dataset/shanghaitech>
<https://www.shanghaitech.edu.cn/eng/>
- [18] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In NIPS, 2015.
- [19] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015.
- [20] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In CVPR, 2016.
- [21] Cong Zhang, Hongsheng Li, Xiaogang Wang, and Xiaokang Yang. Cross-scene crowd counting via deep convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 833–841, 2015.
- [22] Haroon Idrees, Imran Saleemi, Cody Seibert, and Mubarak Shah. Multi-source multi-scale counting in extremely dense crowd images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2547–2554, 2013.
- [23] A. B. Chan, Zhang-Sheng John Liang, and N. Vasconcelos. Privacy preserving crowd monitoring: Counting people without people models or tracking. In 2008 IEEE Conference on Computer Vision and Pattern Recognition, pages 1–7, June 2008.

- [24] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.
- [25] Alex Asenjo. Drones against COVID-19 propagation by controlling capacity in public spaces. UPC
<https://upcommons.upc.edu/handle/2117/352862>
- [26] Géron, A. (2019). Hands-on machine learning with scikit-learn, keras, and tensorflow: Concepts, tools, and techniques to build intelligent systems. O'Reilly Media
- [27] Images data set: Image courtesy of the “Policia Local de Castelldefels”
<https://www.castelldefels.org/>
- [28] Images data set: Image courtesy of “Mars Intelligence” company
<https://www.mars-intelligence.com/>
- [29] Images data set: Icarus research group. EETAC (UPC)
<https://icarus.upc.edu/en>
- [30] Images data set: Juan Manuel Trujillo Gómez
jmtrujillogomez@gmail.com
- [31] Images data set: Unsplash, Internet’s source of freely-usable images.
<https://unsplash.com/>