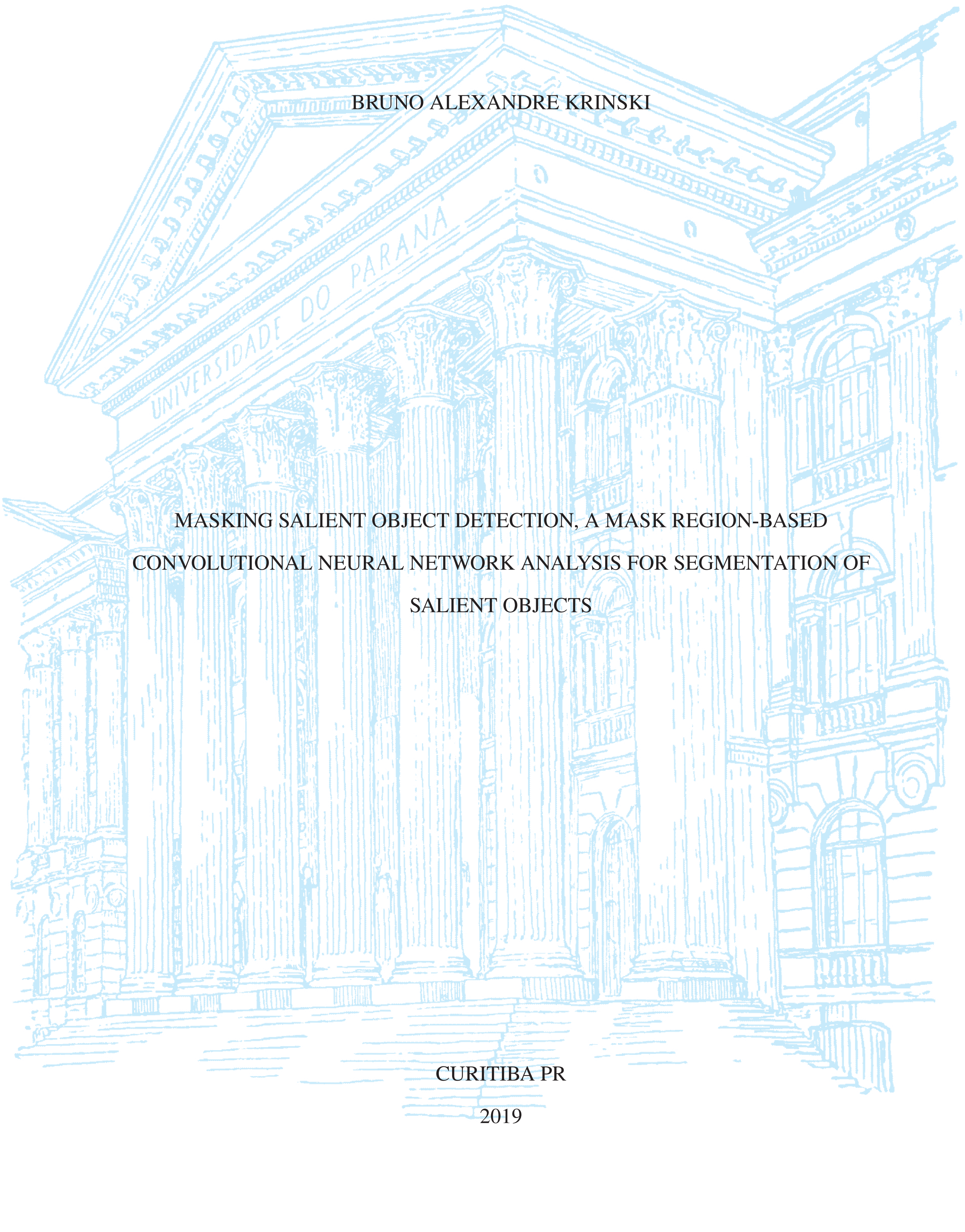
UNIVERSIDADE FEDERAL DO PARANÁ



BRUNO ALEXANDRE KRINSKI

MASKING SALIENT OBJECT DETECTION, A MASK REGION-BASED CONVOLUTIONAL NEURAL NETWORK ANALYSIS FOR SEGMENTATION OF SALIENT OBJECTS

CURITIBA PR

2019

BRUNO ALEXANDRE KRINSKI

MASKING SALIENT OBJECT DETECTION, A MASK REGION-BASED CONVOLUTIONAL NEURAL NETWORK ANALYSIS FOR SEGMENTATION OF SALIENT OBJECTS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Eduardo Todt.

CURITIBA PR

2019

Catalogação na Fonte: Sistema de Bibliotecas, UFPR
Biblioteca de Ciência e Tecnologia

K92m Krinski, Bruno Alexandre
Masking salient object detection, a mask region-based convolutional neural network analysis for segmentation of salient objects [recurso eletrônico] / Bruno Alexandre Krinski. – Curitiba, 2019.

Dissertação - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós- Graduação em Informática, 2019.

Orientador: Eduardo Todt.

1. Redes neurais (Computação). 2. Imagens. 3. Algorítmos computacionais. I. Universidade Federal do Paraná. II.Todt, Eduardo. III.Título.

CDD: 006.32

Bibliotecária: Vanusa Maciel CRB- 9/1928

MINISTÉRIO DA EDUCAÇÃO
SETOR DE CIENCIAS EXATAS
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO INFORMÁTICA - 40001016034P5

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **BRUNO ALEXANDRE KRINSKI** intitulada: **Masking Salient Object Detection, a Mask Region-based Convolutional Neural Network Analysis for Segmentation of Salient Objects**, sob orientação do Prof. Dr. EDUARDO TODT, que após após terem inquirido o aluno e realizado a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.
A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 23 de Agosto de 2019.

[illegible]

EDUARDO TODT
Presidente da Banca Examinadora (UNIVERSIDADE FEDERAL DO PARANÁ)

[illegible]

EDUARDO JAQUES SPINOSA
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

[illegible]

SÍLVIA SILVA DA COSTA BOTELHO
Avaliador Externo (UNIVERSIDADE FEDERAL DO RIO GRANDE)

MINISTÉRIO DA EDUCAÇÃO - UNIVERSIDADE FEDERAL DO PARANÁ - Setor de Ciências Exatas - Coordenação de Pós-Graduação em Informática

For my grandmother, Joana M. Krinski.
(in memoriam)

AGRADECIMENTOS

First and foremost, I would like to thank God for being my strength and guide in the writing of this dissertation. Without Him, I would not have had the wisdom or the physical ability to do so. I express my deep and sincere gratitude to my advisor Eduardo Todt, for allowing me to write this dissertation and providing invaluable guidance throughout this research. It was a great privilege and honor to work and study under his guidance. I am extremely grateful to my parents for their love, prayers, caring, sacrifices, and always supporting my education. I would like to specially thanks to my grandmother Joana M. Krinski. She was one of the most important people in my life and always helped me in times of difficulty. I would also like to thank all friends and colleagues of the Vision, Robotics, and Image (VRI) research group, that helped me to grow as a person and researcher. Thanks to the Coordination for the Improvement of Higher Education Personnel (CAPES) for supporting this work and to the NVIDIA Corporation that donated the Titan Xp used for this research.

RESUMO

Saliências Visuais são regiões ou objetos dentro de uma imagem que chamam a atenção do nosso sistema visual, sendo processados primeiro na compreensão de uma cena. Na computação, encontrar regiões salientes em imagens pode ser usado como pré-processamento em áreas como reconhecimento, detecção e rastreamento de objetos. Pode ser usado para recortar e redimensionar imagens, comprimir e resumir vídeos ou imagens. Na robótica, algoritmos de localização e mapeamento de ambientes também utilizam conceitos de Saliências Visuais. Este trabalho se concentra na área de detecção de objetos salientes, que visa encontrar os objetos salientes de uma imagem e segmentar toda a sua extensão. O resultado desta segmentação é uma imagem binária, representando um Mapa de Saliência, com as regiões de saliência em branco e as regiões de não-saliência em preto. Os primeiros métodos aplicados para resolver este problema foram baseados em características extraídas manualmente. Com o advento da *Convolutional Neural Network* (CNN), melhores resultados foram obtidos porque ela aprende as características apropriadas para o problema. Atualmente, a *Fullly Convolutional Network* (FCN) é o elemento chave das técnicas de ponta no problema da segmentação de objetos salientes. Entretanto, apesar dos resultados promisores alcançados na literatura com redes baseadas em FCNs, a FCN mostrou problemas em algum cenários desafiadores. Muito recentemente, estudos propuseram o uso de uma abordagem baseada na *Mask Region-based Convolutional Neural Network* (Mask-RCNN) para superar tais problemas. Entretanto, não há nenhuma comparação extensiva entre as duas redes na literatura de *Salient Object Detection* (SOD) endorsando a eficácia da Mask R-CNN sobre a FCN para segmentar objetos salientes. Com o objetivo de mostrar a superioridade da Mask R-CNN sobre FCN no contexto de SOD, este trabalho propõe duas arquiteturas Mask R-CNN e realiza uma ampla comparação entre a Mask R-CNN e a FCN no contexto de SOD. Neste trabalho, variações da Mask R-CNN são comparadas com variações da FCN em oito conjuntos de dados amplamente utilizados na literatura. Os resultados deste trabalho mostram melhorias de até 47% no F-measure com a Mask R-CNN em comparação com a FCN. Embora os resultados alcançados neste trabalho não superem os resultados mais recentes da literatura, este trabalho mostra a Mask R-CNN como uma alternativa promissora para o problema de SOD.

Palavras-chave: Saliência, FCN, Mask R-CNN

ABSTRACT

Visual Saliences are regions or objects within an image that draw the attention of our visual system, being processed first in the understanding of a scene. In computing, finding salient regions in images can be used as pre-processing in areas such as object recognition, detection, and tracking. It can be used to crop and resize images, compress and summarize videos or images. In robotics, algorithms of location and mapping of environments also utilize Visual Salience concepts. This work focuses on the area of salient object detection, that aims to find the salient objects of an image and to segment their entire extension. The result of this segmentation is a binary image, representing a Salience Map, with the salience regions in white and the non-salience regions in black. The first methods applied to solve this problem were based on hand-crafted features. With the advent of the Convolutional Neural Network (CNN), better results were obtained because it learns appropriate characteristics for the problem. Nowadays, the Fully Convolutional Network (FCN) is the key element of the state of the art techniques in the problem of segmentation of salient objects. However, besides the promising results achieved in the literature with networks based on FCNs, the FCN showed issues in some challenging scenarios. Fairly recently, studies proposed the use of a Mask Region-based Convolutional Neural Network (Mask R-CNN) based approach to overcome such issues. However, there is no extensive comparison between the two networks in the Salient Object Detection (SOD) literature endorsing the effectiveness of Mask R-CNNs over FCN when segmenting salient objects. Aiming to show the superiority of Mask R-CNNs over FCNs in the SOD context, this work proposes two Mask R-CNN architectures and performs a broad evaluation between the two early mentioned segmentation approaches. In this work, variations of Mask R-CNNs are compared with variations of FCNs in eight datasets widely used in the literature and the findings of this work show improvement from up to 47% in the F-measure with the Mask R-CNN. Although the results achieved in this work do not exceed the most recent results in the literature, this work shows the Mask R-CNN as an promising alternative to the SOD problem.

Keywords: Salience, FCN, Mask R-CNN

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

SOD	Salient Object Detection
CNN	Convolutional Neural Network
FCN	Fully Convolutional Network
PCA	Principal Component Analysis
SLIC	Simple Linear Iterative Clustering
DoG	Difference of Gaussians
GOP	Geodesic Object Proposal
SVM	Support Vector Machine
MLP	MultiLayer Perceptron
MAE	Mean Absolute Error
ILSVRC	ImageNet Large-Scale Visual Recognition Challenge
ReLu	Rectified Linear Units
RNN	Recurrent Neural Network
RoIPool	Region of Interest Pooling
RoIAlign	Region of Interest Alignment
RPN	Region Proposal Network
FPN	Feature Pyramid Network
R-CNN	Region-based Convolutional Neural Network
Fast R-CNN	Fast Region-based Convolutional Neural Network
Faster R-CNN	Faster Region-based Convolutional Neural Network
Mask R-CNN	Mask Region-based Convolutional Neural Network
HED	Holistically-Nested Edge Detector
MCG	Multiscale Combinatorial Grouping
ResNet	Residual Network
GPU	Graphics Processing Unit
FPN	Feature Pyramid Network

SUMÁRIO

1 INTRODUCTION

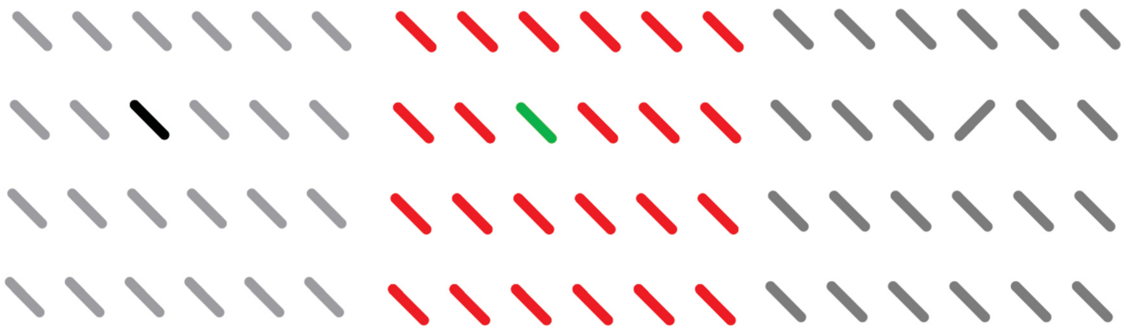
This dissertation presents a study about Visual Salience, explores the leading solutions proposed in the literature to extract Visual Salience regions from images, and performs a comparison study about two methods proposed in the literature to perform segmentation of salient objects. The present chapter introduces the Visual Salience theme and is divided into sections as follows: Section 1.1 presents the Visual Salience concept, Section 1.2 shows how images depict Visual Saliences and the Salient Object Detection (SOD) research field. Section 1.3 presents the objectives and contributions of the present study, and Section 1.4 offers the subjects covered in the following chapters.

1.1 VISUAL SALIENCE

The human brain receives vast amounts of information daily, with the visual system being considered one of the most essential to perform daily tasks. However, the complete understanding of a scene in the real world is a challenging task. Even the human brain suffers limitations to simultaneously identify all interesting elements and objects in a scene [Itti (2007)].

To deal with all information received, the human visual system, as well as the visual system of other primates, chooses elements or objects in the scene to process first through a selection of the most important visual stimuli [Gottlieb et al. (1998)]. The human brain subdivides the image into subsets of regions and chooses the most relevant for first processing or more detailed analysis, repeating the process to all regions in scene accordingly to decreasing order of salience [Itti (2007)].

Visual Salience, or Visual Saliency, is the contextualized characteristic of some objects that makes them stand out from its surrounding regions and attract the attention of the human brain [Itti (2007)]. Light intensity, edge or line orientation, color, motion, and stereo disparity are examples of Visual Salience features that attract human attention.



(a) Example of salience by the difference of intensity (the black bar stands out from the green bars).

(b) Example of salience by the difference of color (the green bar stands out from the red ones).

(c) Example of salience by the difference of orientation (the bar inclined to the opposite direction stands out from the others).

Figura 1.1: Examples of saliency features. Source: Todt (2005).

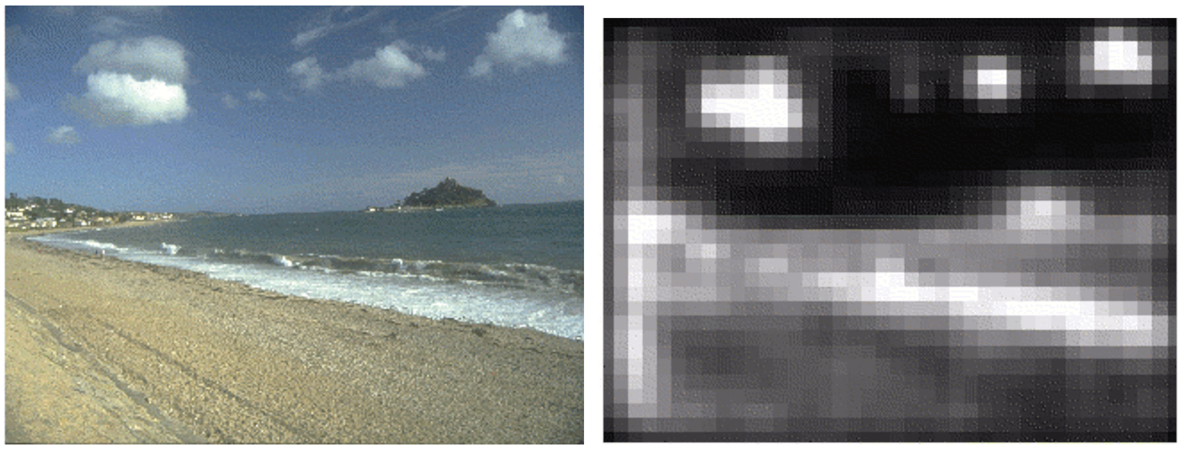
According to Niebur (2007), there are two approaches to select visual stimuli in the attentional process: bottom-up and top-down. The former considers the instantaneous sensory input while the latter thinks the internal state like goals, personal history, and experiences. The present work studies bottom-up Visual Saliences.

Figure 1.1 presents examples of bottom-up saliency features by difference of intensity (the black bar stands out from the gray bars), by difference of color (the green bar stands out from the red bars), and by the difference of orientation (the bar inclined to the opposite direction stands out from the others).

1.2 PROBLEM STATEMENT

Currently, there is a growing demand for computational solutions to solve common day-to-day problems like surveillance systems, autonomous vehicles, applications in industry, medicine, and military field. However, even modern computers have restrictions on bandwidth, memory, and processing power, making it difficult to achieve reasonable time in real-world applications.

Due to such limitations, it is essential to define processing priorities [Niebur (2007)]. As presented in Section 1.1, Visual Saliences are naturally utilized by the human vision system to select essential regions in a scene for priority processing. Aiming to represent Visual Saliences in computer systems, Koch and Ullman (1987) proposed the idea of Saliency Map.



(a) Example of a real-world scene.

(b) Example of Saliency Map. The salient regions (the surf line, the clouds, and the island) receives higher values of white to highlight these regions.

Figura 1.2: Example of a real-world scene and the corresponding saliency map. Source: Niebur (2007).

The Saliency Map is a gray-scale image where each pixel receives a scalar value between 0 and 255 to represent the Visual Saliency level and guide the selection of attended locations based on the spatial distribution of the saliency [Itti et al. (1998)]. Figure 1.2a presents an example of a real-world scene, and Figure 1.2b shows an example of a Saliency Map, as proposed by Koch and Ullman (1987). In Figure 1.2b, the surf line, the clouds, and the island are represented with higher values of white to highlight the salient regions.

In Computer Vision, there are many research directions related to Visual Saliency. The three most relevant are: SOD, Fixation Prediction, and Object Proposal [Borji et al. (2014)]. The SOD aims to find the essential objects in a scene while performs the segmentation of the salient objects. The Fixation Prediction seeks to predict where humans look. The Object Proposal produces a set of bounding box candidates and regions proposals for each salient object in the scene [Borji et al. (2014)]. The present work addresses the SOD problem, the Visual Saliency related topic with the highest growth in recent years [Borji et al. (2014)].

The SOD problem has a well defined evolutionary research line. The first attempts to find saliency regions and generate the Saliency Map applied heuristics to find salient features (difference of color, contrast, and orientation as shown in Figure 1.1) and standard Computer Vision techniques to perform the segmentation of the salient regions [Borji et al. (2014)].

The second generation of studies presented in the SOD literature take advantage of the promising results achieved by Convolutional Neural Network (CNN) [Lecun et al. (1998)] in image classification and detection tasks [Borji et al. (2014)]. To generate a Saliency Map with a CNN, the methods presented in the SOD literature utilized super-pixel and patches strategy to divide the input image into regions, applying the CNN to classify the generated regions as saliency or background regions.

The most recent approaches proposed in the SOD problem are based on Fully Convolutional Network (FCN) [Long et al. (2015)], an evolution of CNNs to perform Semantic Segmentation [Thoma (2016)]. In a FCN, the fully connected layers utilized in the CNN to classify the objects are converted into convolutional layers to perform per-pixel classification [Long et al. (2015)], overcoming problems such as blurriness and inaccurate predictions near the boundaries of salient objects in super-pixels and patches based approaches [Borji et al. (2014)].

1.3 OBJECTIVE AND CONTRIBUTIONS

Recent studies proposed in the SOD literature [Li et al. (2016); Tang et al. (2016); Zhang et al. (2017); Tang and Wu (2016); Xi et al. (2017); Hou et al. (2018); Xie and Tu (2017); Zhang et al. (2018, 2019); Li et al. (2018)] based their studies in FCNs to perform segmentation of salient objects in images. However, besides the promising results achieved by these networks in the SOD problem, FCNs have difficulties in distinguishing salient regions from non-salient regions in some challenging scenarios. Figures 1.3a, 1.3b, 1.3c, and 1.3d presents examples of changeling scenarios. In these images, the background has a high number of objects with different colors, intensities, orientations and illumination.

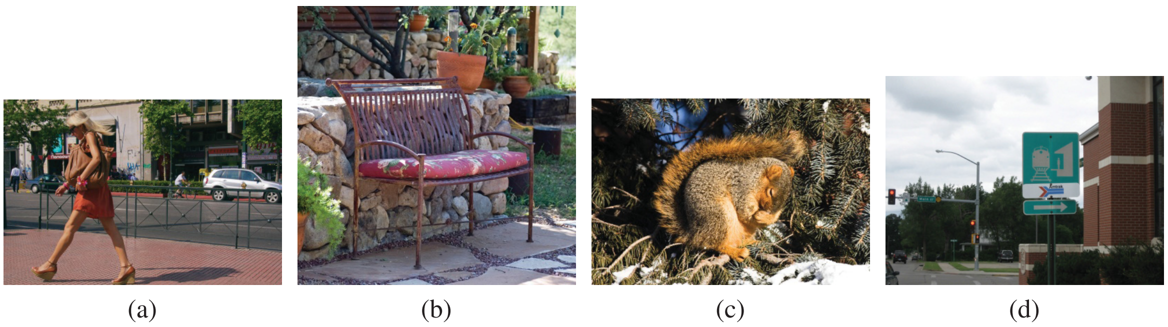

(a) (b) (c) (d)

Figura 1.3: Examples of changeling scenarios. The background has a high number of objects with different colors, intensities, orientations, and illumination. Source: ECSSD Dataset.

Figures 1.4a, 1.4b, 1.4c, and 1.4d presents examples of segmentation results generated by FCNs in preliminary experiments performed in the present work. The FCN achieved impressive results and was able to find the salient object, generating a segmentation involving most of the pixels of the salient objects. However, the FCN wrongly segmented large portions of the image not belonging to the saliency region. Trying to solve similar problems, Nguyen et al. (2019) fairly recently proposed the use of a Mask Region-based Convolutional Neural Network (Mask R-CNN) [He et al. (2017)] approach to SOD. However an extensive comparison between the FCN and the Mask R-CNN was not performed.

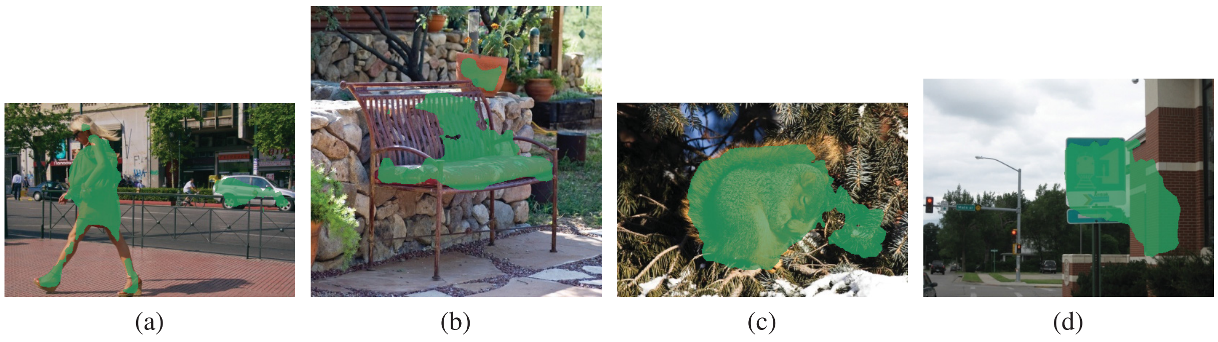


Figura 1.4: Examples of segmentation results generated in preliminary experiments performed in the present work with FCNs. The FCNs achieved impressive results and was able to find the salient objects and generate a segmentation involving most of the salient pixels. However, the FCN wrongly segmented regions not belonging to the saliency object. Source: Author.

Taking in mind the shallow evaluation between the FCN and the Mask R-CNN available in the literature, this work proposes two Mask R-CNN architectures and performs a broad evaluation between the two early mentioned segmentation approaches. The networks are compared by training FCNs and Mask R-CNNs architectures to perform segmentation of salient objects. Three variations of FCN are utilized, two based on Residual Networks (ResNets) and one based on VGG-16 architecture, and two variations of Mask R-CNNs are utilized, both based on ResNets.

Two frameworks containing the implementation of the FCNs and Mask R-CNNs were utilized: KittiSeg [Teichmann et al. (2018)] and Google Object Detection API [Huang et al. (2017)]. Precision, Recall, F-measure, and Mean Absolute Error (MAE) evaluation metrics were employed to evaluate and compare the FCNs and Mask R-CNNs models, following related research [Borji et al. (2014)].

The networks were trained in the MSRA10K [Cheng et al. (2015)] dataset (following similar work in the literature [Nie et al. (2018); Tang and Wu (2016); Zhang et al. (2017); Li et al. (2016); Xi et al. (2017); Tang et al. (2016); Kruthiventi et al. (2016); Zhang et al. (2018, 2019)]) and evaluated in eight datasets of salient objects widely used in the SOD literature: DUT-OMRON [Yang et al. (2013)], ECSSD [Shi et al. (2016)], HKU-IS [Li and Yu (2015)], ICOSEG [Batra et al. (2010)], PASCAL-S [Li et al. (2014)], SED1 [Borji et al. (2015)], SED2 [Borji et al. (2015)] and THUR [Cheng et al. (2014)].

1.4 OUTLINE

The present dissertation proposes a broad evaluation between two segmentation approaches, the FCN and the Mask R-CNN, applied in the segmentation of salient objects context. Chapter 2 provides a general background of Machine Learning, Deep Learning, and CNNs. Chapter 3 reviews state-of-art methods presented in the SOD literature to perform segmentation of salient objects. Chapter 4 presents the variations of the FCN and the Mask R-CNN evaluated in the present work, explaining implementation details. Chapter 5 presents the datasets and metrics utilized to compare the networks. Chapter 6 presents the experiments performed and the results of the FCN and Mask R-CNN comparison. Finally, Chapter 7 summarizes the contributions and discusses possible future works.

2 MAIN CONCEPTS

The present chapter explains the fundamental concepts of the methods and algorithms approached in the following chapters. Section 2.1 explains Machine Learning algorithms and the scene understanding problem. Section 2.2 presents the Artificial Neural Networks. Section 2.3 presents the Deep Learning, a subfield of the Machine Learning which explores Artificial Neural Networks to solve complex learning tasks. Section 2.3 also details a Convolutional Neural Network (CNN) architecture, presents two CNNs models, and explain the Fully Convolutional Network (FCN) and the Mask Region-based Convolutional Neural Network (Mask R-CNN) architectures. Finally, Section 2.4 presents a conclusion of the chapter.

2.1 MACHINE LEARNING AND SCENE UNDERSTANDING

The human beings receives large amount of data through its five senses [Tripathi (2017)] and the human brain learns very quickly how to handle all this data, being easy to recognize objects in a scene by its format, food by its smell and taste or other people by the voice sound. Nowadays, with technology growth, machines have a lot of different sensors to collect data from a variety of sources to understand the surrounding environments [Tripathi (2017)]. However, computer systems have limited processing capacities to handle large amounts of data.

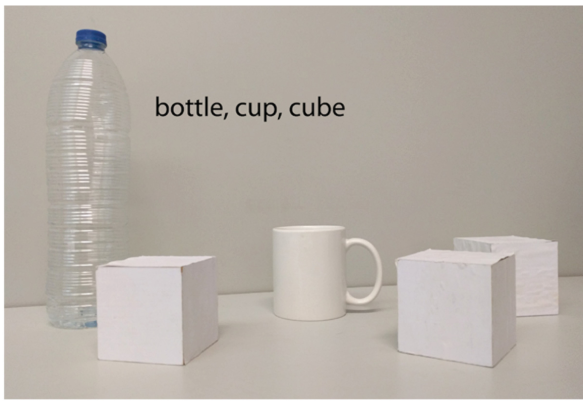
Machine Learning is a subfield of Artificial Intelligence and aims to make computers learn from data without the need of being explicitly programmed [Vishal Maini (2017)]. The main concern of machine learning is the study of algorithms that allow computers to learn similarly to humans [Jeremy Watt (2017)]. The present work applies the Machine Learning in the scene understanding problem, intending to detect and segment salient objects in images.

The scene understanding problem can be divided into classification, detection, semantic segmentation, and instance segmentation problems. The classification is the ability to distinguish different types of objects [Jeremy Watt (2017)]. Figure 2.1a presents a classification example. Each object in the figure receives a label (classification) according to a category. The detection aims to find the image regions where the objects are located, generating a set of points corresponding to the bounding boxes of each object in the image, as presented in Figure 2.1b.

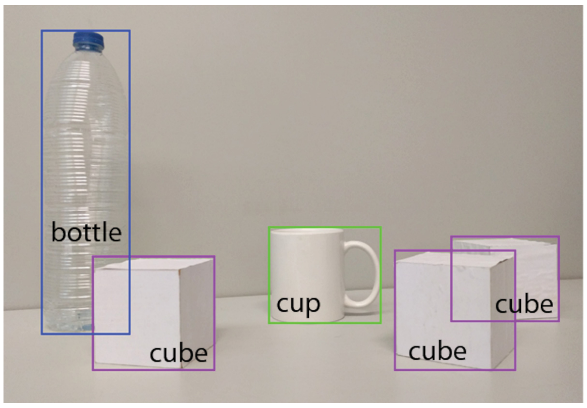
The semantic segmentation aims to cluster regions in the image that belongs to the same object [Thoma (2016)]. In semantic segmentation, every pixel in the image receives a classification according to the category it belongs. Figure 2.1c presents a segmentation example with the scene segmented in four categories: cup, cube, bottle, and background. The semantic segmentation does not identify different object instances.

The instance segmentation is a step further of semantic segmentation and aims to identify the different instances of objects, giving a label (classification) to each segmented object [Garcia-Garcia et al. (2017)]. Different from the semantic segmentation example presented in Figure 2.1c, all instances of the cube are separated by a different color in Figure 2.1d.

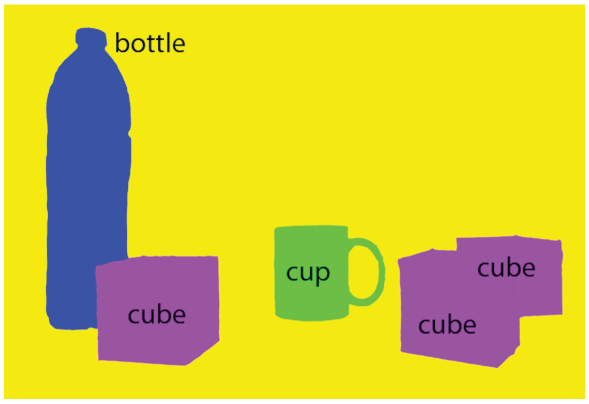
The classification problem and its derivatives (detection, semantic segmentation, and instance segmentation) can be divided in a pipeline of four steps: collecting training data, designing features, model training, and model testing [Jeremy Watt (2017)]. Following paragraphs explain each classification steps.



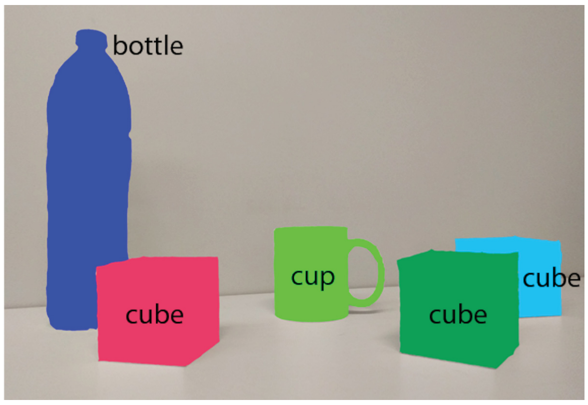
(a) Classification: each object in the image receives a label.



(b) Object Detection: each object receives sets of points corresponding to the bounding boxes.



(c) Semantic Segmentation: each pixel in the image receives a label.



(d) Instance Segmentation: each pixel in the image and each object in the image receives a label.

Figura 2.1: Differences between image classification, object classification, semantic segmentation, and instance segmentation problems. Source: Garcia-Garcia et al. (2017).

2.1.1 Collecting Training Data

Collecting examples of data to be learned is the first step in a classification system. The type of data collected depends on the problem addressed (images, sounds, text, and others) and is referenced as the training set. Generating a good training set is very important in Machine Learning with more extensive and diverse training sets improving the algorithm learning [Jeremy Watt (2017)]. The present work focuses on the scene understanding problem, and the training data is a set of images divided into different datasets.

The classification problem can be divided into two different problems: supervised and unsupervised learning. In unsupervised learning, the system does not know the categories in the scope of the problem, and the learning algorithm tries to learn the categories by clustering similar examples in the training set. Differently, the supervised learning knows the categories of all objects in the training set and aims to establish a rule to classify a new observation into one of the existing categories [Michie et al. (1994)]. The Salient Object Detection (SOD) problem is a supervised learning problem, and for each image, there is a corresponding mask with the label of each pixel.

2.1.2 Designing Features

The second step in a Classification system is the extraction of features to differentiate the types of objects in the training set [Jeremy Watt (2017)]. Exemplifying with a Machine Learning system to differentiate cats from dogs, the features extracted from the objects can be the nose size, ears shape, or weight. The features have to make a proper differentiation between the object being classified.

However, finding useful features is not a trivial task [Jeremy Watt (2017)]. The present work utilizes CNNs to perform the extraction of features from the images. CNNs are Deep Learning algorithms based on artificial neural networks that learn the best features to extract from the images to each classification problem (more details about CNNs are presented in Section 2.3.1).

2.1.3 Model Training

The objective in the training step is to find the best parameters of a learning algorithm, also called classifier, to solve the classification problem based on the collected data. The parameters learned by the classifier algorithm defines a hyperplane, also called decision boundary, to separate the objects in categories.

Figure 2.2 presents an example of a linear hyperplane separating a problem with two classes and two features in the left. However, real-life problems have more categories and features, and the hyperplane which separates the categories is more complex, as presented on the right side of Figure 2.2.

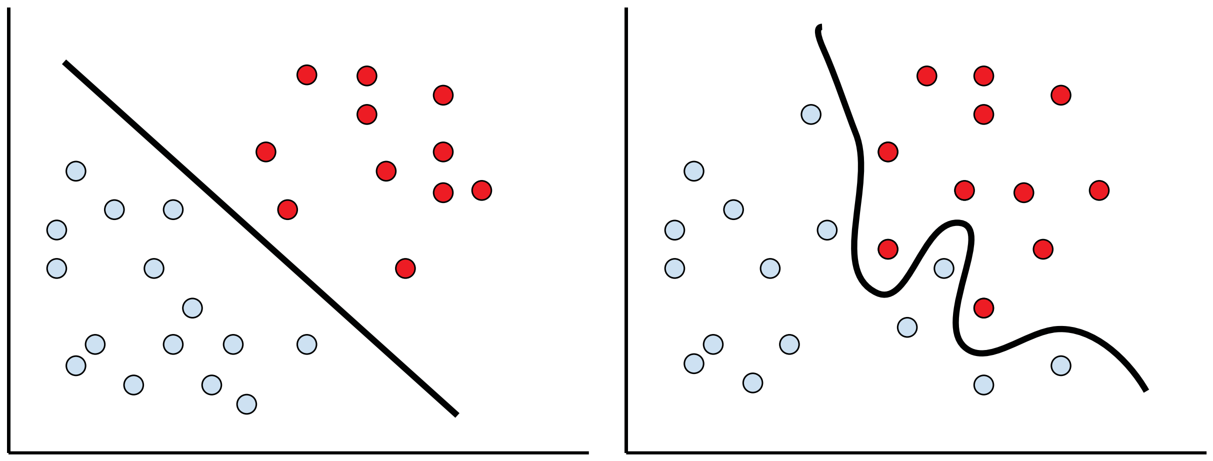


Figura 2.2: Classification example with two categories (red and blue balls) in a 2D space. In the left side, a linear hyperplane divides the two categories. In the right side, the hyperplane that divides the two categories is more complicated. Source: de Oliveira Ceschin (2018).

2.1.4 Model Validation and Test

After training a model, it is necessary to validate the classifier efficacy in the problem [Jeremy Watt (2017)]. The validation is performed to improve the classifier results by pointing weakness in the classification pipeline. Increase the number of examples in each category, rephrase the features and adjust the classifier parameters are examples of actions to improve the classification results [Jeremy Watt (2017)].

With the model trained and validated, the next step is to test the model to get the actual results in the classification problem. Both validation and testing are performed in separated sets of data not utilized in the training step.

2.2 ARTIFICIAL NEURAL NETWORKS

An Artificial Neural Networks is a computational model inspired in the human brain. It is formed by structures called neurons and can find very complex decision boundaries to solve classification problems.

2.2.1 Perceptron

The Perceptron, introduced by Rosenblatt (1958), is the simplest Artificial Neural Network composed by only one neuron. A neuron is composed of inputs, weights, bias, activation function, and output. Figure 2.3 presents the base structure of a neuron. The sets $x_1, \ldots, x_{n-1}, x_n$ and $w_1, w_2, \ldots, w_{n-1}, w_n$ represents the neuron's input and weights respectively. The constant input in Figure 2.3 with value 1, called bias, is utilized to perform a fine adjustment in the decision boundary.

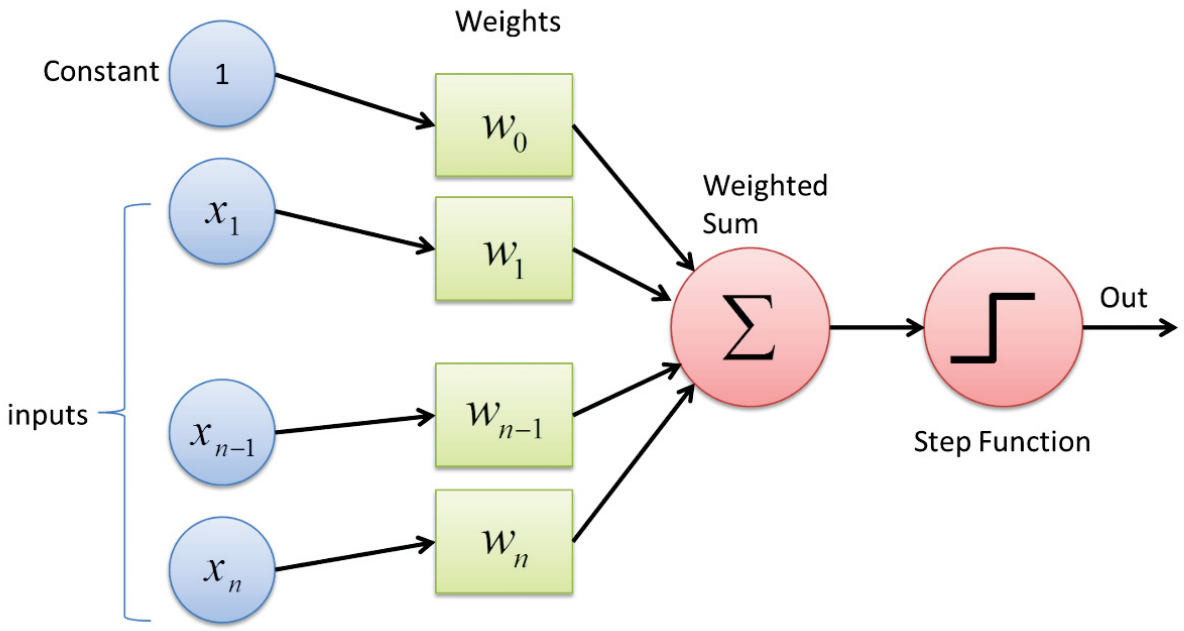


Figura 2.3: Example of Perceptron Structure, the simplest Artificial Neural Network with only one neuron. The sets $x_1, \ldots, x_{n-1}, x_n$ and $w_1, w_2, \ldots, w_{n-1}, w_n$ represents the neuron's input and weights, respectively. Each input is multiplied by the referent weight, and all values are added together to input the step function, also called activation function. The value constant 1 is the bias. In the example, the hardlim function is utilized as the activation function. Source: Sharma (2018).

The operation realized by the Perceptron is presented in Equation 2.1. The input set is multiplied by the weight set and then added together with the bias "b". The sum feeds the activation function represented by "F". The example presented in Figure 2.3 utilizes the hardlim function, where the output is 1 if the sum is higher than 0 or 0 otherwise. There are other functions to be utilized as the activation function like softmax, sigmoid, tanh, and relu.

$$y = F(\sum_{i=1}^{n} w_i \times x_i + b) \tag{2.1}$$

The Perceptron is a supervised learning classifier. So, the output of the activation function, "y"in Equation 2.1, is compared with the expected output of the given input. If the output differs from the expected, an error value is calculated and utilized to update the weights. The training step repeats this process until the error reaches a minimum value [Jeremy Watt (2017)].

The Perceptron has the limitation of being designed to solve linearly separable classification problems [Jeremy Watt (2017)], as the example presented on the left side of Figure 2.2. To found complex decision boundaries, as the example presented on the right side of Figure 2.2, MultiLayer Perceptrons (MLPs), Artificial Neural Networks composed by more than one neuron, are used [Haykin (2009)].

2.2.2 MultiLayer Perceptron (MLP)

The MLP is a Deep Artificial Neural Network composed by more than one Perceptron and is divided into three types of layers: input, hidden, and output layers. In the MLP, the neurons of each layer are connected only to the neurons of the next layer, with the layers structured as follows: The first layer is the input layer and has the number of nodes equal to the number of features extracted from the objects being classified.

Following the input layer, the MLP has a sequence of one or more hidden layers formed by a sequence of neurons (Perceptrons). The number of neurons in each hidden layer is a parameter adjusted depending on the scope of the problem.

The last layer, called output layer, is formed by a sequence of Perceptrons. Each neuron represents an object category in the problem scope and outputs the probability of the input object be of that category [Haykin (2009)]. Figure 2.4 presents an example of MLP structure with the input layer, two hidden layers composed of four neurons, and the output layer composed by one neuron.

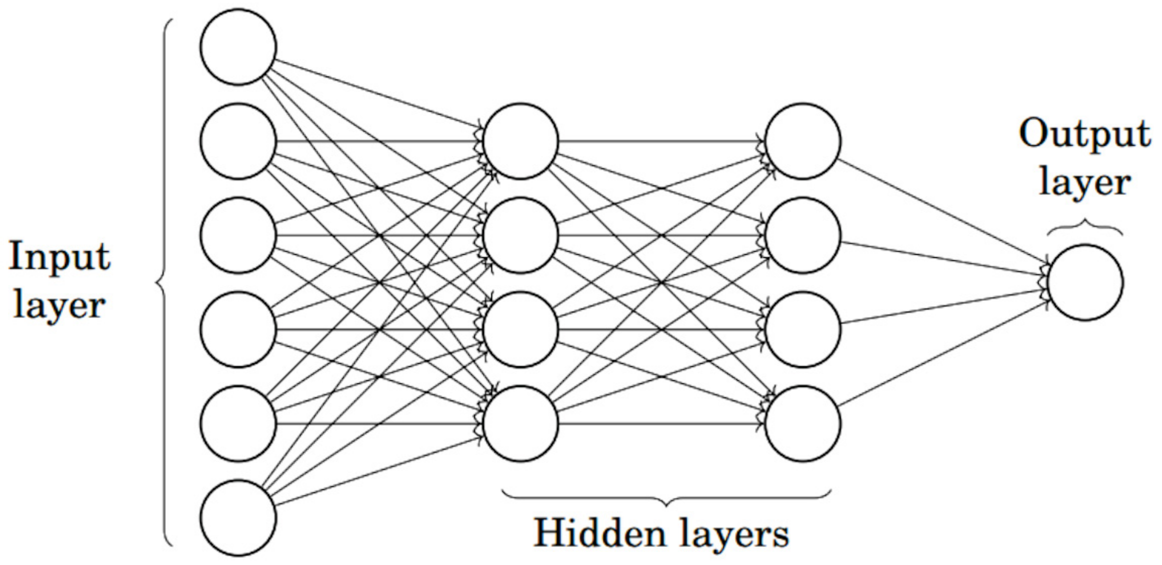


Figura 2.4: Example of MLP structure with the input layer, two hidden layers composed by four neurons, and the output layer composed of one neuron. Source: Pavlovsky (2018).

The MLP is usually trained by an algorithm called back-propagation, proposed by E. Rumelhart et al. (1986). In the back-propagation, a sample goes through all the layers in the MLP and receives a classification in the output layer. The MLP is a supervised learning algorithm, and the true labels are utilized to calculate the classification error.

The error is utilized to update the weights of the output layer. From the new weights of the output layer, an error value is calculated and utilized to update the weights of the back layer. The same is performed to all layers inside the MLP. When the weights of all layers are updated, a new sample from the training set enters the MLP and the training continues until the error reaches a minimum value.

The MLP is a classifier algorithm, and human interaction is necessary to specify the object features to be classified. The next paragraphs explain the CNN, a class of Neural Network with similar concepts of the MLP to learn the best features to extract from the objects without the need for human interaction.

2.3 DEEP LEARNING

Deep Learning is a subfield of Machine Learning based on Artificial Neural Networks to solve complex computer learning problems. Besides the high capacity of Deep Learning in solving classification problems, only in recent years that it was possible to deeply explore Deep Learning algorithms. The advance of powerful and cheapest Graphics Processing Units (GPUs), allied with the advance of the parallel programming, enabled new promising researches in deeper Artificial Neural Networks.

2.3.1 Convolutional Neural Network (CNN)

The CNN is a deep artificial neural network developed for classification of images and object recognition. The first CNN was proposed by Fukushima (1980) for visual pattern recognition and Lecun et al. (1998) successfully applied a CNN in the image pattern recognition problem, proposing an architecture called LeNet to classify handwritten numbers.

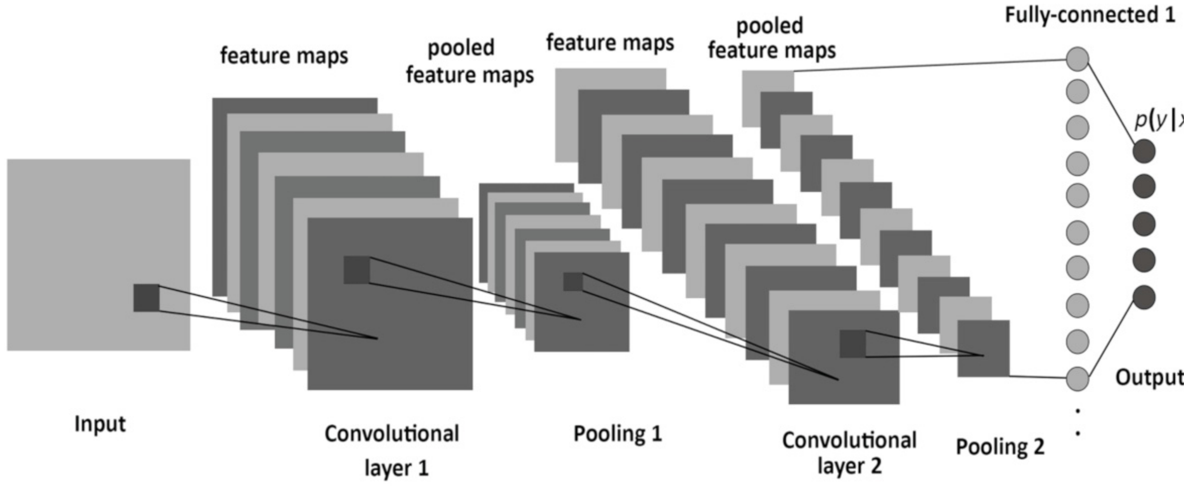


Figura 2.5: Example of generic CNN architecture composed of two convolutional layers and two polling layers. In the end, the network has two fully connected layers. Source: Albelwi and Mahmood (2017).

However, only with AlexNet of Krizhevsky et al. (2012), winner of the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) 2012, the CNNs started to be widely applied in other classification problems. The most significant contribution of Krizhevsky et al. (2012) was to perform the training step of the architecture in GPUs.

Although the impressive learning capacity of CNNs, the training step is highly time-consuming. CNNs has millions of weights to learn in the training step, which takes a significant amount of time to adjust all weights.

However, performing the training step on GPUs opened the possibility of training more complex and more extensive CNNs. Inspired by Krizhevsky et al. (2012), new approaches and architectures [Simonyan and Zisserman (2014); Szegedy et al. (2015); He et al. (2016); Girshick et al. (2013); Girshick (2015); Ren et al. (2017)] was proposed to classification problems and its derivatives.

Figure 2.5 illustrates a generic CNN architecture and its three types of layers: Convolutional, Pooling, and Fully Connected. The input of a CNN model is generally a sequence of images. The images go through a sequence of convolutional and pooling layers to learn features and generate the feature maps.

Pooling layers are applied in the middle of the convolutions to reduce the image dimensionality by filtering irrelevant data. In the end, a sequence of fully connected layers is linked to the network to perform the classification of the features learned in the convolutional layers.

The output of a standard CNN is a 1D vector, where each position in the vector stores a probability of the object being of a determined category. The following paragraphs explain each type of layer in a CNN model.

2.3.1.1 *Convolutional Layer*

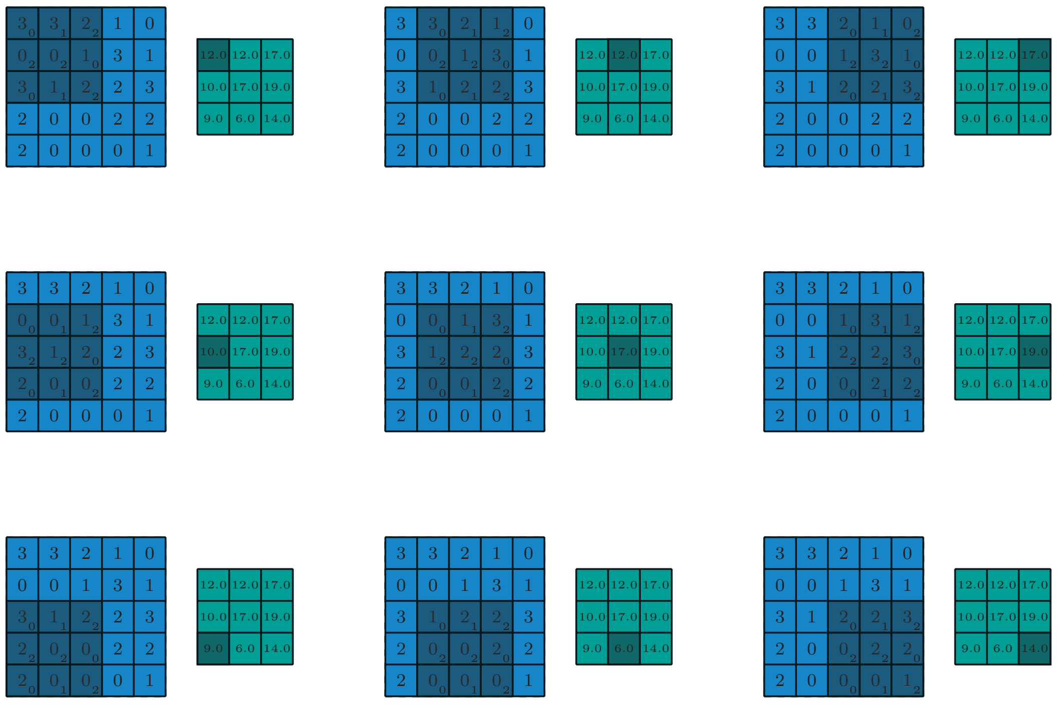


Figura 2.6: Example of a convolution operation. The input image (in light blue) has a size of 5×5 pixels. The kernel (in dark blue) positioned on top of the image has a size of 3×3 pixels and contains the weights learned in the training step. The resulting image (in green) has a size of 3×3 pixels and represents the information learned from the input image. Source: Dumoulin and Visin (2016).

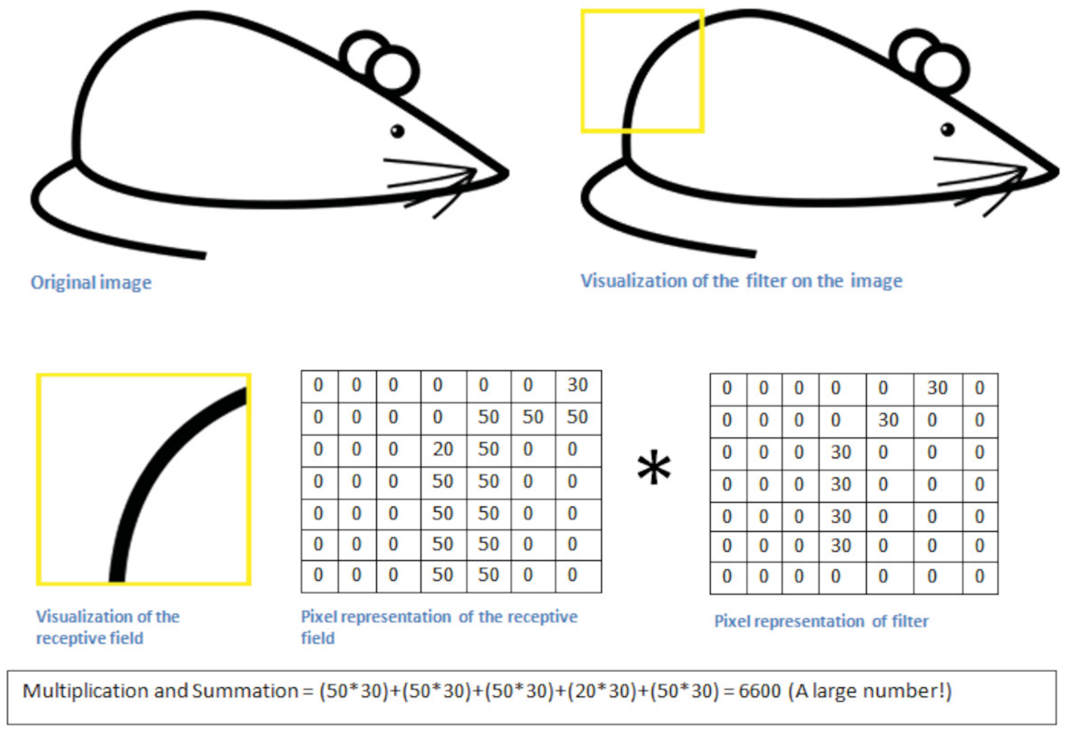
The function of a convolutional layer is to learn features to represent the objects. In the convolutional layer, a filter called kernel or neuron slides through the image. For each location in the image, called receptive field, an elementwise multiplication is performed between the pixels in the image and the weights in the filter [Altenberger and Lenz (2018)].

All multiplications are added together, generating a single value. In general, a CNN model has a sequence of convolutional layers, with the top layers learning more coarse information from the image and the bottom layers learning more refined features [Altenberger and Lenz (2018)].

Figure 2.6 illustrates the convolution process. The input image (in light blue) has a size of 5×5 pixels. The kernel (in dark blue) positioned on top of the image has a size of 3×3 pixels and contains the weights learned in the training step. The resulting image (in green) has a size of 3×3 pixels and represents the information learned from the input image.

Figure 2.7 presents the feature extraction process in a convolutional layer. The image presents an example of a filter designed to extract curves. If the object in the image has a curve similar to the curve in the filter, the output value of the convolution is high. Otherwise, if the object does not present a curve similar to the one in the filter, the output value of the convolution is low [Altenberger and Lenz (2018)].

The output of a convolution operation represents the filter activation, and the feature map contains the image regions with similar curves to the filter. Each convolution layer has a set of filters to represent different image shapes. The weights of each filter are randomly initialized and learned in the training step to best represent the objects being classified [Altenberger and Lenz (2018)].


Original image
Visualization of the filter on the image
Visualization of the receptive field
Pixel representation of the receptive field
*
Pixel representation of filter

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

Figura 2.7: Example of a filter designed to extract curves. If the object presents a curve similar to the one in the filter, the output value of the convolution is high. Otherwise, if the object does not present a curve similar to the one in the filter, the output value of the convolution is low. Source: Deshpande (2018).

Following the convolutional layer, the CNN usually has a Rectified Linear Units (ReLu) layer, an activation layer linked to introducing non-linearity to the convolution operation which is composed only by linear operations (elementwise multiplications and summations). The non-linearity improve the learning capacity of the CNN by generating more complex decision boundaries, i.e., the ReLu layers help the CNN to learn more complex decision functions [Nair and Hinton (2010)]. Usually, nonlinear functions like tahn and sigmoid were applied in CNNs. However, ReLu layers enabled faster training of deep CNNs [Krizhevsky et al. (2012)]. The ReLu layer applies the function $f(x) = max(0, x)$ for each pixel, transforming all negative values to zero [Arora et al. (2017)].

The ReLu layer also helps to reduce the vanishing gradient problem, which occurs in learning models based on gradient-based optimization techniques (e.g., back-propagation) [Glorot et al. (2011)]. In the back-propagation, the top layers are harder to train when compared with the bottom layers. The calculated error tends to be smaller as long as the backpropagation moves backward in the network, and the weights in the top layers are learned slowly. In some cases, if the error is too small, the weights in the top layers may not be updated [Altenberger and Lenz (2018)].

2.3.1.2 Pooling Layer

The pooling layer, or down-sampling layer, is applied to reduce the dimensionality of the feature maps in a way to save the most relevant information from the feature maps. In a pooling layer, the number of weights to be learned are reduced, which helps to speed up the model and to control the overfitting.

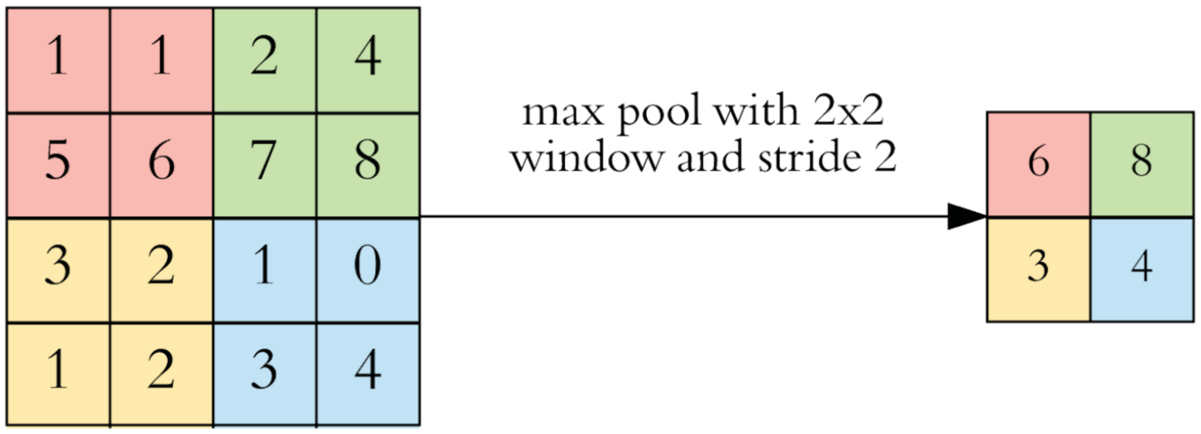


Figura 2.8: Example of max-polling with a filter of size 2×2 pixels and stride 2. The original feature map, with a size of 4×4 pixels, is converted in a feature map of size 2×2 pixels. Source: Dertat (2018).

The overfitting is a problem of learning-based algorithms and occurs when the model is so skilled in the training data that loses the generalization capacity, not being able to classify new examples [Srivastava et al. (2014)]. The dropout layer is another type of layer added in the model during the training step to reduce the overfitting. In a dropout layer, a random set of activation neurons receives zero, forcing the CNN to learn the same concept with different neurons [Srivastava et al. (2014)].

In the pooling layer, a filter with no weights slides through the image applying the pooling operation (maximum, minimum, mean, or sum). In most of the works proposed in the literature, the maximum pooling, also called max-pool, is applied. Figure 2.8 presents an example of max-polling with a filter of a size 2×2 pixels and stride 2. The original feature map, with a size of 4×4 pixels, is converted in a feature map with a size of 2×2 pixels.

2.3.1.3 Fully Connected Layer

The fully connected layer is a MLP with an activation function attached at the end to perform the classification. The fully connected layer works as follows: the input layer takes the feature vector generated by the convolutional layers and passes through a sequence of hidden layers. In the output layer, the neurons usually apply the softmax activation function to convert the output values into probabilities (values between 0 and 1).

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_i}} \tag{2.2}$$

The softmax layer receives a vector "y"output of the last fully connected layer and applies the softmax function, presented in Equation 2.2, for each element "y_i"inside the vector "y". The output of the softmax is a vector of same size of "y"with each position in the vector containing the probability of the object being classified be of that class [Bishop (2006)].

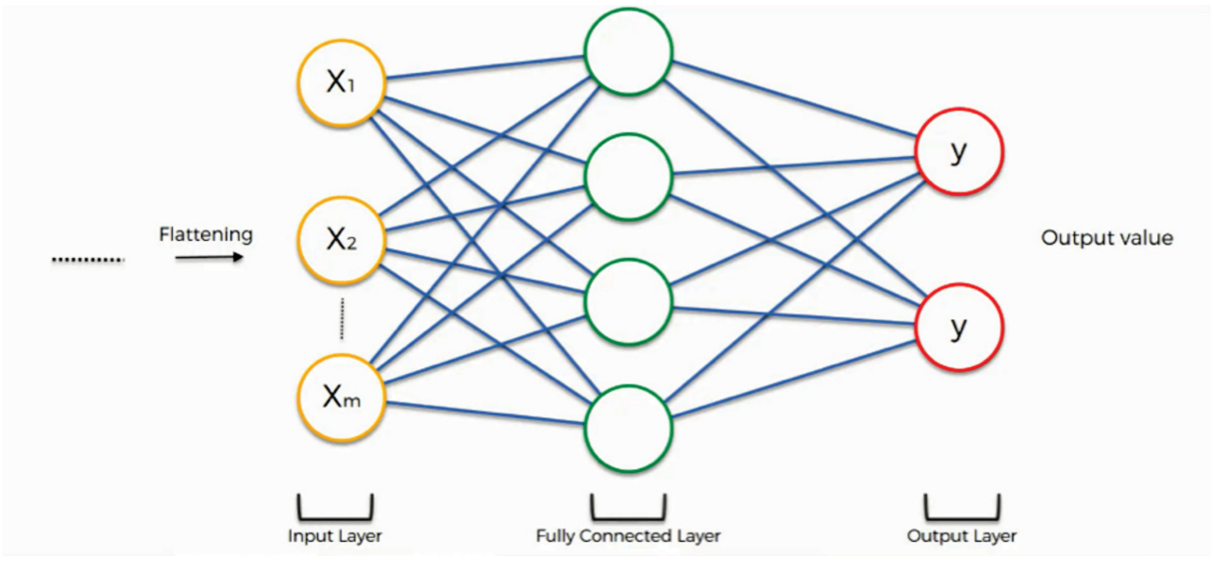


Figura 2.9: The flattening operation converts the feature map generated by the convolutional layers in a feature vector. The feature vector enters in the input layer (represented in yellow). Following the input layer, there is a sequence of fully connected layers (represented in green). The output layer (represented in red) generates the classification results. Source: Team (2019).

Figure 2.9 presents the structure of the fully connected layers in a CNN. The flattening operation converts the feature map generated by the convolutional layers in a feature vector, and the feature vector enters in the input layer (represented in yellow). Following the input layer, there is a sequence of fully connected layers, also called hidden layers in the MLP (represented in green). The output layer (represented in red) generates the classification results.

2.3.1.4 Training a Convolutional Neural Network

Although the impressive results achieved by CNNs solving various classification problems [Altenberger and Lenz (2018)], training the network can be an obstacle. The CNN is trained through the back-propagation algorithm in the same way as the MLP and has millions of weights to learn during the training step. The CNNs also requires a large amount of data to be able to adjust all the network weights. Both factors corroborate to increase the training time. A technique called transfer learning is usually utilized to overcome the training time limitation.

The transfer learning reuses the weights of a CNN, already trained to other problems, to solve a more specific problem. The training step begins with the weights already learned and performs a fine-tuning in the weights of the bottom layers of the CNN [Yosinski et al. (2014)]. In a CNN model, the top layers extract more generic features like edges and curves while the last layers extract more refined features related to the objects in the problem addressed [Yosinski et al. (2014)].

Usually, the models are pre-trained in extensive datasets like the ImageNet [Deng et al. (2009)] and COCO [Lin et al. (2014)], which counts with thousands of images divided into several classes, which reduces the amount of data needed to train a CNN model. After a model being trained in a large dataset, a small dataset is enough to specify the model to another problem [Yosinski et al. (2014)].

2.3.2 CNNs architectures and backbones

In order to evaluate and compare the FCN and the Mask R-CNN architectures in this work, three CNNs models were used as backbones of the FCNs and Mask R-CNNs (i.e., three CNNs models were adapted as FCNs and Mask R-CNNs): the Residual Network (ResNet)-50 [He et al. (2016)], the ResNet-101 [He et al. (2016)] and the VGG-16 [Simonyan and Zisserman (2014)].

2.3.2.1 ResNet

The ResNet architecture, proposed by He et al. (2016), was designed with skip connections to prevent the vanishing gradient problem, allowing the network to be composed with a higher number of layers. The skip connections take the input of an "nth"layer and add to the output of an "(n+m)th"layer, where "m"is the number of skipped layers.

Figure 2.10 presents an example of skipping connection. In the example, the network has two convolutional layers (weight layers). The input of the first convolutional layer ("x") is skipped and added to the output of the second convolutional layer ("$F(x)$"). The sequence of convolutional layers connected by a skip connection is called block, with the ResNet structured by a sequence of blocks.

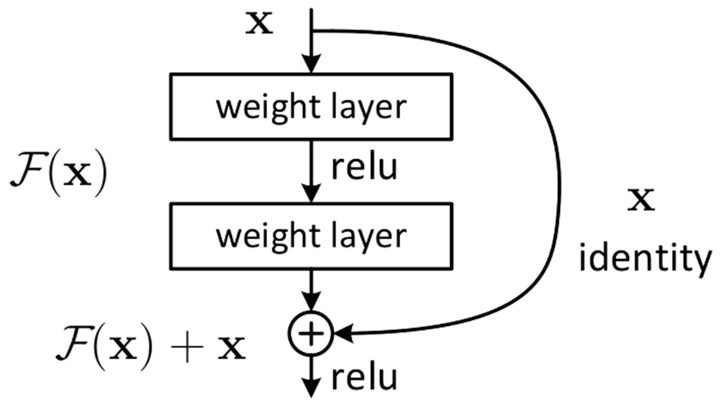
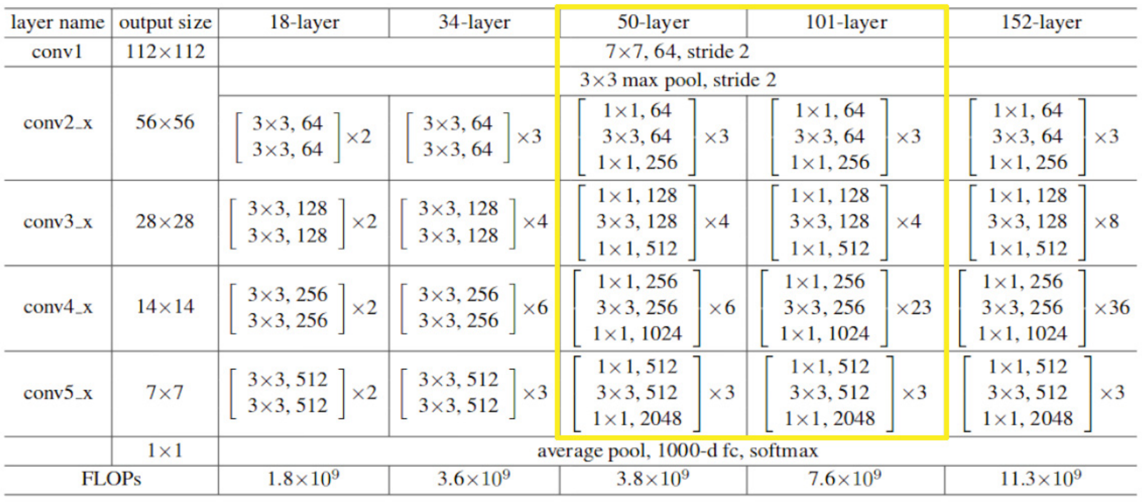

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figura 2.10: Example of a ResNet block with two stacked layers and a skip connection from the input of the first layer ("x") to the output of the last layer ("$F(x)$"). Source: He et al. (2016).

Figure 2.11 presents a table with all versions of the ResNet, with each version structured with a different number of layers: 18, 34, 50, 101, and 152. In this work, it was utilized the ResNets with 50 and 101 layers, highlighted in yellow in Figure 2.11.



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times 2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times 3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times 2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times 8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times 2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times 23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times 36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times 2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figura 2.11: Differences between the five ResNets models, with 18,34,50,101 and 152 layers. The ResNets-50 and 101 in the yellow square were utilized in the present work as FCN and Mask R-CNN backbones. Source: He et al. (2016).

In a ResNet, the input image of size $224 \times 224 \times 3$ enters in a convolutional layer with a kernel of size 7×7, stride 2 and 64 filters, generating a feature map of size $112 \times 112 \times 64$. A max-pooling layer with kernel size 3×3 and stride 2 is applied to reduce the dimensionality to $54 \times 54 \times 64$.

A sequence of four building blocks is attached in the network, with each building block composed by a sequence of blocks with skip connections. In the ResNets 50 and 101, three stacked convolutional layers form these blocks with a kernel of size 1×1, 3×3 and 1×1, respectively. The first convolutional layer of the building block has stride 2 while the other convolutional layers have stride 1.

The first building block has three blocks inside, the second building block has four blocks, the third building block has 6 blocks in the ResNet-50 and 23 blocks in the ResNet-101, and the last building block has three blocks. The number of filters in each layer is presented in Figure 2.11.

An average pooling is attached at the end of the last building block to reduce the feature map dimension, generating a feature vector of size $1 \times 1 \times 2048$, which enters in a fully connected layer. The last ResNet layer is a softmax layer to perform classification.

2.3.2.2 VGG-16

The VGG-16 architecture, proposed by Simonyan and Zisserman (2014), was designed with a stack of convolutional layers and small filters sizes (3×3) instead of one convolutional layer with a large filter size (7×7 or 11×11), a common strategy proposed in other CNNs models [Lecun et al. (1998); Krizhevsky et al. (2012)]. A stack of convolutional layers with small filters sizes generates a decision function more discriminatory by increasing the number of non-linear rectification layers (ReLus).

The VGG-16 approach also reduces the number of parameters in the architecture without decreasing the amount of effective receptive fields (two stacked convolutional layers with kernel of size 3×3 have an effective receptive field of 5×5, tree convolutional layers with kernel of size 3×3 have an effective receptive field of 7×7).

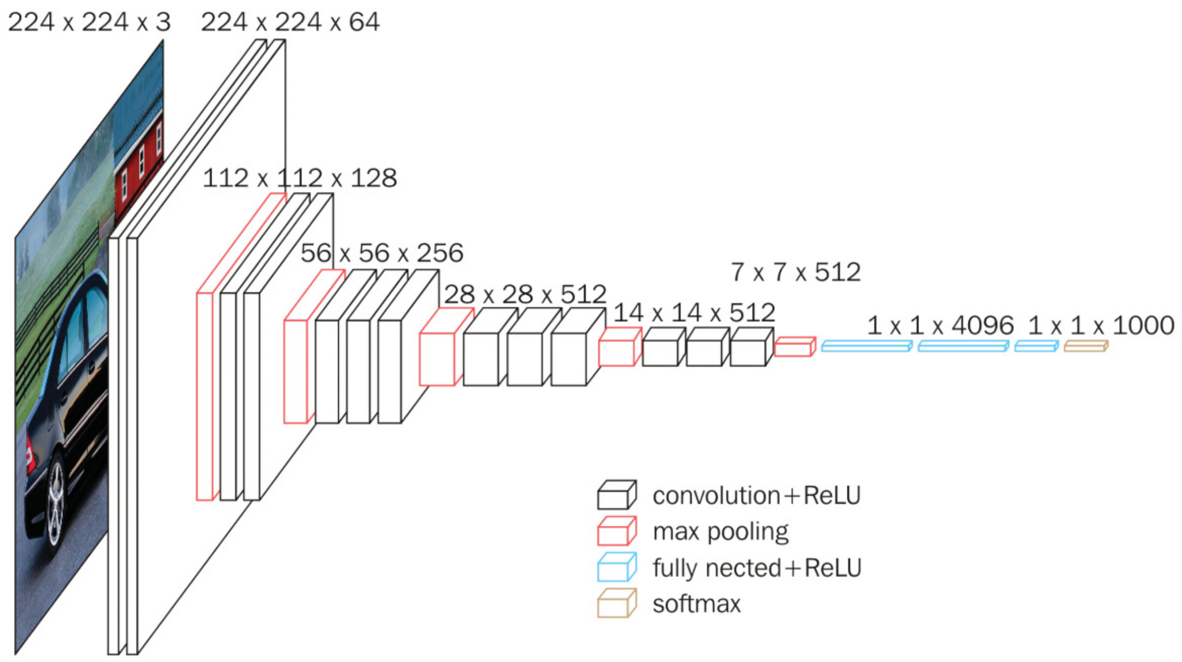


Figura 2.12: The architecture of the VGG-16 network. The black boxes are the convolutional layers and the red blocks are the max-polling layers. In the end, the fully connected layers are presented in blue color and the softmax layer in brown. Source: Saif Ahmed (2017).

The VGG-16 architecture receives an input image of size 224×224 and 3 channels. The image goes through five blocks composed by a sequence of convolutional layers with a max-polling layer at the end of the block. The first two blocks of the VGG-16 have two convolutional layers in sequence, and the remaining blocks have three convolutional layers.

The last block of the VGG-16 has three fully connected layers in sequence, with the first two fully connected layers composed by 4096 neurons and the final fully connected layer formed by 1000 neurons. Figure 2.12 illustrates the VGG-16 architecture and presents all dimensions and numbers of filters in each layer.

Each convolutional layer in the VGG-16 has padding 1 to maintain the dimension value of the previews layers and a filter of a size of 3×3. The max-pooling layers have stride 2, and the convolutional layers have stride 1. A non-linear rectification layer (ReLu) is attached in all hidden layers, and the last layer of the network is a softmax to convert the output vector of the last fully convolutional layer in probabilities.

2.3.3 Fully Convolutional Network (FCN)

The FCN was designed by Long et al. (2015) to perform Semantic Segmentation tasks. Long et al. (2015) converted three CNNs models, AlexNet [Krizhevsky et al. (2012)], GoogleLeNet [Szegedy et al. (2015)], and VGG-16 [Simonyan and Zisserman (2014)] in FCNs by removing the fully connected layers and replacing them to convolutional layers.

Figure 2.13 illustrates the "convolutionalization"[1] process, with a fully connected layer being replaced by a convolutional layer, generating a heatmap with dense prediction values as output, instead of a 1D vector of probabilities. Then, Long et al. (2015) proposed the use of the

[1]Expression utilized by Long et al. (2015) to name the process of replacing the fully connected layers to convolutional layers.

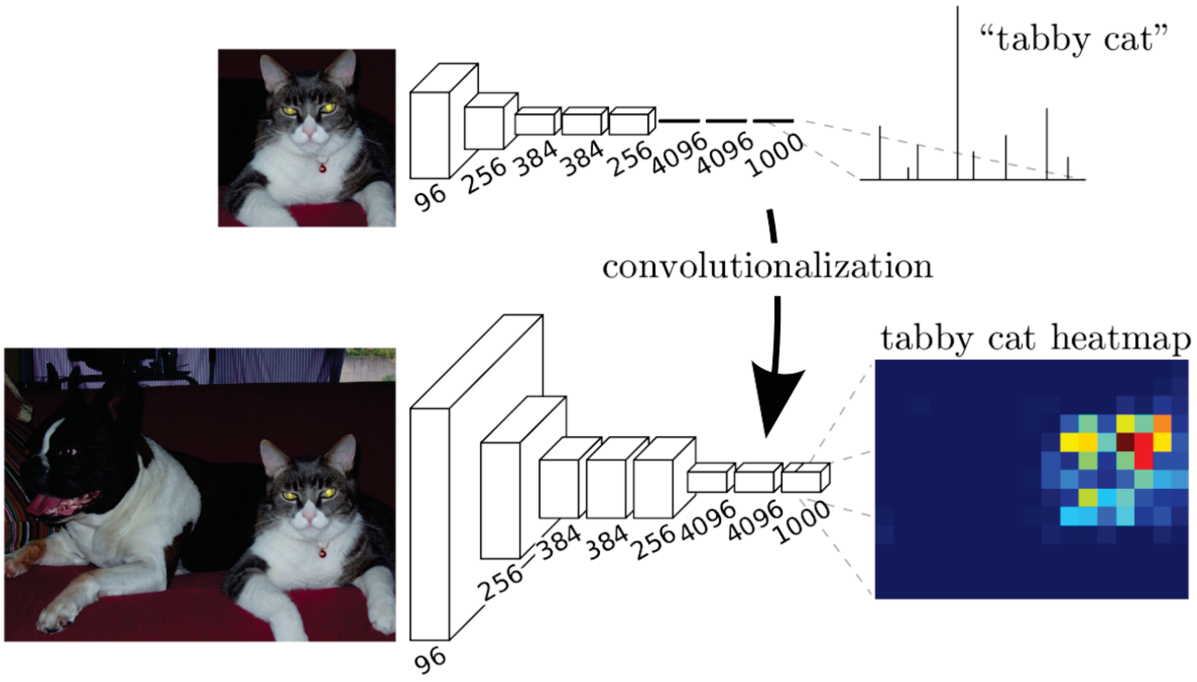


Figura 2.13: Example of the "Convolutionalization"process applied in a CNN model to convert the fully connected layers in convolution layers, generating a heatmap with dense prediction values as output, instead of a $1 \times N$ dimensional vector of probabilities. Source: Long et al. (2015).

bilinear interpolation [Kirkland (2010)], applied in the heatmap, to generate a segmentation mask based on the probabilities of the heatmap. This step is called upsampling.

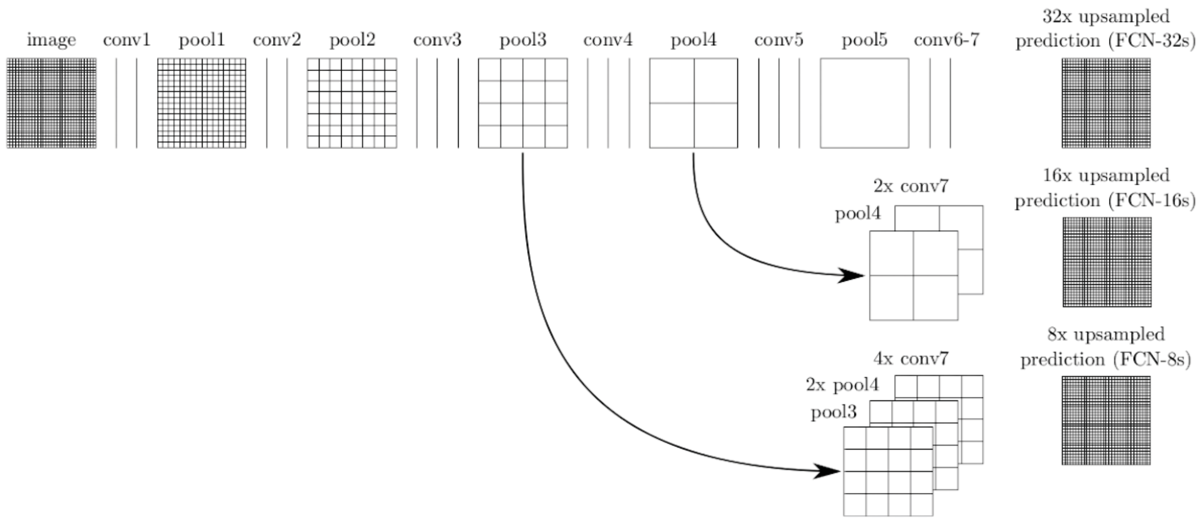


image
conv1
pool1
conv2
pool2
conv3
pool3
conv4
pool4
conv5
pool5
conv6-7
32x upsampled prediction (FCN-32s)
2x conv7
pool4
16x upsampled prediction (FCN-16s)
4x conv7
2x pool4
pool3
8x upsampled prediction (FCN-8s)

Figura 2.14: Differences between FCNs versions proposed by Long et al. (2015). The FCN-32 uses the seventh convolutional layer to generate the heatmap. The FCN-16 adds the fourth polling layer to the seventh convolutional layer to generate the heatmap. The FCN-8 adds the fourth and third polling layers to the seventh convolutional layer to generate the heatmap. Source: Long et al. (2015).

Long et al. (2015) proposed three versions of the FCN, as presented in Figure 2.14. In the first version, called FCN-32, the fully connected layers are converted in convolutional layers. In this version, the heatmap in the seventh convolutional layer (conv-7) suffers an upsampling with a stride of 32 (the heatmap is 1/32 times smaller than the input image) to generate the segmentation mask.

In a second version, called FCN-16, the seventh convolutional layer (conv-7) suffers an upsampling with a stride of 2 and is added to the fourth polling layer (pool-4). An upsampling with a stride of 16 (the pool-4 is 1/16 times smaller than the input image) is applied to generate the segmentation mask.

In the last version, called FCN-8, the seventh convolutional layer (conv-7) suffers an upsampling with a stride of 4 and the fourth polling layer (pool-4) suffers an upsampling with a stride of 2. Both conv-7 and pool-4 are added to the third pooling layer (pool-3). An upsampling with a stride of 8 (the pool-3 is 1/8 times smaller than the input image) is applied to generate the segmentation mask.

In order to generate a segmentation mask with the same size of the input image, a bilinear interpolation is applied in the heatmap. Figure 2.15 presents an example of bilinear interpolation in point "P". Assuming that the points "Q_{11}","Q_{12}","Q_{21}"and "Q_{22}"in the figure are known, the first step interpolates the points in the x-direction "R_1"(Equation 2.3) and "R_2"(Equation 2.4). The points "R_1"and "R_2"are used to interpolate the point P in the y-direction (Equation 2.5).

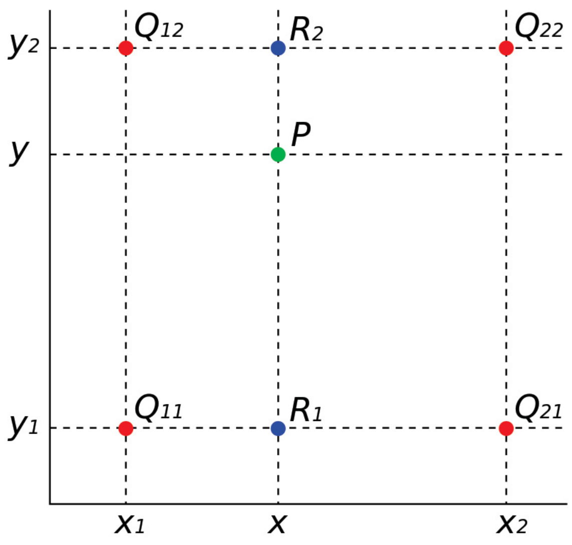


Figura 2.15: Example of bilinear interpolation. Assuming the points Q_{11}, Q_{12}, Q_{21}, and Q_{22} are known, the first step interpolates the points in the x-direction R_1 (Equation 2.3) and R_2 (Equation 2.4). The R_1 and R_2 are used to interpolate the point P in the y-direction (Equation 2.5). Source: Wikipedia (2017).

$$f(R_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \tag{2.3}$$

$$f(R_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \tag{2.4}$$

$$f(P) \approx \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2) \tag{2.5}$$

In a FCN-32, the heatmap is 1/32 times smaller than the input image. In the interpolation with stride 32, a stride of 32 positions is generated between each point in the heatmap, generating a matrix of the same size of the input image. In this new matrix, the only known points are that in the heatmap. So, for every four points, "Q_{11}", "Q_{12}", "Q_{21}", and "Q_{22}", in this new matrix, a point "P"will be interpolated for each combination of the 32 "x"and "y"between "Q_{11}", "Q_{12}", "Q_{21}", and "Q_{22}". When all points in this new matrix are generated for each set of "Q_{11}", "Q_{12}", "Q_{21}", and "Q_{22}", this new matrix will be the segmentation mask generated by the FCN.

Among the FCNs models proposed by Long et al. (2015), the FCN-8 achieved better segmentation results. In the convolutional step, information from the image is lost as the image goes deeper into the network due to the pooling layers. After the classification, the heatmap knows the object position in the image. However, due to the loss of information, it is hard to rebuild the image. The FCN-16 and FCN-8 try to recover information from the top layers of the network to help in the upsampling step. The FCNs evaluated on the present work are based on the FCN-8 strategy.

2.3.4 Mask Regional Convolutional Neural Network (Mask R-CNN)

The Mask R-CNN, developed by He et al. (2017) to Instance Segmentation tasks, is composed by two sub-networks: the Faster Region-based Convolutional Neural Network (Faster R-CNN) to perform object detection and classification, and a FCN to perform object segmentation.

The Mask R-CNN is structured as follows: in the Faster R-CNN, the first block of the network is a sequence of convolutional and pooling layers to generate a feature map. In Ren et al. (2017), the convolutional structure of the ZF [Zeiler and Fergus (2014)] and the VGG-16 [Simonyan and Zisserman (2014)] models are utilized as the base of the Faster R-CNN.

Following the convolutional block, a module called Region Proposal Network (RPN) is linked in the network to generate a set of proposal regions with high probability of being objects. A sliding window is applied in the feature map to generate a set of "k"regions.

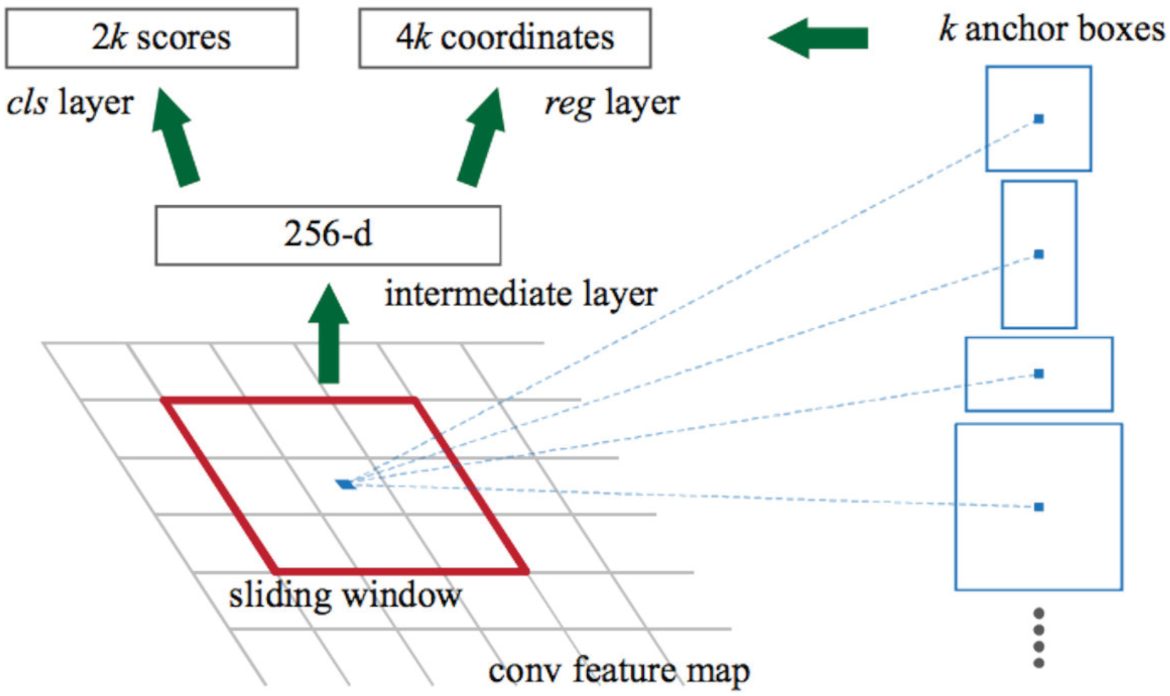


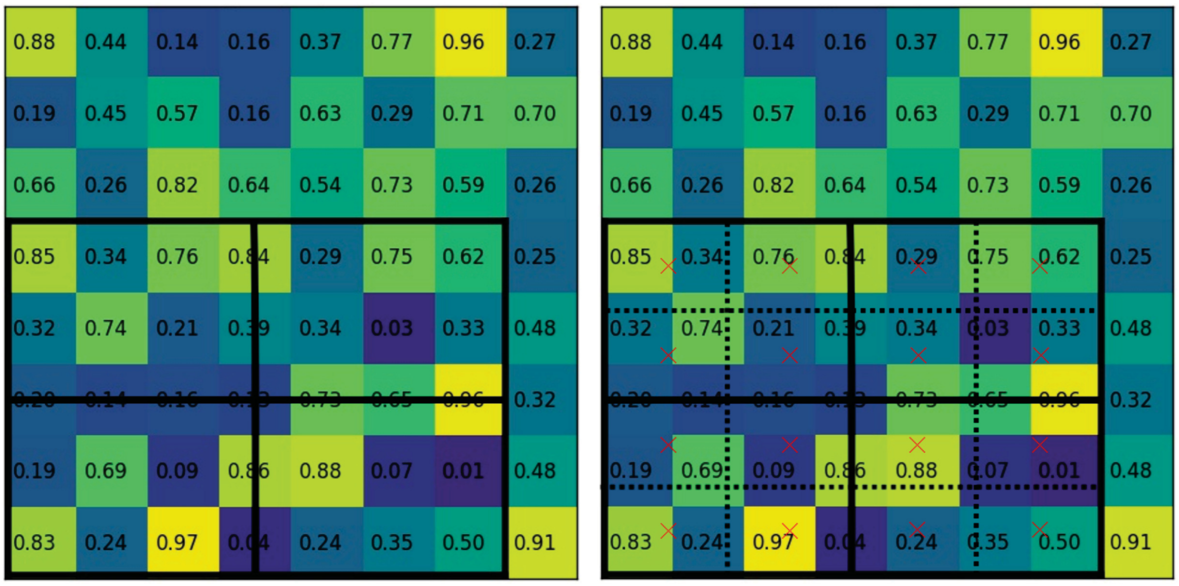
Figura 2.16: RPN architecture. A sliding window (the red square) is applied in the feature map. For each location in the feature map, a set of anchors is generated (the blue squares). The anchors enter in a convolutional layer, followed by two parallel convolutional layers. The first one ("cls") outputs 2k scores, and the second one ("reg") outputs 4k coordinates of bounding boxes of each region proposal. Source: Ren et al. (2017).

Each location (anchor) in the feature map enters the RPN to generate a bounding box (set of four points) and a classification (object or background). For each anchor classified as an object, a set of anchors boxes (calculated in absolute coordinates of the image) is generated with different sizes and aspect ratios.

In Ren et al. (2017), three different sizes (128, 256 and 512) and three different aspect ratios (1:1, 1:2 and 2:1) are used to generate k=9 anchors. Figure 2.16 illustrates the anchor box generation process. The sliding window is represented in the red square, and the proposed anchor boxes are represented in blue.

The RPN has three convolutional layers. The first layer (input layer), which receives the anchor boxes, and two output layers connected in parallel. The first output layer (called "cls") generates a vector of scores (2k scores) with the probabilities of the anchors being an object. The second output layer (called "reg") generates a vector of coordinates (4k coordinates) with the bounding boxes of each proposal region in the form: X_{center}, Y_{center}, width, and height. Figure 2.16 illustrates the RPN model with the anchor boxes generation and the convolutional layers.

The regions with higher probability of being an object enter the Region of Interest Alignment (RoIAlign) layer. As the proposed regions have different sizes and aspect ratios , the RoIAlign operation is applied to generate a feature map with a fixed size through a pooling operation. In the RoIAlign operation, the proposal region inside the feature map is divided $K \times K$ sub-regions, where "K"is the width and height of the output feature map.



(a) The feature map with size 8×8 and the proposed region with size 7×5 (the black square) divided into 2×2 sub-regions with the same size.

(b) The sub-regions are divided into four regions (dotted lines). For each region, an interpolated value is generated (the red "x").

Figura 2.17: Example of the RoIAlign operation. Source: Sardana (2017).

The object region is divided into sub-regions with the same size, as shown in Figure 2.17a. These sub-regions are divided again into four regions, and a bilinear interpolation is applied to generate a value to each region, as shown in Figure 2.17b. The red "x"represents the interpolated values. The higher value for each region is taken to generate the feature map (the normal max pooling operation).

The proposal objects generated in the RPN enters in two fully connected layers to generate a feature vector. The feature vector enters in a softmax and linear regression modules,

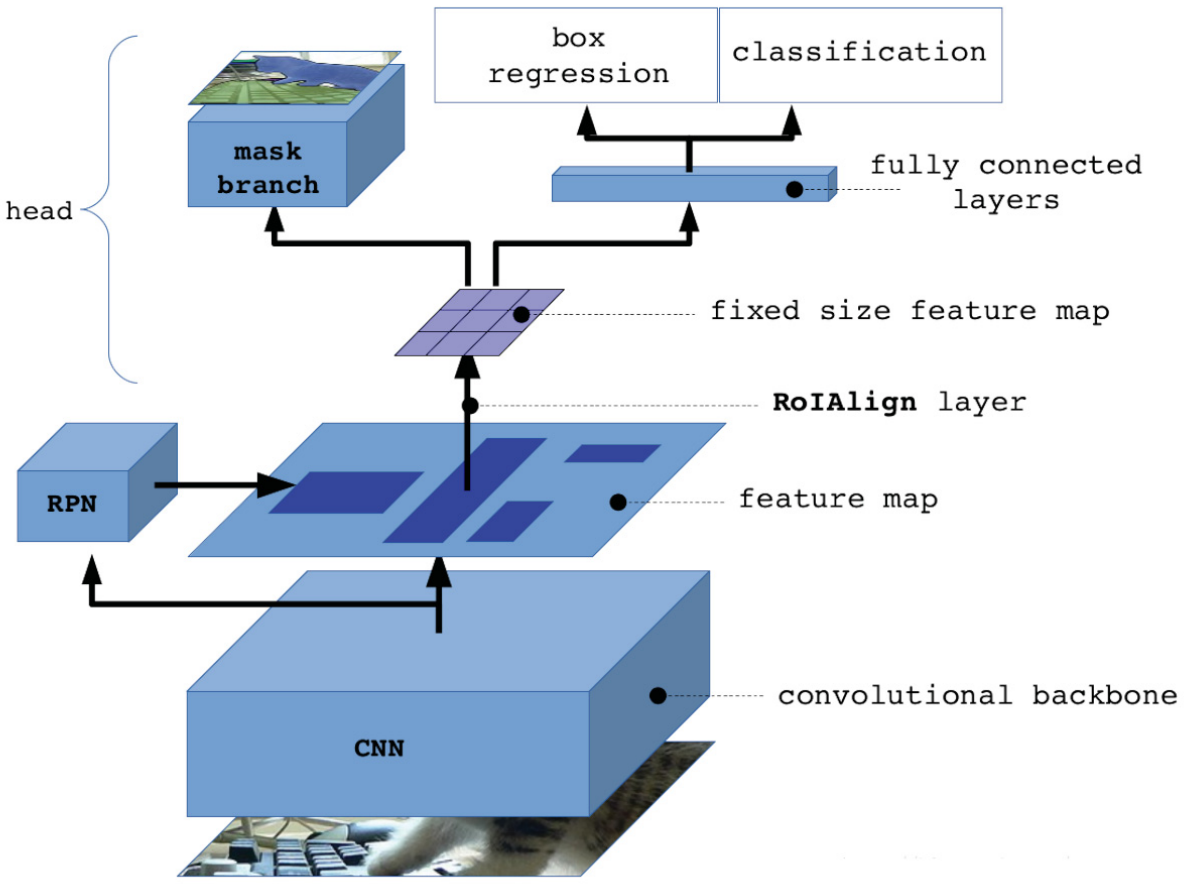


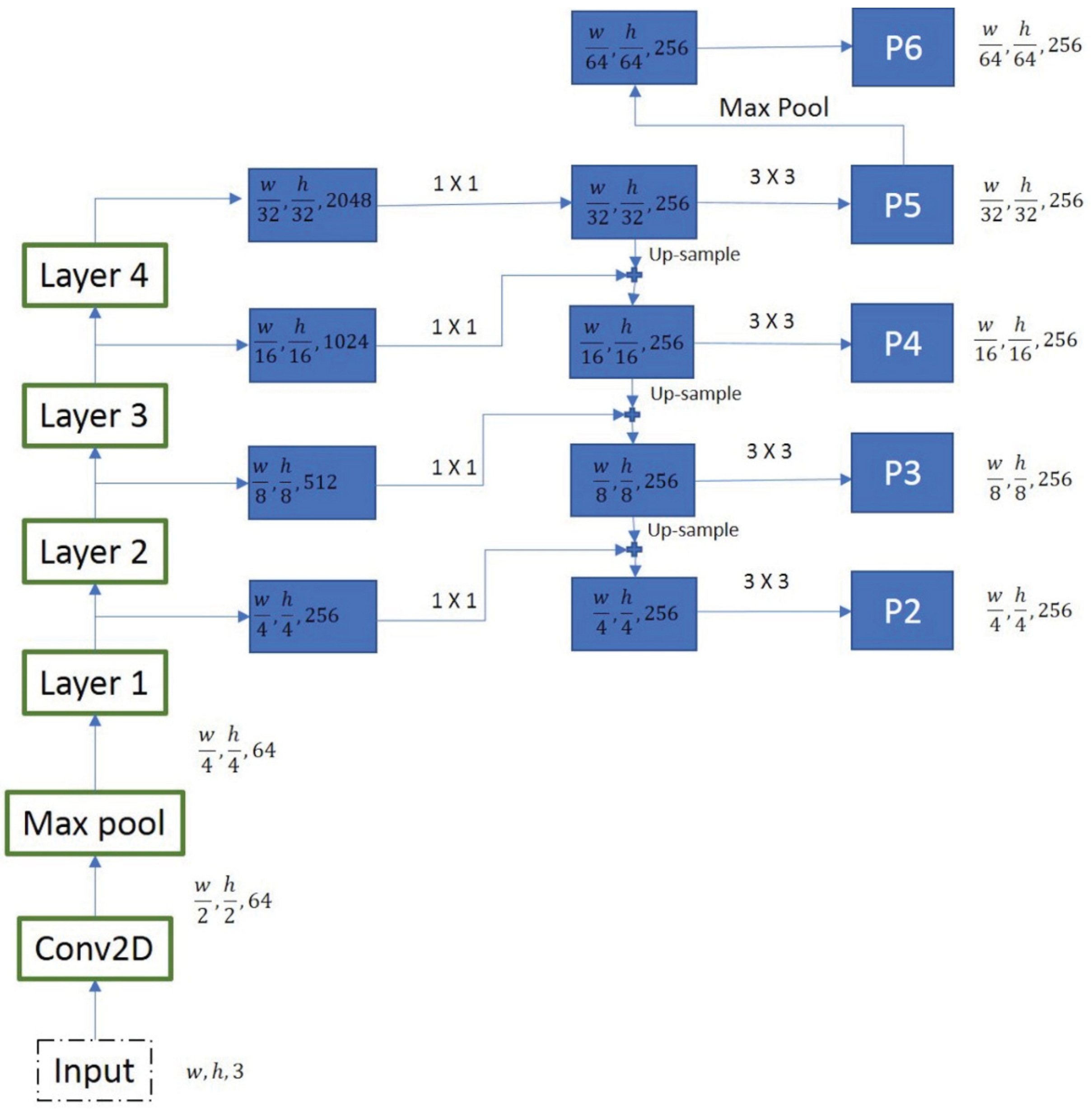
Figura 2.18: A generic Mask R-CNN architecture. The image enters in a convolutional backbone to generate a feature map. The RPN is utilized to generate a set of regions in the feature map with the high probability of being objects. The RoIAlign layer receives the regions and generates a feature map with a fixed size. The feature map with a fixed size enters in two fully connected layers. A softmax and linear regression modules are attached in parallel in the network to perform the object classification and to generate tighter bounding boxes, respectively. The output of the RPN also enters in a segmentation module to generate classification to each pixel in the image. Source: Vinograd (2017).

attached in parallel, to perform object classification and to generate tighter bounding boxes, respectively. The output of the RPN also enters in a segmentation module to generate classification to each pixel in the image. Figure 2.18 presents the complete Mask R-CNN model.

Another modification proposed by He et al. (2017) in the Mask R-CNN is the FPN. The FPN is utilized to include scale information in the Mask R-CNN results. As explained in Section 2.3.3, the image loses information in the convolutional step due to the pooling layers, which hinders the image reconstruction in the segmentation step.

Figure 2.19 presents an example of FPN structure. In the presented example, the network has four blocks of layers: Layer1, Layer2, Layer3, and Layer4, where each layer block is a sequence of convolutional layers. Each layer block divides the width and height of the feature map by two and doubles the number of filters (depth of the feature map).

In the first step of the FPN, the output of each layer block suffers a convolution with a filter of size 1×1 to generate feature maps with the same depth size. With all feature maps normalized, the output of the Layer 4 suffers an upsample operation and is added to the output of the Layer 3. The resulting feature map suffers an upsample and is added to the output of the Layer2. Finally, the resulting feature map suffers an upsample operation and is added to the output of the Layer 1.



$\frac{w}{64},\frac{h}{64},256$
P6
$\frac{w}{64},\frac{h}{64},256$
Max Pool
$\frac{w}{32},\frac{h}{32},2048$
1 X 1
$\frac{w}{32},\frac{h}{32},256$
3 X 3
P5
$\frac{w}{32},\frac{h}{32},256$
Layer 4
Up-sample
$\frac{w}{16},\frac{h}{16},1024$
1 X 1
$\frac{w}{16},\frac{h}{16},256$
3 X 3
P4
$\frac{w}{16},\frac{h}{16},256$
Layer 3
Up-sample
$\frac{w}{8},\frac{h}{8},512$
1 X 1
$\frac{w}{8},\frac{h}{8},256$
3 X 3
P3
$\frac{w}{8},\frac{h}{8},256$
Layer 2
Up-sample
$\frac{w}{4},\frac{h}{4},256$
1 X 1
$\frac{w}{4},\frac{h}{4},256$
3 X 3
P2
$\frac{w}{4},\frac{h}{4},256$
Layer 1
$\frac{w}{4},\frac{h}{4},64$
Max pool
$\frac{w}{2},\frac{h}{2},64$
Conv2D
Input
$w,h,3$

Figura 2.19: A generic Feature Pyramid Network (FPN) structure. Layer1, Layer2, Layer3, and Layer4 are layers blocks of a generic CNN. The output of each layer block suffers an upsample operation and is added to the output of the predecessor layer block, generating four feature maps with different scales. Each feature map suffers a convolution with a kernel 3×3 generating the feature maps P2, P3, P4, and P5. The feature map P5 suffers a max-pooling operation generating the feature map P6. Source: Vinograd (2017).

Each feature map suffers a convolution with a kernel 3×3 generating the final feature map to each layer block ("P2", "P3", "P4", and "P5"in the Figure 2.19). The last feature map ("P5") suffers a max-pooling operation generating a last feature map "P6". The five feature maps enter in the RPN to generate regions proposals of different scales, which helps the next steps of the Mask R-CNN to improve the bounding boxes and segmentation masks results by adding multi-scale information.

2.4 CONCLUSION

The present chapter explained some fundamental concepts and algorithms explored in this work. The chapter explained the CNN, a robust Deep Learning algorithm based on neurons and widely applied in many classification problems. Before the CNNs, the methods utilized to extract features from the images were based on hand-crafted features and were not able to learn the best features to extract from a specific problem. These methods lacked from generalization capacity, not performing well in challenging scenarios. The CNNs has a set of filters that learns the best features to extract, making it easy for the classifier to distinguish the objects in the different categories of the problem being solved. The chapter also presented the FCN and the Mask R-CNN, that are improvements of the standard CNN to perform Semantic Segmentation and Instance Segmentation. The next chapter reviews the main approaches proposed to find saliency regions in images and summarizes relevant studies in the SOD literature.

3 RELATED WORKS

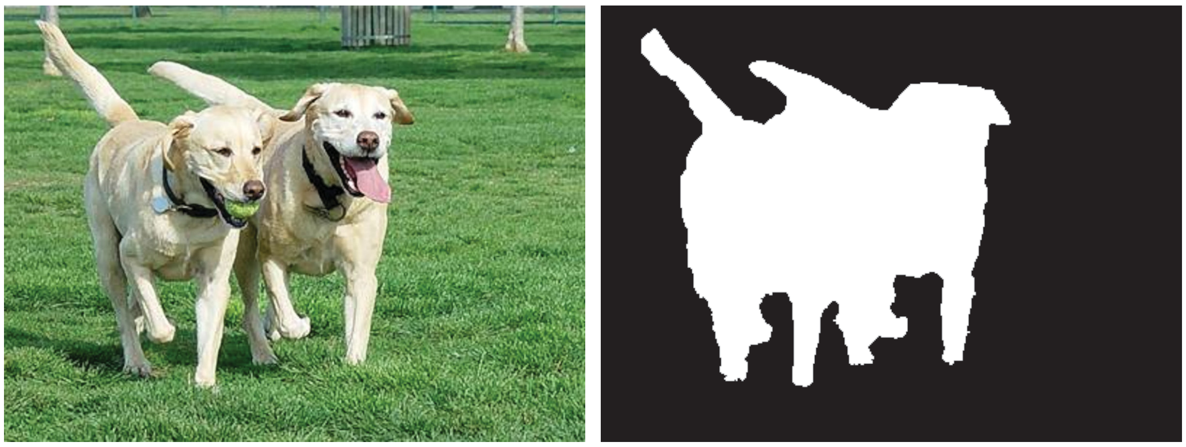
The present chapter details the Salient Object Detection (SOD) problem, reviews the main approaches proposed to find saliency regions in images and summarizes relevant studies in the SOD literature. The subjects covered in this chapter are divided into sections as follows: Section 3.1 presents the SOD problem and the literature approaches. Section 3.2 reviews relevant methods based on hand-crafted features, Section 3.3 addresses methods based on Convolutional Neural Networks (CNNs) [Lecun et al. (1998)], and Section 3.4 targets methods based on Fully Convolutional Networks (FCNs) [Long et al. (2015)]. Finally, Section 3.5 discusses the mentioned approaches.

3.1 SALIENT OBJECT DETECTION

The SOD problem, a subfield of Computer Vision, aims to find Visual Saliency regions in images [Borji et al. (2014)]. It has a wide range of applications in Computer Vision and Image Processing. Recognizing and tracking objects, image cropping and resizing, video and image compression, and summarization are some examples [Lee et al. (2016); Xi et al. (2017)]. In robotics, it can be applied in algorithms to locate robots in unknown environments [Todt (2005); Todt and Torras (2007)].

Achanta et al. (2008) and Zheng et al. (2010) defined the SOD problem as a binary segmentation problem, i.e., pixels in salient regions receive value 255 (white) and pixels in background regions receive value 0 (black). Following the proposed definition, the approaches in the SOD literature are divided into two steps: detection and segmentation.

The detection aims to find the salient objects, and the segmentation aims to generate a segmentation mask containing the salient objects, a classic segmentation problem [Borji et al. (2014)].


(a) Example of a scene containing salient objects (the dogs) in the foreground and non-salient objects in the background (the grass).

(b) Example of a ground-truth mask, with the pixels corresponding to the salient objects (the dogs) in white and the pixels corresponding to the background (the grass) in black.

Figura 3.1: Example of a scene containing salient objects and the expected Saliency Map. Source: Cheng et al. (2015).

Figure 3.1 illustrates a scene containing salient objects and the expected segmentation mask. Figure 3.1a can be divided into two main elements: the dogs in the foreground (salient objects) and the grass in the background (non-salient objects). Figure 3.1b presents the ground-truth mask, with the pixels corresponding to the salient objects (the dogs) in white and the pixels corresponding to the background (the grass) in black.

The approaches proposed in the SOD literature can be divided into three waves [Borji et al. (2014)]: hand-crafted-based, CNN-based, and FCN-based methods. The first attempts to find saliency regions in images were based on hand-crafted features, which addressed the problem through heuristics to find the saliency regions and standard Computer Vision techniques to perform the segmentation of the saliency regions.

Despite the promising results achieved by the hand-crafted-based methods, they lacked from generalization capacity, not performing well in challenging scenarios. To overcome such limitations, CNNs [Lecun et al. (1998)], an approach already applied in other Computer Vision problems, started to be widespread in the SOD literature.

Recent studies in the SOD literature started to explore FCNs [Long et al. (2015)], an improvement of the standard CNNs developed for Semantic Segmentation [Thoma (2016)] problems. The following sections present relevant studies in each wave.

3.2 HAND-CRAFTED METHODS

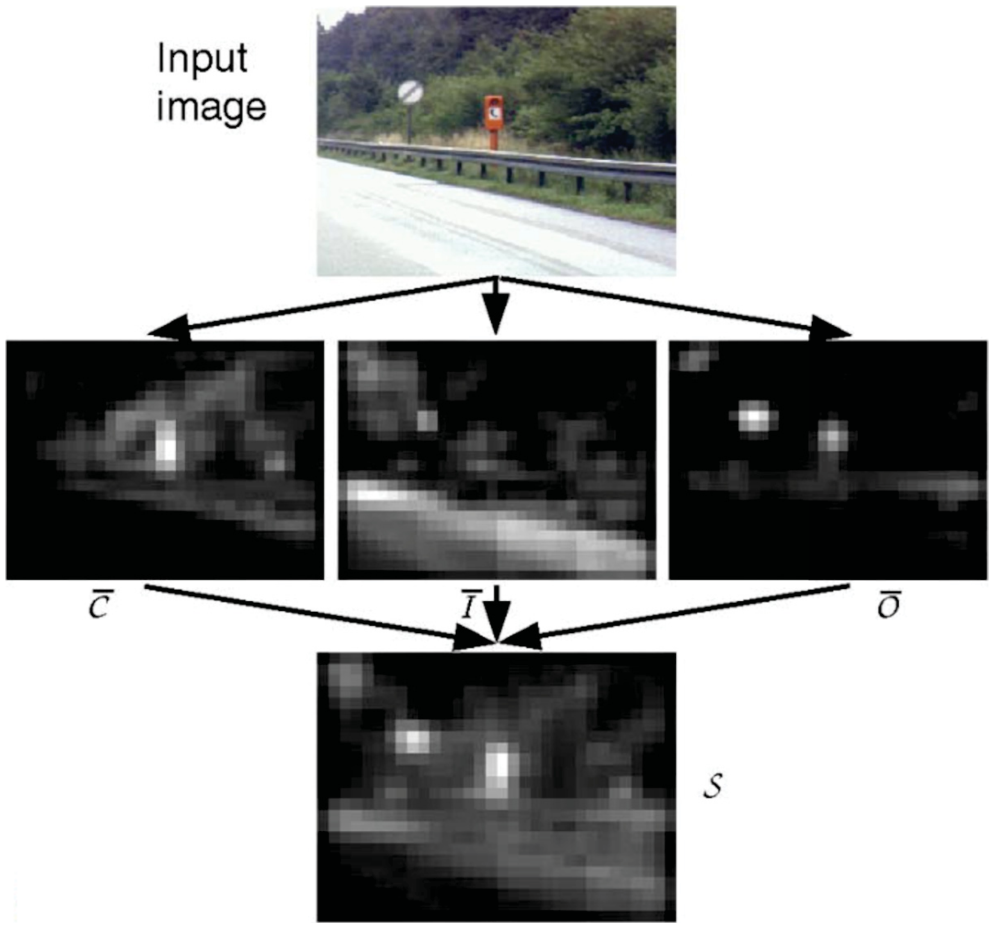


Figura 3.2: Example of a Saliency Map generated by Itti et al. (1998). The three feature maps, "$\overline{C}$","$\overline{I}$"and "$\overline{O}$", encode information of color contrast, intensity contrast and orientation contrast, respectively, and are combined to generate the Saliency Map S. Source: Itti et al. (1998).

One of the earliest studies presented in SOD literature was proposed by Itti et al. (1998) and started a wave of works on the subject. The proposed method applies Gaussian Pyramids [Burt and Adelson (1983)] to generate nine spatial scales. Information of color, intensity, and orientation from the different scales are extracted through a center-surrounding technique.

They generate a set of feature maps for each extracted feature, which are combined to generate the Saliency Map. In the last step, they use a dynamical neural network to select attended locations and refine the final Saliency Map. Figure 3.2 presents an example of a Saliency Map generated by the proposed method. The three feature maps encode information of color contrast, intensity contrast, and orientation contrast, and are combined to generate the Saliency Map.

The hand-crafted approaches can be divided into two categories based on the visual subset explored: blocks (pixels or patches) or super-pixels [Borji et al. (2014)]. In the block-based approaches, each pixel or patch in the image receives a saliency value. In the super-pixel-based approaches, each super-pixels generated by an earlier segmentation step receives a saliency value.

In the block-based category, one of the earliest studies was presented by Achanta et al. (2008). For each pixel in the CIELab color space [McLAREN (1976)], three windows, of different sizes and centered in the pixel, are scrolled through the image, calculating the color and luminance contrast between the pixel and its neighbors.

Three Saliency Maps are generated and combined, generating a final Saliency Map utilized to perform a segmentation of the salient object. Figure 3.3 presents the output of each step of the proposed method. (a) presents the input image and (b) presents the three scales Saliency Maps. (c) presents the final Saliency Map, utilized to segment the saliency object, as presented in (d).

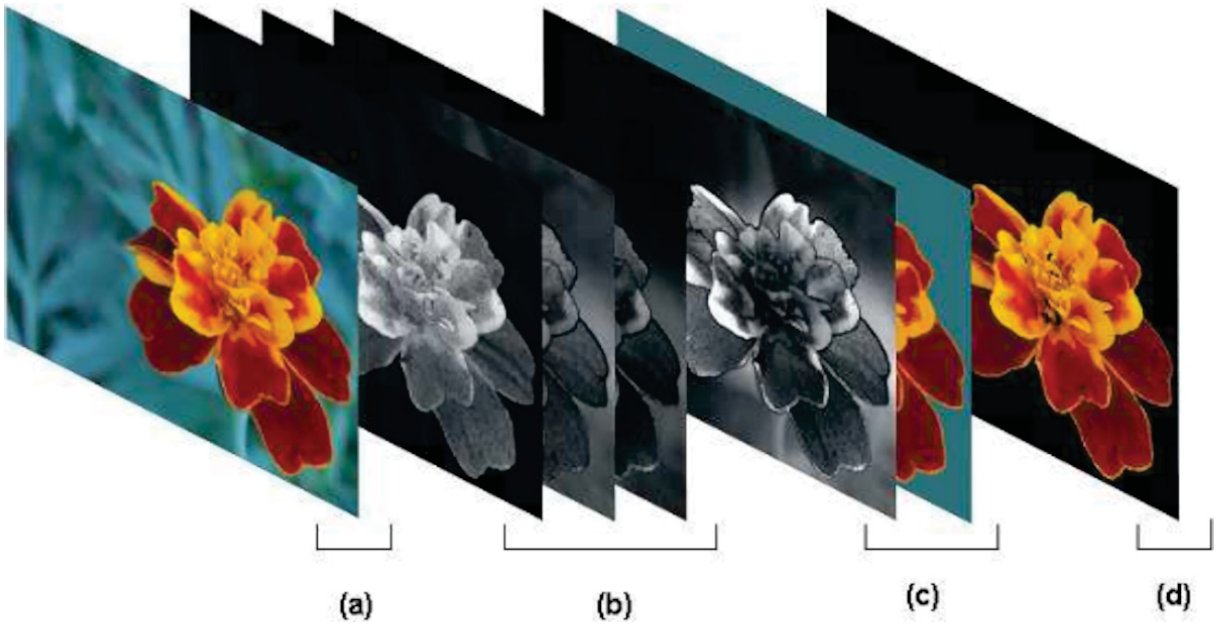
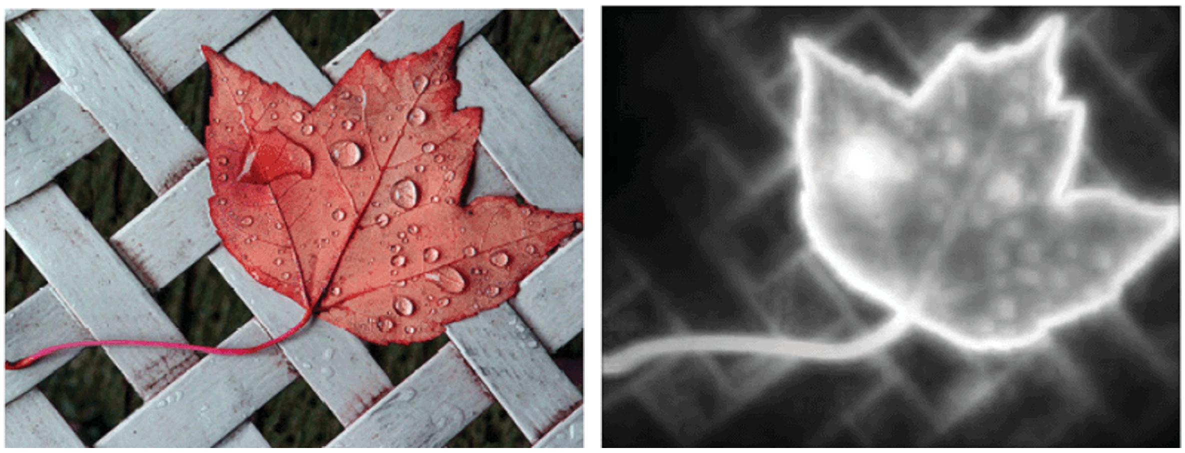


Figura 3.3: Example of a Saliency Map generated by Achanta et al. (2008). The input image is presented in (a) and the three scales Saliency Map are presented in (b). In (c), it is presented the final Saliency Map, utilized to segment the saliency object presented in (d). Source: Achanta et al. (2008).

Another work in the block-based category was presented by Goferman et al. (2012). The authors proposed a context-aware method to find saliency regions in images based on four principles. The first two principles define that regions with colors and patterns differences are more likely to be salient than homogeneous, blurred, or the ones with repeated features.



(a) Example of a scene containing salient objects (the leaf).

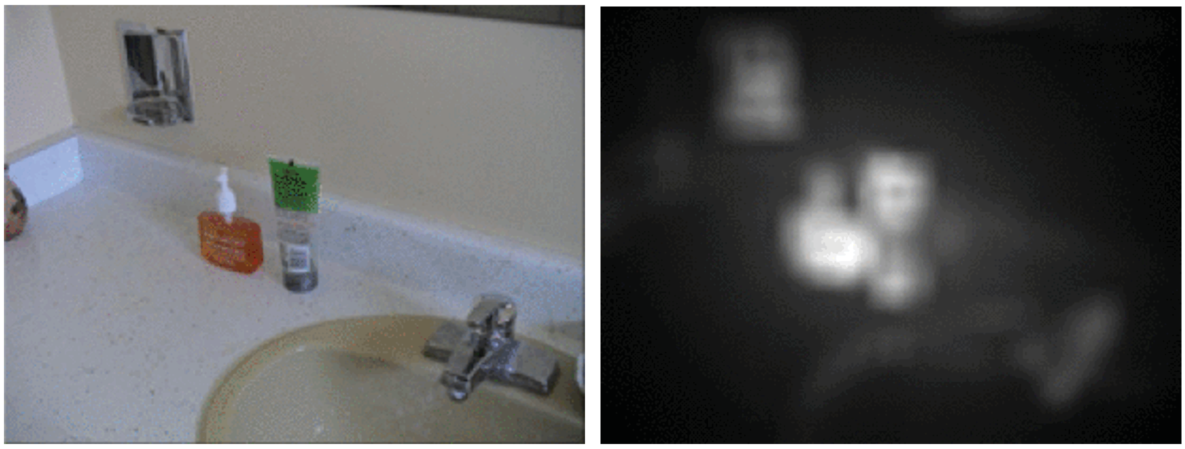
(b) The Saliency Map generated by the method proposed by Goferman et al. (2012).

Figura 3.4: Example of a Saliency Map generated by Goferman et al. (2012). Source: Goferman et al. (2012).

Following these principles, for each pixel "i", a patch "p_i"is created with "i"in the center. The Euclidean Distance [Duda et al. (2000)] is applied to find the "K"most similar patches to "p_i". The pixel "i"is salient if the "K"patches have a high difference from "p_i". The Euclidean Distance is also utilized to calculate the distance between patches, following the third principle, which defines that salient pixels should be grouped.

Different scales are utilized, and the patch "p_i"should be different from all other patches in all scales. The immediate context is added to the salient objects by extracting the most attended localized areas at each scale. The Euclidean Distance to the closest attended pixel is applied to highlight background regions around the salient regions.

The last principle includes prior knowledge about the objects and their location, being utilized as a post-processing step to improve the Saliency Map. Figure 3.4a presents an example of a scene containing a salient object (the leaf), and Figure 3.4b presents the Saliency Map generated by the proposed method.



(a) Example of a scene containing salient objects (the bottles).

(b) The Saliency Map generated by the method proposed by Duan et al. (2011).

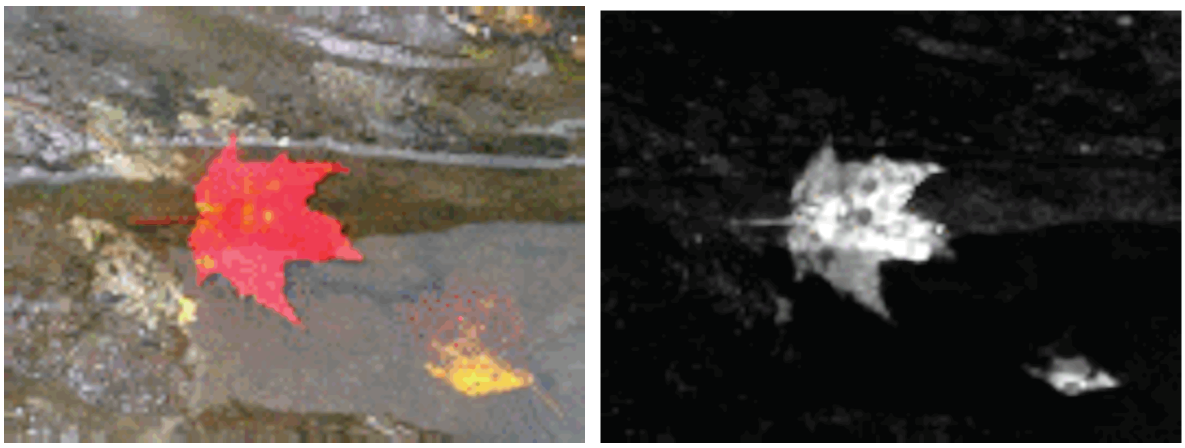
Figura 3.5: Example of a Saliency Map generated by Duan et al. (2011). Source: Duan et al. (2011).

Duan et al. (2011) utilizes spatially weighted dissimilarity between patches to find saliency regions in the image. The proposed framework has three steps: patches generation, dimensionality reduction, and evaluation of the spatially-weighted dissimilarity. In the patch generation step, the input image is divided into non-overlapping patches, forming a grid with each patch represented as a vector of pixels. The Principal Component Analysis (PCA) [F.R.S. (1901)] algorithm is applied in the second step to reduce the patches dimensionality.

Dissimilarities and spatial distance between patches are utilized to calculate the saliency value of each patch. The distance of the patch and the center of the image is also considered to calculate the saliency value of the patch. Figure 3.5a presents an example of a scene containing salient objects (the bottles), and Figure 3.5b presents the Saliency Map generated by the proposed method.

Frequencies in the image are utilized by Achanta et al. (2009) to preserve boundaries of salient objects. Two cut-off values are defined: "w_{lc}"to low frequencies and "w_{hc}"to high frequencies. The Difference of Gaussians (DoG) [Gonzalez (2016)] is applied to create a blurred version of the image, with the "w_{lc}"and "w_{hc}"utilized to select the best parameters of the Gaussian filters. The saliency value of a pixel is calculated as the norm of the difference between the arithmetic mean pixel value in the original image and the pixel value in the Gaussian blurred image.

Aiming to add information of color and luminance to the Saliency Map, the mean pixel value is replaced with the mean image feature vector in CIELab color space. The "L_2"norm is applied over the difference between the mean image feature vector and the pixel value in the Gaussian blurred image. Figure 3.6a presents an example of a scene containing a salient object (the leaf), and Figure 3.6b presents the Saliency Map generated by the proposed method.


(a) Example of a scene containing a salient object (the leaf).

(b) The Saliency Map generated by the method proposed by Achanta et al. (2009).

Figura 3.6: Example of a Saliency Map generated by Achanta et al. (2009). Source: Achanta et al. (2009).

In the super-pixel-based category, Perazzi et al. (2012) proposed an algorithm to identify saliences in images using two contrast measurement: element uniqueness and element distribution. The input image is decomposed in super-pixels through a modification of the Simple Linear Iterative Clustering (SLIC) [Achanta et al. (2010)] algorithm.

Two partial Saliency Maps are constructed based on contrast measures (uniqueness and spatial distribution) and combined to produce the final Saliency Map. Element uniqueness is computed by comparing the super-pixel color and position with all the other super-pixels while element distribution is calculated as the spatial variance of the super-pixel color.

Figure 3.7 presents the result of each step of the proposed algorithm. The input image is presented in (a). The super-pixel representation is presented in (b). Figures (c) and (d) presents the element uniqueness and element distribution extracted from the super-pixels, respectively. The final Saliency Map is presented in (e). Finally, (f) presented the ground-truth mask.

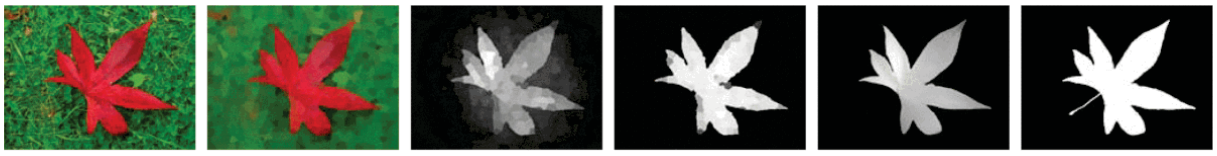

(a) Source image. (b) Abstraction. (c) Uniqueness. (d) Distribution. (e) Saliency. (f) Ground truth.

Figura 3.7: Output of each step of the framework proposed by Perazzi et al. (2012). The input image is presented in (a). The super-pixel representation is presented in (b). In (c) and (d) is presented the element uniqueness and element distribution extracted from the super-pixels, respectively. The final Saliency Map is presented in (e). Finally, (f) presented the ground-truth mask. Source: Perazzi et al. (2012).

Context and shape information are utilized by Huaizu Jiang and Zheng (2011) to produce the saliency map. In the context step, the saliency of a super-pixel is calculated as a function of its spatial neighbors. Gaussian Pyramids are applied to add multiscale information in order to find salient objects independent of its size.

In the shape detection step, an edge detection algorithm is applied to find the contours of the objects, selecting a subset of edge segments close to the salient object. The salient object segmentation is defined as an energy minimization problem [LeCun et al. (2006)] which use the context and shape Saliency Maps to segment the salient object.

Figure 3.8a presents an example of a scene containing a salient object (the squirrel), Figure 3.8b presents the Saliency Map generated in the context step, Figure 3.8c presents the Saliency Map generated in the shape detection step, and Figure 3.8d presents the final Saliency Map generated by the proposed method.

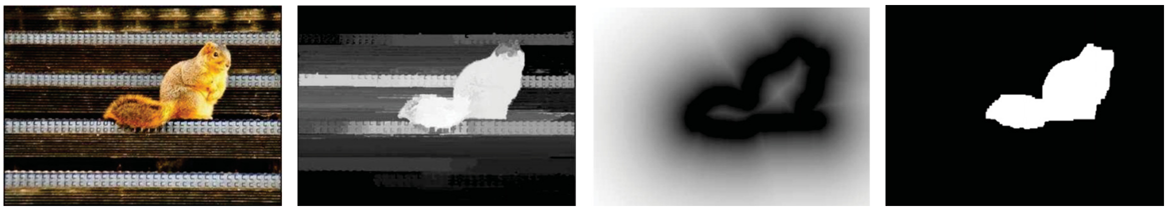
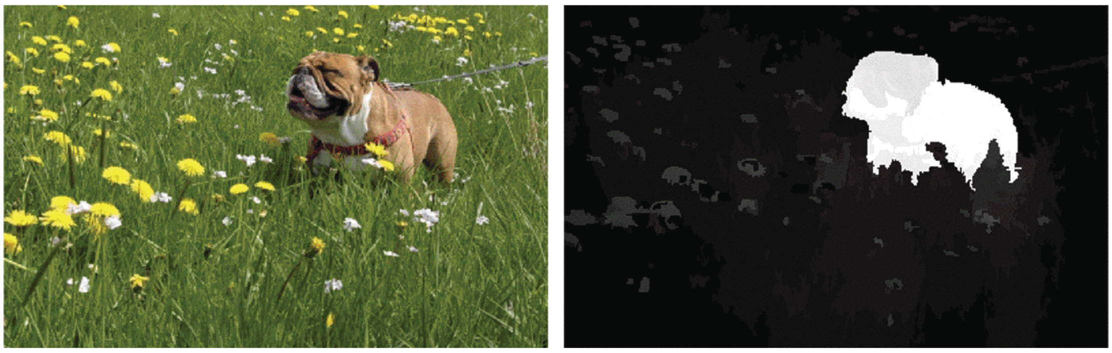

(a) Example of a scene containing salient a object (the squirrel). (b) The Saliency Map generated in the context step. (c) The Saliency Map generated in the shape generation step. (d) The final Saliency Map generated by the proposed method.

Figura 3.8: Examples of Saliency Map generated in each step of the method proposed by Huaizu Jiang and Zheng (2011). Source: Huaizu Jiang and Zheng (2011).

A watershed method [Gonzalez (2016)] is applied by Yan et al. (2013) to generate a set of super-pixels (regions). For each region, a scale value is computed based on its shape uniformities and utilized to produce three image layers. The regions are grouped in layers by a threshold value (one threshold for each layer). A hierarchical inference based on a tree-structure is molded to fuse cues extracted from the three image layers.

In the proposed representation, each tree level represents a layer, and each node represents a region inside the layer. An energy function minimization is applied to compute the optimal saliency values of each region in the proposed tree. Figure 3.9a presents an example of a scene containing a salient object (the dog), and Figure 3.9b presents the Saliency Map generated by the proposed method.



(a) Example of a scene containing a salient object (the dog).

(b) The Saliency Map generated by the method proposed by Yan et al. (2013).

Figura 3.9: Example of a Saliency Map generated by Yan et al. (2013). Source: Yan et al. (2013).

For many years, the studies based on hand-crafted features were considered state of the art in SOD literature. However, these methods lacked from generalization capacity, not performing well in challenging scenarios [Borji et al. (2014)]. The features extracted by the methods presented above (color, luminance, contrast, contour, objectness, luminance, spatial distribution, pattern distinctness) were predefined manually, turning it difficulty to find saliency regions in scenarios with saliency regions not defined by those features.

In order to overcome such limitations, a new category of methods based on Deep Learning and the recent success of CNNs solving other Computer Vision problems started of being applied to find saliency regions. Next section presents relevant studies proposed in SOD literature based on this approach.

3.3 CONVOLUTIONAL NEURAL NETWORKS-BASED METHODS

CNNs were initially proposed by Lecun et al. (1998) to solve image recognition and object classification problems. Krizhevsky et al. (2012) proposed a CNN model for image classification and achieved impressive results on ImageNet [Deng et al. (2009)] dataset, winning the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) 2012 competition.

Following the impressive results achieved by CNNs, studies in the SOD literature started to explore this technique to find salience regions in images. However, as CNNs were not designed for segmentation, it was necessary to attach pre and post-processing steps to overcome such limitation.

In a common strategy presented in the literature, the input image was decomposed in patches or super-pixels, similarly with the hand-crafted-based methods. CNNs were applied to extract features and to classify the patches or super-pixels as saliency or background regions.

In the super-pixel-based category, one of the earliest models was presented by He et al. (2015). The proposed method decomposes each super-pixel in two contrast sequences of color uniqueness and color distribution. Each sequence feeds a different CNN with 1D convolutions, "f_1"and "f_2", in parallel. The outputs of the CNNs are two complementary sequences with hierarchical features utilized to calculate the saliency value of the super-pixel.

Three sequences of "f_1"and "f_2"are structured in parallel, forming a multiscale network with shared weights. Each sequence receives super-pixels of different scales and outputs a Saliency Map. A weighted sum of all Saliency Maps is performed, generating the final Saliency Map. Figure 3.10 presents the structure of the proposed model.

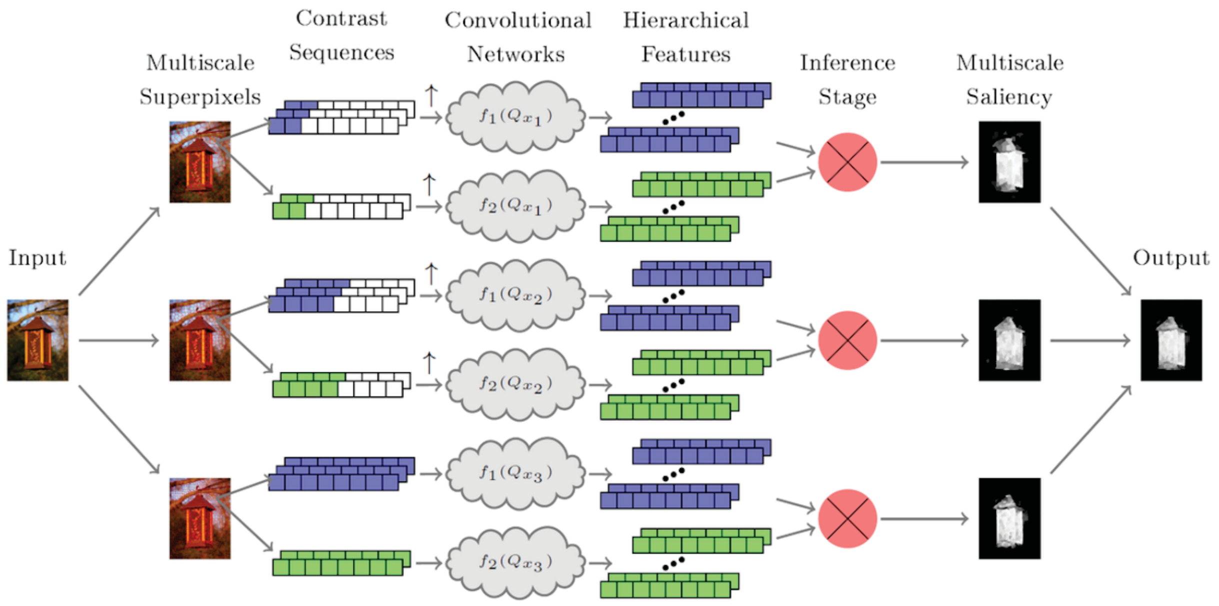


Figura 3.10: The architecture proposed by He et al. (2015). The input image is divided into multiscale super-pixels. Contrast sequences are extracted to feed the CNNs, which outputs hierarchical features utilized to calculate the Saliency Map of each scale. The final saliency map is generated as the weighted sum of the Saliency Maps of all scales. Source: He et al. (2015).

In the patch-based category, Li et al. (2017b) apply the Selective Search [Uijlings et al. (2013)] algorithm to generate a set of squared patches. A CNN model generates a first feature vector containing high-level features extracted from the patches. Low-level information like color contrast, texture, and spatial properties, are extracted from the patches to generate a second feature vector. A Support Vector Machine (SVM) [Hearst et al. (1998)] receives both feature vectors and outputs if the patch is salient or not. The proposed network is presented in Figure 3.11.

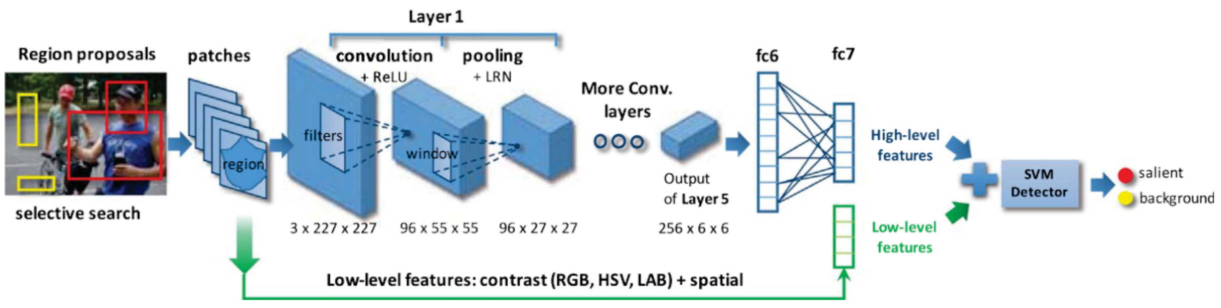


Figura 3.11: The architecture proposed by Li et al. (2017b). Selective Search is applied to segment the input image in patches (regions proposals). Two feature vectors are generated, one with high-level features generated by the CNN and one with low-level features (contrast, texture and spatial properties) generated by hand-crafted methods. The feature vectors are concatenated and utilized as the input of a SVM, which outputs if a particular patch is salient or not. Source: Li et al. (2017b).

Li and Yu (2015) generate a set of non-overlapping regions from the input image. For each region, three windows are created. The first window contains only the region, the second window contains the region and its immediate neighboring regions, and finally, the third window contains the entire image. For each window, a CNN is trained to generate a vector of features.

The three feature vectors feed a MLP [Haykin (2009)] to classify the regions as being salient or not. A complementary step, added by Li and Yu (2016b), integrate hand-craft features

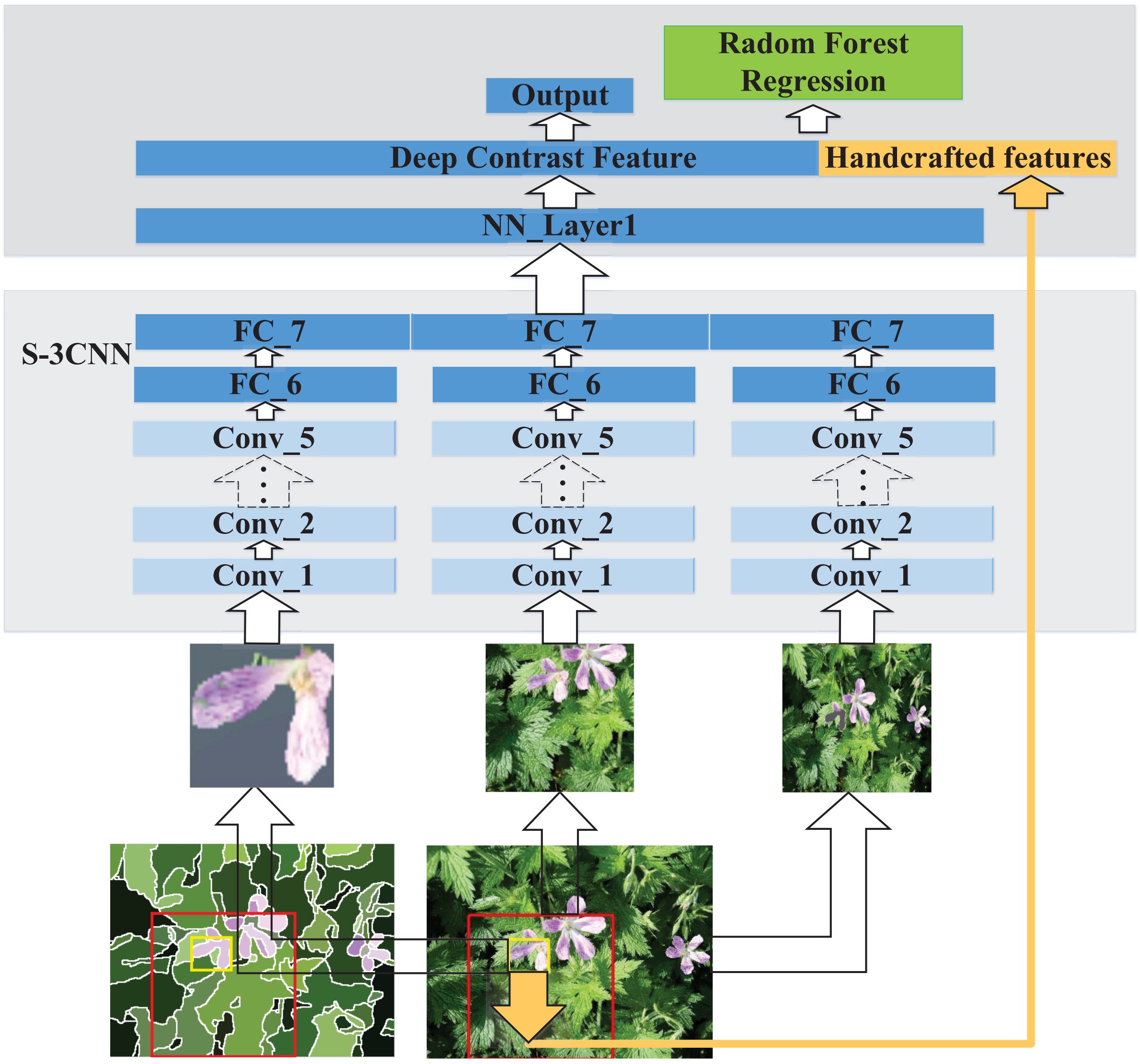


Figura 3.12: The architecture proposed by Li and Yu (2016b). The input image is decomposed into regions. For each region, three windows of different sizes are created, with each window feeding a CNN in parallel. A MultiLayer Perceptron (MLP) is utilized to generate a first Saliency Map based on the high-level features extracted from the CNNs. In parallel, hand-craft features are extracted and concatenated with the high-level features extracted by the CNNs, generating a feature vector utilized as the input of a Random Forest Regressor. Source: Li and Yu (2016b).

with the previews high-level features extracted from the CNNs, generating another feature vector utilized as the input of a Random Forest Regressor [Liaw et al. (2002)].

The final Saliency Map is calculated through the linear combination of the Saliency Maps generated in both steps. The proposed framework is presented in Figure 3.12, illustrating the process with an example of a region extracted from the image.

Wang et al. (2016) apply the Fast Region-based Convolutional Neural Network (Fast R-CNN) [Ren et al. (2017)] architecture to classify the super-pixels as being part of the saliency or background regions, generating a first Saliency Map. In parallel, another Fast R-CNN architecture receives a representation of the image generated by edge-preserving methods, and generates a second Saliency Map to refine the borders of the salient regions in the first Saliency Map.

Low-level features (contrast and backgroundness) and an edge-based propagation method are applied to refine the Saliency Map.

Figure 3.13 presents all the steps of the proposed framework. The input image is presented in (a). The super-pixel and edge regions representations are presented in (b). The Saliency Map learned for each representation is presented in (c). The low-level cues extracted from the input are presented in (d), and the refined Saliency Map is presented in (e). In (f), the edge-based propagation method is applied to generate the final Saliency Map in (g).

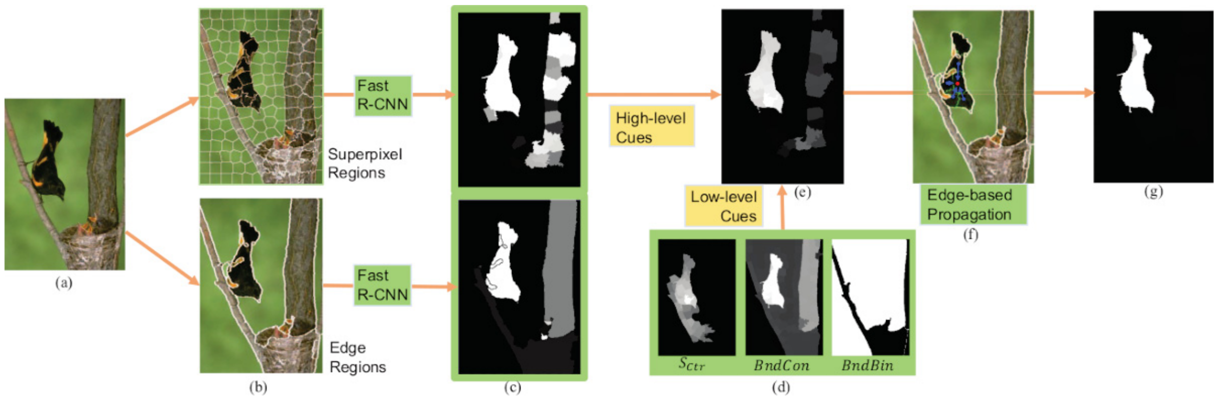


Figura 3.13: The architecture proposed by Wang et al. (2016). The input image is presented in (a). The super-pixel and edge regions representations are presented in (b). The Saliency Map learned for each representation is presented in (c). The low-level cues extracted from the input are presented in (d), and the refined Saliency Map is presented in (e). In (f), the edge-based propagation method is applied to generate the final Saliency Map in (g). Source: Wang et al. (2016).

Another approach explored by CNN-based methods is to fuse local-context information with global-context. The local-context detect high-frequency content while global context suppress the homogeneous regions through holistic contrast and color statistics from the entire image [Wang et al. (2015)]. Zhao et al. (2015) proposed a Neural Network with two branches, as presented in Figure 3.14, to explore this approach.

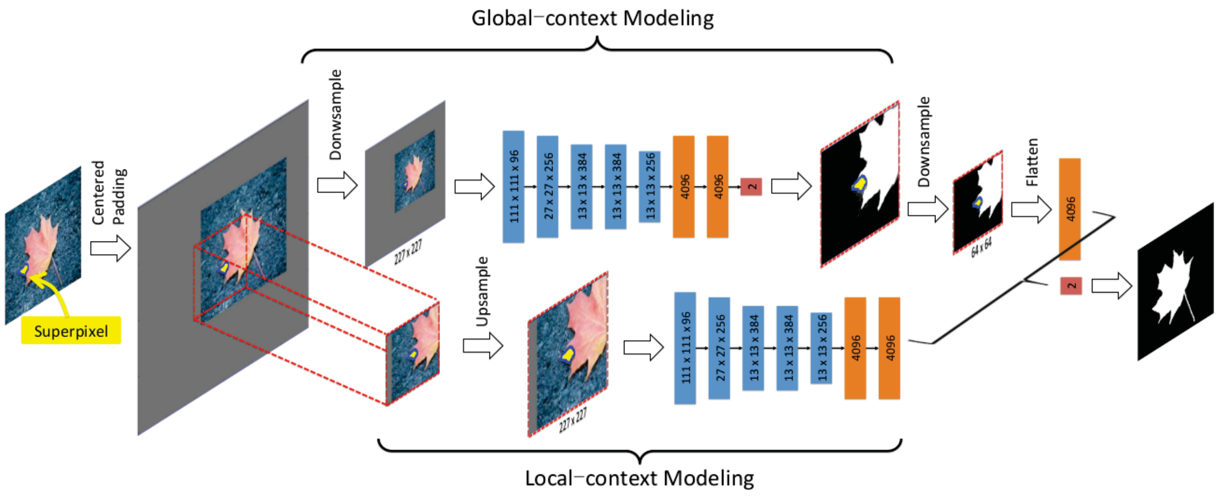


Figura 3.14: The architecture proposed by Zhao et al. (2015). The Global-Context Modeling (upper branch) receives the entire image centered in the super-pixel, and the Local-Context Modeling (bottom-branch) receives the same window with a higher focus in the super-pixel. Information from both contexts is utilized to generate the final Saliency Map. Source: Zhao et al. (2015).

In the proposed method, the first branch (Global-Context Modeling) is designed to extract global-context saliences, while the second branch (Local-Context Modeling) is designed to extract local-context saliences. The SLIC algorithm is applied to extract a sequence of super-pixels from the input image.

The global-context branch receives a window containing the entire image centered in a super-pixel, while the local-context branch receives a window with a higher focus on the super-pixel. The final probability of a super-pixel be salient is based on the probability of a super-pixel be salient in both contexts.

The CNN-based methods proposed in the SOD literature also integrates low-level features extracted from hand-crafted-based methods with high-level features extracted by CNNs. Lee et al. (2016) proposed an encoded low-level distance map (ELD-map) to encode low-level features (colors, color distributions, Gabor filter [Fogel and Sagi (1989)] responses, and location).

The high-level features are extracted by the VGG-16 [Simonyan and Zisserman (2014)]. The output of the VGG-16 and the ELD-map are concatenated to input two fully-connected layers, utilized to classify the super-pixels as being salient or not. Figure 3.15 presents the proposed two-branches framework. The upper-branch presents the encoding step of the ELD-map, and the bottom-branch presents the VGG-16 model to encode high-level features.

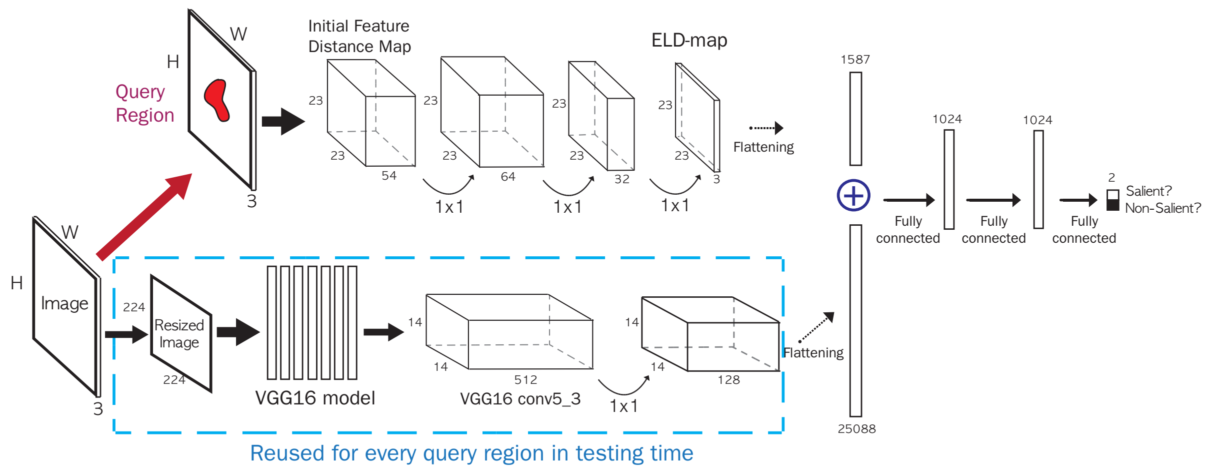


Figura 3.15: The architecture proposed by Lee et al. (2016). The upper-branch presents the encoding step of the ELD-map, and the bottom-branch presents the VGG-16 model to encode high-level features. Both feature vectors are concatenated and utilized as the input of two fully-connected layers to classify the super-pixel as being salient or not. Source: Lee et al. (2016).

The framework proposed by Wang et al. (2015) explores multi-context information and integrates high-level features (extracted by CNNs) with low-level features (extracted by hand-crafted methods). In the local estimation step, the input image feeds a neural network called DNN-L, generating a first Saliency Map.

In parallel, a set of object segments is generated by the Geodesic Object Proposal (GOP) [Krähenbühl and Koltun (2014)] algorithm, and utilized to refine the Saliency Map generated by the DNN-L. From the output of the DNN-L and the object proposals, a feature vector containing global contrast features, geometric information, and local saliency measurements is generated for each potential object. The feature vector is utilized to perform the training of a global context Recurrent Neural Network (RNN) [Gregor et al. (2015)] called DNN-G.

The final Saliency Map is computed through a sum of salient object regions weighted by their saliency values. Figure 3.16 illustrates all steps of the proposed framework. The DNN-L architecture is illustrated in (a). The output of DNN-L in (b) is refined by object proposals (the green block) in (c). The feature vectors extracted from the potential objects and the DNN-L are presented in (d). The DNN-G architecture is illustrated in (e). The salient object regions are presented in (f) and the final Saliency Map in (g).

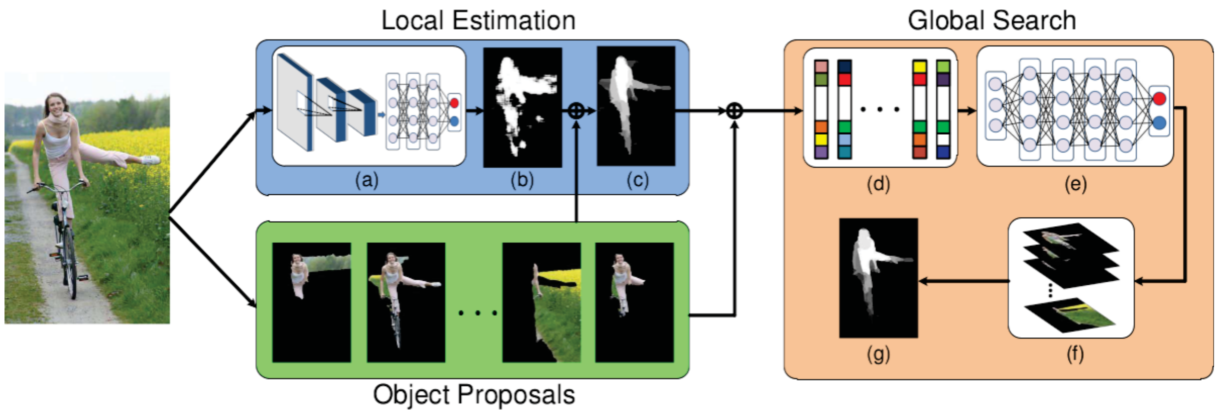


Figura 3.16: The architecture proposed by Wang et al. (2015). The DNN-L architecture is illustrated in (a). The output of DNN-L in (b) is refined by object proposals (the green block) in (c). The feature vectors extracted from the potential objects and the DNN-L are presented in (d). The DNN-G architecture is illustrated in (e). The salient object regions are presented in (f) and the final Saliency Map in (g). Source: Wang et al. (2015).

A different approach, proposed by Kim and Pavlovic (2016), turned the salient detection problem in a multi-class classification problem. The proposed method generate a set of patches in the input image through the Selective Search algorithm. A clustering algorithm is applied to separate the patches into "K"shape classes.

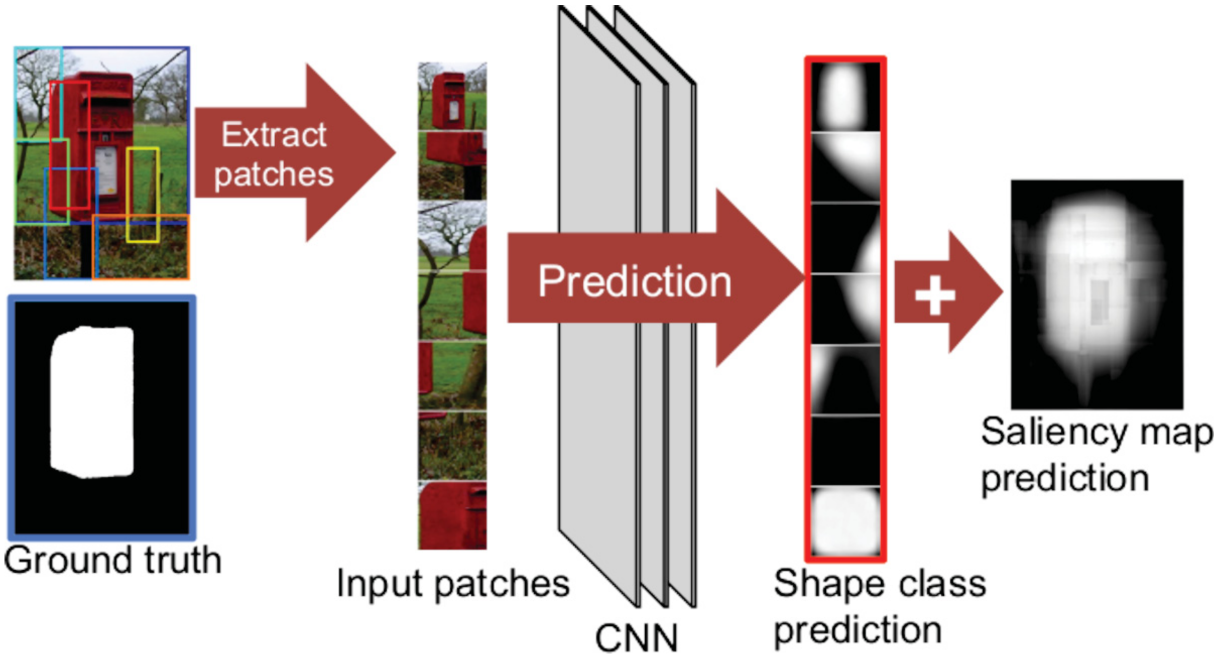


Figura 3.17: The architecture proposed by Kim and Pavlovic (2016). Patches are extracted from the input image, feeding a CNN to predict patches class. The final Saliency Map is calculated as the mean of the most probable masks of each patch shape. Source: Kim and Pavlovic (2016).

To generate the labels of the patches to train the CNN, the Euclidean Distance is applied to find the closest class of each shape. The patches feed a multi-class CNN, and the final Saliency Map is calculated as the mean of the most probable masks of each patch shape. A shape class-based saliency detection with hierarchical segmentation (SCSD-HS) algorithm is proposed to add global-context information in the Saliency Map. Figure 3.17 illustrates each step of the proposed framework.

As presented above, there are several ways to apply CNNs to find saliency regions in images. Fusing high-level features extracted by CNNs with low-level features extracted by hand crafted-based methods, and fusing local-context with global-context information in a two-branched network, are the two most common approaches proposed in the SOD literature.

However, all presented methods needed pre and post-processing steps to perform the segmentation of salient objects, with the super-pixels being the most common strategy explored by the literature due the fact that the block-based algorithms stands out high-contrast edges instead of the saliency regions [Borji et al. (2014)].

Besides the promising results in the CNN-based category, the fully connected layers in CNNs generates blurriness and inaccurate predictions near the boundaries of salient objects in super-pixels-based methods [Borji et al. (2014)].

To overcome such problems, a new approach, already applied in Semantic Segmentation [Thoma (2016)] problems, started to be explored in the SOD literature: FCNs. FCNs performs segmentation in a per-pixel level, which is more accurate and solves the problems generated by the fully connected layers in CNNs. Next section presents relevant methods proposed in SOD literature based on FCNs.

3.4 FULLY CONVOLUTIONAL NEURAL NETWORKS-BASED METHODS

The FCN [Long et al. (2015)] is an improvement of the CNNs developed to perform Semantic Segmentation [Thoma (2016)] tasks. Aiming to convert a CNN in a FCN, the fully connected layers in CNNs models are replaced by convolutional ones, generating as output a 2D vector called heatmap. Instead of a 1d vector generated by CNNs that contains the probabilities of an object be of a determined class, the heatmap contains the probabilities of a region in the image be of a determined class. A detailed explanation about the FCN structure is presented in Chapter 4.

The FCNs also have a upsample or deconvolutional layer utilized to convert the heatmap in a segmentation image of the same size of the input image. There are several ways of performing a upsample or deconvolution in FCNs. The most popular are: transposed convolution [Dumoulin and Visin (2016)], unpooling [Noh et al. (2015)], and bilinear interpolation [Kirkland (2010)].

The impressive results obtained by FCNs in Semantic Segmentation tasks led the studies in the SOD literature to adapt the network to the salience problem. Bianco et al. (2017) explored FCNs to perform segmentation of salient objects and showed the network efficiency in seven state-of-art datasets. Nie et al. (2018) applied FCNs to generate two probability maps: one with the probabilities of a pixel be salient, and one with the probabilities of a pixel be part of the background. Both Saliency Maps are utilized to generate the final probability of a pixel be salient.

Li and Yu (2016a) proposed an end-to-end deep contrast network with two complementary branches: a fully convolutional stream and a segment-wise spatial pooling stream. The fully convolutional stream is a multiscale fully convolutional network (MS-FCN). The MS-FCN receives the raw image to generate a first Saliency Map through a VGG-16 network converted in FCN.

Three extra convolutional layers are attached to each of the first four max-pooling layers of the VGG-16, generating four segmentation outputs. The Saliency Map of each output and the Saliency Map generated in the last layer of the network are utilized to form the final Saliency Map at this branch. The second stream receives a set of super-pixels generated from the input image. For each super-pixel, spatial pooling and saliency estimation are applied to construct a Saliency Map at the super-pixel level.

A convolutional layer is applied to fuse the Saliency Map of both streams, generating a final Saliency Map. A more recent version, proposed by Li and Yu (2018), applied a trained attention module to fuse the Saliency Map of both streams. Figure 3.18 presents the structure of the proposed framework. The upper-branch presents the fully convolutional stream, and the upper-branch presents the segment-wise spatial pooling stream.

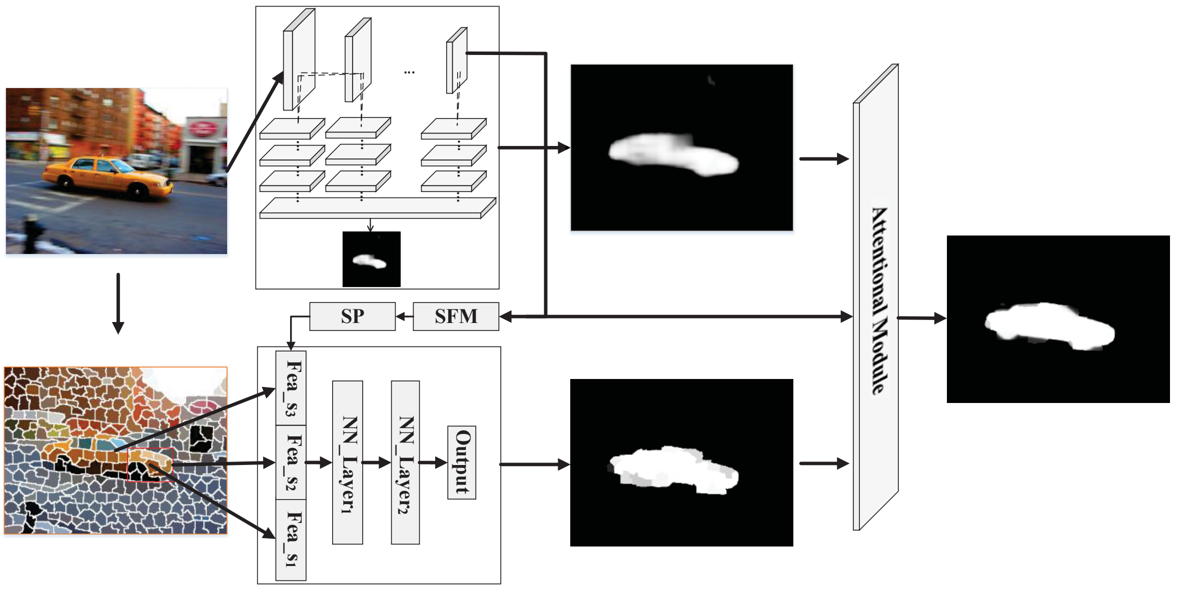
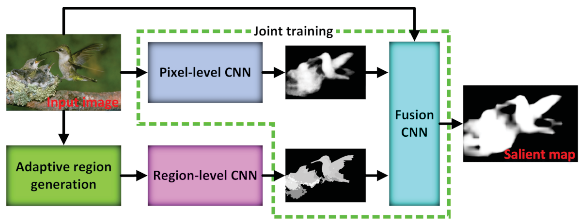
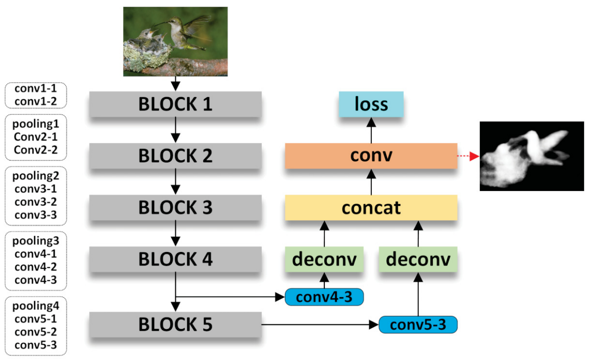


Figura 3.18: The architecture proposed by Li and Yu (2018). The fully convolutional stream (upper-branch) generates a Saliency Map through a VGG-16 network converted in FCN. A super-pixel level Saliency Map is generated in the bottom-branch. An attention module is proposed to fuse the Saliency Map of both streams, generating the final Saliency Map. Source: Li and Yu (2018).



(a) The proposed framework with a FCN generating a pixel-level segmentation, and a CNN generating a region-level segmentation. A fusion CNN is applied to integrate both Saliency Maps.



(b) The proposed FCN. The last block of the VGG-16 is removed. A sequence of convolutional and deconvolutional layers are attached in the last two blocks to generate the Saliency Map.

Figura 3.19: The architecture proposed by Tang and Wu (2016). Source: Tang and Wu (2016).

Tang and Wu (2016) constructed the Saliency Map by fusing the Saliency Map of a FCN with the Saliency Map of a CNN, as presented in Figure 3.19a. The FCN performs a pixel-level segmentation while the CNN performs super-pixel level segmentation. In the pixel-level FCN, the last block of the VGG-16 is removed. A sequence of convolutional and deconvolutional layers are attached in the last two blocks of the VGG-16 to generate the Saliency Map, as presented in

Figure 3.19b. The Saliency Map at pixel-level and the Saliency Map at the super-pixel level are integrated by a fusion CNN.

Zhang et al. (2017) adapted the Residual Network (ResNet)-101 network to perform segmentation, as illustrated in Figure 3.20. The final pooling and the fully connected layers are removed and replaced by multiple parallel dilated convolutional layers to generate a Saliency Map. The SLIC algorithm is applied in the input image to generate super-pixels of three different scales.

The Saliency Map produced by the FCN is utilized to assign a saliency value to each super-pixel. An energy minimization function is applied to refine the saliency value of each super-pixel. In the last step, the Saliency Maps at each scale are utilized to generate the final Saliency Map. Figure 3.20 presents the proposed model.

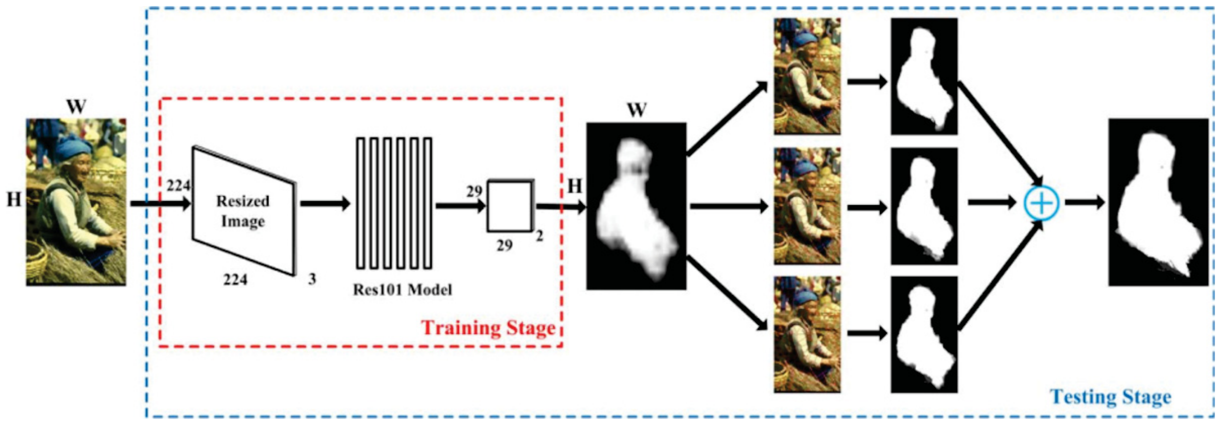


Figura 3.20: The architecture proposed by Zhang et al. (2017). The ResNet-101 model is adapted to perform pixel level segmentation, generating a Saliency Map. The input image is decomposed into three scales of super-pixels. The Saliency Map produced by the FCN is utilized to assign a saliency value to each super-pixel, generating three Saliency Maps, which are utilized to generate the final saliency map. Source: Zhang et al. (2017).

Li et al. (2016) proposed a multi-task architecture to saliency and semantic segmentation tasks in a two-branched network. The feature extraction step is shared between the two branches to improve the saliency detection by adding semantic information on salient objects. In the semantic segmentation branch, a deconvolution step is utilized to produces an image with the "K"objects segmented, where "K"is the number of classes.

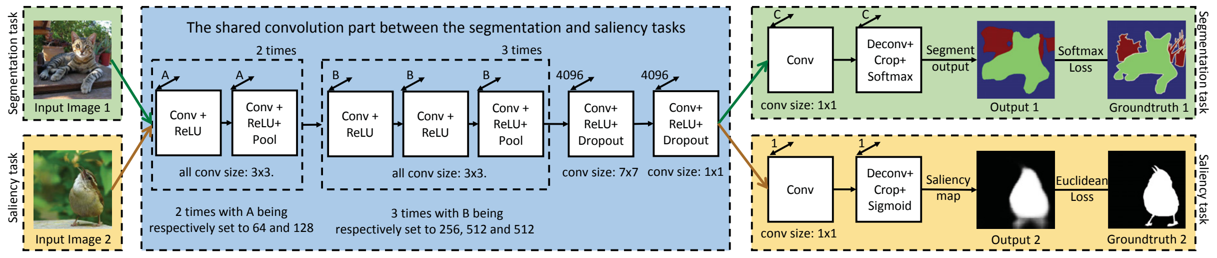


Figura 3.21: The architecture proposed by Li et al. (2016). The input image feeds the feature extraction step (blue rectangle). The network is composed of two outputs, one for Semantic Segmentation task (green rectangle) and one for saliency segmentation task (yellow rectangle). Source: Li et al. (2016).

In the saliency detection branch, a super-pixel graph, with a Laplacian regularized non-linear regression, is applied to refine fuzzy boundaries. Figure 3.21 presents the two branches network. The input image feeds the feature extraction step (blue rectangle). The network is composed of two outputs, one for Semantic Segmentation task (green rectangle) and one for saliency segmentation task (yellow rectangle).

Xi et al. (2017) proposed an end-to-end network with three sequential steps. The VGG-16 network is applied to extract feature maps in the first step. Three fully convolutional layers are added at the end of the VGG-16, making a non-linear regression in the middle step. In the end, bicubic interpolation is applied to generate the Saliency Map. Figure 3.22 presents all the steps of the proposed network. I presents the VGG-16 layers designed to extract features. In II, three convolutional layers are added. In III, bicubic interpolation is performed to restore the Saliency Map size.

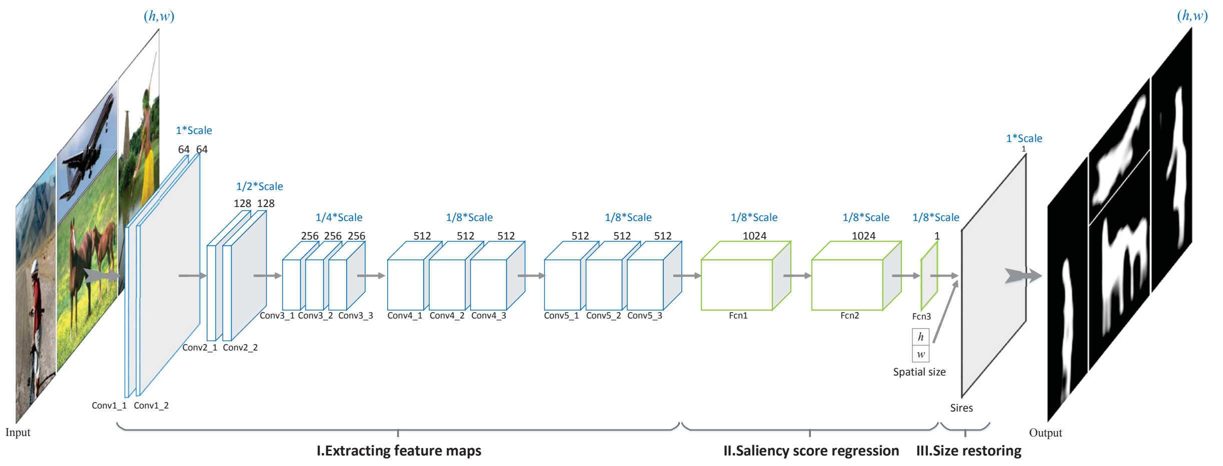


Figura 3.22: The architecture proposed by Xi et al. (2017). I presents the VGG-16 layers designed to extract features. In II, three convolutional layers are added. In III, a bicubic interpolation is performed to restore the Saliency Map size. Source: Xi et al. (2017).

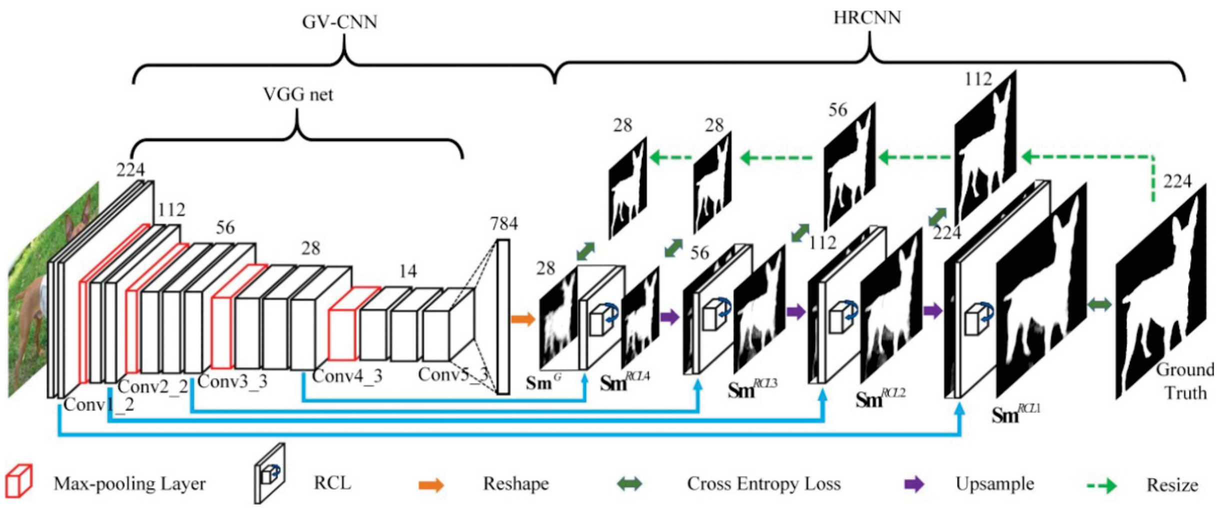


Figura 3.23: The architecture proposed by Liu and Han (2016). The GV-CNN adapts the VGG-16 structure to extract global features (contrast, objectness, and compactness) from the input image, outputting a Saliency Map "Sm^G". The Saliency Map "Sm^G"is refined through a sequence of recurrent convolutional layers in the HR-CNN, adding local information in the Saliency Map. Source: Liu and Han (2016).

Liu and Han (2016) also proposed an end-to-end model called DHSNet to find saliency regions in images. The proposed model is divided into two sequential sub-networks: GV-CNN and HR-CNN. The GV-CNN adapts the VGG-16 structure to extract global features (contrast, objectness, and compactness) from the input image, outputting a Saliency Map "Sm^G". The Saliency Map "Sm^G"is refined through a sequence of recurrent convolutional layers in the

HR-CNN, adding local information in the Saliency Map. Upsampling layers are utilized to recover the size of the Saliency Map. Figure 3.23 presents the proposed model.

Hou et al. (2018) proposed a top-down approach to combine low-level with high-level features. The VGG-16 network is adapted with a modification of the Holistically-Nested Edge Detector (HED) [Xie and Tu (2017)] network for edge and boundary detection of salient objects. The output of each block in the network is concatenated with the output of all previews blocks through the short connections, as presented in Figure 3.24, generating a segmentation mask for each block of the network. All segmentation masks are concatenated with a weighted fusion layer.

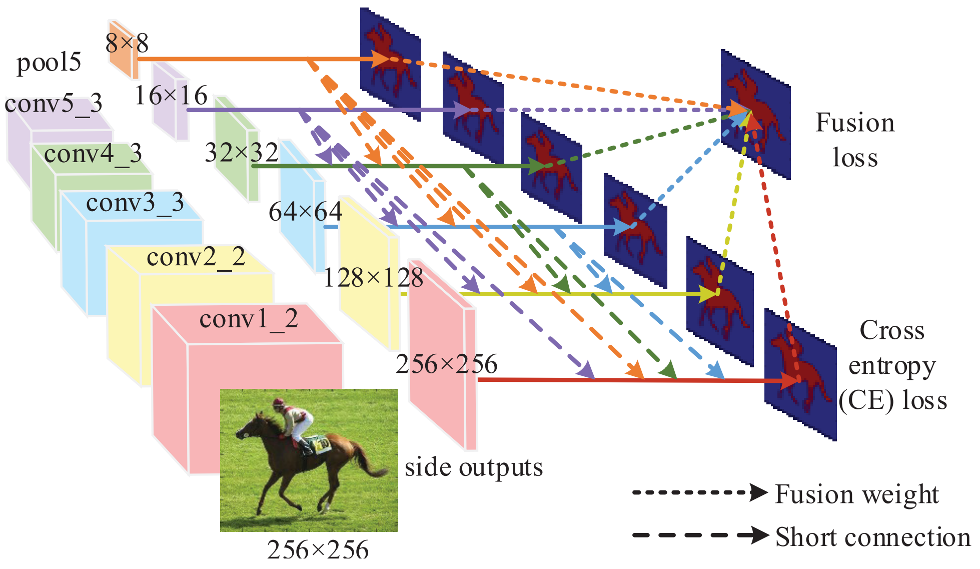


Figura 3.24: The architecture proposed by Hou et al. (2018). The output of each block in the network (represented by a different color) is concatenated with the output of all previews blocks using the HED model. All generated segmentation masks are concatenated with a weighted fusion layer. Source: Hou et al. (2018).

Kruthiventi et al. (2016) proposed a unified framework for salient object detection and eye fixation problems. The proposed architecture is composed of three blocks: one for the saliency segmentation, one for eye fixation prediction and a VGG-16 model shared between both problems. An inception [Szegedy et al. (2015)] module called INCP is linked in the output of the second, fourth, fifth, and sixth max-pooling layers, generating a multiscale representation of the salient objects. A sequence of convolutions is applied to concatenated the INCP outputs, generating a Saliency Map. In the last block of the VGG-16, one last inception module is linked in the network. This last INCP module, followed by a sequence of convolutional layers, is utilized to generate the eye fixation prediction.

Figure 3.25 presents the proposed framework. The grey block in the figure presents the VGG-16 layer shared between the salient object detection and eye fixation problems. The green block presents the INCP modules linked in the max-pooling layers to generate a Saliency Map. The blue block presents the INCP module linked in the last block of the VGG-16 to generate eye fixation prediction. The red block called INCP represents the layers of the inception modules linked in the network.

Tang et al. (2016) also proposed an alteration in the VGG-16 architecture to perform saliency segmentation. The last block of the VGG-16 network (fully connected layers) is removed and, in the remaining blocks, the normal convolutions are replaced by recurrent convolutional layers [Liang and Hu (2015)] to extract contextual information from the images. Convolution and deconvolution layers are linked in the output of each block, as illustrated in Figure 3.26,

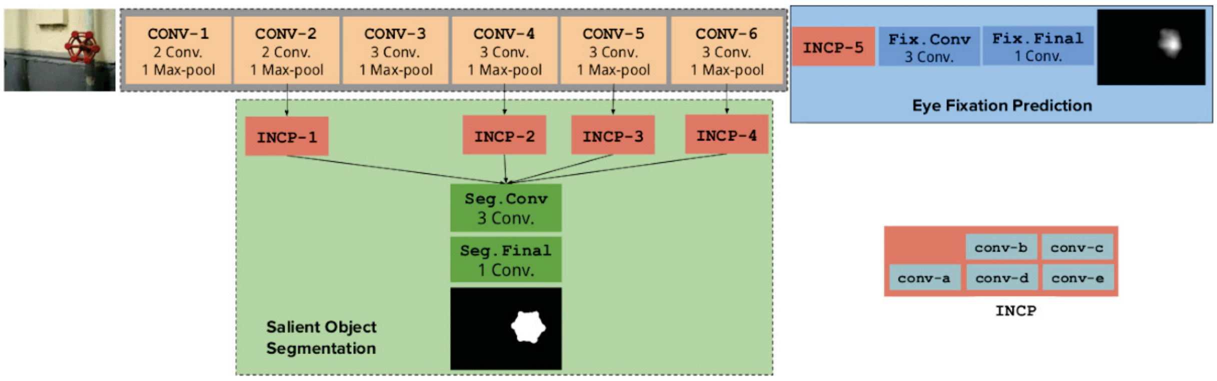


Figura 3.25: The architecture proposed by Kruthiventi et al. (2016). The input image enters in the shared block with the VGG-16 model (the grey block). Then, the network has two outputs: one for saliency segmentation (the green block) and one for eye fixation prediction (the blue block). The red block called INCP represents the layers of the inception modules linked in the network. Source: Kruthiventi et al. (2016).

generating five Saliency Maps that are concatenated and passed by one last convolutional layer to generate the final Saliency Map.

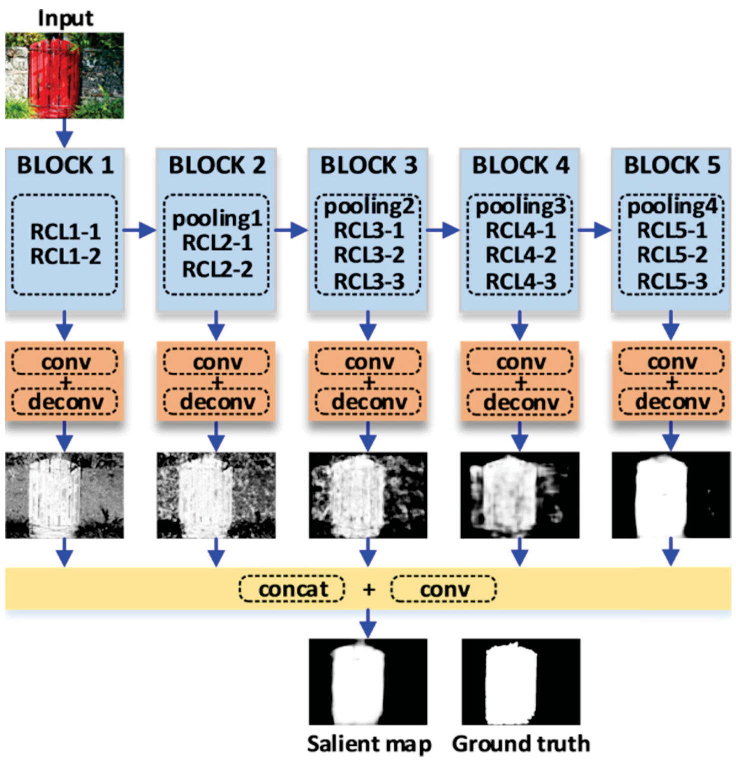


Figura 3.26: The architecture proposed by Tang et al. (2016). The blue blocs are the VGG-16 architecture modified to the SOD problem. The orange blocks are utilized to produce saliency maps, and in the end, all Saliency Maps produced are concatenated to generate the final saliency map. Source: Tang et al. (2016).

Li et al. (2017a) proposed a framework with three steps: estimation of the Saliency Map, detection of salient object contours, and identification of salient object instances. A new network model, called MSRANet, is proposed to produce a Saliency Map and to detect the salient object contours. The MSRANet is composed of three VGG-16 networks with shared weights to segment the input image in three different scales, generating three segmentation masks which are concatenated in the attention module.

Two MSRANets are trained, one for the saliency region detection and one for object contour detection. The result of the saliency region detection feeds the Multiscale Combinatorial Grouping (MCG) [Arbeláez et al. (2014)] algorithm and the MAP-based [Zhang et al. (2016)] subset optimization method to generate a set of object proposals.

The output of the saliency instance segmentation step, generated by fusing the results of the segmentation with the object proposals, is refined with a fully connected CRF model [Krähenbühl and Koltun (2011)]. Figure 3.27 illustrates the proposed framework.

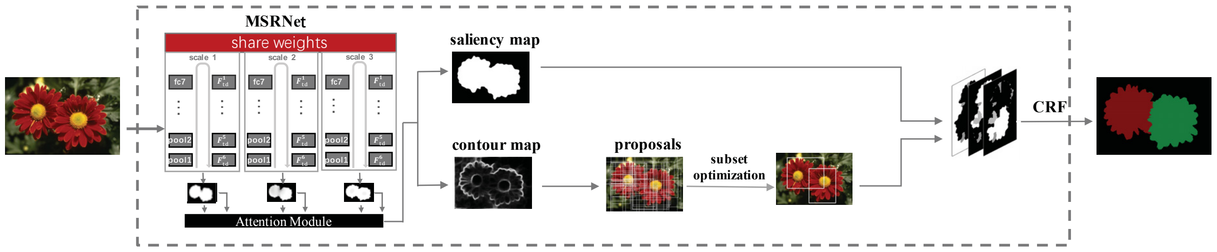


Figura 3.27: The architecture proposed by Li et al. (2017a) with three steps: estimation of the saliency map, detection of salient object contours, and identification of salient object instances. The first two steps are generated by the proposed network (MSRAnet). From the output of the detection of salient object contours step, a set of object proposals are generated. A fully connected CRF model is applied to refine the salient instance segmentation result. Source: Li et al. (2017a).

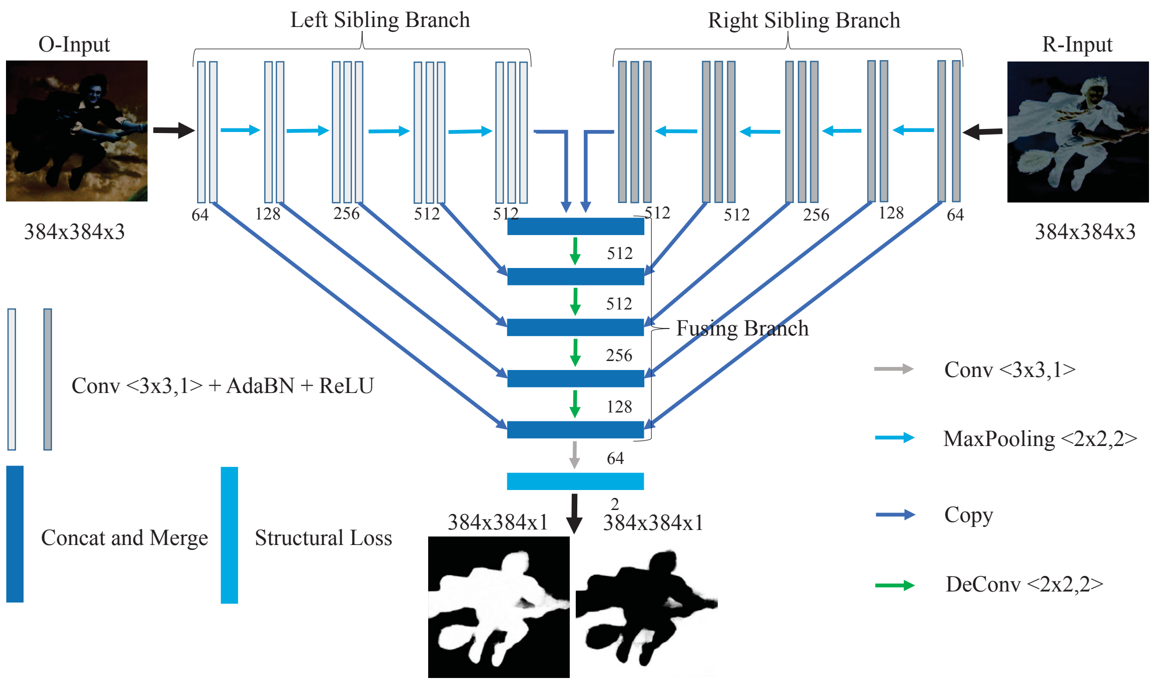


Figura 3.28: The architecture proposed by Zhang et al. (2018). The left sibling branch receives the original input image while the right sibling branch receives the negatived image. A fusing branch is utilized to join the output from the two sibling branches generating the Saliency Map. Source: Zhang et al. (2018).

In order to learn complementary saliency features under the guidance of lossless feature reflection, Zhang et al. (2018) proposed a framework with a symmetrical fully convolutional network. The proposed network has two sibling branches, as presented in Figure 3.28. The left branch receives the original input image, and the right branch receives the input image negatived. The output of the network is generated by a fusion branch, which integrates multi-level features taken from both branches. In Zhang et al. (2019), the authors added a weighted structural loss function to improve the boundaries of the salient objects.

Li et al. (2018) proposed an algorithm to train fully convolutional networks for saliency detection. In the first step of the algorithm, an unsupervised salient object detection model is applied to generate a first Saliency Map for each image in the training dataset. The first Saliency Map, the saliency annotation, and the class distribution are utilized for training a multi-task FCN to generate a pixel-wise segmentation and an image class distribution.

The Saliency Map generated in the first step, the Saliency Map predicted by the network and the class activation map are utilized as the input of a CRF model to refine the final Saliency Map, which is utilized to update the saliency annotation for the next training iteration. The whole framework with all steps is presented in Figure 3.29.

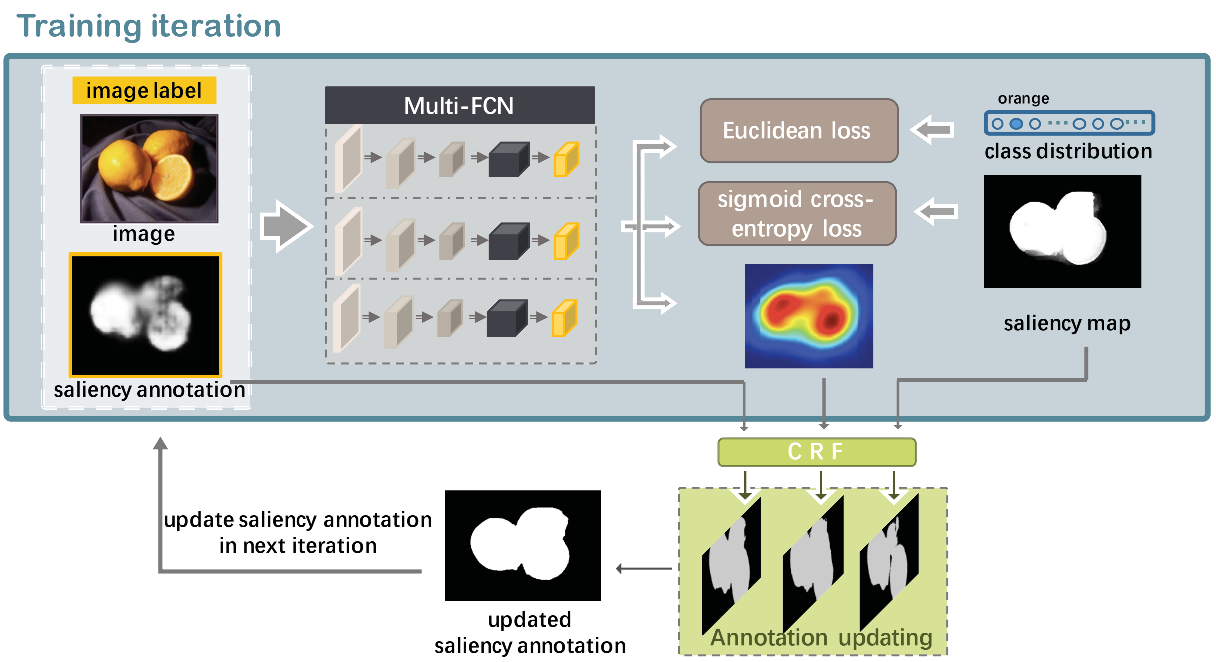


Figura 3.29: The framework proposed by Li et al. (2018). The input image and the saliency annotation enters in the multi-task FCN. The Saliency Map and the class distribution are also utilized for training the network. The first Saliency Map, the Saliency Map predicted by the network and the class activation map are utilized as the input of a CRF model to refine the final Saliency Map which is utilized to update the saliency annotation for the next training iteration. Source: Li et al. (2018).

Nguyen et al. (2019) proposed a framework with two steps to segment salient objects. In the first step, they utilized FCNs to generate a Saliency Map, called explicit Saliency Map, that targets the human common sense of salient objects and focus in the learning of different objects previously defined as salient. In the second step, they utilized an approach based on super-pixels to generate a second Saliency Map, called implicit Saliency Map, that focus in the learning of salient objects not belonging to the list of salient objects defined in the first step. Then, they defined a function that fuses both Saliency Maps. The entire framework is presented in Figure 3.30.

As presented above, FCNs are widely utilized in many different works proposed in the SOD literature and shows promising results when compared with predecessor methods. The proposed works based on FCNs are, in general, well-diversified and has its peculiarities. While some works tried to use different hand-crafted features to refine the Saliency Map produced by the FCN, other works tried to expand the FCN strategy of integrating information from the top layers with the deeper layers to add multiscale information in the Saliency Map.

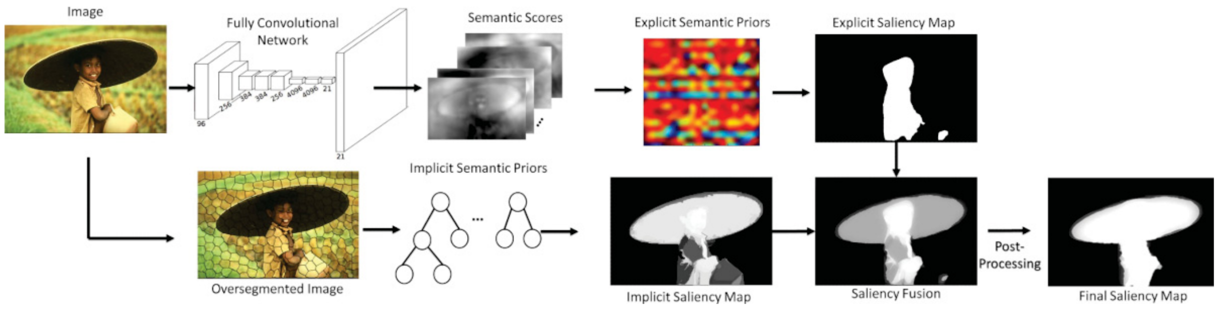


Figura 3.30: In the first step, they utilized FCNs to generate a Saliency Map, called explicit Saliency Map. In the second step, they utilized an approach based on super-pixels to generate a second Saliency Map, called implicit Saliency Map. A saliency function is applied to fuses both Saliency Maps. Source: Nguyen et al. (2019).

When compared with the hand-crafted-based and CNN-based methods, the FCN-based methods are a much more robust approach. However, despite the promising results, the FCN has problems to distinguish salient regions from non-salient regions in some challenging scenarios. Nguyen et al. (2019) fairly recently applied a Mask Region-based Convolutional Neural Network (Mask R-CNN) in the SOD problem to overcome the problems of FCN segmentation. In the first step of the framework proposed by Nguyen et al. (2019), they utilized four variations of FCN and also applied a Mask R-CNN to generate the explicit Saliency Map. However, the authors did not performed an explicit comparison between the FCN and Mask R-CNN. They only compared the entire framework proposed, changing the network utilized to generate the first saliency map. This brief comparison was performed only in one dataset and the entire framework was evaluated only in three datasets.

3.5 CONCLUSION

This chapter reviewed the state-of-art methods in the SOD problem. As presented, the methods proposed in the literature had a significant improvement along the time. The first attempts to find Visual Saliences in images were based on hand-crafted features and started a wave of interest in the field. However, these methods were designed to extract a specific set of features and lacked from generalization capacity.

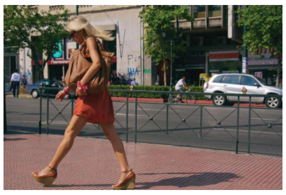
A second attempt to find salience regions in images was based on CNNs, a more robust approach which achieved promising results when compared with the predecessor methods due to the generalization capacity of the CNNs. The weak point of the methods based on CNNs is the fact that the CNN was not designed for segmentation tasks, and pre and post-processing steps were needed to perform segmentation of salient objects, with the super-pixel and patches approaches generating wrong segmentation near the salient objects boundaries.

The most recent methods proposed in the literature are based on FCNs, an evolution of the CNNs designed for Semantic Segmentation tasks. A more detailed discussion about the FCN problems to perform segmentation of salient objects, the Mask R-CNN, and the comparison proposed in this work are presented in the next chapter.

4 PROPOSED APPROACH

This chapter details the Fully Convolutional Network (FCN) and the Mask Region-based Convolutional Neural Network (Mask R-CNN) architectures proposed in this work to perform the segmentation of salient objects. The chapter is divided as follows: Section 4.1.1 presents the FCN, Section 4.1.2 presents the Mask R-CNN, and Section 4.2 ends the chapter.

As presented in Chapter 3, FCNs are widely applied in the recent literature of Salient Object Detection (SOD) to perform segmentation of salient objects. However, besides the promising results achieved by FCNs in the SOD problem, the FCNs have difficulties in distinguishing salient regions from non-salient regions in some challenging scenarios. Figures 4.1a, 4.1b, 4.1c, and 4.1d presents examples of changeling scenarios. In these images, the background has a high number of objects with different colors, intensities, orientations, and illumination. Figures 4.1e, 4.1f, 4.1g, and 4.1h presents the ground truth mask, with the expected salient pixels in white and the expected background pixels in black.


(a) Example of challenging scenario with one salient object.

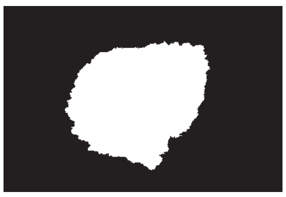
(b) Example of challenging scenario with one salient object.

(c) Example of challenging scenario with one salient object.

(d) Example of challenging scenario with one salient object.

(e) The ground truth.

(f) The ground truth.


(g) The ground truth.

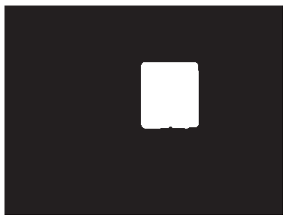
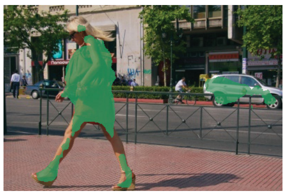
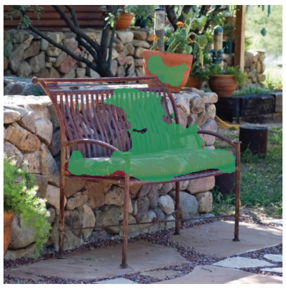

(h) The ground truth.

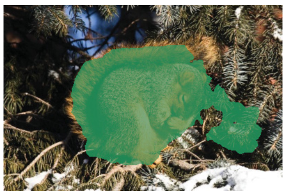
Figura 4.1: Examples of changeling scenarios and the expected ground truth masks. The background has a high number of objects with different colors, intensities, orientations, and illumination. Source: ECSSD Dataset.

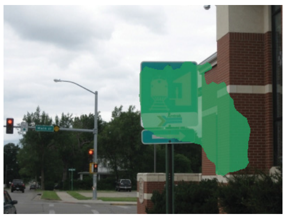
In preliminary experiments performed in this work, the FCNs proposed in this work, found the salient objects, as presented in Figures 4.2a, 4.2b, 4.2c, and 4.2d. In all images, the FCN generated a segmentation mask with most of the salient pixels. However, the FCN wrongly segmented regions in the image not belonging to the saliency regions.

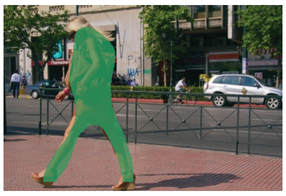
In order to overcome the FCN limitations, Nguyen et al. (2019) fairly recently proposed the use of a Mask R-CNN [He et al. (2017)] approach to SOD. This work proposes the use of two different versions of the Mask R-CNN. Preliminary experiments performed with this two versions of the Mask R-CNN are presented Figures 4.2e, 4.2f, 4.2g, and 4.2h.

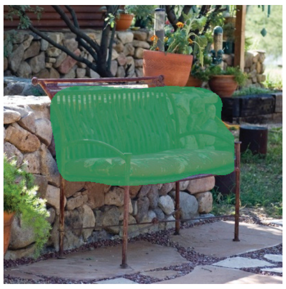

(a) Example of segmentation result generated by a FCN.

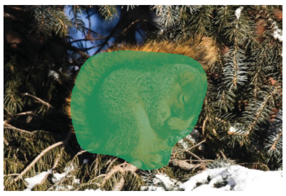

(b) Example of segmentation result generated by a FCN.


(c) Example of segmentation result generated by a FCN.


(d) Example of segmentation result generated by a FCN.


(e) Example of segmentation result generated by a Mask R-CNN.


(f) Example of segmentation result generated by a Mask R-CNN.


(g) Example of segmentation result generated by a Mask R-CNN.

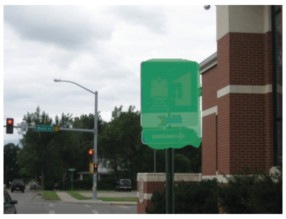

(h) Example of segmentation result generated by a Mask R-CNN.

Figura 4.2: Examples of segmentation result generated by a FCN and a Mask R-CNN in preliminary experiments performed in the present work. The FCN achieved impressive results in all presented images and was able to find all salient objects in the images, generating a segmentation involving most of the salient pixels. However, the FCN wrongly segmented regions in the image not belonging to the saliency regions. The Mask R-CNN improved the segmentation of these salient objects by performing object detection before the segmentation. The segmentation is performed inside the bounding boxes regions, decreasing the rate of false positives. Source: Author.

The Mask R-CNN showed promising segmentation results in the preliminary experiments by generating segmentation masks with fewer background pixels classified as salient. Due to such results, the present work proposes a broad evaluation of the Mask R-CNN in the SOD problem.

The Mask R-CNN is composed of two sub-networks: the Faster Region-based Convolutional Neural Network (Faster R-CNN) [Ren et al. (2017)] and the FCN. The Faster R-CNN is a network designed to object detection tasks and is adapted in the Mask R-CNN with a FCN module to perform segmentation.

The advantage of the Mask R-CNN over a standard FCN is a detection step performed before the segmentation in the Region Proposal Network (RPN), module, generating a set of regions (bounding boxes) with the high probability of being objects. The segmentation is performed inside of the bounding boxes, reducing the false positive error by limiting the segmentation area. Figure 4.3 presents examples of object detection in the RPN. With this step before the segmentation, the Mask R-CNN can outperform the FCN segmentation by limiting the segmentation area to a region close to the salient object.

Taking in mind the shallow evaluation between the FCN and the Mask R-CNN available in the literature, this work proposes a Mask R-CNN architecture, performs a broad evaluation between the two early mentioned segmentation approaches, and aims to investigate the Mask R-CNN effectiveness in the saliency problem by performing a comparison between the Mask R-CNN and the FCN. The next sections presents details about both architectures and the implementation utilized in this work.

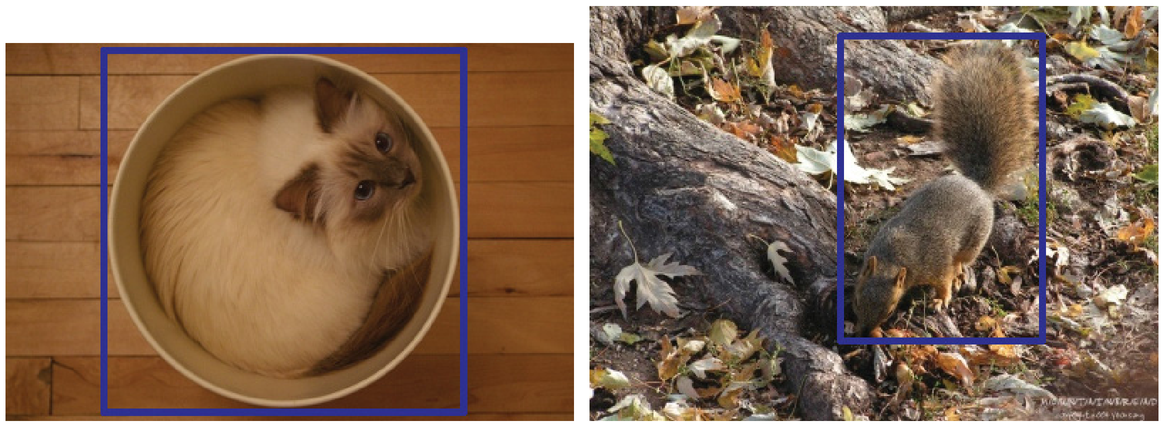

(a) (b)

Figura 4.3: Examples of Mask R-CNN object detection with the RPN. Source: ECSSD dataset.

4.1 PROPOSED NETWORKS AND IMPLEMENTATION DETAILS

Aiming to evaluate and compare the FCNs and Mask R-CNNs, two frameworks containing the implementation of the FCNs and Mask R-CNNs are utilized in the present work: the KittiSeg [Teichmann et al. (2018)] and the Google Object Detection API [Huang et al. (2017)]. Both frameworks are open source and implemented in TensorFlow [Abadi et al. (2015)], an open source machine learning library to build advanced Machine and Deep Learning algorithms.

The KittiSeg framework, winner of the Kitti Road Detection Benchmark 2016 [Fritsch et al. (2013)], was developed to perform segmentation of roads by utilizing FCNs based model. The KittiSeg framework implements the three versions of FCN evaluated in the present work: FCN-Residual Network (ResNet)-50, FCN-ResNet-101, and FCN-VGG-16.

In a preliminary search of FCN implementations over the development of the present work, the KittiSeg stood out based on the following criteria: number of other works based on this implementation [Dung and Anh (2019); Klomann et al. (2017); Severo et al. (2018); S. Bezerra et al. (2018); S. Bezerra and Menotti (2017); S. Bezerra (2018)], ease of use, support from other users (91 issues and 4 pull requests on GitHub [MarvinTeichmann (2019)], site accessed in 05/08/2019), and promising results in preliminary experiments.

The Object Detection API was developed by Google to make it easy to construct, train, and deploy object detection models. The advantages of the Object Detection API over others Mask R-CNN implementation are the number of models for detection and segmentation (roughly 20 detection models and 4 segmentation models, including implementations with the same Convolutional Neural Networks (CNNs) backbones as the KittiSeg: ResNet-50 and ResNet-101).

The Object Detection framework is implemented in the same machine learning (TensorFlow) library as the KittiSeg, and has a significant number of works based on this implementation [Liu and Zhu (2018); Bellotti et al. (2008); Mustamo (2018); Chen et al. (2018)]. Also, there is a large community of users supporting the framework (1243 issues and 368 pull requests on GitHub [Google (2019)], site accessed in 05/08/2019).

4.1.1 Proposed FCNs and implementation details

Figure 4.4 presents the ResNet-50 and ResNet-101 converted in FCN-8 in the KittiSeg implementation. The ResNets are structured by a sequence of four building blocks, with each building block containing a sequence of residual blocks. The difference between each building block is the number of filters in the convolutional layers.

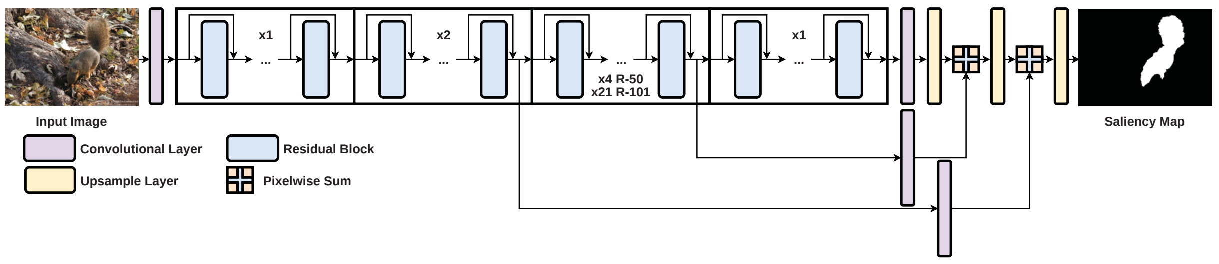


Figura 4.4: The ResNet architectures converted to FCN-8 in the KittiSeg framework. Each residual block (represented in blue) is composed of three convolutional layers in sequence with filters of size 3×3, 1×1, and 3×3, respectively. The input of each residual block is added to its output through the skip connection. R-50 represent ResNet-50 and R-101 represent ResNet-101. The output of the network is the Saliency Map. Source: Author.

The input image goes through the four building blocks of the ResNet, generating a feature map at the end of the last building block. The feature map, which is 1/32 times smaller than the input image, suffers a convolution and a $\times 2$ upsample operation, generating a feature map 1/16 times smaller than the input image.

The output of the third building block enters in a convolutional layer and is added to the feature map. The resulting feature map suffers a $\times 2$ upsample operation and generates a feature map 1/8 times smaller than the input image. The output of the second building block enters in a convolutional layer and is added to the feature map. The resulting feature map suffers a $\times 8$ upsample operation and generates the final segmentation mask.

Figure 4.5 presents the Vgg-16 converted in FCN-8 in the KittiSeg implementation. The blue blocks are convolutional layers, the red blocks are pooling layers, and the yellow blocks are fully connected layers converted in convolutional layers.

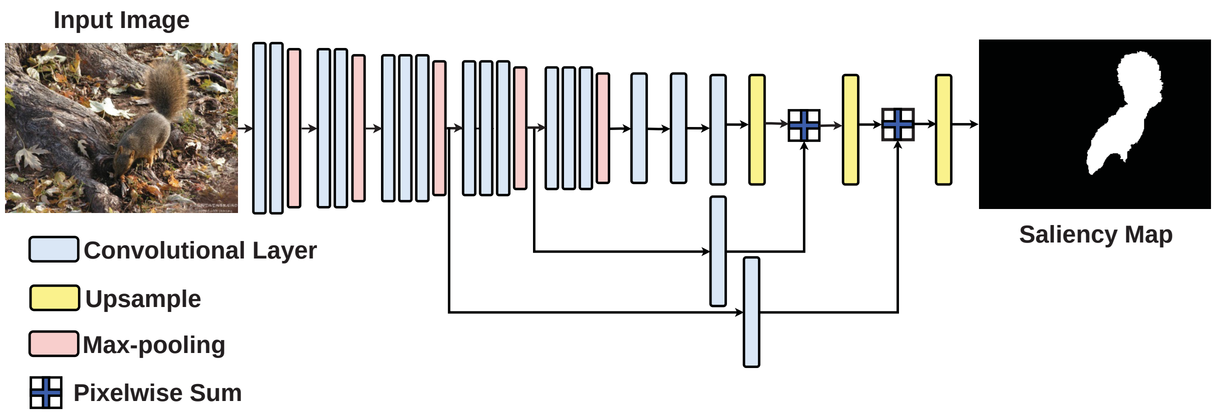


Figura 4.5: The VGG-16 architecture converted in FCN in the KittiSeg framework. The blue blocks are convolutional layers, the red blocks are pooling layers, and the yellow blocks are fully connected layers converted in convolutional layers. Source: Author.

The feature map generated in the fifth pooling layer (last pooling layer), which is 1/32 times smaller than the input image, suffers a convolution and a $\times 2$ upsample operation, generating a feature map 1/16 times smaller than the input image. The output of the fourth pooling layer enters in a convolutional layer and is added to the feature map. The resulting feature map suffers a $\times 2$ upsample operation and generates a feature map 1/8 times smaller than the input image. The output of the third pooling layer enters in a convolutional layer and is added to the feature map.

The resulting feature map suffers a ×8 upsample operation and generates the final segmentation mask.

In Figures 4.4 and 4.6, the white blocks are the building blocks containing a sequence of residual blocks. Each residual block (represented in blue) is composed of three convolutional layers in sequence with filters of size 3×3, 1×1, and 3×3, respectively.

4.1.2 Proposed Mask R-CNNs and implementation details

Figure 4.6 presents the ResNet-50 and ResNet-101 converted in Mask R-CNN in the Object Detection API. The input image goes through the three building blocks of the ResNet, generating a feature map. The feature map enters in the RPN to create a set of proposals regions.

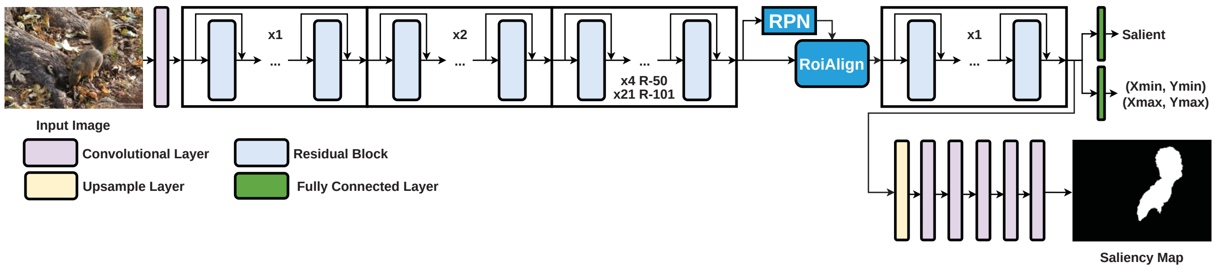


Figura 4.6: The ResNet architectures converted to Mask R-CNN in the Object Detection API. Each residual block (represented in blue) is composed of three convolutional layers in sequence with filters of size 3×3, 1×1, and 3×3, respectively. The input of each residual block is added to its output through the skip connection. R-50 represent ResNet-50 and R-101 represent ResNet-101. Between the third and fourth blocks, the Mask R-CNN has the RPN to generate a set of region proposals and the Region of Interest Alignment (RoIAlign) to perform a max-pooling operation to convert all proposals to a same size feature map. The output of the network is the object classification (Salient), the bounding boxes coordinates, and the Saliency Map. Source: Author.

In the RPN, the feature map inputs a convolutional layer to generate a set of anchors. Each anchor in the feature map enters in two convolutional layers attached in parallel to create a set of bounding boxes and scores to each anchor. A collection of proposals are generated based on the bounding boxes and scores of each anchor.

The proposals are cropped from the feature map (output of the third building block), generating a set of proposal objects. The proposal regions enter in the RoIAlign and a max-pooling layer, generating feature maps of the same size. The feature maps go through a last building block of the ResNet.

The output of the last building block enters in two sequential fully connected layers, attached in parallel, to generate the classifications and the bounding boxes prediction. In parallel, the output of the last building block enters in a upsample layer followed by five convolutional layers to generate the final segmentation.

4.1.3 Transfer Learning

Training a CNN model from scratch is a complicated process, takes a lot of time, and requires thousands of training images. The transfer learning technique enables a CNN being trained in small datasets with a small amount of time. The idea behind the transfer learning is to use a CNN model, already trained in large datasets and reuse the pre-trained weights to fine-tuning the network to specific problems by training the CNN models in small datasets.

Aiming to evaluate the FCNs and Mask R-CNNs models, pre-trained weights from other datasets are utilized in the training step to initialize the network weights. The KittiSeg framework provided the FCNs models with weights initialized in the ImageNet dataset [Deng et al. (2009)].

The ImageNet dataset, designed for visual object recognition software research, contains more than 14 million images divided into more than 20,000 different categories of objects. The Object Detection API provided the Mask R-CNNs models with weights initialized in the COCO dataset [Lin et al. (2014)]. The COCO dataset contains 330 thousand images divided into 80 different categories on objects.

4.2 CONCLUSION

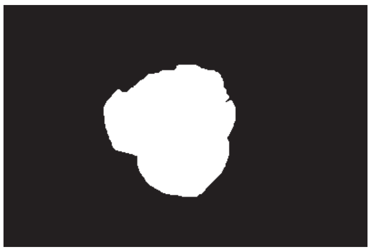
The present chapter explained the proposal of this work of perform a comparison between the Mask R-CNN and the FCN in the Salient Object Detection problem and how the Mask R-CNN can outperform the FCN in the SOD context. This chapter also details the proposed FCN and Mask R-CNN, presenting implementation details, frameworks, and the variations of backbones utilized to compare the FCN and Mask R-CNN. Next chapter presents the datasets and metrics used to evaluate the networks.

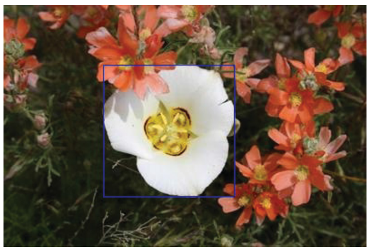
5 DATASETS AND EVALUATION METRICS

The present chapter details the datasets, metrics, and the statistical tests utilized in the present work to compare the Fully Convolutional Network (FCN) and Mask Region-based Convolutional Neural Network (Mask R-CNN). Section 5.1 presents the datasets, Section 5.2 presents the evaluation metrics, Section 5.3 presents the statistical tests, and Section 5.4 ends the chapter.

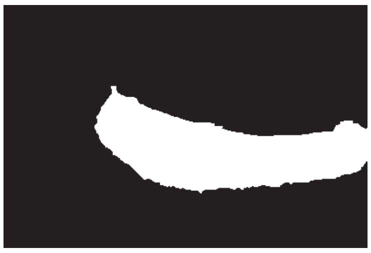
5.1 DATASETS

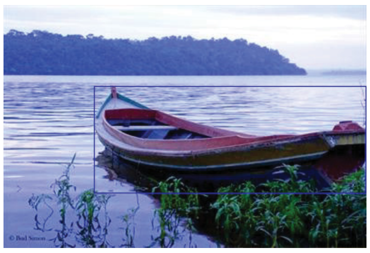
(a) Example of an image containing a salient object.

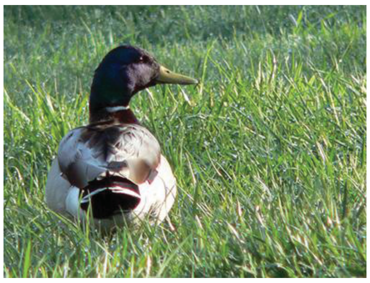

(b) Example of ground truth mask extracted.

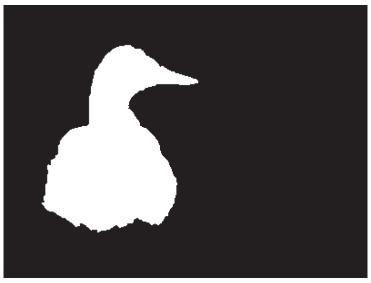

(c) Example of bounding box generated from the ground truth mask.

(d) Example of an image containing a salient object.


(e) Example of ground truth mask extracted.


(f) Example of bounding box generated from the ground truth mask.


(g) Example of an image containing a salient object.


(h) Example of ground truth mask.

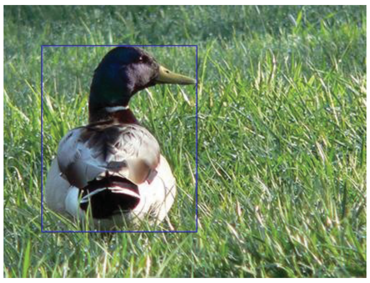

(i) Example of bounding box generated from the ground truth mask.

Figura 5.1: Examples of salient objects extracted from the MSRA10K dataset. Figures 5.1a, 5.1d, and 5.1g presents examples of images containing salient objects, Figures 5.1b, 5.1e, and 5.1h presents examples of the ground truth mask, and Figures 5.1c, 5.1f and 5.1i presents an example of bounding boxes generated from the ground truth masks. Source: Cheng et al. (2015).

Tabela 5.1: Details about the dataset used in this work. The first column presents the dataset name. The second column presents the number of images. Third column presents the number of salient objects in each image. The fourth column presents the width interval of the images. The fifth column presents the height interval of the images. The sixth column presents the average number of pixels. The seventh column presents average number of background pixels. The eighth column presents the average number of pixels in the dataset. The last column presents the proportion of background pixels in relation to salient pixels.

Dataset	**Number of images**	**Number of Salient Objects**	**Width Variation**	**Height Variation**	**Average Number of Salient Pixels**	**Average Number of Background Pixels**	**Average Number of Pixels**	**Proportion of Background Pixels to Salient Pixels**
DUT-OMRON	5166	1 or 2	167 - 401	167 - 401	17,748	101,333	119,082	5.70
ECSSD	1000	1	222 - 400	139 - 400	27,323	87,298	114,621	3.19
HKU-IS	4447	Multiples	100 - 401	100 - 401	21,531	90,590	112,121	4.20
ICOSEG	643	Multiples	284 - 500	200 - 500	38,105	135,700	173,805	3.56
MSRA10K	10000	1 or 2	179 - 400	165 - 400	26,220	91,758	117,978	3.49
PASCALS	2205	Multiples	188 - 681	151 - 511	36,471	138,145	174,617	3.78
SED1	100	1	300 - 300	125 - 465	23,629	63,607	87,237	2.69
SED2	100	2	191 - 300	144 - 300	14,397	52,169	66,567	3.62
THUR	6232	1	300 - 1208	300 - 828	19,098	109,468	128,507	5.73

In order to evaluate the network models, the present work utilized nine datasets of salient objects widely used in the Salient Object Detection (SOD) literature: DUT-OMRON [Yang et al. (2013)], ECSSD [Shi et al. (2016)], HKU-IS [Li and Yu (2015)], ICOSEG [Batra et al. (2010)], MSRA10K [Cheng et al. (2015)], PASCAL-S [Li et al. (2014)], SED1 [Borji et al. (2015)], SED2 [Borji et al. (2015)], and THUR [Cheng et al. (2014)].

The datasets are composed by images containing salient objects with different biases (*e.g.*, number of salient objects, image clutter, center-bias) and the referent ground truth mask with the expected segmentation of the salient objects. The ground truth masks are binary images with the salient regions in white (value 255) and the background with no salient regions in black (value 0).

In order to train the Mask R-CNN, it was necessary to generate the bounding boxes of each salient object in the images. The bounding boxes were generated on top of the ground truth masks utilizing the OpenCV library [Bradski (2000)]. Figures 5.1a, 5.1d, and 5.1g present examples of images containing salient objects, Figures 5.1b, 5.1e, and 5.1h present examples of ground truth mask, and Figures 5.1c, 5.1f and 5.1i present example of bounding boxes generated from the ground truth masks.

Table 5.1 presents details about the number of images, number of salient objects, images sizes, the average number of pixels salient, the average number of background pixels, the average of the total number of pixels, width and height variation, and the proportion of background pixels over salient pixels.

Compared with state-of-art datasets (ImageNet [Deng et al. (2009)] and COCO [Lin et al. (2014)]) utilized in other object classification and detection problems, the saliency datasets are small datasets. The largest saliency dataset found in the present study is the MSRA10K with 10 thousand images.

The images have small dimensions, with the smallest width and height among 100 and 300 pixels, and the highest width and height between 400 and 500 pixels, with the THUR dataset being the dataset with the largest images. Also, in the majority, the datasets utilized have 1 or 2 salient objects in each image.

The last four columns in Table 5.1 present the average number of salient, background, total pixel, and the proportion of background pixels over salient pixels, respectively. In all datasets, the number of pixels labeled as background is significantly high than the number of pixels labeled as salient, *i.e.*, the salient objects represent a small portion of the image, categorizing an unbalanced problem (the categories being classified have a different number of examples).

5.2 EVALUATION METRICS

The architecture models trained in the SOD problem are evaluated and compared through four metrics widely used in the SOD literature: Precision (Equation 5.1), Recall (Equation 5.2), F-measure (Equation 5.3), and Mean Absolute Error (MAE) (Equation 5.4).

The Precision and Recall are metrics applied to evaluate binary classification problems, usually applied in imbalanced problems [Pedregosa et al. (2011)](the number of samples of one category is higher than another category). The SOD problem is a binary segmentation problem with the salient pixels being the positive class and the non-salient pixels being the negative class.

The Precision evaluates the proportion of pixels classified as positive (salient) that are truly positive, and the Recall evaluates the proportion of salient pixels classified as salient [Powers (2008)]. The F-measure is the harmonic average between the Precision and Recall. Equations 5.1 and 5.2 presents the formulas to calculate the Precision and Recall, respectively.

The True Positive is the number of salient pixels classified as salient (correct classification of salient pixels), the False Positive is the amount of non-salient pixels classified as salient (wrong classification of background pixels) and the False Negative is the number of salient pixels in an image classified as background (wrong classification of salient pixels).

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{5.1}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{5.2}$$

$$F - measure = \frac{(1 + \beta^2)Precision * Recall}{\beta^2 Precision + Recall} \tag{5.3}$$

The "β"in the F-measure formula changes the importance of the Precision and Recall. In general, the value 1 is utilized to preserve the equality importance of both. In the SOD literature, "β^2"receives value 0.3 [Borji et al. (2015); Achanta et al. (2009); Cheng et al. (2015); Perazzi et al. (2012)] to increase the Precision importance (i.e. the proportion of pixels classified as positive (salient) that are truly positive).

Attention detection methods aim to locate the position of salients object as accurately as possible, i.e. with high Precision [Zheng et al. (2010)]. As explained in Zheng et al. (2010), the Recall is not much useful for attention detection because high Recall rates are achieved by just selecting a large portion of the image, scarifying the Precision rate.

The MAE verifies how far the predicted result is from the true value and is calculated as the average of the difference between the predicted Saliency Map and the ground truth Saliency Map (both normalized between 0 and 1) [Borji et al. (2015)]. The MAE formula is presented in Equation 5.4. The "W"and "H"are the image width and height, respectively. The "S"is the predicted Saliency Map, and "G"is the ground truth Saliency Map.

$$MAE = \frac{1}{W * H} \sum_{x=1}^{W} \sum_{y=1}^{H} |S(x, y) - G(x, y)| \tag{5.4}$$

5.3 STATISTICAL TESTS

The statistical test aims to provide statistical confidence about the results achieved in the training step. In order to measure the differences in performances of the neural networks evaluated in this work, two statistical tests were performed: Shapiro–Wilk test [SHAPIRO and WILK (1965)] and Wilcoxon signed-rank test [Wilcoxon (1992)].

5.3.1 Shapiro–Wilk Test

The Shapiro–Wilk test is a statistical test to verify if a random distribution, $x_0, x_1, ..., x_n$ comes from a Gaussian distributed population. The Shapiro-Wilk test is calculated as follows:

- Let $x = [x_0, x_1, ..., x_n]$ be a random distribution;
- Let x coming from a normally distributed population be the null hypothesis;
- Let x not coming from a normally distributed population be the alternative hypothesis;

The test statistic W is calculated as follows:

$$W = \frac{(\sum_{i=0}^{n} a_i x_{(i)})^2}{\sum_{i=0}^{n}(x_i - \bar{x})^2} \tag{5.5}$$

Where:

- x_i is a sample at position i;
- $x_{(i)}$ is a sample at position i in the ordered sample values vector (ascending order);
- $\bar{x} = \frac{x_0+x_1+...+x_n}{n}$;
- $(a_0, a_1, ..., a_n)$ are constants calculated with the covariance, variance, and means of a normal distribution sample with size n [Pearson and Hartley (1972)].

5.3.2 Wilcoxon signed-rank test

The Wilcoxon signed-rank test is a non-parametric statistical hypothesis test with the aim to compare two distributions and can be applied to rank the performance difference of two classifiers [Demšar (2006)]. The Wilcoxon signed-rank test was chosen in the present work because it does not require that the distributions compared to be Gaussian distributions. The Wilcoxon test is calculated as follows:

- Let $x = [x_0, x_1, ..., x_n]$ and $y = [y_0, y_1, ..., y_n]$ be two random distribution that will be compared;
- The null hypothesis of the Wilcoxon signed-rank says that the median of the differences between x and y is equal to zero;
- The alternative hypothesis of the Wilcoxon signed-rank says that the median of the differences between x and y is different of zero;
- Compute the absolute difference d between x and y, $d = [|x_0 - y_0|, |x_1 - y_1|, ..., |x_n - y_n|]$;
- Compute the sign s of the difference between x and y, $s = [sgn(x_0 - y_0), sgn(x_1 - y_1), ..., sgn(x_n - y_n)]$;
- Discard from d and s the differences equal to zero;
- Order d in ascending order;
- For each value of d, assign a rank $r = 1, 2, 3, ..., n$. Repeated absolute values receive the average of the ranks they cover;
- Each rank r_i of d_i receives a sign s_i;
- Calculate R^+ as the sum of the positive ranks and R^- as the sum of the negative ranks, both in absolute values;

$$R^+ = |\sum_{i=0}^{n} r_i|, r_i > 0 \tag{5.6}$$

$$R^- = |\sum_{i=0}^{n} r_i|, r_i < 0 \tag{5.7}$$

- Take the smaller value between R^+ and R^-, $T = min(R^+, R^-)$;
- Calculates a z-score as follows:

$$z = \frac{T - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}} \tag{5.8}$$

5.3.3 P-value

In both tests, the statistical significance is measured through the P-value, that contains the probability of achieving the measured statistical value, "W"in the Shapiro-Wilk test and "z"in the Wilcoxon signed-rank test, when the null hypothesis is true. In order to decide to accept or reject the null hypothesis, a significance level $\alpha = 0.05$ is defined [Demšar (2006)]. Then, if the p-value is higher than "α", accept the null hypothesis, or if the P-value is smaller than "α", reject the null hypothesis.

5.3.4 Implementation

This work utilized the implementation of the Scipy [Jones et al. (01)] library in both statistical tests performed. The Scipy is a Python open source library for mathematics, science, and engineering. The Shapiro-Wilk function implemented in the Scipy receives an array "x"and outputs the "W"and the P-value. The Wilcoxon signed-rank test implemented in the Scipy receives two arrays of F-measures "x"and "y", and outputs the $T = min(R^+, R^-)$ and the P-value.

The Scipy implementation of the Wilcoxon signed-rank also support other alternative hypothesis to evaluate if the values in "y"are higher than the values in "x"($x < y$), or the values of "y"are smaller than the values in "x"($x > y$). As presented in Section 5.3.2, the Wilcoxon Signed-rank test performs the operation $x - y$, generating an array of differences "d". Each difference in "d"receives a rank, and each rank receives the signal of the difference. So, if $x < y$, there will be more negative than positive ranks, generating a small "R^+". Meanwhile, if $x > y$, there will be more positive than negative ranks, generating a large "R^+".

When utilizing the alternative hypothesis, the Scipy implementation of the Wilcoxon signed-rank utilizes the "R^+"to calculate the z-score. So, if the values in "x"are smaller than the values in "y", the P-value generated by the Wilcoxon signed-rank function implemented in the Scipy will be small. In the other hand, if the values in "x"are higher than the values in "y", the P-value generated by the Wilcoxon signed-rank function implemented in the Scipy will be high. As the objective of this work is to show that the Mask R-CNN outperforms the FCN in the SOD context, the alternative hypothesis of $x < y$ is utilized.

5.4 CONCLUSION

The present chapter detailed the datasets, metrics, and statistical methods utilized in the present work to compare the FCN and Mask R-CNN. Next chapter presents the experiments realized, the methodology of experiments and the results achieved.

6 EVALUATION AND EXPERIMENTS

Defining the training and test protocol is an essential step to evaluate Machine Learning algorithms. In artificial neural networks evaluation, presented in Chapter 2, this protocol defines the training datasets, the number of images in the training, validation, and test sets. This protocol also defines which images sets are used to training and test the models, and validation protocols to better define networks parameters. The evaluation protocol is essential for a fair comparison between the methods proposed to solve a particular problem.

However, the studies proposed in the SOD literature do not have a standard evaluation protocol defined. Several works follow different protocols, turning it challenging to perform a fair comparison with all studies proposed in the field. After an extensive literature review, this present work follows the protocol most recurrent in the literature.

The evaluation of the FCNs and Mask R-CNNs architectures described in Chapter 4 followed the steps: training, validation, and test. In the training step, the present work follows the protocol proposed by Nie et al. (2018); Tang and Wu (2016); Zhang et al. (2017); Li et al. (2016); Xi et al. (2017); Tang et al. (2016); Kruthiventi et al. (2016); Zhang et al. (2018, 2019).

This protocol uses the MSRA10K [Cheng et al. (2015)] dataset to perform the models training, with the test being performed in the DUT-OMRON [Yang et al. (2013)], ECSSD [Shi et al. (2016)], HKU-IS [Li and Yu (2015)], ICOSEG [Batra et al. (2010)], PASCAL-S [Li et al. (2014)], SED1 [Borji et al. (2015)], SED2 [Borji et al. (2015)], and THUR [Cheng et al. (2014)] datasets, depicting a cross-dataset problem.

Five models were evaluated in the present work: a Residual Network (ResNet)-50 converted in FCN, called FCN-ResNet-50, a ResNet-101 converted in FCN, called FCN-ResNet-101, a ResNet-50 converted in Mask R-CNN, called Mask R-CNN-ResNet-50, ResNet-101 converted in Mask R-CNN, called Mask R-CNN-ResNet-101, and VGG-16 converted in FCN, called FCN-VGG-16.

A validation step is performed before the network's training to find the best number of iterations to train each model and the best learning rate for each model. In the validation, this present work utilizes a K-fold strategy, with "K"equal to five. In the K-fold validation, the training dataset is divided into "K"different sets for training and validation.

Section 6.1 explains in details the validation strategy performed and presents the results obtained. Section 6.2 presents the results achieved in each evaluation dataset in the test step. Section 6.3 presents a statistical analysis performed to compare the FCN and the Mask R-CNN. Finally, Section 6.4 presents a discussion about the results.

6.1 VALIDATION

The first experiment performed in the present study was the models' validation. The validation is an important step in Machine Learning algorithms and is used to adjust the network's parameters (number of training iterations and learning rate) to improve the generalization and preventing the overfitting.

The networks evaluated in this present study were validated through a K-fold strategy with K equal to five. In a K-fold validation with "K"equal to five, the training dataset (MSRA10K) is divided into five sets (folds) of images and trained five times. In the first network training, the training set is composed of four folds (80% of the dataset), with the validation being performed in the fifth fold (20% of the dataset).

In the second network training, the validation fold used in the first training is added to the training set, and one of the folds in the training set (still not used in the validation) is removed from the training set and used to perform the validation in this second network training. The process repeats until all folds being used in training and validation.

Each model evaluated in this present work was validated following the K-fold strategy. For each fold, the networks were trained for 100,000 iterations. For each model evaluated, an initial learning rate was defined through preliminary experiments.

The ResNets 50 and 101 converted in FCN received an initial learning rate of 10^{-6} and the ResNets 50 and 101 converted in Mask R-CNN received an initial learning rate of 10^{-4}. The FCN-VGG-16 received an initial learning rate of 10^{-5} An exponential learning rate decay, similar to the proposed in the literature [Alom et al. (2018)], was utilized to reduce the variance of the learning curves in the last training iterations.

For all networks, the initial learning rate is divided to 10 at the iteration 50,000, generating a learning rate that is divided to 10 again at the iteration 75,000. The higher learning rate in the first iterations force a higher convergence in the firsts training iterations, while the lower learning rate generates in the last iterations generate a fine adjustment in the results.

These learning rates were chosen through preliminary experiments. Initially, the standard learning rates of the KittiSeg and the Object Detection API were utilized to generate preliminary learning curves of the network. Then, different values of learning rates (10^{-3},10^{-4},10^{-5},10^{-6},10^{-7}, and 10^{-8}), higher and lower values , were utilized to generate a stable learning curve to all networks. With the learning rate values utilized, the networks were able to learn very quickly in the first training iterations and converge to an F-measure value with a small standard deviation in the last training iterations. Higher learning rate values generated F-measure values with high standard deviation in each evaluated training iteration, and the networks were not able to converge. With smaller learning rate values, the learning of the networks was too slow, and it would be necessary much more training iterations to the networks being able to converge.

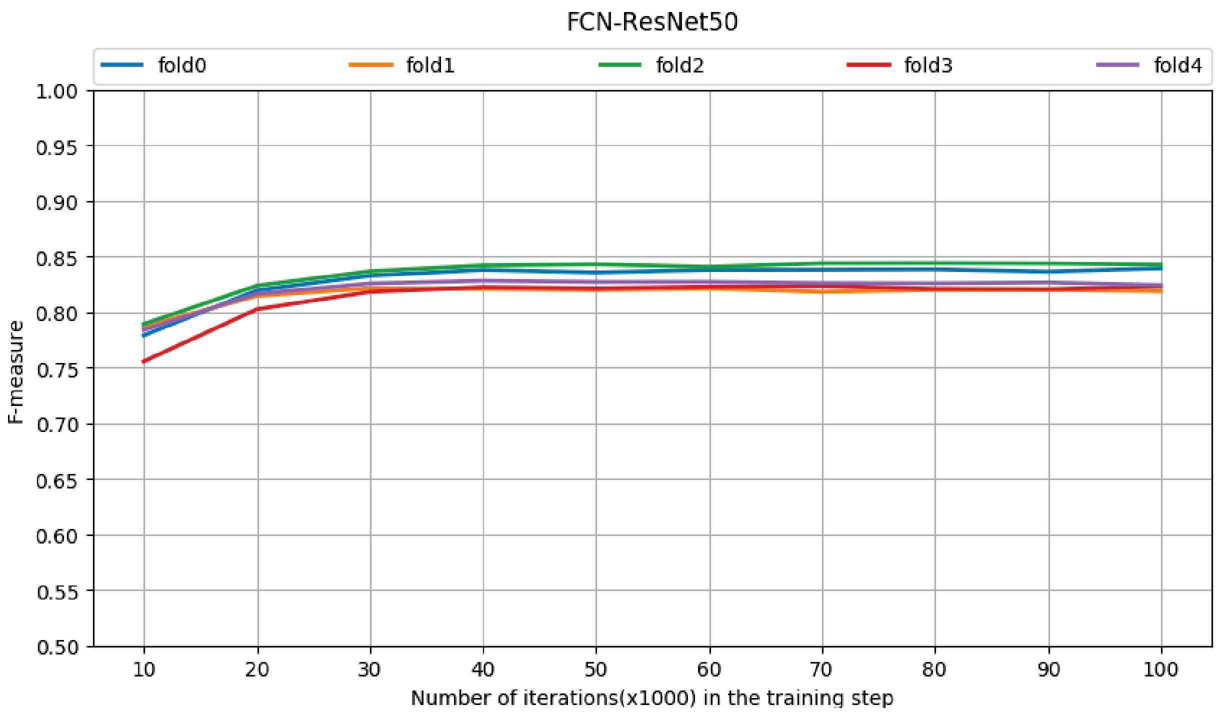


Figura 6.1: The learning curves with the F-measures of each fold generated by the FCN-ResNet-50, which started with an F-measure between 76% and 79% at iteration 10,000 and achieved an F-measure between 82% and 84% in the last iterations. Source: Author.

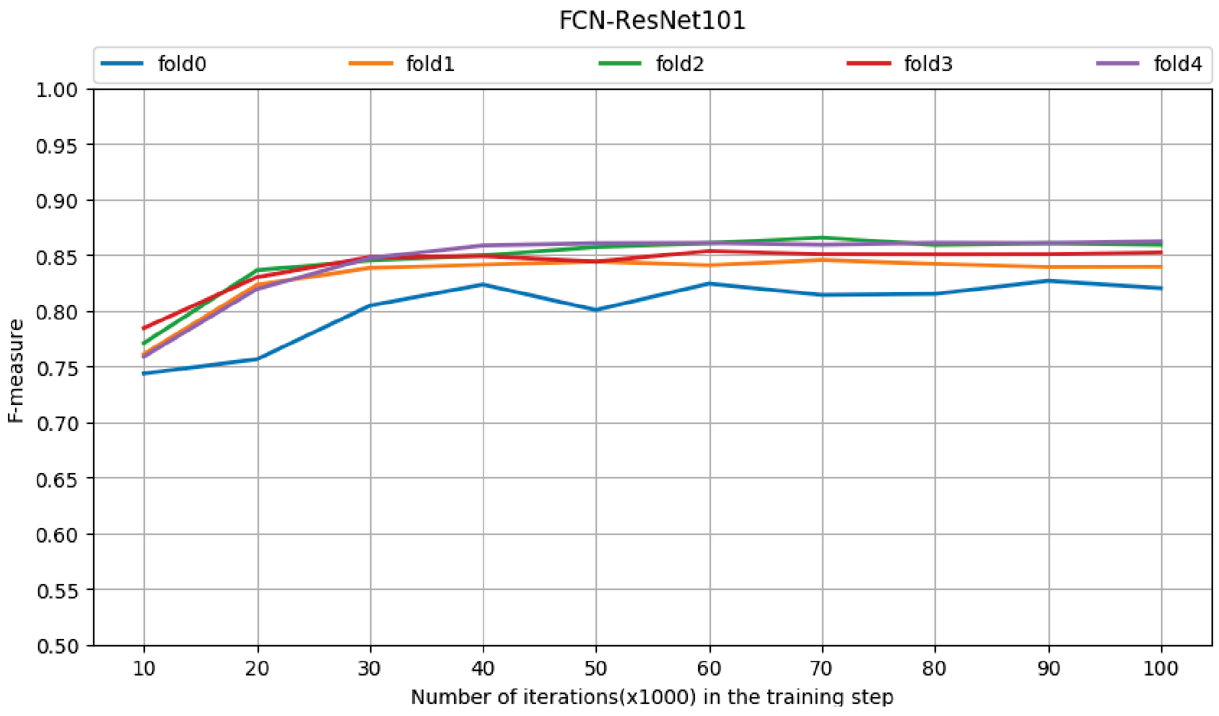


Figura 6.2: The learning curves with the F-measures of each fold generated by the FCN-ResNet-101, which started with an F-measure between 74% and 78% at iteration 10,000 and achieved an F-measure between 82% and 86% in the last iterations. Source: Author.

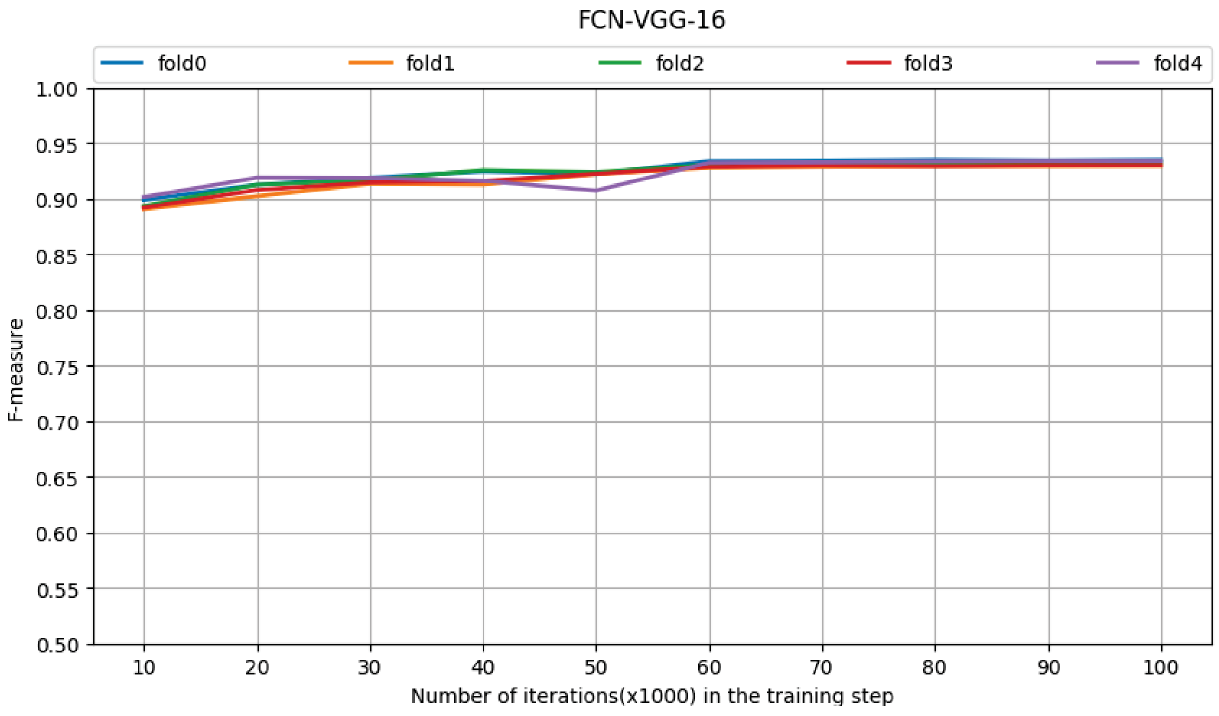


Figura 6.3: The learning curves generated with the F-measures of the folds of the FCN-VGG-16, which started with an F-measure between 89% and 91% at iteration 10,000 and achieved an F-measure between 93% and 94% in the last iterations. Source: Author.

Figure 6.1 presents the learning curves with the F-measures of each fold generated by the FCN-ResNet-50. At iteration 10,000, the fold3 was the worst fold, with an F-measure of 76%, and the fold2 was the best fold, with an F-measure of 79%. At iteration 100,000, the fold1 was the worst fold, with an F-measure of 82%, and the fold2 was the best fold, with an F-measure of 84%.

Figure 6.2 presents the learning curves with the F-measures of each fold generated by the FCN-ResNet-101. At iteration 10,000, the fold0 was the worst fold, with an F-measure of 74%, and the fold3 was the best fold, with an F-measure of 78%. At iteration 100,000, the fold0 was the worst fold, with an F-measure of 82%, and the fold4 was the best fold, with an F-measure of 86%.

Figure 6.3 presents the learning curves with the F-measures of each fold generated by the FCN-VGG-16. At iteration 10,000, the fold3 was the worst fold, with an F-measure of 89%, and the fold4 was the best fold, with an F-measure of 91%. At iteration 100,000, the fold3 was the worst fold, with an F-measure of 93%, and the fold4 was the best fold, with an F-measure o 94%.

In Figure 6.2, it is harder to define the better convergence iteration due to a high variance in the F-measure in the learning curves, mainly in the learning curve of the fold0. In order to define the best iteration to train the models, for each network it was generated the average F-measure curve, and from the average F-measure curve, it was generated an error measurement (1 - F-measure).

Figure 6.4 presents the learning curve with the error of the average F-measure of five folds generated in the validation step of the FCN-ResNet-50, Figure 6.5 presents the learning curve with the error of the average F-measure of five folds generated in the validation step of the FCN-ResNet-101, and Figure 6.6 presents the learning curve with the error of the average F-measure of five folds generated in the validation step of the FCN-VGG-16. As presented in the figures, the FCN-ResNet-50 achieved the lowest error measurement in the iteration 40,000, the FCN-ResNet-101 achieved the lowest error measurement in the iteration 60,000, and the FCN-VGG-16 achieved the lowest error measurement in the iteration 100,000.

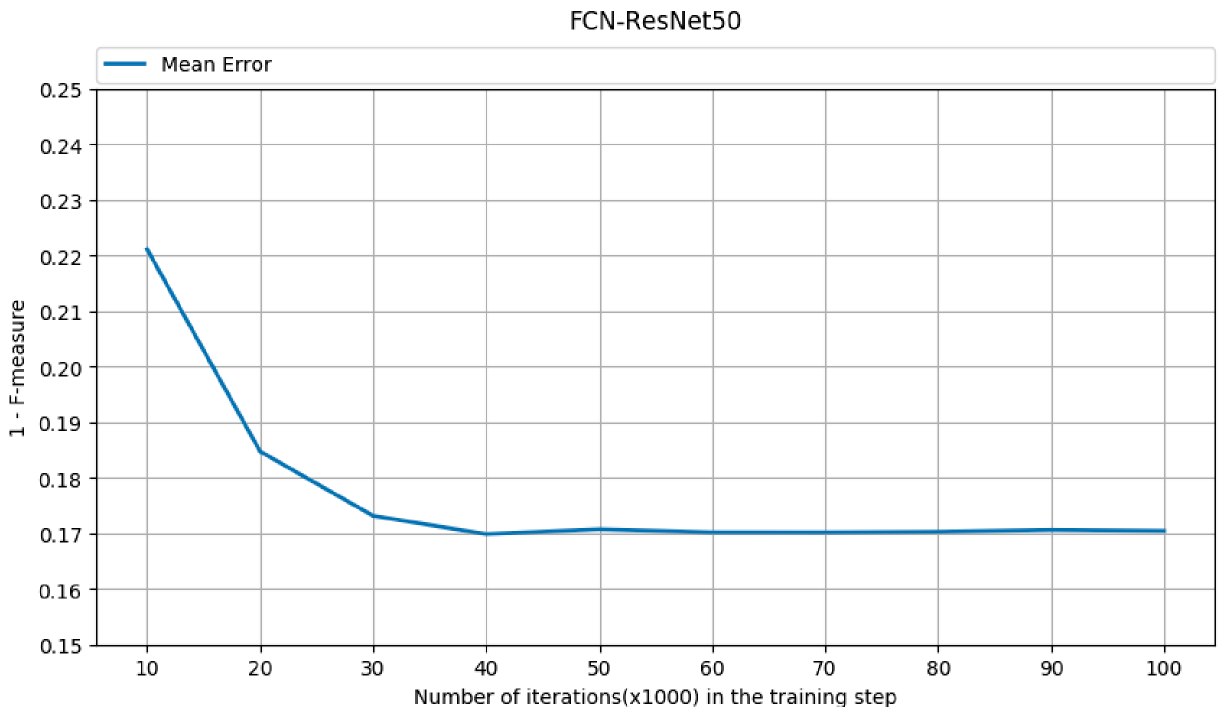


Figura 6.4: The learning curve with the F-measure error generated by the FCN-ResNet-50. As presented in the figure, the iteration 40,000 achieved the lowest error measurement. Source: Author.

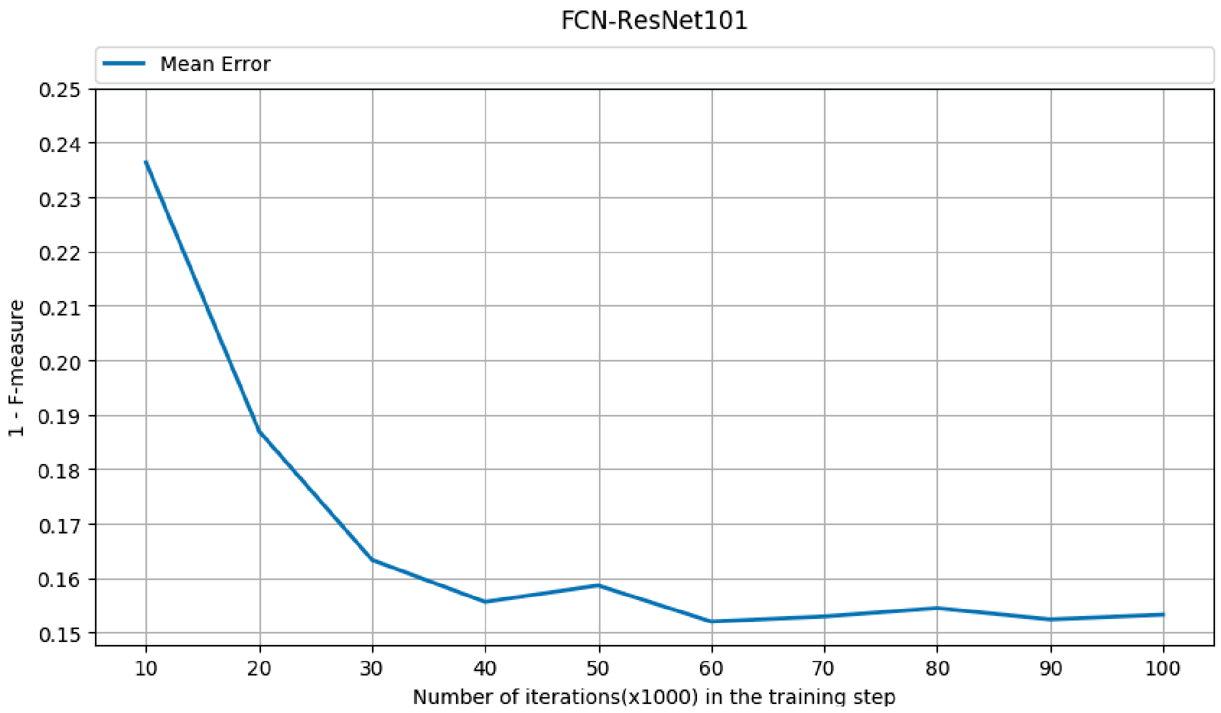


Figura 6.5: The learning curve with the F-measure error generated by the FCN-ResNet-101. As presented in the figure, the iteration 60,000 achieved the lowest error measurement. Source: Author.

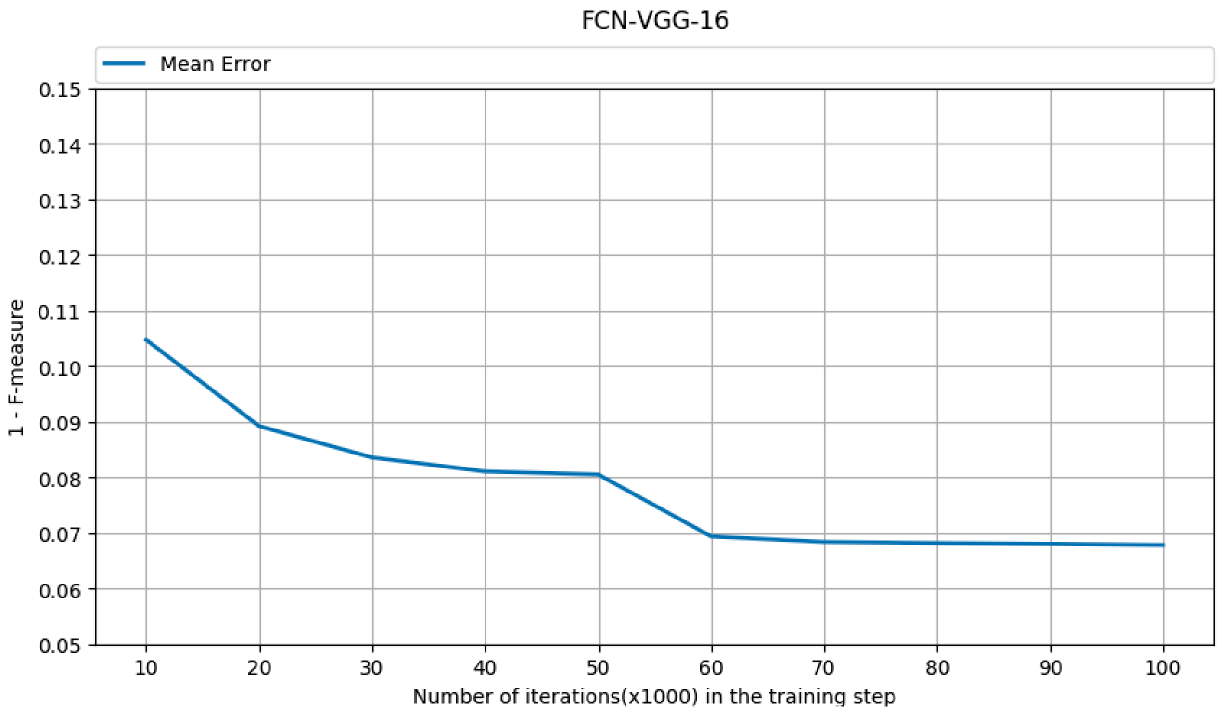


Figura 6.6: The learning curve with the F-measure error generated by the FCN-VGG-16. As presented in the figure, the iteration 100,000 achieved the lowest error measurement. Source: Author.

Figure 6.7 presents the learning curves with the F-measures of each fold generated by the Mask R-CNN-ResNet-50. At iteration 10,000, the fold4 was the worst fold, with an F-measure of 87%, and the fold3 was the best fold, with an F-measure of 88%. At iteration 100,000, all folds generated an F-measure of 89%.

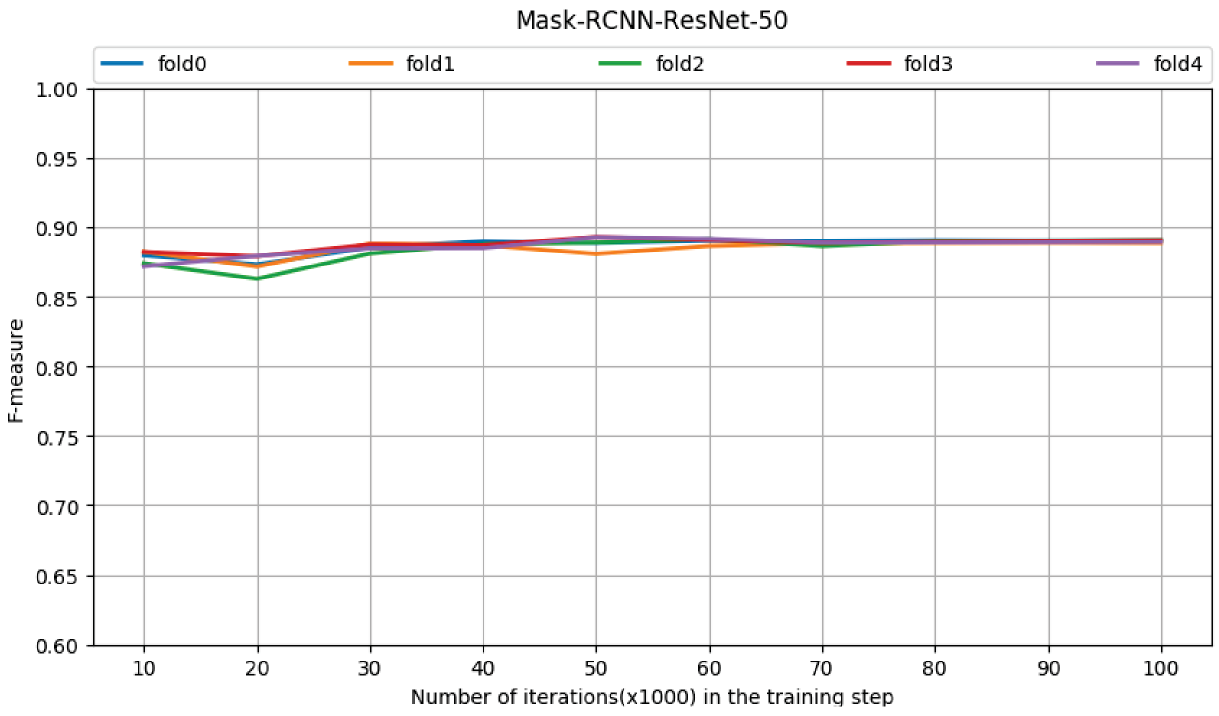


Figura 6.7: The learning curves with the F-measures of each fold generated by the Mask R-CNN-ResNet-50, which started with an F-measure between 87% and 88% at iteration 10,000 and achieved an F-measure of 89% in the last iterations. Source: Author.

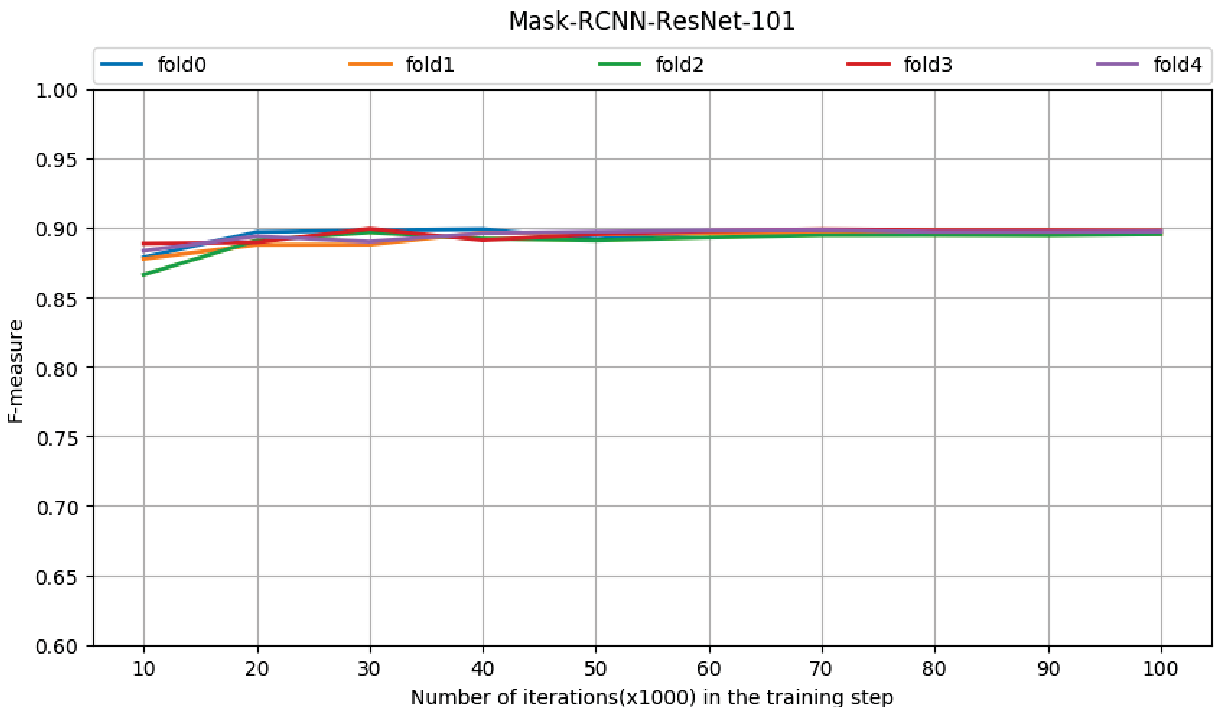


Figura 6.8: The learning curves with the F-measures of each fold generated by the Mask R-CNN-ResNet-101, which started with an F-measure between 86% and 89% at iteration 10,000 and achieved an F-measure of 90% in the last iterations. Source: Author.

Figure 6.8 presents the learning curves with the F-measures of each fold generated by the Mask R-CNN-ResNet-101. At iteration 10,000, the fold2 was the worst fold, with an

F-measure of 86%, and the fold3 was the best fold, with an F-measure of 89%. At iteration 100,000, all folds generated an F-measure of 90%.

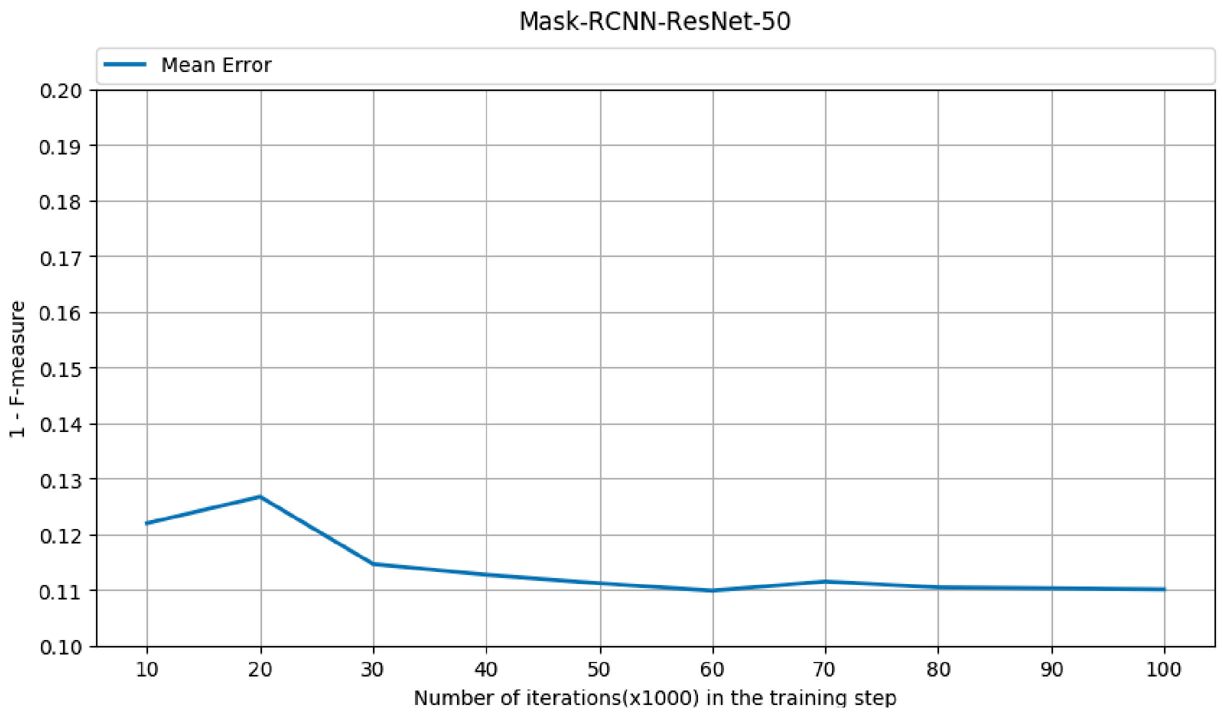


Figura 6.9: The learning curve with the F-measure error generated by the ResNet-50 converted in Mask R-CNN. As presented in the figure, the iteration 60,000 achieved the lowest error measurement. Source: Author.

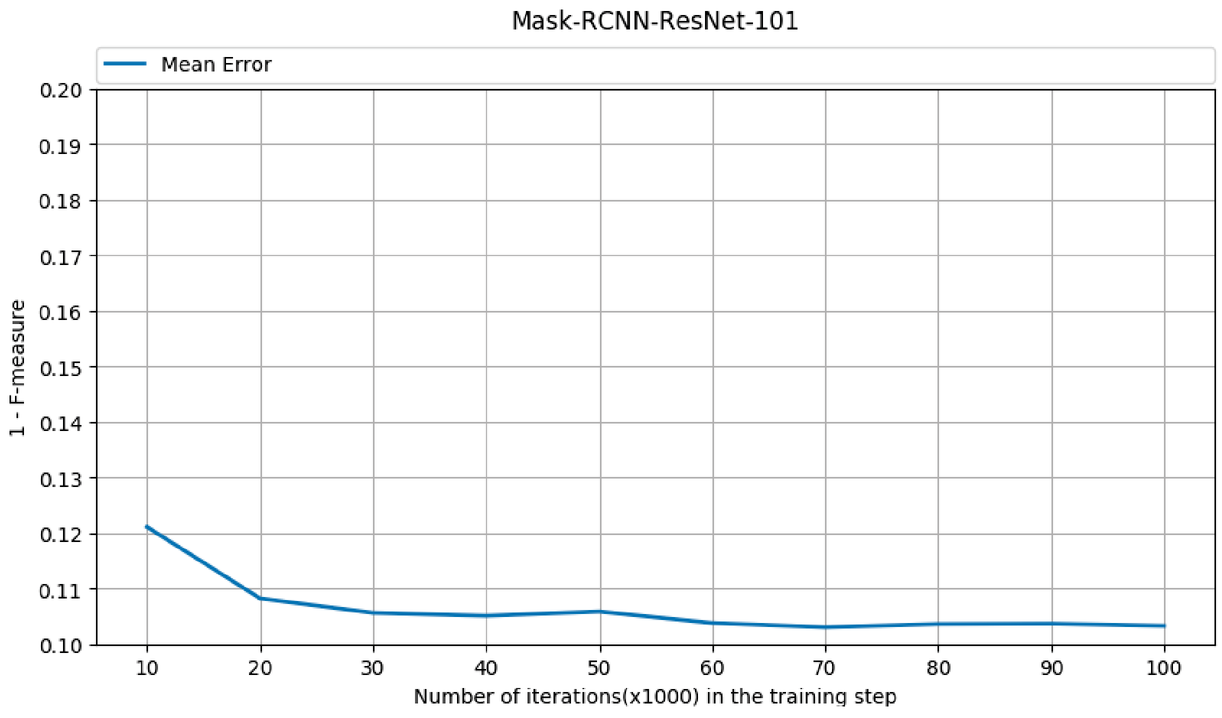


Figura 6.10: The learning curve with the F-measure error generated by the Mask R-CNN-ResNet-101. As presented in the figure, the iteration 70,000 achieved the lowest error measurement. Source: Author.

Figure 6.9 presents the learning curve with the error of the average F-measure of five folds generated in the validation step of the Mask R-CNN-ResNet-50 and figure 6.10 presents the learning curve with the error of the average F-measure of five folds generated in the validation step of the Mask R-CNN-ResNet-101. As presented in the figure, the Mask R-CNN-ResNet-50 achieved the lowest error measurement in the iteration 60,000, and the Mask R-CNN-ResNet-101 achieved the lowest error measurement in the iteration 70,000.

In the validation step, the Mask R-CNNs already showed great improvement when compared with the FCNs. With the ResNet-50 backbone, the Mask R-CNN showed an improvement between 5% and 7% of F-measure over the FCN, and with the ResNet-101, the Mask R-CNN showed an improvement between 4% and 8% of F-measure over the FCN. The FCN with VGG-16 backbone achieved the best results in the validation step. However, the FCN with VGG-16 backbone was the network that needed a higher number of training iterations to converge. The next section presents the tests performed, measuring the Mask R-CNN improvement over the FCN and which network achieved better results in each dataset.

6.2 EVALUATION TESTS

Aiming to evaluate the models in the test datasets, DUT-OMRON, ECSSD, HKU-IS, ICOSEG, PASCAL-S, SED1, SED2, and THUR (datasets presented in Chapter 5.1), each model was trained five times. The entire MSRA10K was utilized as a training set, and each model was training with the number of iterations defined in the validation step. So, the FCN-ResNet-50 was trained for 40,000 iterations, the FCN-ResNet-101 was trained for 60,000 iterations, the FCN-VGG-16 was trained for 100,000 iterations, the Mask R-CNN-ResNet-50 was trained for 60,000 iterations, the Mask R-CNN-ResNet-101 was trained for 70,000 iterations.

Each model was trained five times to generate a fair comparison between the evaluated methods. Besides the initial weights being fixed due to the transfer learning technique, the final weights can be different. It happens because the images are randomly chosen in batches during the training step, and the dropout layers attached in the training step. So, there are training results that can achieve better results than others. The average of multiple training is capable of leveling those training differences, allowing a fairer comparison of the results.

After performing the five training of a specific model, each training was evaluated in the eight test datasets, generating five F-measures, Precisions, Recalls, and MAEs per dataset. The final F-measure, Precision, Recall, and MAE of a test dataset are calculated as the average of the five training.

As presented in Section 6.1, the FCN-VGG-16 achieved the best results in the validation step, showing the VGG-16 as a promising backbone in the SOD problem. However, the Object Detection API did not implement a Mask R-CNN with a VGG-16 backbone. So, in order to simulate the Mask R-CNN with a VGG-16 backbone, it were utilized the bounding boxes generated by the ResNets converted in Mask R-CNNs to evaluate the possibility of improving the FCN with VGG-16 backbone, generating two new solutions named as FCN-VGG-16-R50 and FCN-VGG-16-R101, to represent the FCN-VGG-16 improved by the Mask R-CNN-ResNet-50 bounding boxes and the FCN-VGG-16 improved by the Mask R-CNN-ResNet-101 bounding boxes, respectively. Table 6.1 presents the average F-measure, Precision, Recall, and MAE achieved per dataset and the standard deviation of each model.

Tabela 6.1: Comparison between FCN-ResNet-50, FCN-ResNet-101, FCN-VGG-16, Mask R-CNN-ResNet-50, Mask R-CNN-ResNet-101, FCN-VGG-16-R50, and FCN-VGG-16-R101. The best F-measure and MAE results are highlighted in blue, red, and teal.

Network	Metric	DUT-OMRON		ECSSD		HKU-IS		ICOSEG		PASCAL-S		SED1		SED2		THUR	
		Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
FCN ResNet-50	**F-measure**	0.4999	0.0163	0.7902	0.0102	0.7618	0.0098	0.7488	0.0100	0.7141	0.0082	0.6410	0.0263	0.4083	0.0265	0.6178	0.0107
	Precision	0.6815	0.0172	0.8872	0.0169	0.8702	0.0183	0.8264	0.0170	0.8238	0.0159	0.8909	0.0132	0.8486	0.0293	0.6913	0.0192
	Recall	0.4411	0.0355	0.6975	0.0411	0.6659	0.0377	0.6926	0.0477	0.6466	0.0330	0.4971	0.0372	0.2646	0.0230	0.6499	0.0499
	MAE	0.1067	0.0015	0.0927	0.0054	0.0738	0.0031	**0.0879**	0.0072	0.1047	0.0034	0.1298	0.0090	0.1487	0.0050	0.0999	0.0020
Mask R-CNN ResNet-50	**F-measure**	**0.6964**	0.0023	**0.8383**	0.0022	**0.7979**	0.0017	**0.7495**	0.0019	**0.7850**	0.0017	**0.8861**	0.0051	**0.7873**	0.0081	**0.6680**	0.0019
	Precision	0.7060	0.0034	0.8671	0.0028	0.8299	0.0030	0.7929	0.0016	0.8287	0.0025	0.9052	0.0057	0.8733	0.0037	0.6617	0.0026
	Recall	0.7646	0.0039	0.8137	0.0040	0.7781	0.0054	0.7356	0.0038	0.7722	0.0048	0.8644	0.0044	0.6813	0.0095	0.8014	0.0041
	MAE	**0.0765**	0.0012	**0.0743**	0.0010	**0.0679**	0.0004	0.0927	0.0007	**0.0896**	0.0008	**0.0539**	0.0018	**0.0801**	0.0023	**0.0912**	0.0007
FCN ResNet-101	**F-measure**	0.4424	0.0327	0.7665	0.0176	0.7353	0.0216	0.6820	0.0287	0.6919	0.0156	0.5382	0.0520	0.3150	0.0444	0.5763	0.0191
	Precision	0.6410	0.0484	0.8970	0.0222	0.8658	0.0323	0.7694	0.0612	0.8174	0.0322	0.7879	0.0683	0.6299	0.1203	0.6581	0.0487
	Recall	0.3796	0.0463	0.6592	0.0308	0.6320	0.0358	0.6306	0.0503	0.6135	0.0287	0.4420	0.0640	0.2441	0.0637	0.6088	0.0439
	MAE	0.1078	0.0080	0.0949	0.0044	0.0757	0.0044	0.1080	0.0097	0.1070	0.0042	0.1527	0.0075	0.1862	0.0227	0.1021	0.0077
Mask R-CNN ResNet-101	**F-measure**	**0.7072**	0.0033	**0.8465**	0.0024	**0.8066**	0.0023	**0.7575**	0.0036	**0.7897**	0.0005	**0.9002**	0.0008	**0.7911**	0.0039	**0.6765**	0.0018
	Precision	0.7166	0.0051	0.8712	0.0028	0.8384	0.0042	0.7974	0.0044	0.8347	0.0024	0.9200	0.0020	0.8745	0.0052	0.6650	0.0030
	Recall	0.7739	0.0028	0.8258	0.0049	0.7853	0.0044	0.7442	0.0050	0.7754	0.0036	0.8710	0.0026	0.6844	0.0097	0.8227	0.0045
	MAE	**0.0739**	0.0013	**0.0694**	0.0007	**0.0653**	0.0005	**0.0886**	0.0013	**0.0872**	0.0006	**0.0480**	0.0004	**0.0812**	0.0014	**0.0880**	0.0007
FCN VGG-16	**F-measure**	**0.6952**	0.0036	**0.8638**	0.0031	**0.8411**	0.0034	**0.8065**	0.0032	**0.7981**	0.0022	0.8583	0.0058	**0.5914**	0.0063	0.7225	0.0006
	Precision	0.8162	0.0049	0.9616	0.0016	0.9546	0.0018	0.9113	0.0023	0.9133	0.0028	0.9596	0.0028	0.9531	0.0036	0.7678	0.0021
	Recall	0.5975	0.0095	0.7041	0.0071	0.6717	0.0078	0.6722	0.0065	0.6712	0.0064	0.7114	0.0104	0.3813	0.0080	0.7081	0.0056
	MAE	0.0673	0.0002	**0.0804**	0.0015	**0.0640**	0.0012	**0.0941**	0.0016	**0.0919**	0.0012	0.0791	0.0023	**0.1345**	0.0012	0.0740	0.0004
FCN VGG-16-R50	**F-measure**	0.6841	0.0045	0.8544	0.0033	0.8224	0.0041	0.7883	0.0043	0.7859	0.0036	0.8598	0.0060	0.5571	0.0077	0.7183	0.0005
	Precision	0.8108	0.0039	0.9617	0.0019	0.9538	0.0007	0.9167	0.0019	0.9131	0.0021	0.9630	0.0022	0.9069	0.0124	0.7707	0.0016
	Recall	0.5719	0.0091	0.6881	0.0066	0.6443	0.0078	0.6384	0.0066	0.6489	0.0061	0.7083	0.0104	0.3510	0.0088	0.6857	0.0054
	MAE	0.0663	0.0005	0.0833	0.0014	0.0675	0.0012	0.0977	0.0016	0.0937	0.0012	0.0788	0.0023	0.1380	0.0014	0.0728	0.0003
FCN VGG-16-R101	**F-measure**	0.6876	0.0038	0.8578	0.0027	0.8258	0.0032	0.7866	0.0028	0.7853	0.0030	**0.8609**	0.0060	0.5657	0.0130	**0.7235**	0.0011
	Precision	0.8154	0.0044	0.9630	0.0013	0.9567	0.0012	0.9164	0.0024	0.9118	0.0017	0.9638	0.0020	0.8866	0.0078	0.7754	0.0011
	Recall	0.5747	0.0083	0.6911	0.0063	0.6462	0.0066	0.6371	0.0057	0.6493	0.0052	0.7089	0.0106	0.3541	0.0121	0.6901	0.0051
	MAE	**0.0659**	0.0005	0.0823	0.0014	0.0671	0.0011	0.0984	0.0014	0.0938	0.0011	**0.0784**	0.0024	0.1383	0.0025	**0.0717**	0.0004

With the ResNet-50 backbone, the Mask R-CNN overcame the FCN F-measure in 37.9% in the SED2, 24.51% in the SED1, and 19.65% in the DUT-OMRON datasets. In the worst case, the Mask R-CNN overcame the FCN F-measure in 0.07% in the ICOSEG dataset. With exception of the ICOSEG dataset, the Mask R-CNN achieved lower MAEs when compared with the FCN. Figure 6.11 presents the F-measure difference between the FCN-ResNet-50 and the Mask R-CNN-ResNet-50, with the Mask R-CNN achieving the best F-measure results in all datasets.

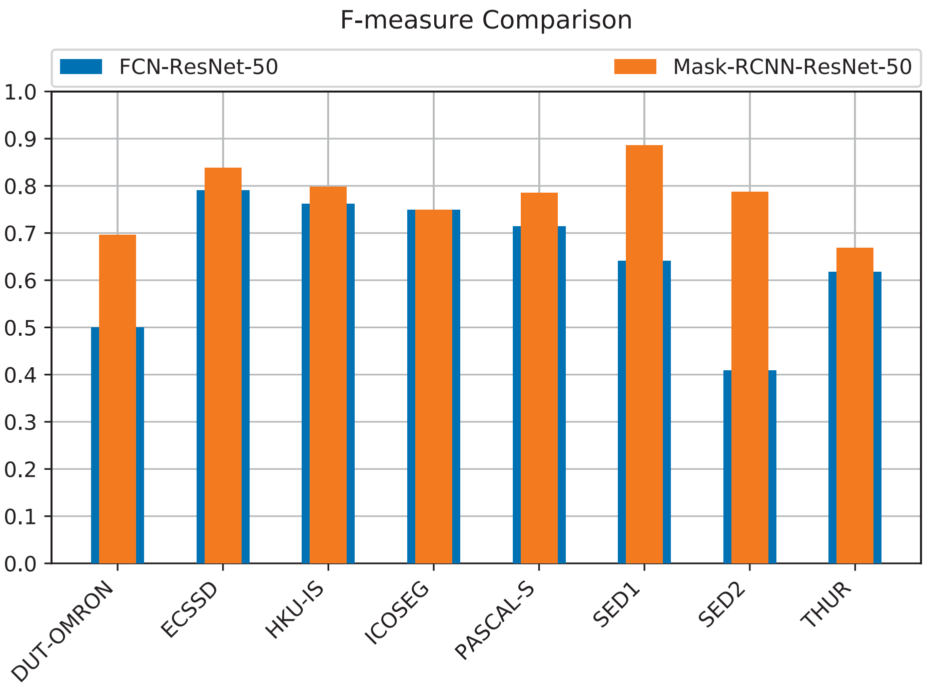


Figura 6.11: F-measure comparison between the FCN and Mask R-CNN utilizing the ResNet-50 backbone, with the Mask R-CNN overcoming the FCN F-measure in all datasets. Source: Author.

The Mask R-CNN with ResNet-50 backbone has not improved the F-measure and MAE results in the ICOSEG dataset due to the high complexity of the ground truth annotations. Figure 6.12 presents examples of ground truth masks from the MSRA10K dataset utilized to train the evaluated models, and Figure 6.13 presents examples of ground truth masks from the ICOSEG dataset. As presented in the images, the annotation in the ICOSEG dataset is more complex, and detailed than the annotation in the MSRA10K dataset, and even the state of art methods have difficulties to generate segmentation masks with this precision level. As presented in Table 6.1, the Mask R-CNN achieved a promising F-measure value, that is, the Mask R-CNN can find the salient objects and generate a segmentation masks that covers most of the salient pixels. However, due to the high detailed annotations, that highlight tiny details from the objects, it is harder to improve the F-measure and MAE metrics.

Figura 6.12: Examples of ground truth masks from the MSRA10K dataset. Source MSRA10K dataset.

Figura 6.13: Examples of ground truth masks from the ICOSEG dataset. Source ICOSEG dataset.

With the ResNet-101 backbone, the Mask R-CNN results were even better when compared with the FCN. The FCN-ResNet-101 achieved worst results when compared with the FCN-ResNet-50. However, when converted in Mask R-CNN, the ResNet-101 achieved better results when compared with the Mask R-CNN-ResNet-50.

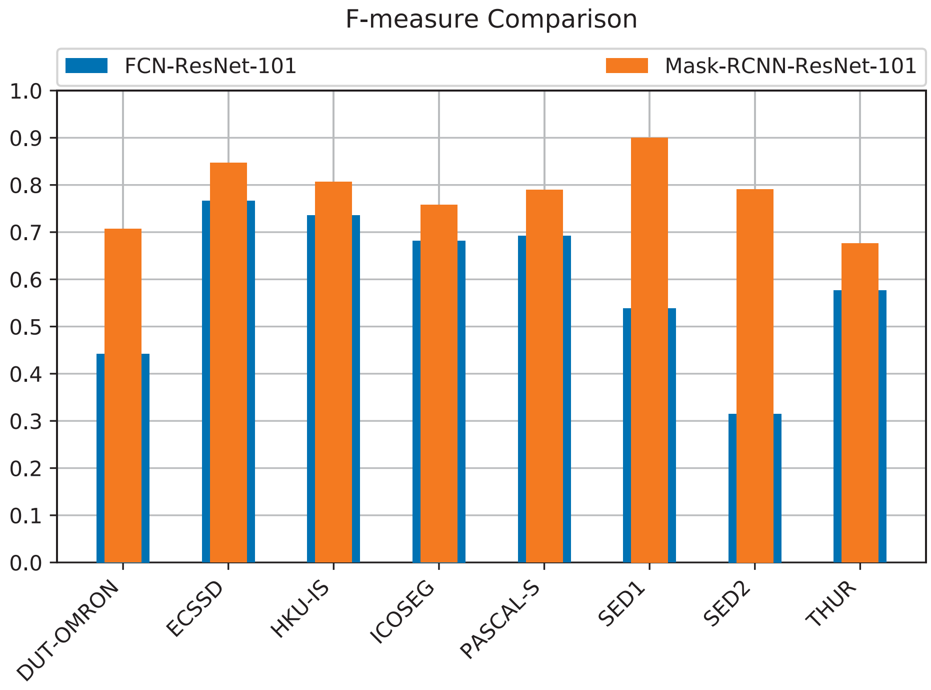


Figura 6.14: F-measure comparison between the FCN and Mask R-CNN utilizing the ResNet-101 backbone, with the Mask R-CNN overcoming the FCN F-measure in all datasets. Source: Author.

In the worst case, the Mask R-CNN overcame the FCN F-measure in 7.13% in the HKU-IS dataset and in the best cases, the Mask R-CNN overcame the FCN F-measure in 47.61%

in the SED2, 36.20% in the SED1, and 26.48% in the DUT-OMRON datasets. Also, the Mask R-CNN achieved lower MAEs in all datasets when compared with the FCN. Figure 6.14 presents the F-measure difference between the FCN-ResNet-101 and the Mask R-CNNResNet-101, with the Mask R-CNN achieving the best F-measure results in all datasets.

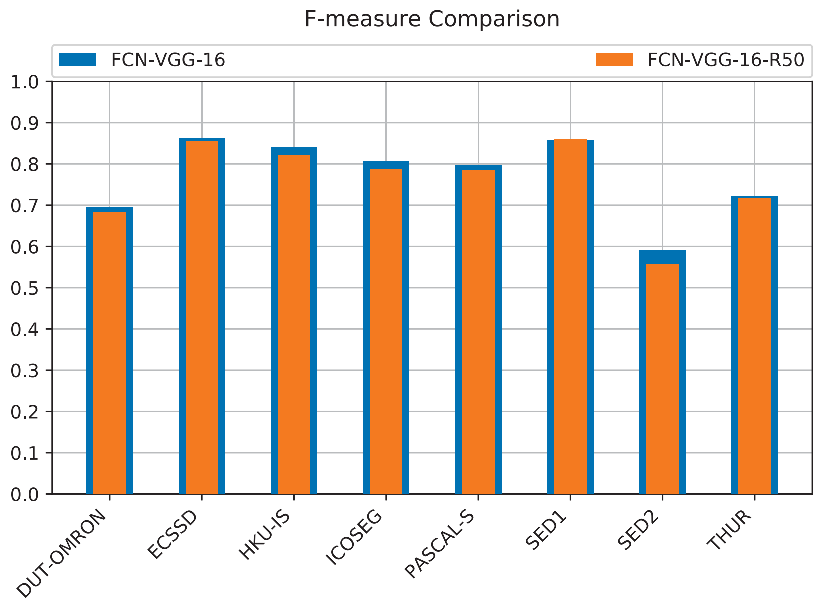


Figura 6.15: F-measure comparison between the FCN with VGG-16 backbone and the FCN with VGG-16 backbone improved by ResNet-50 converted in Mask R-CNN bounding boxes, with the bounding box strategy improving the results in the SED1 dataset. Source: Author.

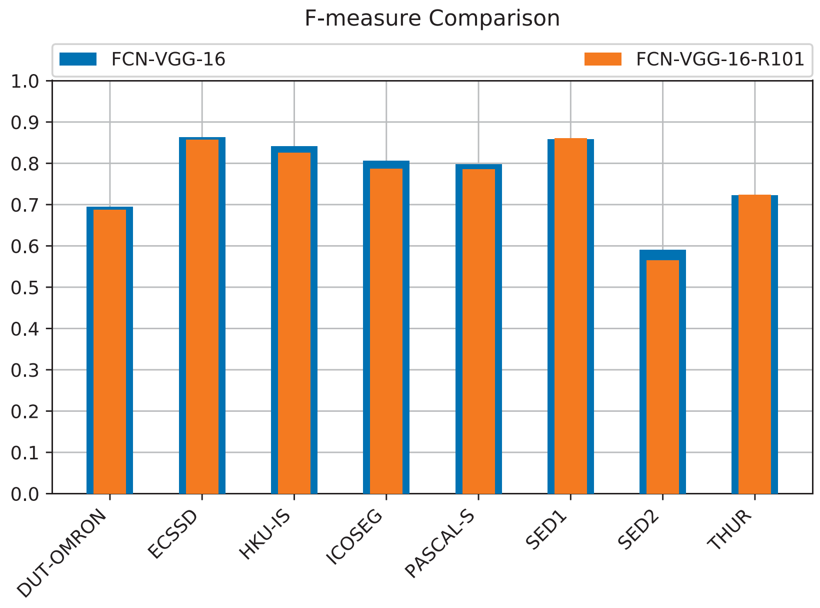


Figura 6.16: F-measure comparison between the FCN with VGG-16 backbone and the FCN with VGG-16 backbone improved by ResNet-101 converted in Mask R-CNN bounding boxes, with the bounding box strategy improving the results in the SED1 and THUR datasets. Source: Author.

The original FCN-VGG-16 results remained as the best results in the majority of the datasets. Applying the bounding boxes generated by the Mask R-CNNs improved the F-measures only of the SED1 and THUR datasets, and improved the MAE of the DUT-OMRON, SED1, and THUR datasets. Figure 6.15 and Figure 6.16 presents the F-measure difference between the FCN-VGG-16 and the FCN-VGG-16-R50, and the F-measure difference between the FCN-VGG-16 and the FCN-VGG-16-R101, respectively.

As presented above, the FCN-VGG-16 achieved the best F-measures among the FCNs and the Mask R-CNN-ResNet-101 achieved the best F-measures among the Mask R-CNNs. Because of that, the two architectures were compared and Figure 6.17 presents the F-measure difference between the FCN-VGG-16 and the Mask R-CNN-ResNet-101. The Mask R-CNN-ResNet-101 outperformed the FCN-VGG-16 F-measures in 1.2% in the DUT-OMRON, in 4.19% in the SED1, and 19.97% in the SED2 datasets. In the other datasets, the Mask R-CNN achieved very close F-measures to the FCN-VGG-16. The Mask R-CNN-ResNet-101 also generated better MAEs in the ECSSD, ICOSEG, PASCAL-S, SED1, and SED2 datasets.

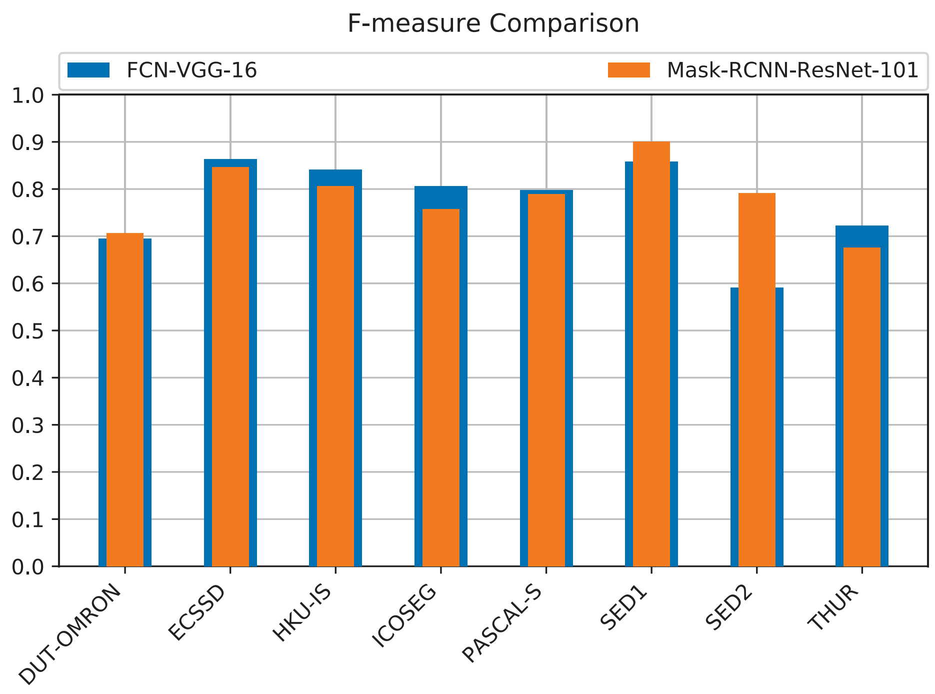
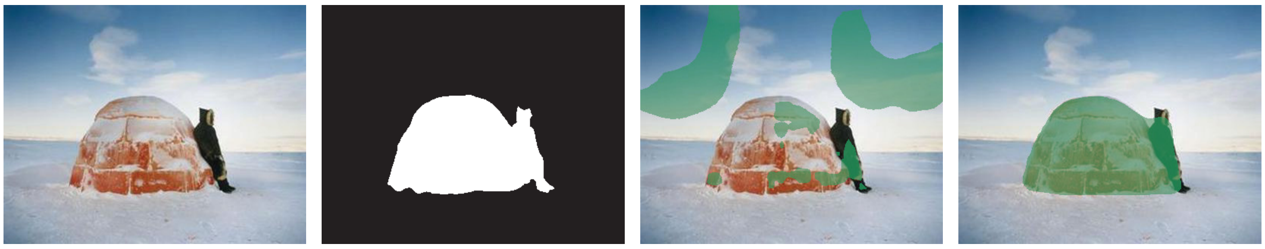


Figura 6.17: F-measure comparison between the FCN with VGG-16 backbone and the Mask R-CNN with ResNet-101 backbone. The Mask R-CNN-ResNet-101 achieved F-measures higher than the FCN-VGG-16 in the DUT-OMRON, SED1, and SED2 datasets. Source: Author.

As presented in Chapter 4, the expected result of the Mask R-CNN in the SOD problem was an improvement in the segmentation Precision. The Mask R-CNN performs the segmentation inside the region proposals, reducing the possibility of performing the segmentation in regions far from the object and decreasing the false positive error. However, as presented in Table 6.1, the higher contribution of the Mask R-CNN was in the Recall, meaning that the Mask R-CNN drastically reduced the false negative error.

Figure 6.19 shows examples of segmentation generated by the FCN-ResNet-50, and examples of segmentation generated by the Mask R-CNN-ResNet-50 and Figure 6.18 shows examples of segmentation generated by the FCN-ResNet-101 and examples of segmentation generated by the Mask R-CNN-ResNet-101. As presented, with both backbones, the Mask R-CNN significantly decreased the false negative error when compared with the FCN, and also, as presented in 6.18, the Mask R-CNN also decreased the false positive error when compared with the FCN.



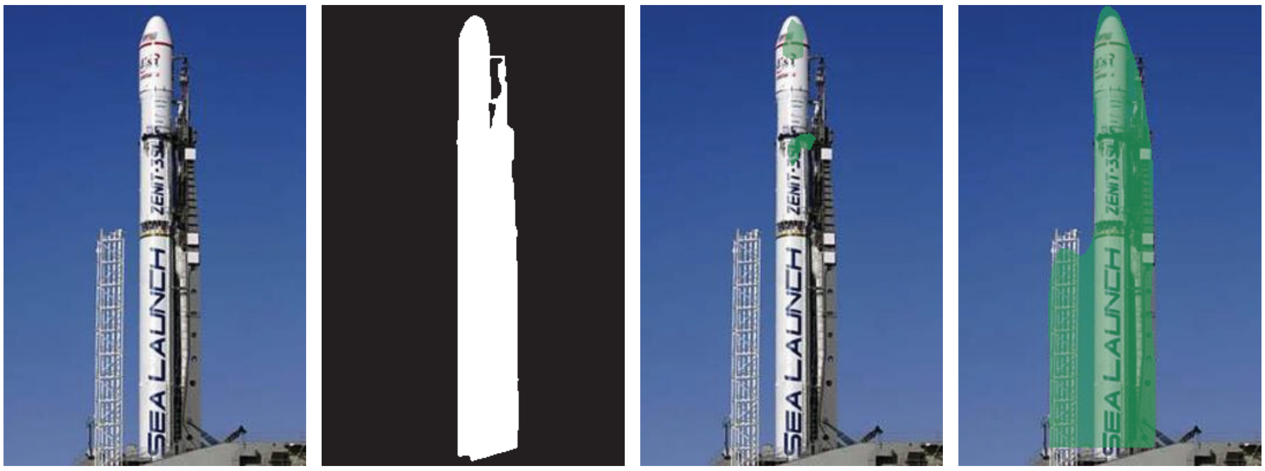
(a) Example of an image containing one salient object. Source: DUT-OMRON dataset.

(b) The respective Saliency Map (the ground truth). Source: DUT-OMRON dataset.

(c) The Saliency Map generated by the FCN with ResNet-101 backbone (in teal). Source: Author.

(d) The Saliency Map generated by the Mask R-CNN with ResNet-101 backbone (in teal). Source: Author.

Figura 6.18: Comparison between FCN-ResNet-101 and Mask R-CNN-ResNet-101 segmentation.



(a) Example of an image containing one salient object. Source: DUT-OMRON dataset.

(b) The respective Saliency Map (the ground truth). Source: DUT-OMRON dataset.

(c) The Saliency Map generated by the FCN with ResNet-50 backbone (in teal). Source: Author.

(d) The Saliency Map generated by the Mask R-CNN with ResNet-50 backbone (in teal). Source: Author.

Figura 6.19: Comparison between FCN-ResNet-50 and Mask R-CNN-ResNet-50 segmentation.

6.3 STATISTICAL ANALYSIS

In order to perform the statistical analysis, the average F-measure is utilized as input of the statistical tests performed. The F-measure is the harmonic average between the Precision and Recall and encodes information from false positive and false negative errors. Both implementations in the Scipy library receive arrays of F-measures calculated with the mean F-measure per image. As explained in Section 6.2, each model was trained five times and tested five times, generating a total of five F-measures for each test image. So, for each test dataset, it was generated an array with the size equal to the number of images in that dataset. Each position of that array represents an image in the dataset and contains the mean F-measure of five training for that image.

The first statistical test performed in this work was the Shapiro-Wilk test to verify if the F-measure results achieved in each evaluated method follows a normal distribution or not. The input of the Shapiro-Wilk test implemented in the Scipy is an array x with the F-measure distributions generated by the evaluated methods. The W and the P-value are outputs generated by the Shapiro-Wilk test implemented in the Scipy library and are presented in Table 6.2.

Tabela 6.2: The W and the P-value achieved in the Shapiro-Wilk test. All P-values achieved results smaller than the significance level ($\alpha = 0.05$), and the null hypothesis can be rejected for all evaluated methods in all datasets, i.e., the distributions are not normally distributed.

Method	**DUT-OMRON**		**ECSSD**		**HKU-IS**		**ICOSEG**		**PASCAL-S**		**SED1**		**SED2**		**THUR**	
	W	P	W	P	W	P	W	P	W	P	W	P	W	P	W	P
FCN ResNet-50	0.9134	0.0	0.7630	1.3132e-35	0.7686	0.0	0.8007	2.2967e-27	0.7964	0.0	0.8524	1.4578e-08	0.8802	1.8518e-07	0.9013	0.0
FCN ResNet-50	0.8921	0.0	0.7674	2.3507e-35	0.7887	0.0	0.8366	3.8872e-25	0.8101	4.2038e-45	0.8802	1.8510e-07	0.8278	1.9586e-09	0.9114	0.0
FCN VGG-16	0.7993	0.0	0.6933	3.7423e-39	0.7056	0.0	0.7792	1.4826e-28	0.7447	0.0	0.7275	2.4810e-12	0.8423	6.2718e-09	0.7986	0.0
FCN VGG-16-R-101	0.7866	0.0	0.6660	2.2900e-40	0.7155	0.0	0.7948	1.0618e-27	0.7337	0.0	0.7348	3.7938e-12	0.8615	3.2501e-08	0.7801	0.0
FCN VGG-16-R-101	0.7831	0.0	0.6694	3.2093e-40	0.7129	0.0	0.7921	7.4400e-28	0.7345	0.0	0.7379	4.5390e-12	0.8578	2.3483e-08	0.7739	0.0
Mask R-CNN ResNet-50	0.7955	0.0	0.7517	3.1033e-36	0.8004	0.0	0.8309	1.6196e-25	0.7169	0.0	0.6352	2.1169e-14	0.8559	1.9873e-08	0.8321	0.0
Mask R-CNN ResNet-101	0.7881	0.0	0.7407	8.0308e-37	0.7977	0.0	0.8290	1.2193e-25	0.7126	0.0	0.7554	1.3087e-11	0.7686	3.0011e-11	0.8241	0.0

Tabela 6.3: Wilcoxon signed-rank test results. The Mask R-CNNs with ResNets backbones achieved P-values smaller than the significance level. However, using the bounding boxes generated by the Mask R-CNN didn't improve the FCN-VGG-16 results, generating P-values higher than the significance level.

Method	**Method Compared**	**DUT-OMRON**	**ECSSD**	**HKU-IS**	**ICOSEG**	**PASCAL-S**	**SED1**	**SED2**	**THUR**
		P	P	P	P	P	P	P	P
FCN-ResNet-50	**Mask R-CNN-ResNet-50**	0.0	1.1480e-09	3.9601e-15	0.9999	8.2508e-51	1.1158e-14	1.2057e-17	3.4194e-84
FCN-ResNet-101	**Mask R-CNN-ResNet-101**	0.0	1.6570e-11	1.9459e-31	3.6840e-11	4.1217e-64	8.3135e-15	5.5738e-18	1.1402e-139
FCN-VGG-16	**FCN-VGG-16-R50**	1.0	1.0	1.0	1.0	1.0	0.9995	0.9999	0.9999
FCN-VGG-16	**FCN-VGG-16-R101**	1.0	1.0	1.0	1.0	1.0	0.9999	0.9999	0.9314
FCN-VGG-16	**Mask R-CNN-ResNet-101**	0.9999	0.9999	1.0	1.0	1.0	0.0002	1.6499e-11	1.0

The null hypothesis of the Shapiro-Wilk test says that the input distribution follows a normal distribution. A significance level $\alpha = 0.05$ is defined [Demšar (2006)]. Then, if the P-value is higher than α, accept the null hypothesis, or if the P-value is smaller than α, reject the null hypothesis. As shown in Table 6.2, all P-values achieved results smaller than the significance level, and the null hypothesis can be rejected for all evaluated methods in all datasets, i.e., the distributions are not normally distributed.

After performing the Shapiro-Wilk test and proof that the distributions do not follow a normal distribution, the Wilcoxon signed-rank test is performed to compare the methods. The Wilcoxon signed-rank test implemented in the Scipy library receives two arrays of F-measures x and y. The F-measures generated by the FCN are represented as x and the F-measures generated by the Mask R-CNN are represented as y.

The Wilcoxon signed-rank test is applied in this work to evaluate if $x < y$. The null hypothesis of the Wilcoxon signed-rank says that the median of the differences between x and y is equal to zero, which means that the Mask R-CNN did not outperforms the FCN. In the alternative hypothesis, the values in x are smaller than the values in y, generating small P-values and the null hypothesis can be rejected, which means that the Mask R-CNN outperformed the FCN.

Table 6.3 presents the P-values generated by the Wilcoxon signed-rank test. A significance level $\alpha = 0.05$ is also defined. If the P-value is higher than α means that the values in x are similar or higher than the values in y, and the null hypothesis is accepted. Otherwise, if the P-value is smaller than α means that the values in x are smaller than the values in y and the null hypothesis is rejected.

The FCN-ResNet-50, when compared with the Mask R-CNN-ResNet-50, achieved P-values smaller than the significance level ($\alpha = 0.05$) in all datasets, except in the ICOSED. So, with the exception of the ICOSEG dataset, the null hypothesis can be rejected in all datasets, and the Mask R-CNN-ResNet-50 can be considered statistically better for the SOD problem than the FCN-ResNet-50.

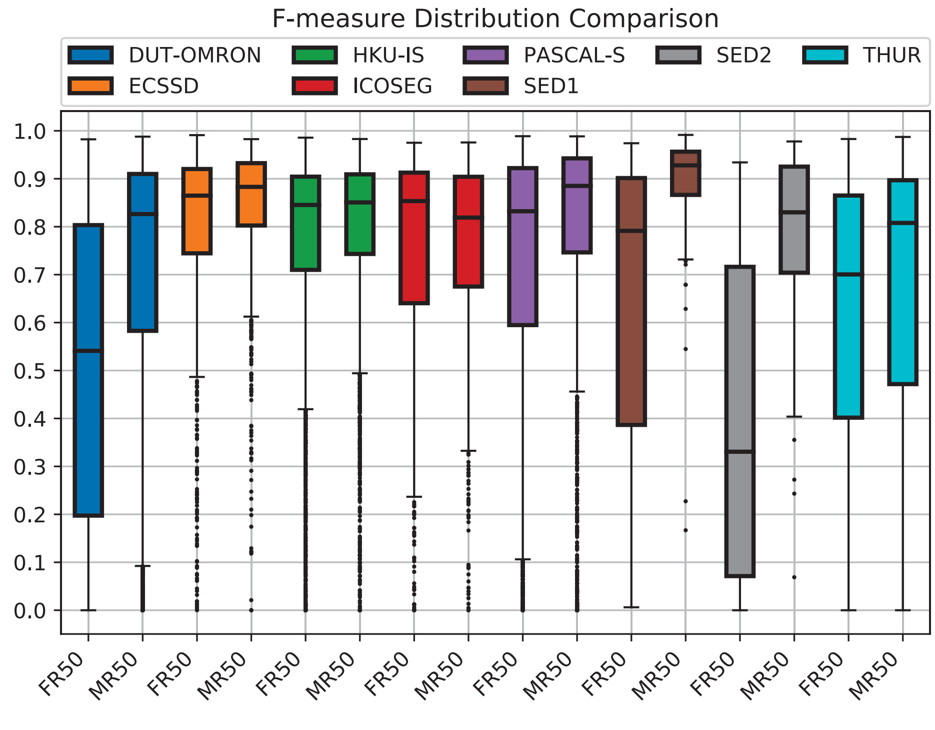


Figura 6.20: Box-plot comparison between the FCN-ResNet-50 (FR50) and the Mask R-CNN-ResNet-50 (MR50). Source: Author.

Figure 6.20 presents the box-plot comparison between the FCN-ResNet-50 and the Mask R-CNN-ResNet-50. The box-plot shows that both networks, FCN and Mask R-CNN, achieved F-measures very close to zero, and also achieved F-measures very close to one. However, the medians of the F-measures in the Mask R-CNN results are superior to the medians of the F-measures in the FCN results.

With the exception of the ICOSEG dataset, the Mask R-CNN achieved superior F-measure medians in all datasets. Also, the Mask R-CNN generated smaller box-plots, indicating that the Mask R-CNN is more robust to the different difficulties of each image of the dataset and generates F-measure closer to the median, while the F-measures generated by the FCN are more scattered. The Mask R-CNN also generated more outliers than the FCN.

The FCN-ResNet-101, when compared with the Mask R-CNN-ResNet-101, achieved P-values smaller than the significance level ($\alpha = 0.05$) in all datasets. So, the null hypothesis can be rejected in all datasets, and the Mask R-CNN-ResNet-101 can be considered statistically better for the SOD problem than the FCN-ResNet-101.

Figure 6.21 presents the box-plot comparison between the FCN-ResNet-101 and the Mask R-CNN-ResNet-101. Similar to the ResNet-50 backbone, the medians of the F-measures in the Mask R-CNN results are superior to the medians of the F-measures in the FCN results. The Mask R-CNN achieved superior F-measure medians and generated smaller box-plots in all datasets. The Mask R-CNN also generated more outliers than the FCN.

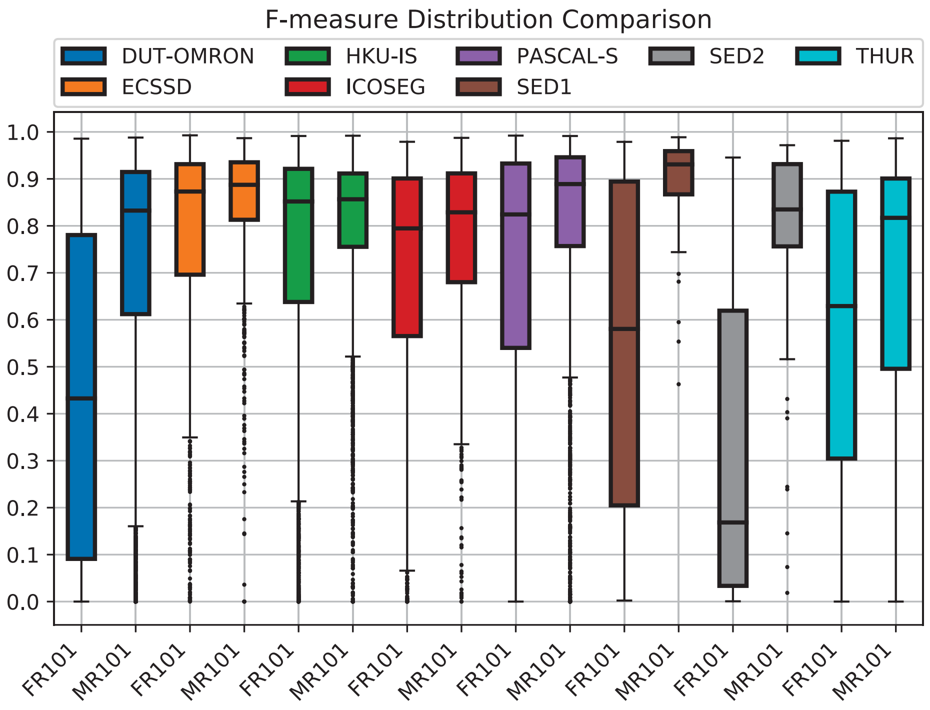


Figura 6.21: Box-plot comparison between the FCN-ResNet-101 (FR101) and the Mask R-CNN-ResNet-101 (MR101). Source: Author.

The FCN-VGG-16, when compared with the FCN-VGG-16-R50 and the FCN-VGG-16-R101, achieved P-values higher than the significance level ($\alpha = 0.05$) in all datasets, and the null hypothesis cannot be rejected. That is, using the bounding boxes generated by the Mask R-CNN has not improved the FCN-VGG-16 results.

Figure 6.22 and Figure 6.23 present the box-plots comparison between the FCN-VGG-16 and the FCN-VGG-16-R50 and the FCN-VGG-16-R101. As expected, applying the bounding

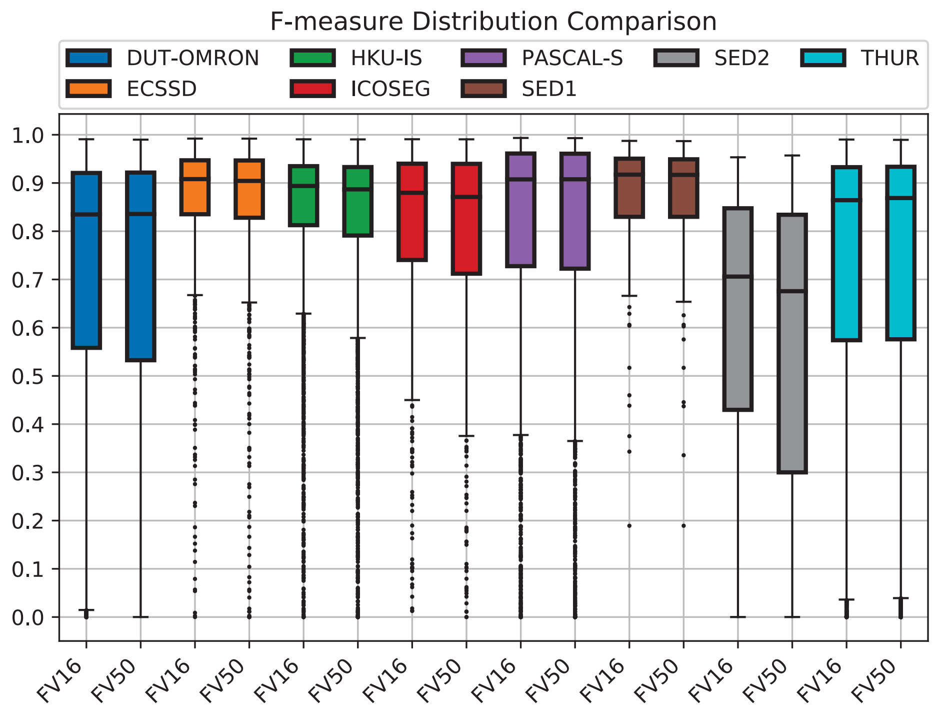


Figura 6.22: Box-plot comparison between the FCN-VGG-16 (FV16) and the FCN-VGG-16-R50 (FV50). Source: Author.

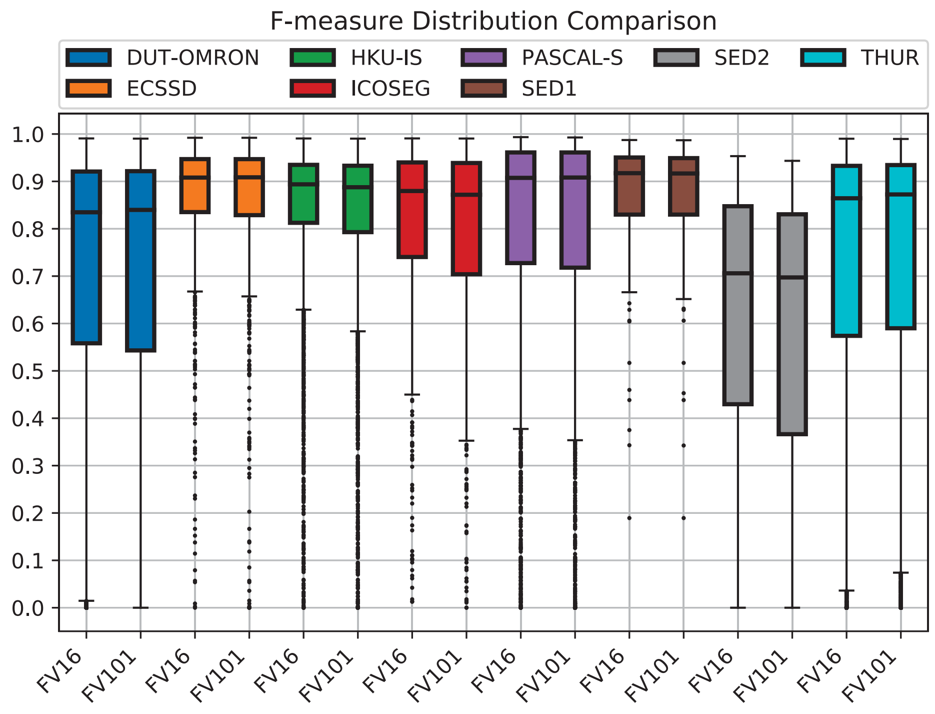


Figura 6.23: Box-plot comparison between the FCN-VGG-16 (FV16) and the FCN-VGG-16-R101 (FV101). Source: Author.

boxes generated by the Mask R-CNN has not generated muny variations in the F-measure distributions of the FCN-VGG-16, and the box-plot generated by the FCN-VGG-16 improved with the bounding boxes are very similar with the original. The higher differences are in the size of the box-plots. The bounding box strategy worsened some results of the original FCN-VGG-16, making the first quartile and the minimum F-measures of the bounding box strategy a little bit lower when compared with the original.

As the FCN-VGG-16 achieved the best results among the FCNs and the Mask R-CNN-ResNet-101 achieved the best results among the Mask R-CNNs, they were also compared with the Wilcoxon signed-rank test. The Mask R-CNN-ResNet-101 generated F-measures statistically higher than the FCN-VGG-16 only in the SED1 and SED2 datasets, with P-values smaller than the significance level ($\alpha = 0.05$). So, the null hypothesis can be rejected only in these two datasets.

Figure 6.24 presents the box-plot comparison between the FCN-VGG-16 and the Mask R-CNN-ResNet-101. The Mask R-CNN achieved higher medians in the SED1 and SED2 datasets. However, the FCN achieved higher F-measures in the other datasets. In the DUT-OMRON, PASCAL-S, SED1, and SED2 datasets, the Mask R-CNN generated smaller box-plots than the FCN, and as a consequence of this distribution with F-measures closer to the median, the Mask R-CNN also generated more outliers than the FCN. In the ECSSD, HKU-IS, ICOSEG, and THUR datasets, the FCN generated smaller box-plots and more outliers than the Mask R-CNN.

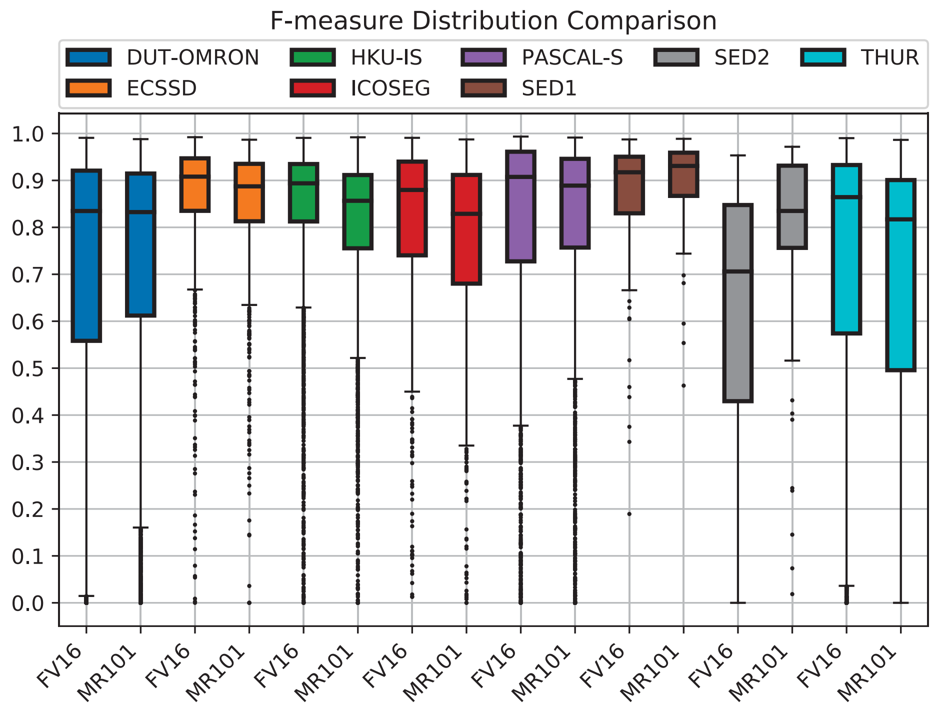


Figura 6.24: Box-plot comparison between the FCN-VGG-16 (FV16) and the Mask R-CNN-ResNet-101 (MR101). Source: Author.

6.4 CONCLUSION

As presented in this chapter, the Mask R-CNN impressively outperformed the FCN with the ResNet-50 and the ResNet-101 backbones, and as showed by the Wilcoxon signed-rank test, the results generated by the Mask R-CNNs are statistically higher than the results generated by the FCN.

The most impressive result was in the SED2 dataset, with an improvement of 47.61% with the ResNet-101 backbone. The VGG-16 also showed to be a promising backbone, with the FCN-VGG-16 achieving the highest F-measures in many datasets. However, the Mask R-CNN-ResNet-101 outperformed all evaluated FCNs in the DUT-OMRON, SED1, and SED2. Also, the strategy of applying the Mask R-CNNs bounding boxes in the FCN-VGG-16 segmentation masks has not improved the results, and the VGG-16 will be explored as a Mask R-CNN backbone in future works.

Among the evaluated FCNs, the ResNets achieved very poor segmentation results. The Mask R-CNN strategy of detecting regions of the image with a high probability of being objects, and performing the segmentation inside the detected regions, was able to improve the results of the ResNets, generating results close to the FCN with VGG-16 backbone.

Also, as explained in the Chapter 4, when compared with the FCN, it was expected a reduction of the false positive error with the object detection module of the Mask R-CNN. However, the Mask R-CNN highly improved the Recall, reducing the false negative error. Next chapter presents the final thoughts and discusses possible future works.

7 CONCLUSION

Fully Convolutional Networks (FCNs) have been achieving promising results in the Salient Object Detection (SOD) literature. However, preliminary experiments with standard FCNs showed that the FCNs have limitations to perform the segmentation of salient objects in some changeling scenarios. Fairly recently, the Mask Region-based Convolutional Neural Network (Mask R-CNN) was used in the SOD literature aiming to overcome such limitations. However, no extensive comparison between the FCN and the Mask R-CNN was presented in the literature. Taking in mind the shallow evaluation between the FCN and the Mask R-CNN available in the literature, this work proposed a Mask R-CNN architecture and performed a broad evaluation between the the two early mentioned segmentation approaches.

The present work compared three versions, FCN-Residual Network (ResNet)-50, FCN-ResNet-101, and FCN-VGG-16, with two versions of the Mask R-CNN, Mask R-CNN-ResNet-50 and Mask R-CNN-ResNet-101, through extensive experiments on eight public datasets using four metrics (F-measure, Precision, Recall, and Mean Absolute Error (MAE)).

In the experiments performed in this work, the Mask R-CNN impressively outperformed the FCN. With the ResNet-50 as the backbone, the Mask R-CNN achieved superior F-measure in seven of eight datasets when compared with the FCN with the same backbone, and achieved superior MAE in seven of eight datasets. With the ResNet-101 as the backbone, the Mask R-CNN achieved superior F-measure and MAE in all datasets when compared with the FCN with the same backbone.

The FCN-VGG-16 achieved the highest F-measures among all evaluated FCNs and was compared with the Mask R-CNN with ResNet-101 backbone, that achieved the best results among the evaluated Mask R-CNNs. The Mask R-CNN-ResNet-101 achieved higher F-measure in three of eight datasets, with improvements from up to 20%. The Mask R-CNN-ResNet-101 also improved the MAE in five of eight datasets when compared with the FCN-VGG-16.

The experiments performed in this work showed that the Mask R-CNN is a promising architecture to be explored in the SOD problem. The Region Proposal Network (RPN) attached before the segmentation module in the Mask R-CNN improved the segmentation results by significantly decreasing the false negative error, which highly increased the Recall metric.

Although the Mask R-CNNs evaluated in this work are outperformed by state-of-art results [Liu et al. (2019)] based on enhanced FCNs, further research enhancing the Mask R-CNNs in a similar way could provide competitive results, with the findings of this work endorsing that the Mask R-CNN is a promising alternative to solve the SOD problem.

7.1 FUTURE WORKS

Besides the promising results achieved by the Mask R-CNNs, there are new improvements and alternatives to be explored in future works.

- As presented in Chapter 4, the implementation utilized in this work has not implemented the Feature Pyramid Network (FPN), a robust module attached at the beginning of the Mask R-CNN that provides multiscale information to the RPN and the segmentation module. Versions of the Mask R-CNN with the FPN could be evaluated in the SOD problem in future works.

- As presented, the VGG-16 achieved the most promising results among all backbones evaluated. Future works could utilize implementations of the Mask R-CNN with the VGG-16 backbone and evaluate if the VGG-16 when converted in Mask R-CNN can outperform the results achieved in this work with the ResNets converted in Mask R-CNN.

- This work compared standard versions of the FCN with standard versions of the Mask R-CNN. However, the works proposed in the SOD literature developed new versions of the FCN, and evaluated a bunch of modifications on top of the FCN. Aiming to achieve results even better, future works could explore these new versions and modifications on top of Mask R-CNNs.

- Aiming to explore the SOD problem on embedded platforms for mobile robotic applications, Mask R-CNNs implemented with small networks models, such as Mobile-Net [Howard et al. (2017)], could be explored in future works.

REFERÊNCIAS

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Achanta, R., Estrada, F., Wils, P., and Süsstrunk, S. (2008). *Salient Region Detection and Segmentation*, pages 66–75. ICVS'08. Springer-Verlag, Berlin, Heidelberg.

Achanta, R., Hemami, S., Estrada, F., and Susstrunk, S. (2009). Frequency-tuned salient region detection. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1597–1604.

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süsstrunk, S. (2010). Slic superpixels.

Albelwi, S. and Mahmood, A. (2017). A framework for designing the architectures of deep convolutional neural networks. *Entropy*, 19(6):242.

Alom, M. Z., M. Taha, T., Yakopcic, C., Westberg, S., Hasan, M., C Van Esesn, B., Awwal, A., and Asari, V. (2018). The history began from alexnet: A comprehensive survey on deep learning approaches. *CoRR*.

Altenberger, F. and Lenz, C. (2018). A non-technical survey on deep convolutional neural network architectures. *CoRR*, abs/1803.02129.

Arbeláez, P., Pont-Tuset, J., Barron, J., Marques, F., and Malik, J. (2014). Multiscale combinatorial grouping. In *Computer Vision and Pattern Recognition*.

Arora, R., Basu, A., Mianjy, P., and Mukherjee, A. (2017). Understanding deep neural networks with rectified linear units. *ArXiv*, abs/1611.01491.

Batra, D., Kowdle, A., Parikh, D., Luo, J., and Chen, T. (2010). iCoseg: Interactive co-segmentation with intelligent scribble guidance. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3169–3176.

Bellotti, F., Berta, R., Margarone, M., and De Gloria, A. (2008). oDect: an RFID-based object detection api to support applications development on mobile devices. *Software: Practice and Experience*, 38(12):1241–1259.

Bianco, S., Buzzelli, M., and Schettini, R. (2017). A fully convolutional network for salient object detection. In Battiato, S., Gallo, G., Schettini, R., and Stanco, F., editors, *Image Analysis and Processing - ICIAP 2017*, pages 82–92, Cham. Springer International Publishing.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

Borji, A., Cheng, M.-M., Jiang, H., and Li, J. (2014). Salient object detection: A survey. *CoRR*, abs/1411.5878.

Borji, A., Cheng, M.-M., Jiang, H., and Li, J. (2015). Salient object detection: A benchmark. *IEEE TIP*, 24(12):5706–5722.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Burt, P. and Adelson, E. (1983). The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31(4):532–540.

Chen, S.-T., Cornelius, C., Martin, J., and Chau, D. H. (2018). Robust physical adversarial attack on faster R-CNN object detector. *arXiv preprint arXiv:1804.05810*, 2(3):4.

Cheng, M.-M., Mitra, N., Huang, X., and Hu, S.-M. (2014). Salientshape: group saliency in image collections. *The Visual Computer*, 30(4):443–453.

Cheng, M.-M., Mitra, N. J., Huang, X., Torr, P. H. S., and Hu, S.-M. (2015). Global contrast based salient region detection. *IEEE TPAMI*, 37(3):569–582.

de Oliveira Ceschin, F. J. (2018). Need for speed: Analysis of brazilian malware classifiers' expiration date. Master's thesis, Universidade Federal do Paraná, Curitiba, Paraná.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Dertat, A. (2018). Applied deep learning - part 4: Convolutional neural networks. `https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2`. Accessed in 08/01/2018.

Deshpande, A. (2018). A beginner's guide to understanding convolutional neural networks. `https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/`. Accessed in 08/01/2018.

Duan, L., Wu, C., Miao, J., Qing, L., and Fu, Y. (2011). Visual saliency detection by spatially weighted dissimilarity. *CVPR 2011*, pages 473–480.

Duda, R. O., Hart, P. E., and Stork, D. G. (2000). *Pattern Classification (2Nd Edition)*. Wiley-Interscience, New York, NY, USA.

Dumoulin, V. and Visin, F. (2016). A guide to convolution arithmetic for deep learning. *ArXiv e-prints*.

Dung, C. V. and Anh, L. D. (2019). Autonomous concrete crack detection using deep fully convolutional neural network. *Automation in Construction*, 99:52 – 58.

E. Rumelhart, D., E. Hinton, G., and J. Williams, R. (1986). Learning representations by back propagating errors. *Nature*, 323:533–536.

Fogel, I. and Sagi, D. (1989). Gabor filters as texture discriminator. *Biological Cybernetics*, 61(2):103–113.

Fritsch, J., Kuehnl, T., and Geiger, A. (2013). A new performance measure and evaluation benchmark for road detection algorithms. In *International Conference on Intelligent Transportation Systems (ITSC)*.

F.R.S., K. P. (1901). LIII. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202.

Garcia-Garcia, A., Orts, S., Oprea, S., Villena-Martinez, V., and Rodríguez, J. G. (2017). A review on deep learning techniques applied to semantic segmentation. *CoRR*, abs/1704.06857.

Girshick, R. (2015). Fast R-CNN. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 1440–1448, Washington, DC, USA. IEEE Computer Society.

Girshick, R. B., Donahue, J., Darrell, T., and Malik, J. (2013). Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In Gordon, G., Dunson, D., and Dudík, M., editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA. PMLR.

Goferman, S., Zelnik-Manor, L., and Tal, A. (2012). Context-aware saliency detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(10):1915–1926.

Gonzalez, R. C. (2016). Digital image processing.

Google (2019). Object detection api github. `https://github.com/tensorflow/models/tree/master/research/object_detection`. Accessed in 05/08/2019.

Gottlieb, J. P., Kusunoki, M., and Goldberg, M. E. (1998). The representation of visual salience in monkey parietal cortex. *Nature*, 391:481–484.

Gregor, K., Danihelka, I., Graves, A., and Wierstra, D. (2015). Draw: A recurrent neural network for image generation. In *ICML*.

Haykin, S. (2009). *Neural Networks and Learning Machines*. Number v. 10 in Neural networks and learning machines. Prentice Hall.

He, K., Gkioxari, G., Dollar, P., and Girshick, R. (2017). Mask R-CNN. In *The IEEE International Conference on Computer Vision (ICCV)*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *CVPR*, pages 770–778. IEEE Computer Society.

He, S., Lau, R. W. H., Liu, W., Huang, Z., and Yang, Q. (2015). Supercnn: A superpixelwise convolutional neural network for salient object detection. *International Journal of Computer Vision*, 115(3):330–344.

Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., and Scholkopf, B. (1998). Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28.

Hou, Q., Cheng, M., Hu, X., Borji, A., Tu, Z., and Torr, P. H. S. (2018). Deeply supervised salient object detection with short connections. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861.

Huaizu Jiang, Jingdong Wang, Z. Y. T. L. and Zheng, N. (2011). Automatic salient object segmentation based on context and shape prior. In *Proceedings of the British Machine Vision Conference*, pages 110.1–110.12. BMVA Press. http://dx.doi.org/10.5244/C.25.110.

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., and Murphy, K. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3296–3297.

Itti, L. (2007). Visual salience. *Scholarpedia*, 2(9):3327. revision #72776.

Itti, L., Koch, C., and Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(11):1254–1259.

Jeremy Watt, Reza Borhani, A. K. K. (2017). *Machine Learning Refined, Foundations, Algorithms, and Applications*. Cambridge University Press.

Jones, E., Oliphant, T., Peterson, P., et al. (2001–). SciPy: Open source scientific tools for Python. [Online; accessed in 08/01/2018].

Kim, J. and Pavlovic, V. (2016). A shape preserving approach for salient object detection using convolutional neural networks. *ICPR 2016*, pages 609–614.

Kirkland, E. J. (2010). *Bilinear Interpolation*, pages 261–263. Springer US, Boston, MA.

Klomann, M., Englert, M., Rehberger, A., Dietze, A., Geier, T., Rieger, S., Grimm, P., and Jung, Y. (2017). Netflincs: A hybrid cloud-based framework to allow context-based detection and surveillance. In *2017 23rd International Conference on Virtual System Multimedia (VSMM)*, pages 1–8.

Koch, C. and Ullman, S. (1987). *Shifts in Selective Visual Attention: Towards the Underlying Neural Circuitry*, pages 115–141. Springer Netherlands, Dordrecht.

Krähenbühl, P. and Koltun, V. (2011). Efficient inference in fully connected crfs with gaussian edge potentials. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 24*, pages 109–117. Curran Associates, Inc.

Krähenbühl, P. and Koltun, V. (2014). Geodesic object proposals. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, pages 725–739, Cham. Springer International Publishing.

Krizhevsky, A., Sutskever, I., and E. Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25.

Kruthiventi, S. S., Gudisa, V., Dholakiya, J. H., and Venkatesh Babu, R. (2016). Saliency unified: A deep architecture for simultaneous eye fixation prediction and salient object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324.

LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F. (2006). A tutorial on energy-based learning. *Predicting structured data*, 1(0).

Lee, G., Tai, Y., and Kim, J. (2016). Deep saliency with encoded low level distance map and high level features. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 660–668.

Li, G., Xie, Y., and Lin, L. (2018). Weakly supervised salient object detection using image labels. In *AAAI*.

Li, G., Xie, Y., Lin, L., and Yu, Y. (2017a). Instance-level salient object segmentation. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 247–256.

Li, G. and Yu, Y. (2015). Visual saliency based on multiscale deep features. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5455–5463.

Li, G. and Yu, Y. (2016a). Deep contrast learning for salient object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 478–487.

Li, G. and Yu, Y. (2016b). Visual saliency detection based on multiscale deep cnn features. *IEEE Transactions on Image Processing*, 25(11):5012–5024.

Li, G. and Yu, Y. (2018). Contrast-oriented deep neural networks for salient object detection. *CoRR*, abs/1803.11395.

Li, H., Chen, J., Lu, H., and Chi, Z. (2017b). Cnn for saliency detection with low-level feature integration. *Neurocomputing*, 226:212 – 220.

Li, X., Zhao, L., Wei, L., Yang, M., Wu, F., Zhuang, Y., Ling, H., and Wang, J. (2016). Deepsaliency: Multi-task deep neural network model for salient object detection. *IEEE Transactions on Image Processing*, 25(8):3919–3930.

Li, Y., Hou, X., Koch, C., Rehg, J. M., and Yuille, A. L. (2014). The secrets of salient object segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 280–287.

Liang, M. and Hu, X. (2015). Recurrent convolutional neural network for object recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Liaw, A., Wiener, M., et al. (2002). Classification and regression by random forest. *R news*, 2(3):18–22.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham. Springer International Publishing.

Liu, J.-J., Hou, Q., Cheng, M.-M., Feng, J., and Jiang, J. (2019). A simple pooling-based design for real-time salient object detection. In *IEEE CVPR*.

Liu, M. and Zhu, M. (2018). Mobile video object detection with temporally-aware feature maps. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Liu, N. and Han, J. (2016). Dhsnet: Deep hierarchical saliency network for salient object detection. *Proc. IEEE Conf. Comput. Vision Pattern Recog*, pages 678–686.

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440.

MarvinTeichmann (2019). Kittiseg github. `https://github.com/MarvinTeichmann/KittiSeg`. Accessed in 05/08/2019.

McLAREN, K. (1976). Xiii—the development of the cie 1976 (l* a* b*) uniform colour space and colour-difference formula. *Journal of the Society of Dyers and Colourists*, 92(9):338–341.

Michie, D., Spiegelhalter, D. J., Taylor, C. C., and Campbell, J., editors (1994). *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, Upper Saddle River, NJ, USA.

Mustamo, P. (2018). Object detection in sports: Tensorflow object detection api case study. *no. January*.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA. Omnipress.

Nguyen, T. V., Nguyen, K., and Do, T. (2019). Semantic prior analysis for salient object detection. *IEEE Transactions on Image Processing*, 28(6):3130–3141.

Nie, G., Guo, Y., Liu, Y. F. C. N. f. S. O. D., and Wang, Y. (2018). Real-time salient object detection based on fully convolutional networks. In Wang, Y., Wang, S., Liu, Y., Yang, J., Yuan, X., He, R., and Duh, H. B.-L., editors, *Advances in Image and Graphics Technologies*, pages 189–198, Singapore. Springer Singapore.

Niebur, E. (2007). Saliency map. *Scholarpedia*, 2(8):2675. revision #147400.

Noh, H., Hong, S., and Han, B. (2015). Learning deconvolution network for semantic segmentation. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 1520–1528, Washington, DC, USA. IEEE Computer Society.

Pavlovsky, V. (2018). Introduction to artificial neural networks. `https://www.vaetas.cz/posts/introduction-artificial-neural-networks/`. Accessed in 08/01/2018.

Pearson, E. S. E. S. and Hartley, H. O. (1972). *'Biometrika' tables for statisticians / Vol.2*. London : Cambridge University Press for the 'Biometrika' Trustees. Statistical mathematics. Tables (BNB/PRECIS).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Perazzi, F., Krähenbühl, P., Pritch, Y., and Hornung, A. (2012). Saliency filters: Contrast based filtering for salient region detection. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 733–740.

Powers, D. (2008). Evaluation: From precision, recall and f-factor to roc, informedness, markedness & correlation. *Mach. Learn. Technol.*, 2.

Ren, S., He, K., Girshick, R., and Sun, J. (2017). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386.

S. Bezerra, C. (2018). Uma abordagem de segmentação semântica de Íris para fins biométricos usando aprendizagem profunda. Master's thesis, Universidade Federal do Paraná, Curitiba, Paraná.

S. Bezerra, C., Laroca, R., R. Lucio, D., Severo, E., F. Oliveira, L., Britto, A. S., and Menotti, D. (2018). Robust iris segmentation based on fully convolutional networks and generative adversarial networks. In *2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 281–288.

S. Bezerra, C. and Menotti, D. (2017). Fully convolutional neural network for occular iris semantic segmentation. In XIII Workshop de Visão Computacional (WVC).

Saif Ahmed, Shams Ul Azeem, Q. H. (2017). *Machine Learning with TensorFlow 1.x*. Packt Publishing.

Sardana, N. (2017). Instance segmentation. `https://tjmachinelearning.com/lectures/1718/instance/`. Accessed in 08/01/2018.

Severo, E., Laroca, R., Bezerra, C. S., Zanlorensi, L. A., Weingaertner, D., Moreira, G., and Menotti, D. (2018). A benchmark for iris location and a deep learning detector evaluation. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7.

SHAPIRO, S. S. and WILK, M. B. (1965). An analysis of variance test for normality (complete samples)†. *Biometrika*, 52(3-4):591–611.

Sharma, S. (2018). What the hell is perceptron? the fundamentals of neural networks. `https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53`. Accessed in 08/01/2018.

Shi, J., Yan, Q., Xu, L., and Jia, J. (2016). Hierarchical image saliency detection on extended cssd. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(4):717–729.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9.

Tang, Y. and Wu, X. (2016). Saliency detection via combining region-level and pixel-level predictions with cnns. In *ECCV*.

Tang, Y., Wu, X., and Bu, W. (2016). Deeply-supervised recurrent convolutional neural network for saliency detection. In *ACM Multimedia*.

Team, S. (2019). Convolutional neural networks (cnn): Step 4 - full connection. `https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-4-full-connection`. Accessed in 05/23/2019.

Teichmann, M., Weber, M., Zöllner, J. M., Cipolla, R., and Urtasun, R. (2018). Multinet: Real-time joint semantic reasoning for autonomous driving. *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1013–1020.

Thoma, M. (2016). A survey of semantic segmentation. *CoRR*, abs/1602.06541.

Todt, E. (2005). *Visual Landmark Detection For Navigation in Outdoor Environments*. PhD thesis, Universitat Politècnica de Catalunya, Barcelona, Catalunya.

Todt, E. and Torras, C. (2007). Outdoor landmark-view recognition based on bipartite-graph matching and logistic regression. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4289–4294.

Tripathi, A. (2017). *Practical Machine Learning Cookbook*. Packt Publishing.

Uijlings, J. R. R., van de Sande, K. E. A., Gevers, T., and Smeulders, A. W. M. (2013). Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171.

Vinograd, B. (2017). Instance embedding: Segmentation without proposals. `https://towardsdatascience.com/instance-embedding-instance-segmentation-without-proposals-31946a7c53e1`. Accessed in 08/01/2018.

Vishal Maini, S. S. (2017). *Machine Learning for Humans*. online.

Wang, L., Lu, H., Ruan, X., and Yang, M.-H. (2015). Deep networks for saliency detection via local estimation and global search. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3183–3192.

Wang, X., Ma, H., and Chen, X. (2016). Salient object detection via fast R-CNN and low-level cues. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 1042–1046.

Wikipedia (2017). Bilinear interpolation. `https://en.wikipedia.org/wiki/Bilinear_interpolation`. Accessed in 08/01/2018.

Wilcoxon, F. (1992). *Individual Comparisons by Ranking Methods*, pages 196–202. Springer New York, New York, NY.

Xi, X., Luo, Y., Li, F., Wang, P., and Qiao, H. (2017). A fast and compact saliency score regression network based on fully convolutional network. *CoRR*, abs/1702.00615.

Xie, S. and Tu, Z. (2017). Holistically-nested edge detection. *International Journal of Computer Vision*, 125(1):3–18.

Yan, Q., Xu, L., Shi, J., and Jia, J. (2013). Hierarchical saliency detection. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1155–1162.

Yang, C., Zhang, L., Lu, Huchuan, R. X., and Yang, M.-H. (2013). Saliency detection via graph-based manifold ranking. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3166–3173. IEEE.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3320–3328, Cambridge, MA, USA. MIT Press.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, pages 818–833, Cham. Springer International Publishing.

Zhang, J., Dai, Y., and Porikli, F. (2017). Deep salient object detection by integrating multi-level cues. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–10.

Zhang, J., Sclaroff, S., Lin, Z. L., Shen, X., Price, B. L., and Mech, R. (2016). Unconstrained salient object detection via proposal subset optimization. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5733–5742.

Zhang, P., Liu, W., Lu, H., and Shen, C. (2018). Salient object detection by lossless feature reflection. In *IJCAI*.

Zhang, P., Liu, W., Lu, H., and Shen, C. (2019). Salient object detection with lossless feature reflection and weighted structural loss. *IEEE Transactions on Image Processing*, PP:1–1.

Zhao, R., Ouyang, W., Li, H., and Wang, X. (2015). Saliency detection by multi-context deep learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zheng, N., Tang, X., Shum, H., Wang, J., Liu, T., Sun, J., and Yuan, Z. (2010). Learning to detect a salient object. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 33:353–367.