UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Eduardo Rocha de Andrade

Segmentação Semântica Sob Desbalanceamento Extremo por Imagens Vazias

Semantic Segmentation Under Extreme Imbalance Towards Full-background Images

Campinas
2021

Eduardo Rocha de Andrade

Semantic Segmentation Under Extreme Imbalance Towards Full-background Images

Segmentação Semântica Sob Desbalanceamento Extremo por Imagens Vazias

Dissertação apresentada à Faculdade de Engenharia Elétrica e Computação (FEEC) da Universidade Estadual de Campinas (UNICAMP) como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica, na Área da Engenharia da Computação.

Dissertation presented to the School of Electrical and Computer Engineering of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Electrical Engineering, in the area of Computer Engineering.

Supervisor: Levy Boccato

ESTE TRABALHO CORRESPONDE À VERSÃO FINAL DA DISSERTAÇÃO DEFENDIDA PELO ALUNO EDUARDO ROCHA DE ANDRADE, E ORIENTADA PELO PROF. DR. LEVY BOCCATO

Campinas

2021

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Rose Meire da Silva - CRB 8/5974

Rocha de Andrade, Eduardo, 1992-
An24s Semantic segmentation under extreme imbalance towards full-background images / Eduardo Rocha de Andrade. – Campinas, SP : [s.n.], 2021.

Orientador: Levy Boccato.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Aprendizado de máquina. 2. Segmentação de imagens. 3. Segmentação de múltiplos objetos. 4. Aprendizado profundo. I. Boccato, Levy, 1986-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Segmentação semântica sob desbalanceamento extremo por imagens vazias
Palavras-chave em inglês:
Machine learning
Image segmentation
Multi-object segmentation
Deep learning
Área de concentração: Engenharia de Computação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora:
Levy Boccato [Orientador]
Hélio Pedrini
Tiago Fernandes Tavares
Data de defesa: 20-12-2021
Programa de Pós-Graduação: Engenharia Elétrica

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0001-7544-6822
- Currículo Lattes do autor: http://lattes.cnpq.br/0565655408463869

COMISSÃO JULGADORA - DISSERTAÇÃO DE MESTRADO

Candidato: Eduardo Rocha de Andrade RA: 208913

Data da Defesa: 20 de dezembro de 2021

Título da Tese: "Semantic Segmentation Under Extreme Imbalance Towards Full-background Images.”.

Prof. Dr. Levy Boccato
Prof. Dr. Hélio Pedrini
Prof. Dr. Tiago Fernandes Tavares

A ata de defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no SIGA (Sistema de Fluxo de Dissertação/Tese) e na Secretaria de Pós-Graduação da Faculdade de Engenharia Elétrica e de Computação.

To my parents

Acknowledgements

This work certainly represents a chapter in my life and it would never be possible without the help and support of so many people. To all of them, which would be too exhaustive to list, I would like to deeply express my appreciation.

To some, who provided especial support, I would like to especially express my gratitude:

To my parents, Fernando Rocha de Andrade and Ligia Rocha de Andrade, for raising me and providing me with the required education to get here.

To my companion and friend, Cinthia Kleiner, for all the love and support and for having the patience to listen to my ideas.

To my professor, Levy Boccato, for accepting to be my supervisor in this work and for providing the guidance I needed throughout all this time.

"The impediment to action advances action. What stands in the way becomes the way."

– Marcus Aurelius

Resumo

O panorama de visão computacional mudou significativamente com o advento das redes neurais convolucionais e técnicas de aprendizado profundo. Estas poderosas ferramentas não apenas aprimoraram tarefas tradicionais como classificação de imagem, segmentação semântica e detecção de objeto, mas também possibilitaram novas aplicações de visão computacional, principalmente na àrea generativa, como, por exemplo, na geração de imagens sintéticas e transferência de estilo. Entretanto, em aplicações reais, as condições de contorno podem divergir significativamente daquelas mais vistas na literatura. Especificamente para segmentação semântica, imagens que não contêm nenhum objeto de interesse, denotadas como imagens vazias, podem corresponder a uma porção grande da base de dados, resultando em um severo desbalanceamento de classe. Esse cenário particular é o tema deste trabalho, no qual analisamos as duas abordagens mais comuns: segmentação em um único estágio e classificação-segmentação em dois estágios. Como principal contribuição, nós propomos uma nova modificação na arquitetura de redes neurais de formato encoder-decoder. Tal modificação, apesar de pequena, é capaz de utilizar contexto semi-global e mecanismos de atenção para melhorar a eficácia de redes de segmentação de estagio único em condições extremamente desbalanceadas a favor de imagens vazias. Adicionalmente, propomos uma função custo auxiliar para imagens de *foreground* que, além de estabilizar o processo de treinamento, permite que a rede se concentre em objetos pequenos mesmo na presença de um grande número de imagens vazias. Ambas as propostas foram avaliadas em duas bases de dados de distintas características e demonstraram ganhos em IoU de 15 e 25% contra os melhores competidores de um e dois estágios, respectivamente. Finalmente, a fim de melhor compreender os mecanismos internos de nossa arquitetura, estudos de ablação foram realizados, demonstrando forte concordância com nossas suposições iniciais.

Palavras-chaves: Segmentação Semântica; Aprendizado Profundo; Aprendizado de Máquina; Inteligência Artificial; Reconhecimento de Padrões

Abstract

The landscape of computer vision tasks has been significantly changed in the past decade with the advent of convolution neural networks and deep learning techniques. Such powerful tools not only improved traditional tasks such as image classification, semantic segmentation and object detection, but also unlocked new computer vision applications altogether, specially in the generative field, such as image generation and style transfer. Nonetheless, in real applications, boundary conditions might diverge significantly from those found in the literature. Specifically for semantic segmentation, images with no object of interest – namely empty images – may comprise a big part of the dataset, resulting in a stark class imbalance. This particular, yet common, scenario is the subject of this work, where we analyze both the single-stage segmentation and two-stage classification-segmentation pipelines – the two most common deep learning approaches to tackle this problem. We propose a novel modification for encoder-decoder segmentation networks as our main contribution. This relatively simple yet powerful layer takes advantage of semi-global context and attention mechanisms to improve the efficacy of single-stage encoder-decoder segmentation models in extremely unbalanced conditions. Additionally, we propose an auxiliary segmentation loss for foreground images, which stabilizes the training process and allows the network to focus on small objects even under strong imbalance towards the background class. Both proposals are evaluated on two different datasets, showing IoU gains of up to 15 and 25% against its strongest single- and two-stage competitors, respectively. Finally, in order to better comprehend the underlying mechanisms of our architecture, ablation studies were performed, which showed a strong agreement with our initial assumptions.

Keywords: Semantic Segmentation; Deep Learning; Machine Learning; Artificial Intelligence; Pattern Recognition.

List of Figures

List of Tables

List of Symbols

General

CNN Convolutional Neural Network

CPU Central Processing Unit

CV Computer Vision

FLOPs Floating Point Operations

GPU Graphics Processing Unit

MRI Magnetic Resonance Imaging

OS Operational System

RAM Random-access Memory

RBG Red, Green and Blue Image Channels

SS Semantic Segmentation

SSEI Semantic Segmentation with *Empty* Images

Layers

BiFPN Bidirectional Feature Pyramid Network

BN Batch Normalization

DWConv Depth-wise Convolution

ELU Exponential Linear Unit

FAM Feature Alignment Module

FC Fully-connected Layer

FCNet Fully Convolutional Network

FPN Feature Pyramid Network Layer

FSM Feature Selection Module

GN Group Normalization

HC Hypercolumns Layer

IN Instance Normalization

LN Layer Normalization

MBConv Mobile Inverted Residual Convolutional Block

MLF Multi-level Fuse Block

MLP Multi-layer Perceptron

PAN Pyramid Attention Network

PReLU Parametric Rectified Linear Unit

ReLU Rectified Linear Unit

SE Squeeze-and-Excitation

Metrics

FN False Negatives

FP False Positives

N Negatives Samples

P Positives Samples

TN True Negatives

TP True Positives

CE Cross-entropy Loss

IoU Intersection Over Union

mIoU Mean Intersection Over Union of Each Class

Networks

FaPN Feature-aligned Pyramid Network

OCRNet Object Contextual Representitaion Network

PANet Pyramid Attention Network

PSPNet Pyramid Scene Parsing Network

ResNet Residual Neural Network

SENet Squeeze-and-Excitation Network

VGG Visual Geometry Group Network

Contents

1 Introduction

In this chapter, we begin by briefly explaining the target problem of this work. In the sequence, we address the motivations behind its investigation and present the objectives and contributions of this work. Finally, the main research questions we aim to answer are introduced and the organization of this document is explained with the purpose of providing the reader with further clarity.

1.1 Problem Characterization

The advent of convolutional neural networks (CNNs) (LECUN *et al.*, 1989) has definitively changed the landscape of computer vision (CV) applications. Initially, most deep learning techniques were applied to the traditional task of image classification, which consists in classifying an entire image into one of a set of pre-defined classes. Naturally, with the breakthroughs observed in popular CV benchmarks such as ImageNet (RUSSAKOVSKY *et al.*, 2015), studies quickly started to adapt deep classification models to dense prediction tasks. Among those, object detection – which aims to provide localization (as boxes) for instances of each class in the image – semantic segmentation – which consists in classifying every individual pixel of the input images as belonging to one of a set of predefined classes – and instance segmentation – that further extends semantic segmentation results to also distinguishing between instances of the same class – were the ones that experienced major improvements due to the emergence of such models. Figure 1.1 overviews the traditional dense prediction tasks in CV.

Nonetheless, real-world applications present themselves with several peculiarities, which, in some cases, diverge significantly from the canonical scenarios that are mostly studied in the literature. **Semantic segmentation with *empty* images** (SSEI) is one of such cases and constitutes the main scope of this work.

Empty images can be defined as images that have no object of interest and, consequently, all their pixels belong to the background class[1]. Although in semantic segmentation the background class is intrinsically handled, when *empty* images are present they usually account for a high percentage of the dataset, which leads to dire class imbalance between background and foreground pixels, which, ultimately, hinders the CNN's performance.

[1] Due to a lack of better terminology, we refer to them as *empty*, *negative* or *background* images. Correspondingly, images that have at least one pixel belonging to a foreground class are referred to as *foreground*, *positive* or *non-empty* images.

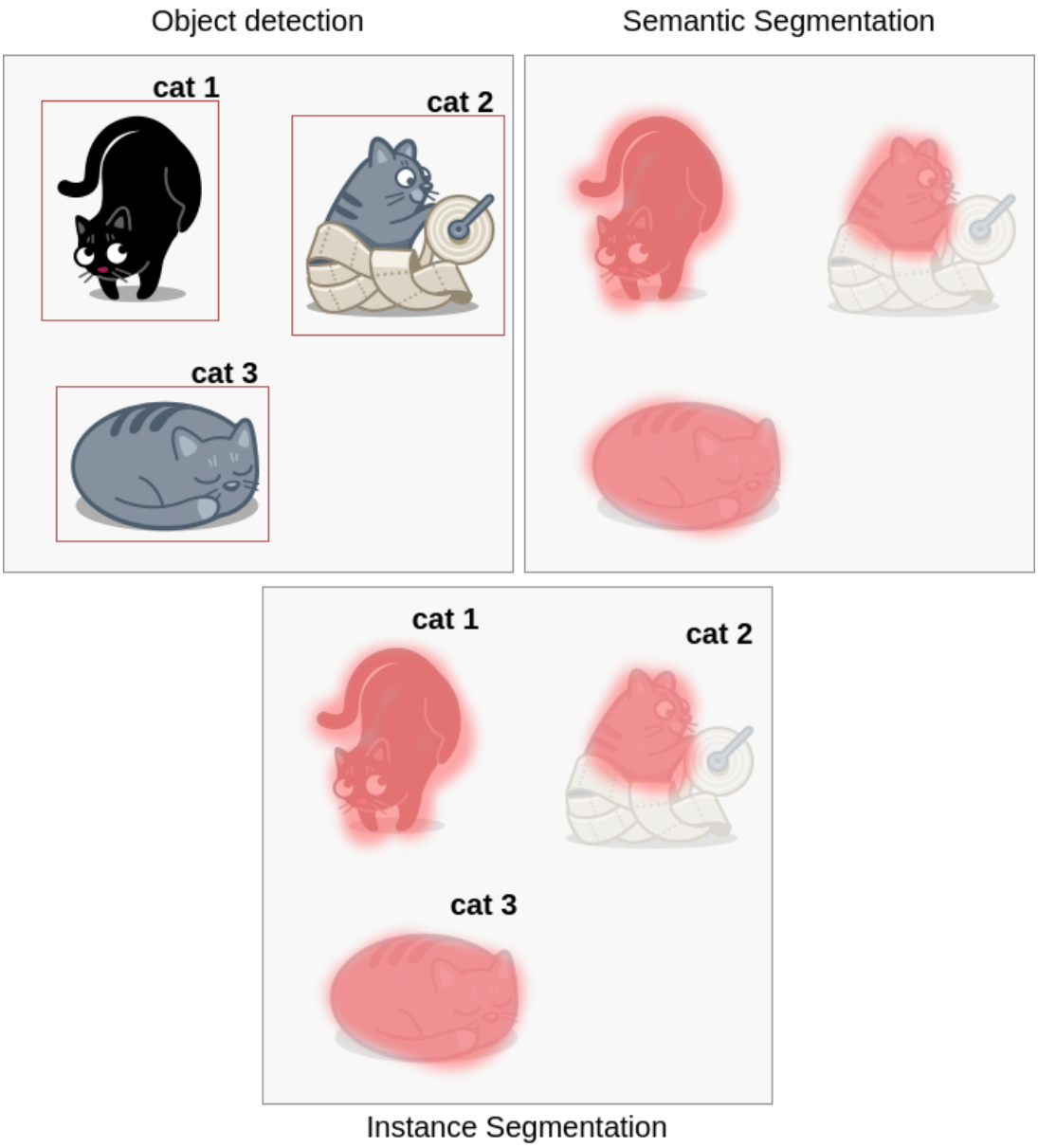

Figure 1.1 – Dense prediction tasks in CV.

A possible strategy to address this task involves a two-stage scheme. First, a classifier is used to predict the presence of one of the foreground classes at image-level; then, a semantic segmentation model is used to classify the *positive* images at pixel-level. Nevertheless, such approach presents some limitations of its own such as error propagation, bigger memory footprint and may be prohibitive in applications that demand better explainability, such as medical imaging, given the first classifier's coarse prediction. An overview of the two most common approaches to address the problem of semantic segmentation with *empty* images is given in Figure 1.2.

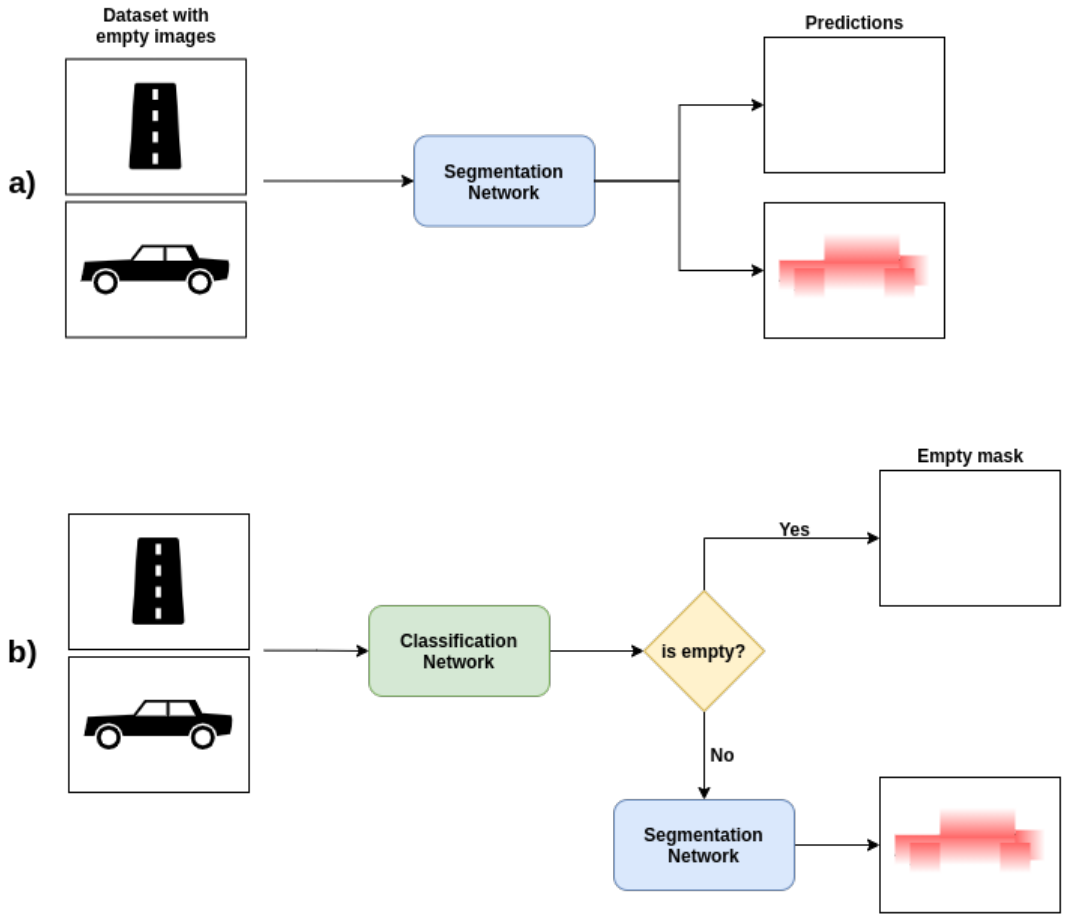

Figure 1.2 – **Single- and two-stage approaches for segmentation with empty images.** In a), the single-stage approach is shown where a single segmentation network is trained for segmenting both *empty* and *non-empty* images. In this case, the network performs the task of image-level *foreground* classification implicitly. Alternatively, in b), a classification network is first used to filter out the *empty* images and only *foreground* images proceed to the segmentation network.

1.2 Problem Motivation

The motivations that encourage us to investigate the problem of semantic segmentation with *empty* images essentially are the observed lack of coverage in the literature and its common presence in practical applications.

We hypothesize that the first reason mainly stems from the fact that one of the major factors that have driven the growth of the machine learning field in the past years is the standardisation of databases. By using the same publicly available datasets, reproducibility is facilitated and authors are able to easily compare their results and, consequently, build upon previous works. Nonetheless, as a side effect, new works start to get specifically tailored for the major databases and, in some cases, may not be representative of the general task. Particularly pertinent to the scope of this work, popular datasets for

semantic segmentation, such as PASCAL-VOC (EVERINGHAM *et al.*, 2010), CityScapes (CORDTS *et al.*, 2016) and MS-COCO (LIN *et al.*, 2015), are all majorly designed in a way that all their images have at least one of the *foreground* classes, disregarding *empty* images that are often found in practical applications. Additionally, SSEI is naturally handled by the *background* class and given the simplicity of two-stage pipelines, the task does not attract as much academic interest for researchers as the standard semantic segmentation. We argue, however, that the *default* approaches to the problem are sub-optimal and can be significantly improved upon scrutiny and by adopting more refined and especially-tailored strategies.

No less important, the high-frequency of *empty* images in real-world applications is our second motivation. In medicine, for example, many diseases, such as pneumonia, pneumothorax, tumors, carcinoma and melanoma, can be detected by analyzing X-rays and CT-scans images. Nevertheless, the occurrence of the disease in the exam may be quite scarce when compared to the cases of healthy patients, which can generate intrinsic class imbalance towards *empty* or healthy images in medical datasets. Similarly, in industrial processes, the event of interest may be related to rare anomalies or defects in a production line that must be detected. *Empty* images are also often present in large-scale sensor data such as satellite images and seismic imaging. Given its frequent and important occurrence in practical applications and its absence in the deep learning literature, the particular case of semantic segmentation with unbalanced data towards *empty* images was chosen as the topic of this work.

1.3 Objectives and Contributions

In this work, we propose to study and compare common single- and two-stages approaches on the problem of semantic segmentation with *empty* images, which despite being very common in real applications is seldom explored in the literature. Furthermore, we propose a new network architecture that employs attention mechanisms to incorporate global and semi-global context, which enhances its image- and pixel-level predictions. In addition, an auxiliary segmentation loss based on the Dice coefficient (SORENSEN, 1948) is proposed for *foreground* images, which aims to reduce the class imbalance problem at batch level. Finally, we perform a series of ablation experiments to investigate our proposed architecture and loss, providing empirical evidence to support our understanding of such layers.

Accordingly, the main contributions can be summarized as follows:

1. Proposal of architectural modifications that can be applied to modern encoder-

decoder networks to specially tackle the problem of semantic segmentation with *empty* images.

2. Investigation on the role of simple image-level batch formulation schemes, which are seldom used in segmentation, when *background* images are also taken into account.

3. Proposal of an auxiliary scale-invariant segmentation loss function to be used along image-level batch sampling schemes to cope with the class imbalance in single-stage approaches.

4. Investigation and comparison of the two most common approaches to solve the problem – single- and two-stage – providing important insights for data science practitioners.

1.4 Research Questions

In this section, we list the main questions that drive our investigation and research:

1. What are the difficulties particularly imposed by the presence of *empty* images in the dataset?

2. Considering class imbalance, how single-stage approaches fare against their more complex two-stage counterparts?

3. Is it possible to employ "smart" image-level batch sampling schemes, commonly used in image classification tasks, to mitigate class imbalance in semantic segmentation with *empty* images?

4. Can a especially-tailored architecture for the task at hand improve the results when compared to the *default* approaches?

1.5 Text Structure

The remainder of this dissertation is aimed to provide the reader with a good introduction to the problem at hand and its corresponding theoretical basis. Then, the proposed contributions are presented along with the experimental setup and the methodology used throughout this work. Finally, the results are presented and discussed followed by the conclusion and future works.

More objectively, the document is structured as follows:

- Chapter 2 presents a formal and more in-depth introduction to the tasks covered in this dissertation.
- Chapter 3 overviews the most common operations and layers used in CNNs, such as the multi-layer perceptron, the convolution, activation functions and the batch normalization, providing the foundation for the following chapters.
- Chapters 4 and 5 provide a bibliographic review of the most famous and relevant neural networks for image classification and semantic segmentation, respectively. Also, in Chapter 5, we briefly overview popular evaluation metrics and loss functions used to train segmentation networks.
- Chapter 6 introduces the proposed framework and other contributions proposed by this work.
- Chapter 7 explains the experimental procedure and methodology utilized in the performed experiments.
- Chapter 8 presents and discusses the experimental results and their implications.
- Chapter 9 concludes this document by summarizing its most important findings and enumerating possible future works.

2 Problem Definition

We start this chapter by briefly introducing the computer vision tasks of Image Classification and Semantic Segmentation under a *deep learning* perspective. Then, the target topic of this work – semantic segmentation with empty images – is presented along with a section focused on class imbalance, which often occurs when empty images are introduced.

2.1 Image Classification

Image classification is a classical tasks in computer vision. It consists in determining which class c among a predefined set of classes $\mathcal{C}$ an input image belongs to. Formally, for an input RGB image $\mathbf{X} \in \mathbb{R}^{H \times W \times 3}$ of height H and width W, the classification model is expected to produce a function $f : \mathbb{R}^{H \times W \times 3} \rightarrow \{c_1, c_2, ..., c_k\}$ mapping it to a categorical output $c \in \mathcal{C}$. In practice, the image classes are encoded as a strictly increasing sequence of natural numbers, i.e., $\{1, 2, 3, ..., c\} \in \mathbb{N}^*$, which are denoted as *ground-truth labels* $\hat{y}$. The model's class estimate, on the other hand, is denoted by $\hat{y} = f(\mathbf{X})$.

The applications for image classification are very wide ranging from satellite imagery to medical imaging. Particularly for the latter, Rajpurkar *et al.* (2017) utilized chest X-rays to train a CNN capable to detect 14 lung diseases such as Pneumonia, Pneumothorax, Edema, Fibrosis and Hernia, and reports F1-score performance exceeding that of an average radiologist. Another example is (HOSSEINI-ASL *et al.*, 2016), which uses a 3D-CNN to diagnose Alzheimer's disease in MRI scans reporting state-of-the-art accuracy over 99%. The paper also reports a pre-training in an autoencoder fashion to first learn generic brain structure features and, then, fine-tunes the model using deep-supervision to detect the Alzheimer's disease. Pratt *et al.* (2016) reported an accuracy metric of 75% when detecting diabetic retinophaty using a 13 layers CNN trained on 90,000 digital images of the fundus of the eye.

Aside from medical applications, Pritt and Chern (2017) used an ensemble of CNNs to classify 63 classes of facilities, such as hospital, airport, park and zoo, in satellite imagery. The paper reports overall accuracy over 83% and, for some specific classes (15 among the 63 available), over 95%.

ImageNet (RUSSAKOVSKY *et al.*, 2015) – one of the most important datasets and challenges in computer vision – comprises over 1 million images and over 1000 different classes, from several dog and cat breeds to general objects and vehicles. Notably, prior to

CNNs, the best scoring solution was (SANCHEZ; PERRONNIN, 2011) achieving 50.9% top-1 accuracy. Currently, modern deep convolutional networks such as (HE *et al.*, 2016; XIE *et al.*, 2017; PHAM *et al.*, 2020a; HU *et al.*, 2018; TAN; LE, 2019) can easily surpass the 80% accuracy milestone and, recently, (PHAM *et al.*, 2020b) achieved the 90% mark.

2.2 Semantic Segmentation

Along with image classification and object detection, semantic segmentation is one of the most common tasks in computer vision. Similarly to image classification, a sample must be assigned to one of $c_k \in \mathcal{C}$ predefined classed. However, instead of classifying the entire image, in semantic segmentation the prediction occurs at pixel-level. Thus, all pixels are expected to be individually assigned to one of the c foreground classes or a background class $\mathcal{B}$.

Formally, the segmentation model implements a function $f : \mathbb{R}^{H \times W \times 3} \rightarrow \{\mathcal{C} \cup \mathcal{B}\}^{H \times W}$ capable of mapping an input image $\mathbf{X} \in \mathbb{R}^{H \times W \times 3}$ into a segmentation mask $\mathbf{Y} \in \{\mathcal{C} \cup \mathcal{B}\}^{H \times W}$ whose pixels have integer values corresponding to their class encoding, which includes the background class $\mathcal{B}$.

As for its applications, the joint classification and localization capabilities of semantic segmentation models make such class of algorithms very useful, enabling their widespread adoption in several areas. In diagnostic medicine, for example, Pereira *et al.* (2016) designed a 11-layered CNN able to segment tumors on brain MRI scans, which obtained the first place in the BRATS 2013 challenge[1]. Brain tumor segmentation is very important since, by providing precise boarders for the brain and tumor regions, it guides the brain surgeon on the removal of the cancerigenous mass, avoiding eliminating excessive eloquent brain tissue, which can cause limb weakness, numbness and cognitive impairment according to Ker *et al.* (2018). In 2019, The Society for Imaging Informatics in Medicine (SIIM) and the American College of Radiology (ACR) hosted a semantic segmentation competition on Kaggle[2] to predict Pneumothorax[3] on lung X-ray images. The competition[4] had more than 1,400 participating teams and the top solutions achieved Dice coefficient (SORENSEN, 1948) greater than 86%. Apart from medical applications, AirBus[5] hosted a competition to segment ships in satellite imagery. Severstal[6] offered $120,000 in prizes for contestants to segment defects in images of steel sheets, claiming

[1] More information at: <https://www.smir.ch/BRATS/Start2013>
[2] Popular website that promotes several Artificial Intelligence (AI) competitions with prizes.
[3] Pneumothorax is a pathology caused by the presence of air in the pleural cavity of the lung, which may be life-threatening and requires drainage.
[4] More information at: <https://www.kaggle.com/c/siim-acr-pneumothorax-segmentation/overview>
[5] More information at: <https://www.kaggle.com/c/airbus-ship-detection>
[6] More information at: <https://www.kaggle.com/c/severstal-steel-defect-detection/overview>

a good algorithm could help them improve their production quality. TGS[7] also prized a total of $100,000 to the top-4 solutions to correctly predict salt deposits in sonar images. Chen *et al.* (2018) proposed to use dilated convolutions (YU; KOLTUN, 2015) along with a thin decoder, allowing their model to better handle high-frequency information and granting the state of the art in the 2012 Pascal-VOC (EVERINGHAM *et al.*, 2010) image segmentation challenge with almost 90% mean Intersection Over Union (mIoU) without any post-processing. Some examples of the results offered by such powerful algorithm are displayed in Figure 2.1.

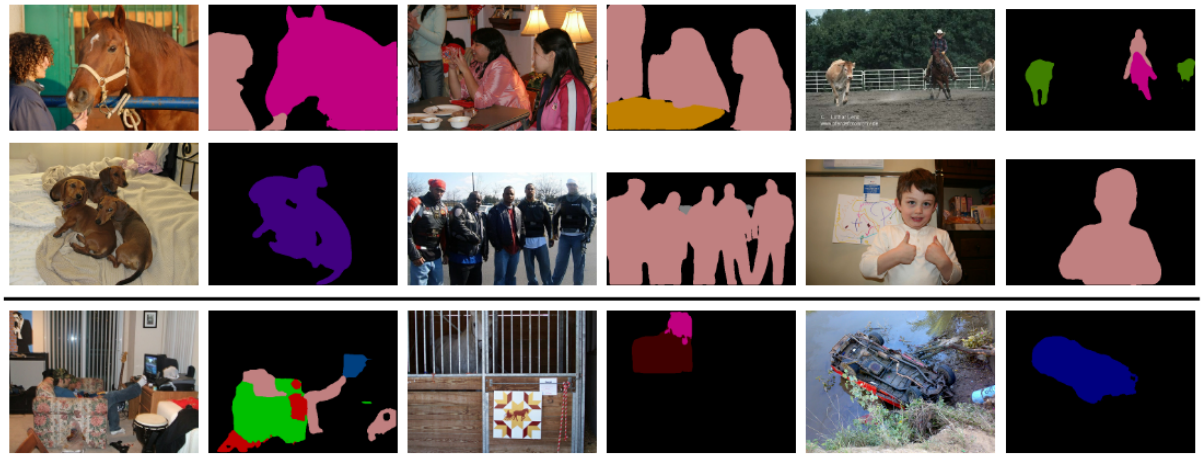

Figure 2.1 – Examples of some predictions of the DeepLabV3+ model on the Pascal-VOC 2012 dataset. Figure from (CHEN *et al.*, 2018).

2.3 Segmentation with Empty Images

Ever since the deep learning technology was disseminated into the field of semantic segmentation, many novel network architectures and loss functions have been consistently proposed in an incredible fast pace, allowing solutions with ever-growing quality. Part of such success is due to the good quality and availability of recent image databases such as MS-COCO (LIN *et al.*, 2015), Pascal-VOC (EVERINGHAM *et al.*, 2010) and Cityscapes (CORDTS *et al.*, 2016), which allow public access to a high number of labeled data and enables researchers to benchmark and compare their solutions on a common basis. Nevertheless, as a consequence, many models and innovations are exclusively designed to work on problems with a set of characteristics similar to those of the aforementioned datasets and, consequently, many particularities of semantic segmentation that are often found in real applications are neglected or do not receive sufficient attention. As an illustration of the given point, it is not rare to find top solutions on the benchmarks that perform poorer than expected in unusual domains and, in many

[7] More information at: <https://www.kaggle.com/c/tgs-salt-identification-challenge/overview>

cases, simple architectures can perform equally well with much less parameters. Both SIIM-ACR Pneumothorax[8] and Severstal: Steel Defect Detection[9] competitions, for example, had winner solutions based on simple encoder-decoder architectures, such as U-Net (RONNEBERGER *et al.*, 2015) and FPN (LIN *et al.*, 2016), whereas networks based on dilated convolutions (YU; KOLTUN, 2015) – that were the state-of-the-art solutions for the aforementioned benchmarks at the time – achieved significantly lower scores. In this particular example, we hypothesize that while dilated convolutions may be very suited for natural images where objects' borders are well defined, they bring no gain on data whose objects' contours are more subtle or defined by texture, such as X-ray or seismic imaging.

In this section, we introduce the problem of semantic segmentation with empty images. In contrast to the aforementioned datasets from the literature, many real applications have considerably less samples to train deep learning models. Furthermore, it is also common that many of such samples are completely *empty*, i.e., all of their pixels belong to the *background* class. Consequently, such tasks usually present higher levels of class imbalance towards the *background* class and, thus, require a careful approach. Considering the importance of class imbalance in the scope of our work, we dedicate Section 2.4 for the analysis of such property.

Although the problem of segmentation with empty images represents a particular instance of the general semantic segmentation, it can also be interpreted as the combination of two separate tasks. First, a binary classification is set to classify images as either *foreground* or *background*. Then, a conventional semantic segmentation problem is defined only for the *foreground* images, i.e., images that have at least one of their pixels belonging to a *foreground* class. In this case, the overall model is expected to produce a classification function $f : \mathbb{R}^{H \times W \times 3} \rightarrow \{\mathcal{B}, \mathcal{F}\}$ and a segmentation function $g : \mathbb{R}^{H \times W \times 3} \rightarrow \{\mathcal{B}, 1, ..., c\}^{H \times W}$. The function f is responsible for first determining whether the input image $\mathbf{X} \in \mathbb{R}^{H \times W \times 3}$ belongs to the *foreground* class $\mathcal{F}$ or the *background* class $\mathcal{B}$. If the label $\mathcal{F}$ is assigned, then the image is used as input to the segmentation function g, which is responsible for assigning a class $c \in \mathcal{C}$ for each pixel, producing a segmentation mask $\mathbf{Y} \in \{\mathcal{B}, 1, ..., c\}^{H \times W}$. Otherwise, an empty segmentation mask $\mathbf{Y}^{\varnothing} \in \mathcal{B}^{H \times W}$ is assigned to the image. The contrast between single- and two-stage approaches to solve the task of SSEI was illustrated in Figure 1.2 in the previous chapter.

[8] <https://www.kaggle.com/c/siim-acr-pneumothorax-segmentation/overview>
[9] <https://www.kaggle.com/c/severstal-steel-defect-detection/overview>

2.4 Class Imbalance

For essentially all deep learning applications, data quality is of utmost importance. Although some characteristics are very task specific, there are several others that could be generically desired for a training set. Particularly for computer vision, the following can be listed:

- *Size*: the number of samples (images) in the dataset. In general, deep neural networks are very data intensive and dataset size is very important to avoid overfitting. Such dependency is specially true for deeper models with high number of learnable parameters. Interesting enough, some recent works such as (XIE *et al.*, 2020) and (DOSOVITSKIY *et al.*, 2020) have shown that even datasets with more than 1 million images such as ImageNet (RUSSAKOVSKY *et al.*, 2015) are not enough to exploit the full capability of big models.

- *Diversity*: this characteristic is related to the training dataset's distribution. In CNNs, invariance to scale, rotation, pose, style, brightness and other photometric effects are learned through data. Hence, a diverse dataset is very important and, consequently, data augmentation techniques can be an important asset to ensure the network receives relevant variations of the samples.

- *Class balancing*: this property accounts for the difference in the number of samples of each class. Ideally, a training dataset is expected to be well balanced between its classes. Otherwise, given a disproportional signal feedback at the learning phase, a deep learning model may bias its predictions towards the majority class. Unfortunately, unbalanced datasets are reasonably common in real-world applications.

Even though all the aforementioned items are inter-connected, the latter, in particular, is very important to the scope of this work due to the significant growth in the number of samples for the *background* class when *empty* images are incorporated as part of the training set. Thus, a thorough examination of such dataset property is of great interest to our work.

For a binary classification problem, the images are divided into two categories or classes. In such scenario, class imbalance occurs when one class, namely *minority class*, has considerably less samples than the other (*majority*). Such phenomenon is very common in real applications and its causes can be either classified as *intrinsic* or *extrinsic*. The former category is related to the natural distribution/frequency of the data; for instance, in medical imaging is frequent to have considerably less X-rays or MRI images of a certain disease than those associated with healthy patients. Other examples include *rare events*

such as manufacturing defects, natural disasters and security violations. The latter, on the other hand, is usually associated to data collection or other external factors, e.g., privacy or economical reasons (JOHNSON; KHOSHGOFTAAR, 2019). In addition, the *minority* class usually stands as the one that capture most of the application's interest, e.g disease or rare event. Thus, being able to correctly handle these categories is paramount in some contexts. Analogously, the definition can be extended to a multi-class form through class decomposition (Wang; Yao, 2012).

Early in the 1990's, Anand *et al.* (1993) studied how the backpropagation algorithm (RUMELHART *et al.*, 1986) performed under class imbalance situations. It was found that, in such scenarios, the net gradient for the *minority* class was significantly lower than that of its *majority* counterpart. Consequently, since the gradients are directly used to update the network's weights, the algorithm could easily learn features from the dominant class while the *minority* class would converge in a much slower pace and with higher associated error. Furthermore, as aforementioned, CNNs are very data intensive. Thus, having a category under-represented in the dataset often results in sub-optimal *size* and *diversity* for that category, which may hinder the network's capacity to learn representative and diverse features to correctly classify images of the dominated class. Even in *Big Data*, the recent works (Bauder *et al.*, 2018) and (BAUDER; KHOSHGOFTAAR, 2018) show that working with rare classes can be very difficult and require deliberate handling.

Finally, not only class imbalance can degenerate the model's performance, but one must be careful when selecting the performance metrics for a problem with unbalanced dataset since some metrics can be easily skewed towards the dominant class, occluding the analysis of the dominated categories and, ultimately, hampering an unbiased evaluation of the algorithm.

In the sequence, we present the under- and over-sampling techniques (BRANCO *et al.*, 2015; HE; MA, 2013; JOHNSON; KHOSHGOFTAAR, 2019), which are commonly used to tackle the class imbalance issue in classification tasks. These two simple techniques are very important to our work since we propose to adapt them for the SSEI context.

2.4.1 Sampling Techniques

Considering a binary classification scenario for a training set of N_{total} samples with a *minority class ratio* r_{min}, i.e., the number of images belonging to the minority class N_{min} divided by the total number of samples N_{total}, where $r_{min} << 50\%$, a single training epoch would contain significantly more samples from the majority class Maj. The under-sampling technique simply consists in modifying the dataset exploited in an

epoch to provide a more balanced data representation.

From a practical perspective, at the beginning of each epoch, one would randomly select N_{min} examples from the majority class so that the training set for that epoch would be comprised of $2 \cdot N_{min}$ samples (all the images from the minority class plus the same number of randomly selected samples from the majority class). Since most deep learning models train over multiple epochs, the majority class is exploited in its entirety whilst providing a balanced representation for the network. If desired, one can take a more careful approach and make sure that the batches provided to the network are also balanced.

On the opposite direction, the over-sampling technique randomly selects **with replacement** a total of $N_{maj} = N_{total} - N_{min}$ samples from the minority class so that the epoch training set is balanced (and, as consequence, bigger than the original training set). Especially when applying sampling techniques, data augmentation is an important asset to increase the diversity of the examples.

In semantic segmentation, on the other hand, since the annotations are provided at pixel-level, a single image is usually comprised of pixels from more than one class, making it difficult to apply sampling techniques. Therefore, such methods are avoided and modifications to the loss function, such as Focal Loss (LIN *et al.*, 2017), are used instead.

Particularly in the SSEI context, since most of the imbalance occurs between the background class and the set of all others classes, we can divide the images in the training set as *background*, where all pixels belong to the background class, and *foreground*, where at least one pixel belongs to one of the non-background classes. By doing so, we propose to mitigate the class imbalance towards empty images by applying one of the sampling techniques previously described. In Chapter 8, we provide a better description of this application and perform some experiments to evaluate its usefulness.

3 Elementary Operations in CNNs

This chapter brings a short introduction on the basic operations used in convolutional neural networks, which are the building blocks for almost any modern CNN discussed in the next chapters.

3.1 Multi-layer Perceptron

The Multi-layer Perceptron (MLP), also known as **feedforward network**, is one of the most emblematic models in deep learning. Its main goal is to approximate a function $\overset{*}{f}$ and is usually used to classify samples or perform regression to a target numeric value. The MLP is comprised by one or more *hidden layers* and a final *output layer*, generally a linear or logistic regression, depending on the task (GOODFELLOW *et al.*, 2016).

The *hidden layer* is usually a linear transformation followed by a non-linear activation function ϕ. The complete transformation $f(\mathbf{x}; \mathbf{W}, \mathbf{b}) = \mathbf{z} = \phi(\mathbf{x} \cdot \mathbf{W}^\intercal + \mathbf{b})$, parameterized by a weight matrix $\mathbf{W} \in \mathbb{R}^{C_{out} \times C_{in}}$ and a bias vector $\mathbf{b}^{C_{out}}$, maps an input vector of size C_{in}, which can be either the actual input features $\mathbf{x}^0$ or the output from a previous *hidden layer* $\mathbf{z}^{i-1}$, to a non-linear representation $\mathbf{z}^i$ where the superscript i represents the layer number. The final network, then, corresponds to the sequence of layers from the input to the output. For a four-layer network, for instance, the overall transformation is represented by $f(\mathbf{x}) = f^4(f^3(f^2(f^1(x))))$ and is expected to approximate $\overset{*}{f}$ given the optimization algorithm finds suitable values for the parameters $\mathbf{W}^i, \mathbf{b}^i, i \in [1, 4]$ (GOODFELLOW *et al.*, 2016).

In the context of CNNs, the *hidden layer* is also known as *dense* or *fully-connected* layer – in contrast to the convolutional layer that has sparse connections – and is commonly used at the final portion of the model in classification tasks. In the next section, we explain in detail the non-linear activation functions ϕ that are most used in deep convolutional networks.

3.2 Activation Function

The non-linear activation functions are a essential tool in deep neural networks. Without them, MLPs and CNN would not be able to capture non-linear interactions between data. Even simple non-trainable non-linear functions as the ReLU (NAIR;

HINTON, 2010) when applied on top of linear *dense* layers can lead to powerful descriptors, modeling complex data while remaining conceptually simple at its essence. In this section, we introduce the most common activation functions for *hidden layers*. A summary of these functions is presented in Figure 3.1.

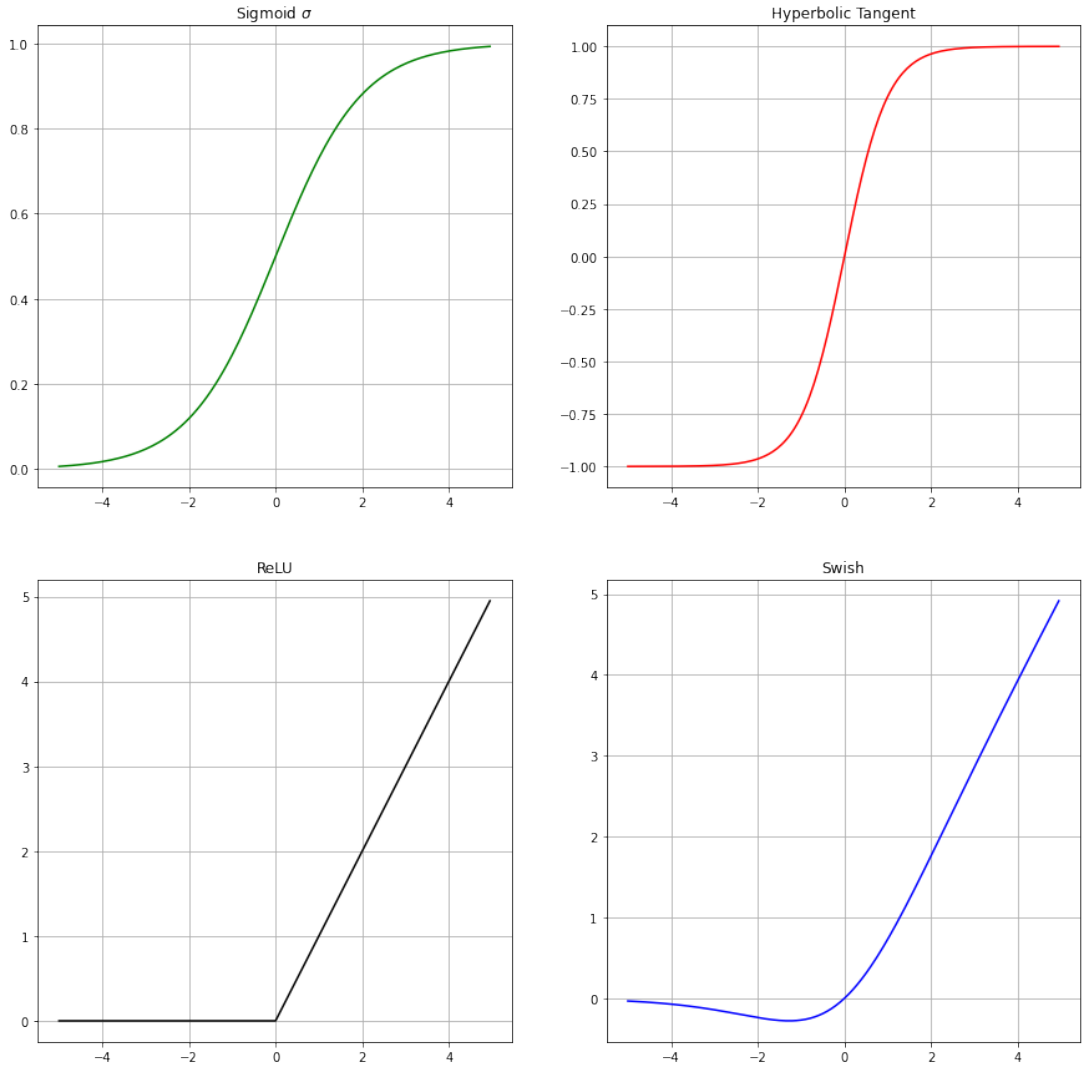

Figure 3.1 – Summary of the main activation functions for *hidden layers*.

3.2.1 Sigmoid

The sigmoid function, defined by Equation (3.1), is very important in the context of deep learning since it is commonly used to convert the final network activation or *logits* into a probability distribution for binary classification problems.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.1}$$

It has also been used in the past as activation for the *hidden layers*. However, since it saturates for both low and high values of x, the first derivative converges to zero

at these regions, which impacts the gradient's flow and, ultimately, hinders the network training. Also, the sigmoid function is not centered at zero, i.e., $\sigma(0) = 1/2$, which skews the activations' distribution as it propagates through the network.

3.2.2 Hyperbolic Tangent

The hyperbolic tangent, defined in Equation (3.2), has a similar behavior to that of the sigmoid function. In fact, they are related by $\tanh(x) = 2 \cdot \sigma(2x) - 1$ (GOODFELLOW *et al.*, 2016).

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{3.2}$$

The hyperbolic tangent was used as an activation function for *hidden layers* as an alternative for the sigmoid since it is centered at zero, i.e., $\tanh(0) = 0$, which maintains the activations' distribution throughout the network. Nevertheless, similarly to the sigmoid function, it also saturates for extreme values of x, consequently, restricting the unit's operating region around $-3 < x < 3$. Thus, its usage has been deprecated in modern CNN architectures.

3.2.3 ReLU

The rectified linear unit (ReLU) (NAIR; HINTON, 2010) is a simple piecewise linear function that restricts activations to positive values. It is simply defined by $g(x) = max(x, 0)$. Albeit simple, it should be noted that its first-order derivative is 1 for all non-negative values, which avoids saturation and enables the gradient flow whenever the unit is active. Also, its second derivative is 0 almost everywhere, which avoids second-order effects. It has replaced the sigmoid and hyperbolic tangent as the main activation function for *hidden layers* in modern network architectures (GOODFELLOW *et al.*, 2016).

Despite the fact that its derivatives are not defined for $x = 0$, the ReLU function does not preclude optimization via gradient-based learning algorithms. Computer methods are usually subject of numerical errors, which means that, in most cases, the value never actually reaches 0; instead, it remains as a very small number avoiding the critical point of $x = 0$ for the derivatives. Additionally, the ReLU function has its derivatives defined for both the *positive* and *negative* parts of its domain, which allows, in practice, software implementations to simply assign the derivate of $x = 0$ numerically equal to one of its well defined sides (GOODFELLOW *et al.*, 2016).

3.2.4 ReLU Variants

A well known drawback of the ReLU function is that for negative values its gradient is always zero, which means that, if a *hidden unit* falls into a bad optimization region[1], it can get stuck into that point indefinitely since it will operate on the left side of the curve for most input samples and, consequently, there will be no gradients to update its parameters. Hence, the unit will be deemed "dead" – a problem known as *dead neuron*.

In order to alleviate this problem, some variations of the ReLU function have been proposed, which are explained in the sequence.

3.2.4.1 Leaky-ReLU

The main idea behind the Leaky-ReLU (MAAS *et al.*, 2013) function is to replace the flat negative side of the curve by a small negative slope, which prevents the *dead neuron* problem. Equation (3.3) demonstrates how a small parameter α can be used to create a non-zero slope for the negative side of the curve. In practical applications, α is chosen as a small number such as 0.01.

$$g(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha \cdot x, & \text{otherwise} \end{cases} \tag{3.3}$$

3.2.4.2 PReLU

The Parametric ReLU (HE *et al.*, 2015) is very similar to the Leaky-ReLU (MAAS *et al.*, 2013) in the sense that it adds negative slope to the left side of the function's domain. However, the PReLU proposes that the parameter α of Equation (3.3) should be learned by the network instead of receiving a fixed pre-determined value prior to training.

3.2.4.3 ELU

The Exponential Linear Unit (CLEVERT *et al.*, 2016) adds a log curve to the negative side of the ReLU's domain as given by:

$$g(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha \cdot (e^x - 1), & \text{otherwise} \end{cases} \tag{3.4}$$

where, similarly to the Leaky-ReLU, α is a fixed hyperparameter.

[1] This issue is usually caused by either a very large learning rate or a bad initialization, e.g., a large negative value for the bias term.

The advantages of the ELU against the Leaky-ReLU and PReLU is that it provides a noisy-robust deactivate state for the unit.

Figure 3.2 shows a comparison between the ReLU and its variants. In particular, Figure 3.2(b) focuses on the negative side of these functions, where the differences appear.

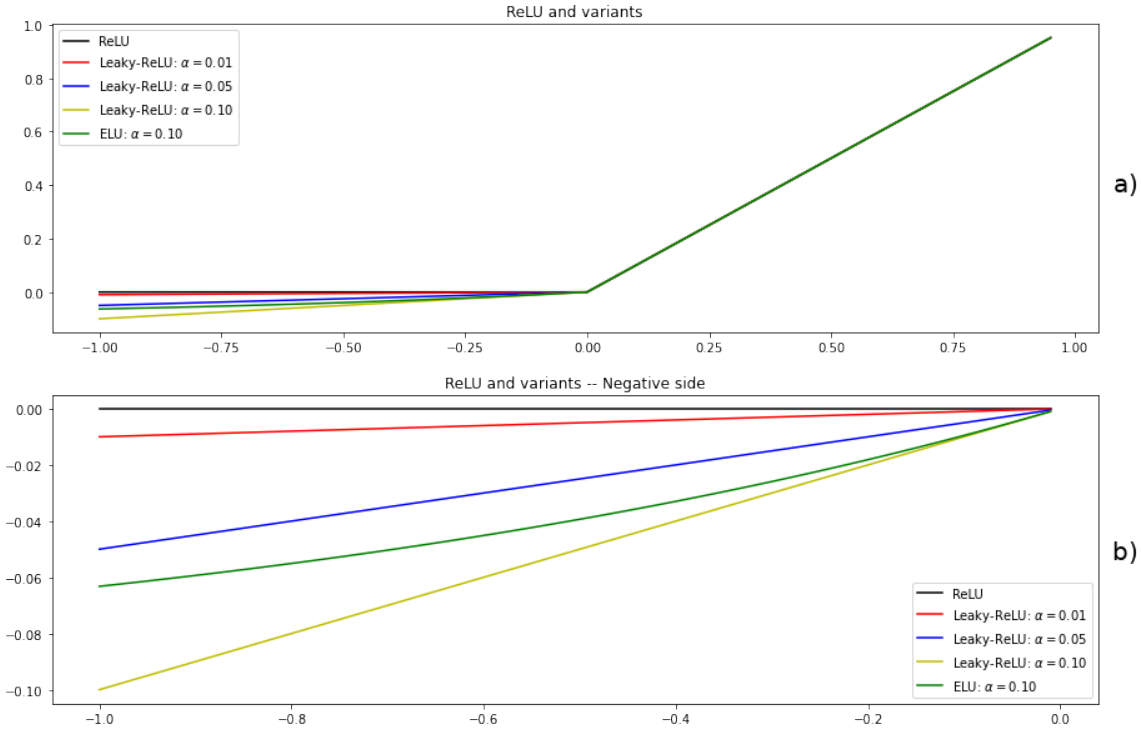

Figure 3.2 – **Comparison between ReLU and its variants.** The Parametric ReLU (HE *et al.*, 2015) is a special case of the Leaky-ReLU (MAAS *et al.*, 2013) when the parameter α is learned by the network and, thus, is not displayed in this figure.

3.2.5 Swish

Even though the ReLU variants propose ideas to fix the function's main issues, their results are inconsistent and may vary from dataset to dataset. The Swish function (RAMACHANDRAN *et al.*, 2017a; RAMACHANDRAN *et al.*, 2017b) was proposed in 2017 as an attempt to definitively replace for the ReLU function. It is defined by $g(x) = x \cdot \sigma(x)$ and closely resembles the ReLU curve as can be observed in Figure 3.1. Nonetheless, it is non-monotonic and bounded in the left side of its domain, which, consequently, alleviates the *dead neuron* problem whilst being self-gated by the sigmoid function of the input value. The authors claim consistent improvements over the ReLU and its variants across several deep learning tasks.

3.3 Convolution

As the name suggests, the convolution operation is the main foundation of the CNNs. The convolution is a function that operates on two real-valued sequences producing another sequence as output. Essentially, it can be interpreted, for a single time-step or index t, as the integral or summation of all the products between a fixed sequence x and a second sequence w, usually of smaller size than x and commonly named *kernel*, that is evaluated at time-step t but offset from $-\infty$ to ∞. The convolution formulation in its continuous form, which is denoted by $s(t) = (x * w)(t)$, is shown in Equation (3.5) (GOODFELLOW *et al.*, 2016).

$$s(t) = \int x(a) \cdot w(t-a) \cdot da \tag{3.5}$$

In most fields of knowledge such as physics, however, data cannot be acquired continuously. Alternatively, as occurs in digital images and time-series, data is usually sampled at constant intervals in a discrete fashion (GOODFELLOW *et al.*, 2016). Consequently, Equation (3.5) can be altered to obtain the discrete convolution:

$$s(t) = \sum_{a=-\infty}^{\infty} x(a) \cdot w(t-a) \tag{3.6}$$

Nevertheless, in real applications, the sequences are finite and the summation of Equation (3.6) cannot go on indefinitely. Therefore, in practice, the operations occur only in the valid positions where the input sequence is defined, which is known as the *VALID* convolution format. Alternatively, one can also pad the edges of the input vector with 0 or any other value in order to forcefully get an output sequence of the same size as the input, which is known as the *SAME* format and is commonly used in modern CNN architectures (DUMOULIN; VISIN, 2018). Figure 3.3 shows an example of an unidimensional convolution operation, in the *SAME* format, i.e., zero-padding of size 1 for kernel size equals 3, between an input sequence x and a kernel w. The sequence x remains fixed as the kernel slides across. For each position, the output unit is computed as the summation over all products between input's units and their corresponding kernel's units.

Convolution can also be extended to a *N-D* multidimensional form by sliding the kernel across all dimensions, as described for a 2-D image I and kernel K by Equation (3.7) (GOODFELLOW *et al.*, 2016). Particularly, the 2-D convolution is very useful for images and audio spectrograms, whereas 3-D convolution can be used on other types of data such as point-clouds and MRIs (Magnetic Resonance Imaging), which are composed of volume elements (voxels), the 3-D counterparts of the pixels.

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n) \cdot K(i-m, j-n) \tag{3.7}$$

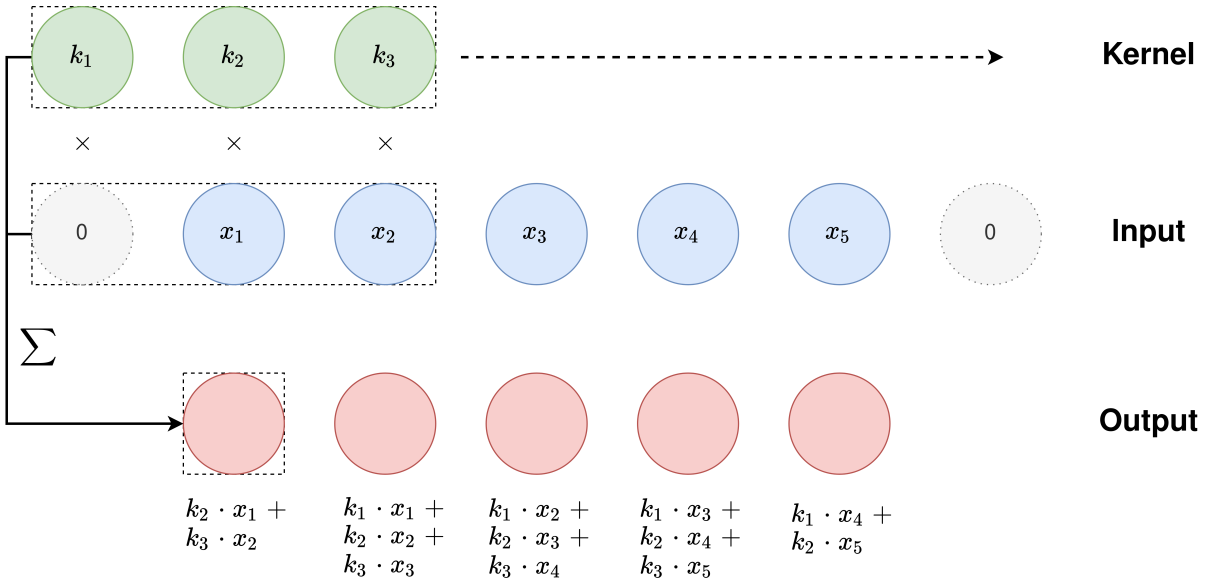

Figure 3.3 – **1D Convolution Operation:** Example of a 1D convolution with sequence and kernel with size of 5 and 3 units, respectively. This figure uses padding equals to 1, which yields the *SAME* convolution format. Figure inspired in (GOODFELLOW *et al.*, 2016).

There are many reasons why convolutions have become the standard operation used in Machine Learning algorithms for computer vision tasks, but certainly the most important are **sparse connectivity**, **parameter sharing** and **equivariance to translation** (GOODFELLOW *et al.*, 2016).

3.3.1 Sparse Connectivity

Unlike dense matrix multiplications, which are the basis for traditional neural network layers (*dense/fully-connected layer*), convolution only uses spatially-local responses to compute an output unit. Consequently, for each input unit, the number of interactions is constant and strictly defined by the *kernel size* k, which is usually orders of magnitude smaller than the image's resolution (typical values are 1[2], 3, 5 and 7). Hence, convolutions has $O(n \times k)$ runtime, where n is the number of input units. The matrix multiplication of *fully-connected layers*, on the other hand, have complexity of $O(n \times m)$ since each input unit is connected to all of the output nodes m. As an example, for images of resolutions as low as 256×256, dense matrix multiplication have more than 4B connections and, consequently, its usage easily become prohibitive (GOODFELLOW *et al.*, 2016). Figure 3.4 provides visual intuition on how the number of connections grow in both matrix multiplications and convolutions.

[2] In deep learning applications, convolutions with kernel size 1x1 are typically used to adapt the number of channels of a feature map.

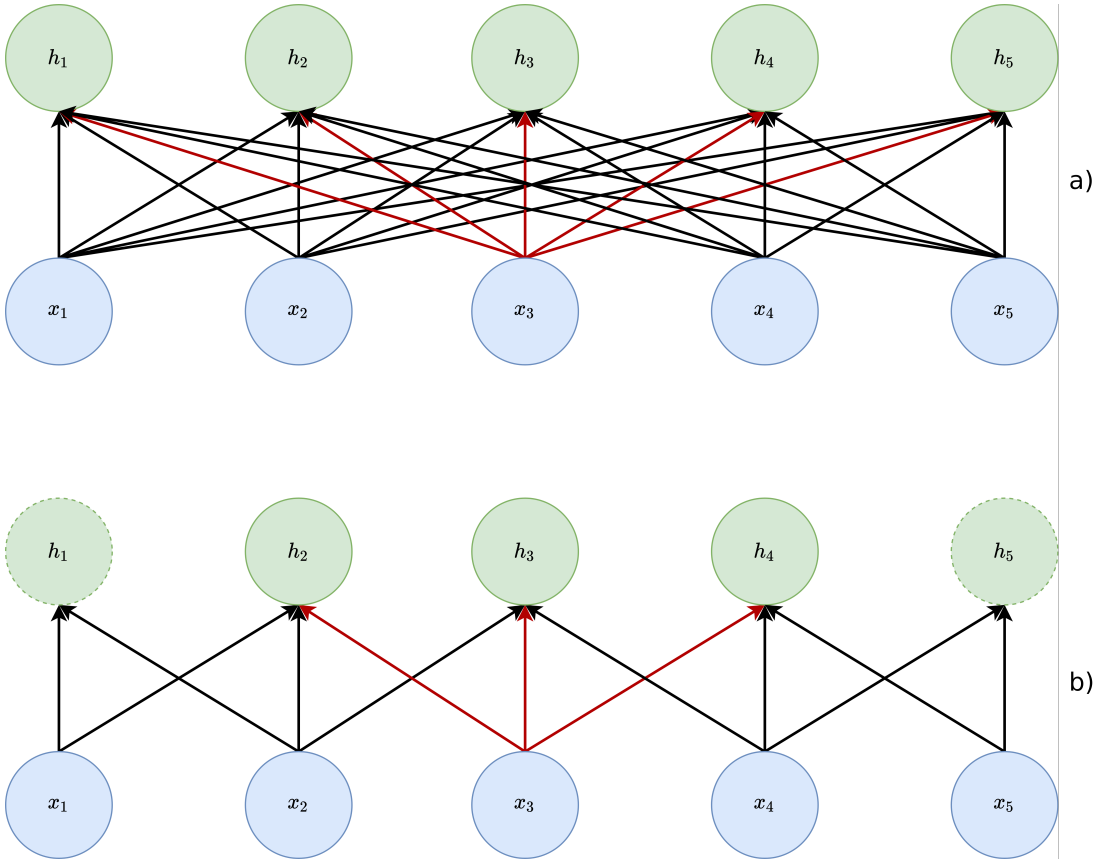

Figure 3.4 – **Matrix Multiplication vs. Convolution:** Number of connections and parameters for the *dense* or *fully-connected layer* increase linearly with the number of units, whereas for the convolution it is fixed and equals to the kernel size, in this case 3. Dashed circles show units that only exist when padding is used. Figure inspired in (GOODFELLOW *et al.*, 2016).

3.3.2 Parameter Sharing

In *dense layers*, the number of parameters is equal to the number of connections, meaning that every interaction is unique and no weight $w_{i,j} \in \mathbf{w}$ is ever reused. On the contrary, in convolutions the parameters (*kernel values*) are shared across all image positions, even though we have $n \times k$ interactions. Therefore, the total number of parameters is only equal to k. For images or other types of data where patterns are sometimes repetitive, sharing parameters leads to memory and statistical efficiency. For instance, convolutional kernels that capture textures or edges are not specific to a single image region, instead they can easily be useful in many different image parts as shown in Figure 3.5, where a simple kernel $\mathbf{k} = [-1, 1]$ is used to detect vertical edges in the entire image. In this example, the input image (left) has resolution of 280×320 pixels. Therefore, a total of $280 \times 319 \times 3 = 267,960$[3] floating point operations are required to compute the operation, but only 2 parameters are required for the whole image (GOODFELLOW *et*

[3] Here we use 319 in the calculation instead of 320 because we are neglecting the borders. Also 3 operations, two multiplications and one addition, are required per output pixel.

Figure 3.5 – **Vertical Edge Detector Example:** In this image, a simple hand-designed convolutional kernel was employed to detect vertical edges across the entire image. Image inspired in (GOODFELLOW *et al.*, 2016).

al., 2016).

3.3.3 Translation Equivariance

The property of equivariance means that if the input is affected by a transformation, the output will be affected by the exact transformation. Particularly, the equivariance of a function $f(x)$ to a transformation g occurs if $f(g(x)) = g(f(x))$. For convolutions, the equivariance is present when g is a translation transformation, e.g., horizontal and/or vertical shifts of the input image (GOODFELLOW *et al.*, 2016). For example, considering a synthetic image $\mathbf{X} \in \mathbb{R}^{3\times5}$ and a vertical edge detector kernel $\mathbf{K} \in \mathbb{R}^{3\times2}$, their *VALID* convolution is given by:

$$\mathbf{X} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix} \tag{3.8a}$$

$$\mathbf{K} = \begin{bmatrix} -1 & 1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix} \tag{3.8b}$$

$$(\mathbf{X} * \mathbf{K})(i,j) = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 1 & 0 & -1 & 0 \\ 1 & 0 & -1 & 0 \end{bmatrix} \tag{3.8c}$$

If we translate the image $\mathbf{X}$ horizontally by 1 pixel, we have:

$$\mathbf{X}' = I(\mathbf{X}) = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \tag{3.9a}$$

so that

$$(\mathbf{X}' * \mathbf{K})(i,j) = \begin{bmatrix} 0 & 1 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 1 & 0 & -1 \end{bmatrix}, \tag{3.9b}$$

which is essentially the same as translating the result of Equation (3.8c) by 1 horizontal pixel, i.e., $(I(\mathbf{X}) * \mathbf{K})(i,j) = I(\mathbf{X} * \mathbf{K})(i,j)$.

3.3.4 Receptive Field

It is well known that convolution is a powerful tool in computer vision due to its core inductive bias that, for a pixel (i,j), the most important information are present in the neighbourhood of such pixel, $\mathcal{N}_{i,j}$. Nonetheless, considering such ability of only attending to a small field spatially located around the kernel, it is natural to expect that convolutions are not able to capture long range dependencies when using small kernel sizes. This limitation, however, can be overcome by stacking multiple convolutions on top of each other as depicted in Figure 3.6, considering the simple scenario of two layers with *kernel size* equals 3.

By using multiple convolutions operating on the output of its predecessor, each new unit indirectly increases its receptive field, i.e., the area of the original image that is "perceived" by an unit (or, in other words, the area that affects the output of an unit), and, consequently, is able to also capture more complex features and long range dependencies.

3.3.5 Convolutions in CNNs: The Conv Layer

The convolutional layer – the main building block of CNNs – is, as the name suggests, based on the convolution operation. In traditional computer vision applications, the kernel values are fixed and designed depending on the application at hand. For example, the Sobel filter (FELDMAN *et al.*, 1969) can be used to detect borders in the image. In CNNs, however, the kernel values are not previously fixed. Instead, they are initialized pseudo-randomly and learned by the network via gradient-based optimization algorithms during the training phase. Thus, ideally, the network learns – through data – the best values for each convolution filter (GOODFELLOW *et al.*, 2016).

Another substantial difference from the traditional application is that the conv layer operates simultaneously on multiple input channels, i.e., sub-images. This means that for a RGB image $\mathbf{X} \in \mathbb{R}^{H \times W \times 3}$, a 3x3 convolutional kernel is extrapolated to have the same number of channels as the input image $\mathbf{X}$, i.e., $\mathbf{K} \in \mathbb{R}^{3 \times 3 \times 3}$. For each channel,

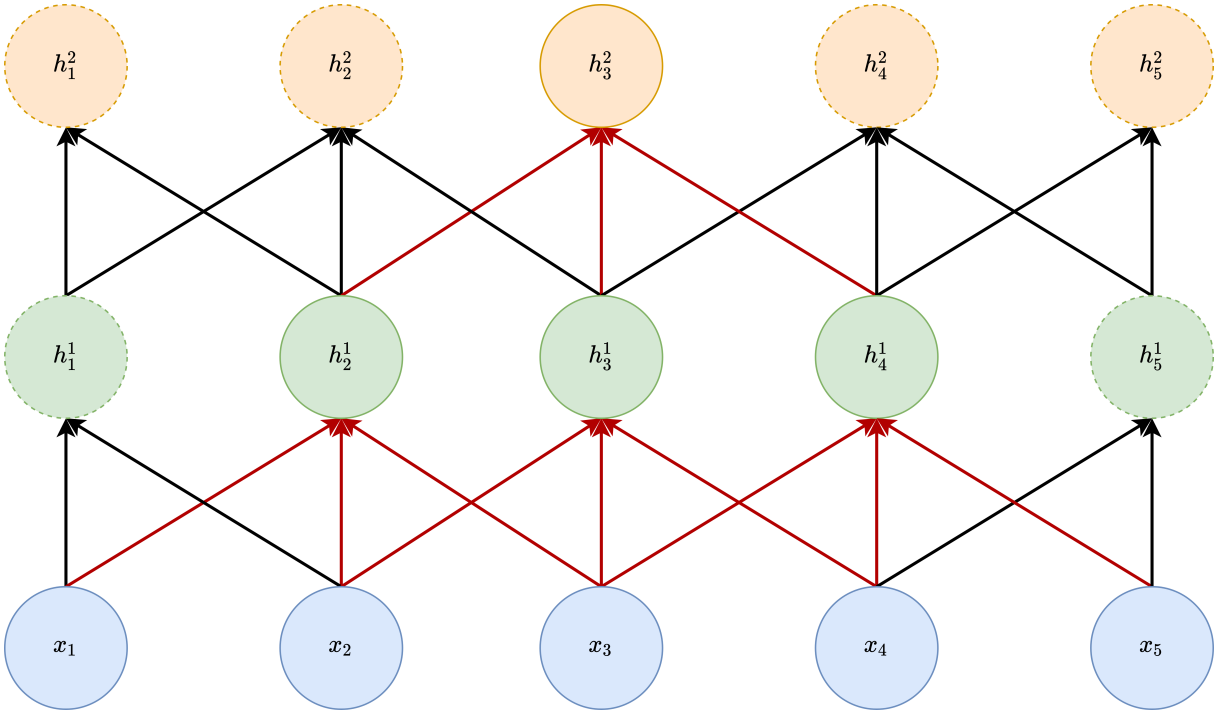

Figure 3.6 – **Convolution's Receptive Field:** Deeper convolutional layers naturally increase their receptive field, which is usually useful to capture long-range dependencies and more complex features. Dashed circles show units that only exist when padding is used. Figure inspired in (GOODFELLOW *et al.*, 2016).

the operation proceeds exactly as presented in Section 3.3 and the final result is obtained by summing each channel's individual result. The complete equation for a single output channel can be described as follows:

$$(\mathbf{X} * \mathbf{K})(i, j, k) = \sum_k \sum_m \sum_n I(m, n, k) \cdot K(i - m, j - n, k) \tag{3.10}$$

Additionally, the convolutional layer can have multiple output channels, which are computed independently, one for each kernel of the convolutional layer. Thus, the number of learned parameters in a single layer can be expressed by $i \times h \times w \times o$, where i, h, w and o are the number of channels in the input image, the height and width of the convolution kernel and the number of output channels, respectively. Figure 3.7 shows the operation for an output composed of two channels.

In practice, in modern deep CNNs, the convolutional layer is usually followed by an activation function and a normalization layer, which comprises a convolution block. Also, after a certain number of blocks, CNNs usually reduce the feature map's spatial dimension, which is commonly implemented via pooling. Those concepts are better explained in the sequence.

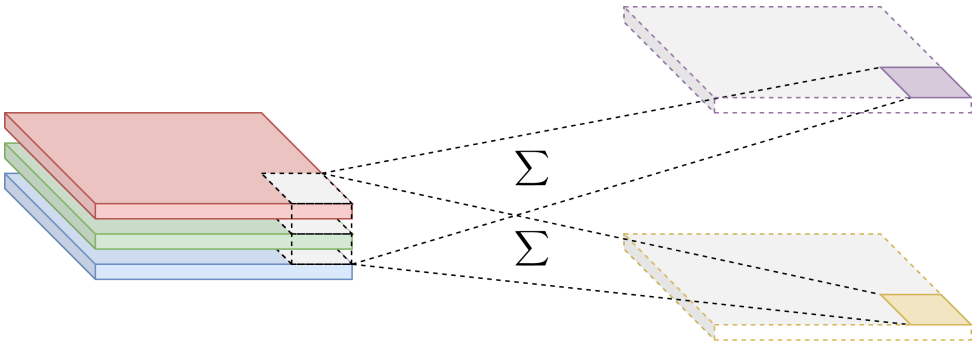

Figure 3.7 – **The Convolutional Layer.** Each output channel is generated using all input and kernel channels.

3.4 Pooling

Even for input resolutions as small as 224x224 pixels, the computational cost of using several convolutional layers is large. Also, since small kernel sizes are typically used to ease the computational burden, a large number of convolutional layers is required in order to increase the network's receptive field enough to capture global class-sensitive information. Consequently, the *pooling layer* was designed to reduce the feature map dimension throughout the network's forward propagation. By doing so, each convolution after the pooling layer operate on a smaller input feature map, which addresses both the aforementioned issues.

Similarly to the convolution, the *pooling* operation utilizes a sliding window algorithm that sweeps across the input feature map and, for each position, outputs the result of a *kernel* or *function* that operates on the local input values. The sub-sample ratio r can be determined as a function of the *padding* p, *kernel size* k and *stride* s used in the operation. For the PyTorch library (PASZKE *et al.*, 2017), the output size is given as follows:

$$H_{out} = f(k, s, p) = \left\lfloor \frac{H_{in} + 2p - k}{s} + 1 \right\rfloor \tag{3.11}$$

where H_{out} and H_{in} are the output and input height, respectively, and the operator $\lfloor \cdot \rfloor$ represents the mathematical *flooring* or rounding to the closest smaller integer. The width component can be determined analogously.

The two most common *pooling layers* are the *max pooling* and *average pooling*. As the name suggests, the former uses a *kernel* that returns the maximum value, whereas the latter performs the average of its input values.

Alternatively to *pooling layers*, some CNN architectures, such as the ResNet (HE *et al.*, 2016) family, resort to *strided* convolutions – where the kernel displacement

between consecutive positions during convolution can be greater than one – to reduce the input dimensions. Small differences aside, the result of adding a stride greater than 1 to a convolution is similar to adding an *average pooling layer* with the same stride prior to the convolution.

3.5 Batch Normalization

As neural networks grow deeper, many gradient-related issues may arise. As presented in Section 3.4, many non-linear activation functions, e.g., hyperbolic tangent and sigmoid function, saturate for extreme values of x. Consequently, the gradients tend to be close to zero, which especially affects deep networks, since the chain-rule is used to estimate the derivatives, and, thus, the gradients from layer to layer are multiplied and diminish exponentially, a problem known as the *vanishing gradient*, which, ultimately, slows down the training procedure and damages the model's performance (IOFFE; SZEGEDY, 2015).

A possible solution to the aforementioned problem would be to use non-saturating activation functions such as the ReLU (NAIR; HINTON, 2010) and Swish (RAMACHANDRAN *et al.*, 2017a; RAMACHANDRAN *et al.*, 2017b) along with careful initialization algorithms (GLOROT; BENGIO, 2010). Nonetheless, these functions are neither symmetric nor centered at $x = 0$, which skews the activation's distribution as data propagates throughout the network, and, eventually, can lead to numerical instabilities. Hence, small learning rate values are also required for stability (IOFFE; SZEGEDY, 2015).

The batch normalization layer (IOFFE; SZEGEDY, 2015) addresses those issues by performing standardization within a batch of samples, i.e., normalizing to zero mean and unit standard deviation. Also, after normalization, the layer applies an affine transformation defined by a learned set of parameters to, possibly, revert the normalizing operation. In CNNs, the normalization layer is usually added after each convolution in order to prevent the activation's norm of growing too much, which can lead to instabilities.

Formally, for a channel k, the standardization can be computed as follows:

$$\hat{x}^k = \frac{x^k - \mathrm{E}[x^k]}{\sqrt{\mathrm{Var}[x^k]}}, \tag{3.12}$$

where the expectation E and variance Var are computed on the mini-batch of images $\mathcal{B}$. Given the standardized input $\hat{x}$, the affine transformation is also performed by channel:

$$\hat{y}^k = \gamma^k \cdot \hat{x}^k + \beta^k, \tag{3.13}$$

where $\gamma^k \in \mathbb{R}$ and $\beta^k \in \mathbb{R}$ are parameters learned by the network and K is the total number of channels in the layer.

A major drawback of this approach is that the model's output is directly coupled with the mini-batch statistics, which is not desired at test time or inference. In order to solve this, Ioffe and Szegedy (2015) proposed to accumulate the mini-batch statistics – average and standard deviation – during training by means of exponential moving averages. Then, during inference, the acconlineumulated averages, which can be perceived as a good estimation of the training set's statistics, are used in Equation (3.12), which eliminates the cross-dependency between mini-batch samples.

Notoriously, the usage of batch normalization layers after convolutions smoothens the loss landscape (SANTURKAR *et al.*, 2019) and allows for larger mini-batch sizes and learning rates (DE; SMITH, 2020; BJORCK *et al.*, 2018), significantly reducing the training time (IOFFE; SZEGEDY, 2015). Additionally, since the expectation and variance of a mini-batch are not an exact representation of the entire training set's statistics, batch normalization layers provide a stochastic noise effect, which can be helpful to prevent overfitting (HOFFER *et al.*, 2017; LUO *et al.*, 2019). On the other hand, if a small mini-batch size is employed, its statistics can be non-representative of the whole training set, which can severely impact the model's performance (WU; HE, 2018).

Following (IOFFE; SZEGEDY, 2015), several different normalization layers have been proposed to alleviate the mini-batch dependency, such as Layer Normalization (BA *et al.*, 2016), Instance Normalization (ULYANOV *et al.*, 2016), Batch Renormalization (IOFFE, 2017) and Group Normalization (WU; HE, 2018). Even networks specifically designed to not require normalization layers while retaining their advantages have been recently proposed (BROCK *et al.*, 2021). Nevertheless, since most of modern convolution neural networks still rely on the batch normalization layer (IOFFE; SZEGEDY, 2015), these variants are not the main focus of this work and, thus, are not covered in this material.

4 Convolutional Neural Networks for Image Classification

In the last decade, CNNs (LECUN *et al.*, 1989) have been one of the standard tools present in almost every computer vision engineer's repertoire. This chapter is aimed to discuss the most relevant image classification CNN architectures to the scope of this work.

4.1 Residual Neural Networks (ResNets)

As aforementioned, most CNNs are created based on convolutional blocks that are usually comprised by a convolution, followed by a normalization layer and an activation function. The blocks are commonly stacked on top of each other forming a *layer*[1], which are interleaved by pooling layers to reduce the computational cost whilst increasing the network's receptive field allowing for more complex features to be captured. Nevertheless, even with intermediate normalization layers and good parameter initialization, gradient-vanishing problems still may occur for very deep networks hampering their performance. Figure 4.1 shows that a 56-layer network may present a higher associated test error than a 20-layer one. Contrary to what one might suspect, the issue is not caused by *overfitting* as the same phenomenon occurs for the train error (Figure 4.1, left-side graph), suggesting an optimization cause (HE *et al.*, 2016).

4.1.1 The Residual Connection

The core intuition of ResNet (HE *et al.*, 2016) lies on the fact that, since the weights are learned through data, a deeper network should always provide a superior or, at least, equal associated *train* error to that of a shallow network. Nonetheless, as observed in Figure 4.1, this does not always occur, which suggests that regressing to an identity mapping function via training is not trivial for the common non-linear convolutional blocks (HE *et al.*, 2016).

Accordingly, the main contribution of Residual Neural Network (ResNet) (HE *et al.*, 2016) was to propose a new convolutional block that can easily perform the identity mapping if required. In practice, this is easily implemented by providing a residual connection between input and output of the block via element-wise summation, as depicted

[1] In this case, the convolutional layer that we refer is not a single convolution operation but rather a collection of stacked convolutional blocks.

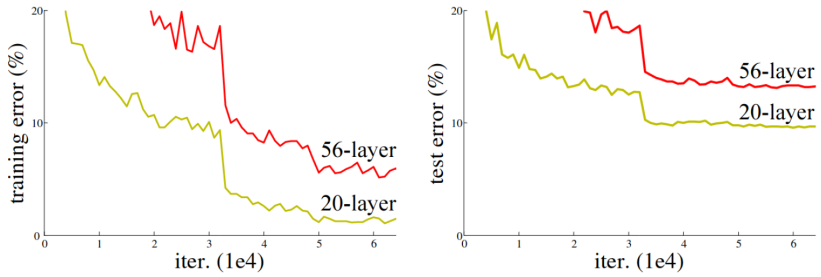

Figure 4.1 – **Depth comparison.** Comparison between a 20- and 56-layers networks trained on CIFAR-10 (KRIZHEVSKY *et al.*, 2009) for image classification task. Image extracted from (HE *et al.*, 2016).

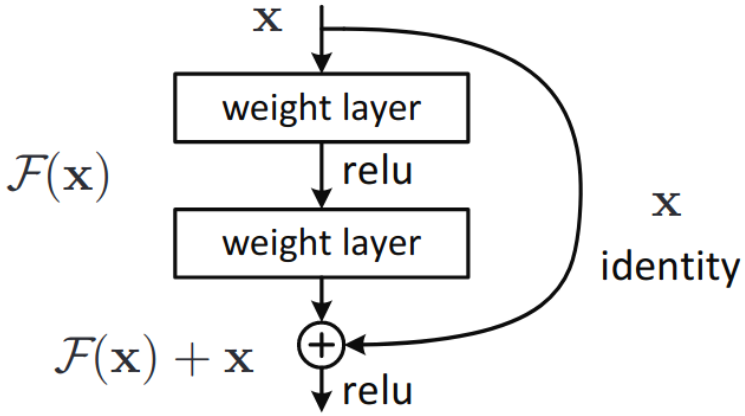

Figure 4.2 – **The residual connection.** The input of the convolutional block is directly added to its output, providing a *shortcut*. Image extracted from (HE *et al.*, 2016).

in Figure 4.2. This "shortcut" not only acts as an identity function but also provides an unobstructed path for the gradients, which also ease the optimization (HE *et al.*, 2016).

4.1.2 The Residual Blocks

There are two types of convolutional blocks used in residual networks: the *basic* and *bottleneck* blocks. The former is used in the two most shallow designs of the ResNet family – ResNet-18 and 34 –, whereas the second is used in the deeper and wider designs, such as ResNet-50, 101 and 152. We start by first introducing the basic block and elucidating its intrinsic limitations on wider networks, which is also the key motivation behind the bottleneck block.

4.1.2.1 The Basic Block

The basic block is comprised by two 3x3 convolutional layers followed by batch normalization (IOFFE; SZEGEDY, 2015) and ReLU (NAIR; HINTON, 2010) stacked

on top of each other. Additionally, the *shortcut* connection is employed by performing element-wise summation between the block's input and output as depicted in the left side of Figure 4.3. Since, in the basic block, the 3x3 convolutional layers do not modify the number of channels, the total number of parameters in the block is given by the product of the number of convolutions, the kernel width and height and input and output channels[2].

Particularly for two basic block with 512 and 2048 channels[3] we have $2 \cdot 3^2 \cdot 512^2 \approx 4.7$M and $2 \cdot 3^2 \cdot 2048^2 \approx 75.5$M parameters, respectively. It is easy to notice that the quadratic dependency on the number of channels is a limiting factor for wider networks, which are usually advised for datasets with a large number of classes such as ImageNet (1000 classes) (RUSSAKOVSKY *et al.*, 2015).

4.1.2.2 The Bottleneck Block

The bottleneck block was proposed to mitigate the parameter growth issue by temporally reducing the number of channels. Such aspect is accomplished by using a 1x1 convolution to reduce the number of channels by a ratio r prior to its more demanding 3x3 counterpart. After the features are extracted by the spatial 3x3 kernel, another pointwise 1x1 convolution is used to expand the number of channels again by the same factor r.

By employing a ratio $r = 4$, for example, the total number of parameters for the same 2048 channels layer is now given by $1^2 \cdot 2048 \cdot 2048/4 + 3^2 \cdot (2048/4)^2 + 1^2 \cdot 2048/4 \cdot 2048 \approx$ 4.5M parameters, which is almost 17 times smaller than the 75.5M parameters required by a basic block of the same width. The bottleneck block is depicted in the right-side of Figure 4.3.

4.1.3 Network Architecture

The ResNet (HE *et al.*, 2016) family is comprised by a stem block, the main *layers* and a final global pooling and fully connected layer.

The stem block is a simple 7x7 convolution with stride 2 followed by batch normalization (IOFFE; SZEGEDY, 2015), ReLU (NAIR; HINTON, 2010) and a 3x3 max pooling also with stride 2. The block aims to extract the initial 64 feature maps from the input image whilst reducing its dimension by a factor of 4 at the same time.

Following, four *convolutional layers* are applied where the first convolution of each *layer* usually doubles the number of channels and halves the feature map's dimension. More specifically, each *convolutional layer* is comprised of a number l_i of residual blocks

[2] In this case we disregard the batch normalizations' (IOFFE; SZEGEDY, 2015) parameters since they are order of magnitudes lower than that of the convolutions.

[3] 512 and 2048 channels are, respectively, the width of the last layer of ResNet-34 and 50.

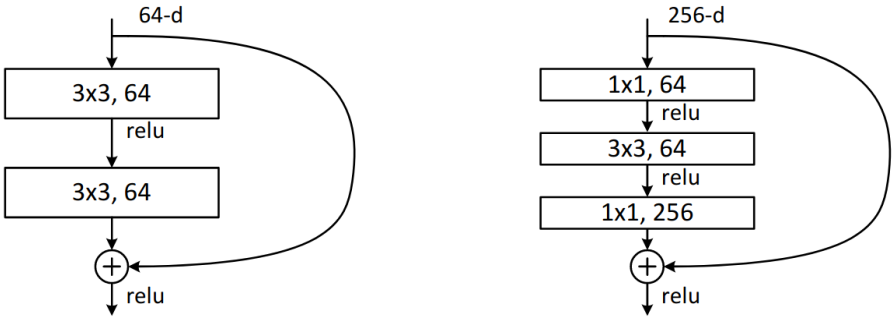

Figure 4.3 – **The residual blocks.** Left: the basic residual block, where two 3x3 convolutions are used without dimensional reduction. Right: The bottleneck residual block, two 1x1 convolutions are used to reduce and expand the number of the channels, before and after the 3x3 convolution, respectively. In both blocks, a 1x1 convolution may be employed to the *shortcut* connection in order to match the number of channels if required. Image extracted from (HE *et al.*, 2016).

where the number of blocks used and their type varies for each specific network in the family. The same occurs for the number of channels in the layer. In general, ResNets-18 and 34 use the basic residual block and have their width $w_i = \{64, 128, 256, 512\}$ for each *layer* i respectively. On the other hand, ResNets-50, 101 and 152 use the bottleneck residual block and have their width per layer configuration as $w_i = \{256, 512, 1024, 2048\}$. Notably, as the number of channels doubles at each *convolutional layer* transition, a pointwise convolution is used in the *shortcut* connection in order to adapt the width of the block's input.

Finally, a global average pooling is applied to reduce the 2D feature maps into a unidimensional vector, which is feed to a *dense* layer to match the number of classes in the classification task. The complete architecture for the ResNet-34 and a plain 34-layer network is shown in Figure 4.4.

4.2 Squeeze-and-Excitation Networks

Since the pioneer work of LeNet-5 (Lecun *et al.*, 1998), many different approaches to improve the CNNs' performance have been attempted, either by simply replacing some operators such as the activation functions or pooling layer, or by converting the common *convolution-activation-pooling* sequence into specially-tailored designs, e.g., (SZEGEDY *et al.*, 2015; TAN; LE, 2019). Nonetheless, improving channel representativity and reducing redundant features maps was only first explored in 2018 by the Squeeze-and-Excitation Network (SENet) (HU *et al.*, 2018).

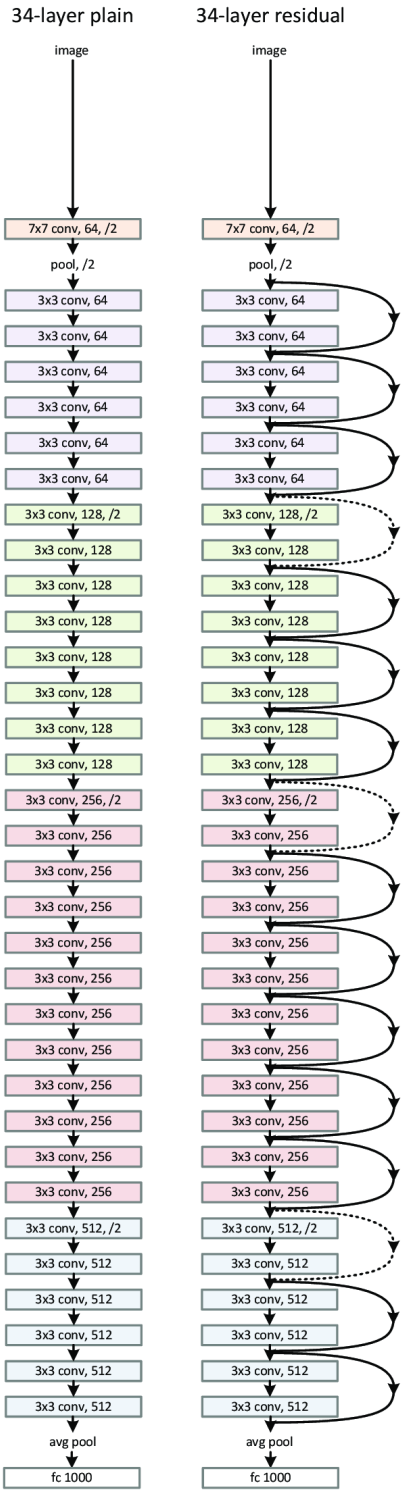

Figure 4.4 – **Comparison between a plain and residual 34-layer networks.** Right: the dashed arrow represents a 1x1 convolution employed to the *shortcut* connection in order to match the number of channels. Image adapted from (HE *et al.*, 2016).

Hu *et al.* (2018) proposed a new block called Squeeze-and-Excitation (*SEBlock*) responsible for enhancing the channel's representativity of feature maps by capturing the inter-channel dependencies, reducing redundant features and enabling the network to focus on more sensitive information. The SEBlock can be interpreted as a self-attention block, which helps the network to better allocate resources towards more informative components of the input signal (HU *et al.*, 2018). The block is comprised of two stages: *squeeze* and *excitation*.

4.2.1 Squeeze

The goal of the *squeeze* stage is to create a vector $\overset{*}{x} \in \mathbb{R}^C$ containing a single scalar representation for each of the 2D feature maps $\mathbf{X} \in \mathbb{R}^{H \times W \times C}$. This is accomplished by applying a global pooling operation μ, which is defined for a single channel $c \in C$ according to Equation (4.1). The *squeezed* vector $\overset{*}{x}$ is a useful *global* representation of the features that each channel attempts to capture and it can be perceived as the channel-wise statistics of the feature map $\mathbf{X}$.

$$\mu(\mathbf{X}_c) = \frac{1}{W \cdot H} \cdot \sum_{i=1}^{H} \sum_{j=1}^{W} \mathbf{X}_c(i, j) \tag{4.1}$$

4.2.2 Excitation

Once a statistical representation for each channel is attained, the *excitation* stage of the *SEBlock* is utilized to map the inter-channel dependencies and re-calibrate the original feature map $\mathbf{X}$ accordingly. The correlation between channels can be easily captured by a fully-connected layer or an 1x1 convolution f. However, such operations are only able to capture linear dependencies. Thus, a non-linear activation function such as ReLU (NAIR; HINTON, 2010) and a second 1x1 convolution are also used. In practice, in order to reduce the number of parameters, a bottleneck layer with ratio r similar to that of ResNet (HE *et al.*, 2016) is used, i.e., the first 1x1 convolution f_1 reduces the number of channels from C to C/r , which is, then, increased back to C in the second convolution f_2. Finally, a *sigmoid* activation function σ, defined by Equation (3.1), is applied to the result, bounding its range between 0 and 1, which is, then, multiplied by the original feature map $\mathbf{X}$ in order to re-calibrate its channels.

Formally, the *excitation* transformation $\mathbf{F}_{ex}$ is defined as follows:

$$\mathbf{F}_{ex}(\overset{*}{x}, \mathbf{W}_1, \mathbf{W}_2) = \sigma(\mathbf{W}_2 \cdot \delta(\mathbf{W}_1 \cdot \overset{*}{x})), \tag{4.2}$$

where $\mathbf{W}_1 \in \mathbb{R}^{\frac{C}{r} \times C}$ and $\mathbf{W}_2 \in \mathbb{R}^{C \times \frac{C}{r}}$ are the f_1 and f_2 convolutions' weights, respectively, and δ represents the ReLU (NAIR; HINTON, 2010) activation function. The complete

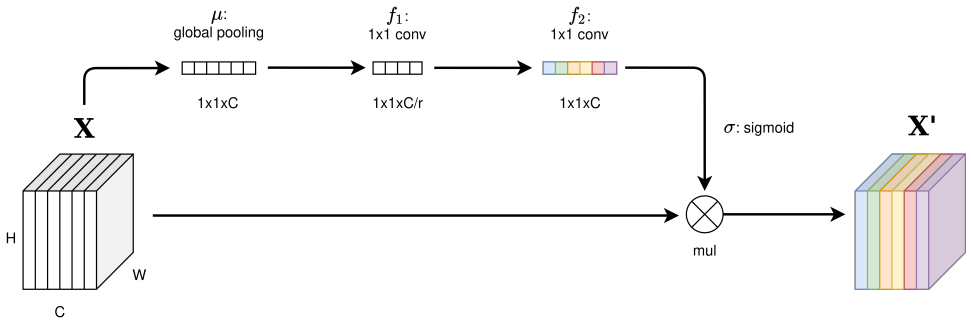

Figure 4.5 – **Squeeze-and-Excitation Block:** The original feature map $\mathbf{X}$ is subject to a self-attention transformation based on its own content, which allows it to reduce feature redundancy and to focus on most sensitive information along the channel dimension. ReLU activation function between f_1 and f_2 is omitted. Figure inspired in (HU *et al.*, 2018).

SEBlock operation can be written as:

$$\mathbf{X}' = \mathbf{F}_{SE}(\mathbf{X}) = \mathbf{X} \odot \mathbf{F}_{ex}(\mu(\mathbf{X})), \tag{4.3}$$

where $\odot$ stands for the Hadamard product. The whole operation is depicted in Figure 4.5.

4.2.3 SENet Architecture

The SENet architecture is designed based on the ResNet (HE *et al.*, 2016) family. In fact, even the depth configuration is the same, i.e., SENet-50, 101, 152. The main difference between ResNet (HE *et al.*, 2016) and SENet (HU *et al.*, 2018) is that the latter employs a Squeeze-and-Excitation block at the end of each bottleneck residual block as evinced in Figure 4.6.

4.3 EfficientNet

Prior to EfficientNet (TAN; LE, 2019), most works scaled the network's parameters arbitrarily and independently. The ResNet (HE *et al.*, 2016) model family, for example, generally doubles the network depth while keeping the width and input resolution constant. Nonetheless, this approach can be sub-optimal since the design parameters are interdependent. For example, by increasing the input resolution, one should also consider scaling up width and depth. The increased width or number of channels per layer allows the network to capture more fine grained features, better exploiting the higher level of detail from bigger input sizes. Additionally, with high resolutions, the apparent size of objects in the image increase, requiring a higher receptive field, which can be attained by increasing the depth of the network.

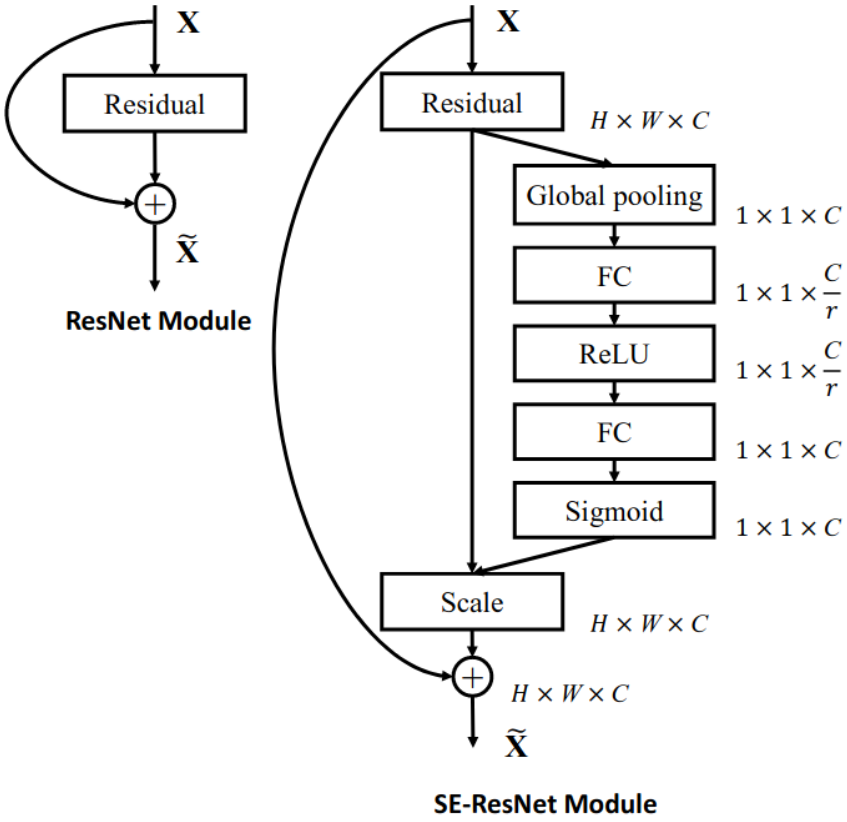

Figure 4.6 – **Comparison between the residual block and SE-augmented residual block.** Image extracted from (HU *et al.*, 2018).

In the remainder of this section we present the compound scaling system introduced by EfficientNet (TAN; LE, 2019), its basic convolutional block and the final network architecture.

4.3.1 Compound Scaling

Tan and Le (2019) proposed to use a compound scaling system, which scales network's depth, width and input resolution together according to:

$$\text{depth: } d = \alpha^{\phi}$$
$$\text{width: } w = \beta^{\phi}$$
$$\text{resolution: } s = \gamma^{\phi}$$

where ϕ is a user defined parameter that controls the overall scale of the network and α, β and γ are constants that control along with ϕ the depth, width and resolution growth, respectively.

Considering that the computational cost of convolutions in CNNs usually dominate that of other layers, by using the compound scaling, the FLOPs is expected to increase proportionally to $(\alpha \cdot \beta^2 \cdot \gamma^2)^\phi$ (TAN; LE, 2019). Accordingly, the authors restrained $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$ so that the FLOPs increase would be 2^ϕ. By fixing $\phi = 1$ and performing grid-search on the baseline model EfficientNet-B0 (TAN; LE, 2019), the best values of $\alpha = 1.2$, $\beta = 1.1$ and $\gamma = 1.15$ were found. The following models, from B1 to B7, were created accordingly by scaling up the value of ϕ.

Remarkably, despite not included in the compound scaling formulation, the authors also scale the dropout (SRIVASTAVA *et al.*, 2014) probability since regularization should also increase for higher model capacities.

4.3.2 MBConv Block

The compound scaling system is paramount when scaling up networks. Nonetheless, having a good convolutional block as the main network's building block is also critical.

Inspired on MobileNetV2 (SANDLER *et al.*, 2018), the EfficientNet's main block is based on the mobile inverted residual block (MBConv) (SANDLER *et al.*, 2018; TAN *et al.*, 2019), which is similar to the residual block of the ResNet (HE *et al.*, 2016) family, but inverts the channel bottleneck. The residual bottleneck block (HE *et al.*, 2016) reduces the number of channels by a ratio r in the first convolution (1x1), allowing a second convolution (3x3) to operate on a reduced width, which is, then, expanded back by the third convolution (1x1). Conversely, the inverted residual block (SANDLER *et al.*, 2018) operates on a lower default width, which is increased for the 3x3 convolution. Additionally, it replaces the 3x3 convolution by its depthwise separable (VANHOUCKE, 2014; CHOLLET, 2017) counterpart, thus significantly reducing the number of parameters learned by the network.

EfficientNet (TAN; LE, 2019) further extends the MBConv by adding attention with the SE block (HU *et al.*, 2018) and using Swish (RAMACHANDRAN *et al.*, 2017a; RAMACHANDRAN *et al.*, 2017b) as activation function. The adapted MBConv block is displayed in Figure 4.7.

4.3.3 Network Architecture

From the MBConv block of Figure 4.7, the baseline network architecture, EfficientNet-B0 (TAN; LE, 2019), was designed via neural-architecture search methods (TAN *et al.*, 2019). The complete EfficientNet-B0 architecture is presented in Table 4.1.

The subsequent models, B1 to B7, can be derived from Table 4.1 by changing the resolution s, width w and depth d as a consequence of increasing ϕ. The characteristics

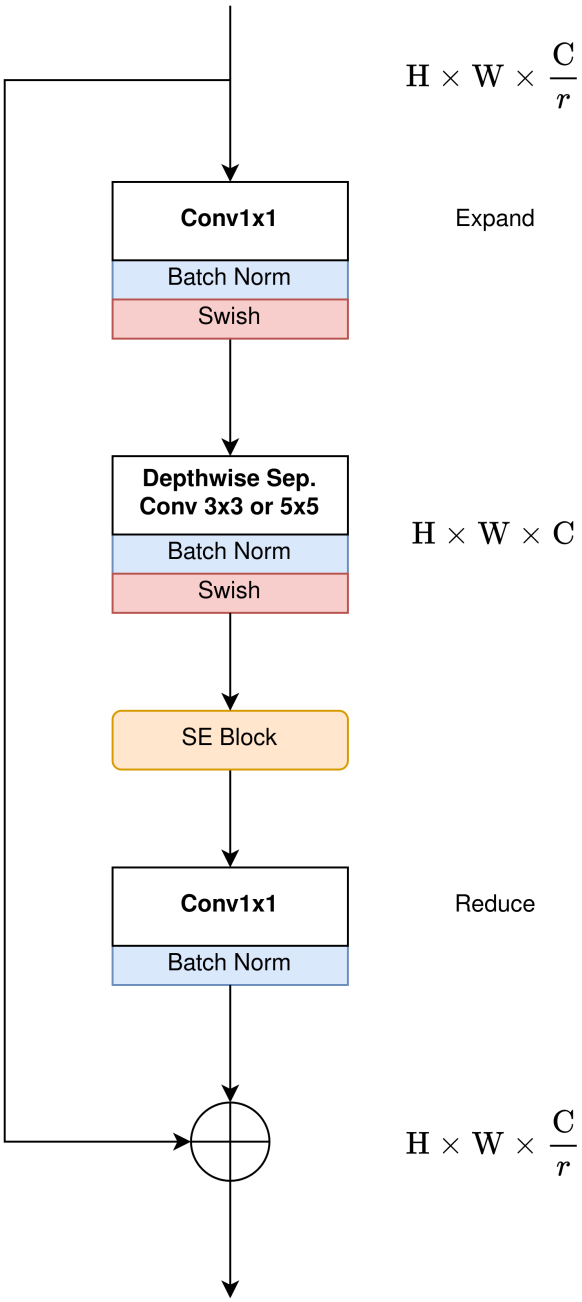

Figure 4.7 – **MBConv Block variant used in EfficientNet.** Figure inspired in (TAN; LE, 2019).

Table 4.1 – EfficientNet-B0 (TAN; LE, 2019) Architecture

Stage	**Operator**	**Kernel**	**Expansion Ratio** r	**Resolution** s	**# Channels** w	**# Layers** d
1	Conv	3x3	-	224^2	32	1
2	MBConv	3x3	1	112^2	16	1
3	MBConv	3x3	6	112^2	24	2
4	MBConv	5x5	6	56^2	40	2
5	MBConv	3x3	6	28^2	80	3
6	MBConv	5x5	6	28^2	112	3
7	MBConv	5x5	6	14^2	192	4
8	MBConv	3x3	6	7^2	320	1
9	Conv & Pooling & FC	1x1	-	7^2	1280	1

Table 4.2 – EfficientNet (TAN; LE, 2019) model family

Model	Multiplier			Dropout prob.
	Width w	Depth d	Resolution s	
B0	1.0	1.0	1.00	0.2
B1	1.0	1.1	1.07	0.2
B2	1.1	1.2	1.16	0.3
B3	1.2	1.4	1.34	0.3
B4	1.4	1.8	1.70	0.4
B5	1.6	2.2	2.04	0.4
B6	1.8	2.6	2.36	0.5
B7	2.0	3.1	2.68	0.5

of the complete EfficientNet (TAN; LE, 2019) family are summarized in Table 4.2.

In this chapter, we covered the most relevant deep networks used in image classification tasks and the rationale behind their designs. Those architectures are also the foundation for many downstream tasks in computer vision such as segmentation, object detection and keypoint estimation. In the sequence, we show how these different types of encoders can be employed to tackle the task of semantic segmentation.

5 Convolutional Neural Networks for Semantic Segmentation

Given the great results achieved by CNNs in image classification tasks in the last decade, especially in famous computer vision benchmark challenges such as ImageNet (RUSSAKOVSKY *et al.*, 2015), it was only a matter of time for this technology to be adopted to many other computer vision tasks such as semantic segmentation and object detection. Particularly for segmentation, many approaches were designed to adapt the coarse prediction of CNNs to denser regions, ultimately progressing to pixel-wise inference.

In this chapter, a few segmentation architectures that are most relevant for our work are listed and explained. Additionally, we briefly introduce a few popular evaluation metrics and loss functions employed in segmentation models.

5.1 Hypercolumns

Hypercolumns (HARIHARAN *et al.*, 2014a) is one of the first efforts to adapt image classification architectures for the purpose of object segmentation and other denser prediction tasks. Although this work appeared in the same period as *fully convolutional networks* (LONG *et al.*, 2014), the scope proposed by Hariharan *et al.* (2014a) is restricted to predicting the segmentation mask only for pre-selected regions with high likelihood of having an object. Thus, an input image must first be pre-processed by a region proposal algorithm and, only then, each region patch extracted from the original image can be segmented by Hariharan *et al.* (2014a). Even with such limitation, Hypercolumns is quite important to the development of our work since part of our solution is based on its architecture design and rationale.

The idea behind Hypercolumns consists in realizing that, unlike image classification, semantic segmentation tasks heavily rely on precise feature localization and high-frequency information. Common CNNs' architectures for image classification are comprised of alternating layers of convolutional blocks[1] and pooling operations. Consequently, feature maps' resolution is usually halved at each new layer, reducing the computational cost for subsequent convolutions and increasing their receptive fields at the expense of sacrificing high-frequency information and precise feature localization.

[1] We used the term convolutional block to refer to a series of convolutions, activation functions and, optionally, batch normalization layers stacked on top of each other.

5.1.1 The *Hypercolumn* Layer

In order to preserve the precise localization of features throughout the network's forward pass, Hariharan *et al.* (2014a) propose to concatenate, for each pixel x_i^L at spatial position i of the last layer L, the activations of all feature maps at the same position for all[2] layers above L creating a *hypercolumn*[3] at pixel i. Unlike regular CNN architectures for which only the last/deepest layer is used for classification, Hypercolumns allows the network to also incorporate intermediate activations, which are richer in high-frequency content, into the classifier layer.

In practice, since each intermediate feature map has a different spatial resolution, defining a hypercolumn with respect to each pixel position i is ambiguous. Thus, to solve this issue, a bi-linear upsampling layer is applied to each feature map prior to concatenation in order to match the resolution. If desired, a 1x1 convolution can also be applied in order to change the number of channels of each feature map, which also allows a summation merge strategy to be used instead of concatenation. The hypercolumn concept can be better understood with the aid of Figure 5.1.

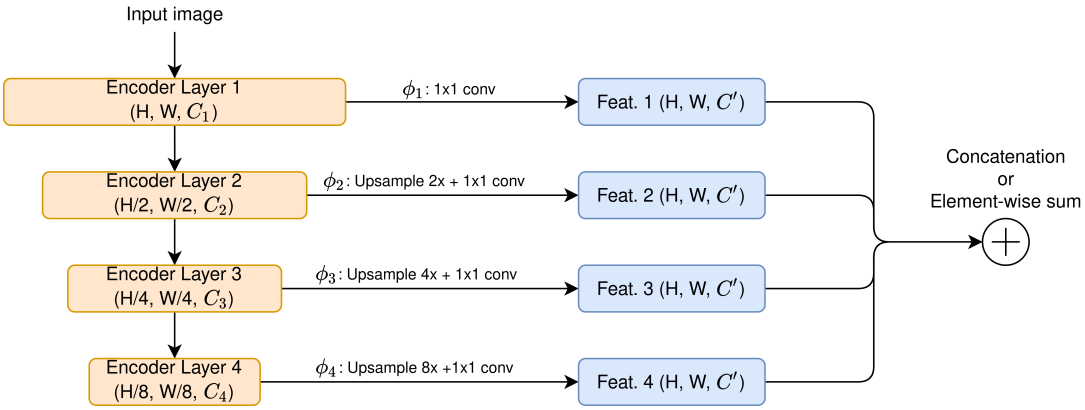

Figure 5.1 – **Hypercolumn concept.** The hypercolumn layer first adapts each feature map's resolution and number of channels with bi-linear upsampling and 1x1 convolutions, respectively. Then, the feature maps are fused, either by concatenation or element-wise sum, into a common representation, which is the input of the pixel-wise classifier. Figure inspired by Hariharan *et al.* (2014a).

5.2 Fully Convolutional Network

In the three-year period that succeeded 2012, several works were published proposing different methods to bridge the gap between coarse and dense predictions

[2] In practice, not all layers are considered. The first convolutional layer is usually discarded since it has very low semantic content.

[3] The name *hypercolumn* comes from neuroscience where it is used to described a set of V1 neurons forming a column, which are sensitive to edges at multiple frequencies and orientations.

(PINHEIRO; COLLOBERT, 2013; GANIN; LEMPITSKY, 2014; CIREsAN *et al.*, 2012; Farabet *et al.*, 2013; HARIHARAN *et al.*, 2014b; HARIHARAN *et al.*, 2014a). Among them, the *fully convolutional network* (FCNet) (LONG *et al.*, 2014) is worth mentioning given its capacity of producing a segmentation mask for an entire image of any resolution at once without the need of any extra complicated pre- or post-processing.

Contemporary to FCNet, many works employed local classifiers (Farabet *et al.*, 2013; HARIHARAN *et al.*, 2014b) or patchwise classification (GANIN; LEMPITSKY, 2014; Feng Ning *et al.*, 2005; CIREsAN *et al.*, 2012; PINHEIRO; COLLOBERT, 2013) to perform segmentation on regions extracted from the input image. Nonetheless, such approaches suffer significantly from the trade-off between context and precise localization – as the patch size diminishes, the local classifiers are able to predict more complex and precise borders and edges at the expense of lacking context and semi-global information, which are important semantic cues for classification. Figure 5.2 depicts an example of how context can be helpful for classification, where a) shows the entire image of a marina with cars and boats, while in b) two regions – depicting a boat and a car – are extracted from the original image. Without additional context, it is really challenging to classify the vehicles in the patches, whereas in c), with extra context, it is clearly easier.

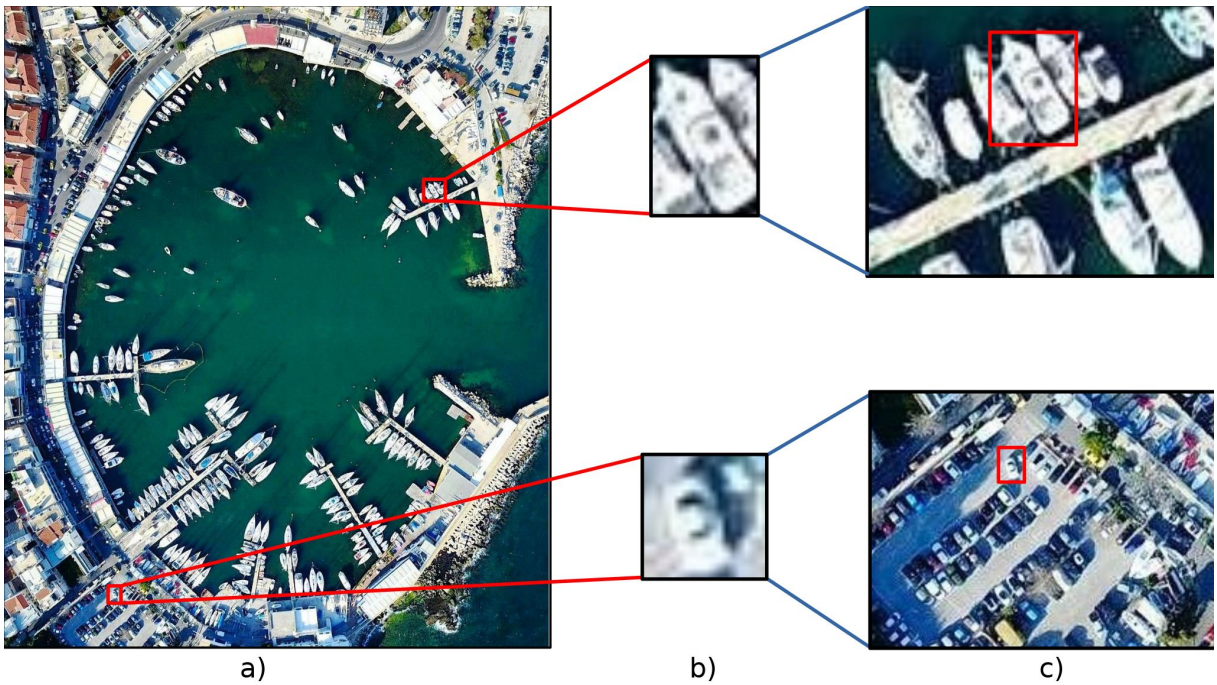

Figure 5.2 – **Importance of context and semi-global information.** This example evinces how context and semi-global image information can be important semantic cues for pixel classification.

5.2.1 From Dense Layers to Convolutions

The key concept behind FCNet simply consists in removing the global pooling layer and replacing the last *dense* layers of common image classification architectures by convolutions[4]. By performing such modifications, the network is able to predict a heatmap $\mathbf{H} \in \mathbb{R}^{\frac{H}{S} \times \frac{W}{S} \times C}$ for each class $c \in \mathcal{C}$ instead of a distribution as depicted in Figure 5.3. Although the output heatmap has a resolution smaller than the original image by a stride factor of S due to pooling operations, it can be upsampled back via bi-linear interpolation.

The network, then, can be trained end-to-end with dense supervision, which is simply accomplished by computing the loss function at each individual pixel position of the output heatmap. The result is then averaged and backpropagated (RUMELHART *et al.*, 1986) similarly to any other CNN training process.

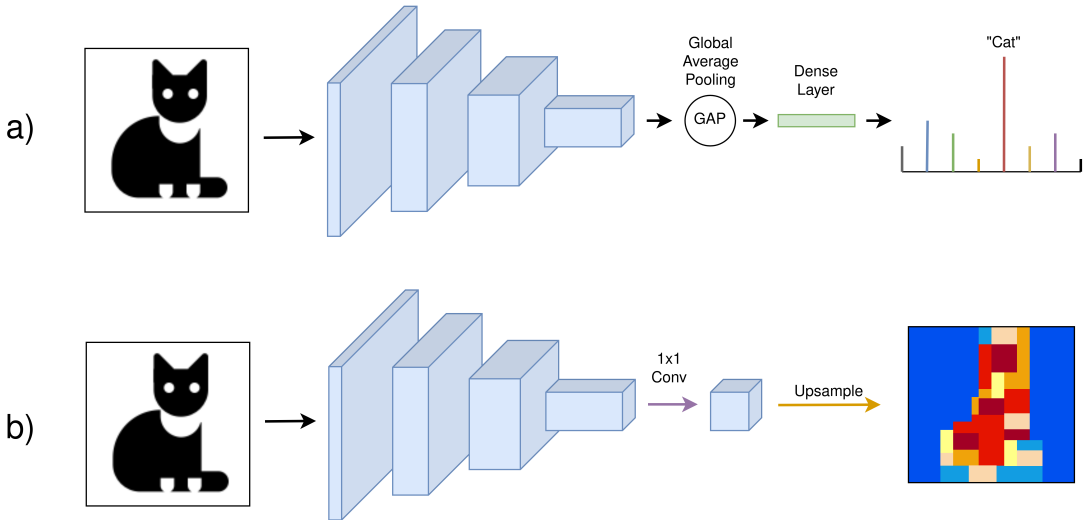

Figure 5.3 – **Fully convolutional network.** When the final pooling and fully-connected layers of a) are replaced by convolutions, the network is able to output a heatmap, as in b), enabling end-to-end dense training by averaging the loss at every output pixel. Figure inspired in (LONG *et al.*, 2014).

5.2.2 Decoder

Another prominent structural modification proposed by Long *et al.* (2014) was the addition of a decoder network to upsample the encoder's output heatmap in a step-wise fashion. Furthermore, Long *et al.* (2014) realized that bi-linear upsampling could be generalized by convolutions with fractional stride $1/f$, i.e., transposed convolutions (DUMOULIN; VISIN, 2018), where the weights of the neighbourhood $\mathcal{N}_i$ of a pixel i could also be learned via backpropagation (RUMELHART *et al.*, 1986). Such approach is able

[4] Usually a 1x1 kernel is employed since the goal is to obtain a channel-wise linear combination for each spatial position, reducing the number of channels to the number of segmentation classes.

to preserve more fine-grained details when compared to a simple bi-linear interpolation with high factor S.

Additionally, after every upsample step, the result is added to the corresponding encoder's feature map of same resolution. Such connection is commonly denoted as *skip-connection* and helps to recover some fine-grained details, such as borders and edges, that are lost during pooling operations. The FCNet's simple decoder is depicted in Figure 5.4, whilst its intermediate results are shown in Figure 5.5 evincing the step-wise detail reconstruction.

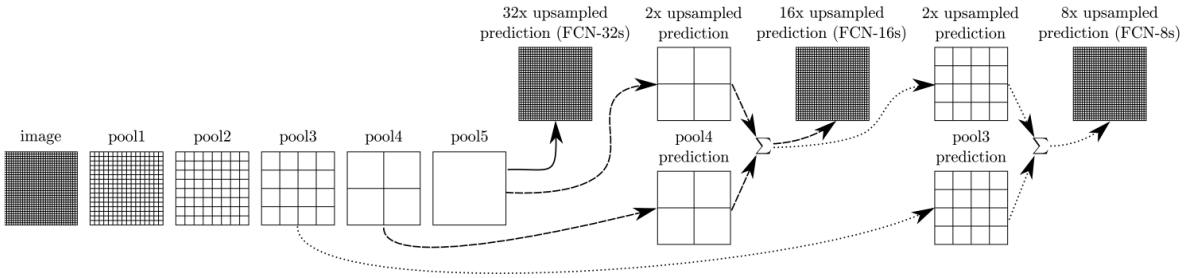

Figure 5.4 – **The slender decoder of FCNet.** The encoder's output feature map is upsampled by 2x steps. In each step, the encoder's feature map of corresponding resolution is used as skip-connection to recover high-frequency information. Figure extracted from (LONG *et al.*, 2014).

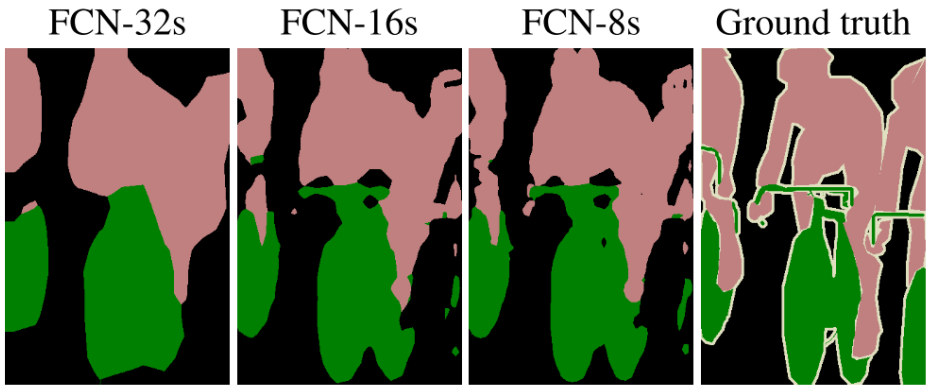

Figure 5.5 – **Intermediate segmentation results of FCNet.** Each image shows the corresponding segmentation result for each stride of Figure 5.4. Figure extracted from (LONG *et al.*, 2014).

5.3 U-Net

Following the breakthrough of *fully convolutional networks* for semantic segmentation (LONG *et al.*, 2014), many different types of decoder network were proposed. One of the most prominent is U-Net (RONNEBERGER *et al.*, 2015).

5.3.1 Bigger Decoder

From the simple decoder of Figure 5.4, Ronneberger *et al.* (2015) increased the number of up-sampling[5] steps and added two additional 3x3 convolutions followed by ReLU activation functions after each skip-connection, providing a better merge between encoder and decoder's feature maps, as can be observed in Figure 5.6. Additionally, U-Net (RONNEBERGER *et al.*, 2015) increases the number of channels in the decoder's layers, which allows context information to be better propagated to the high-resolution layers located at the end of the decoder, which, consequently, improves the network's classification capabilities. As a result, the encoder and decoder networks are more or less symmetric and the network's architecture is shaped as a letter 'U' – hence the name U-Net.

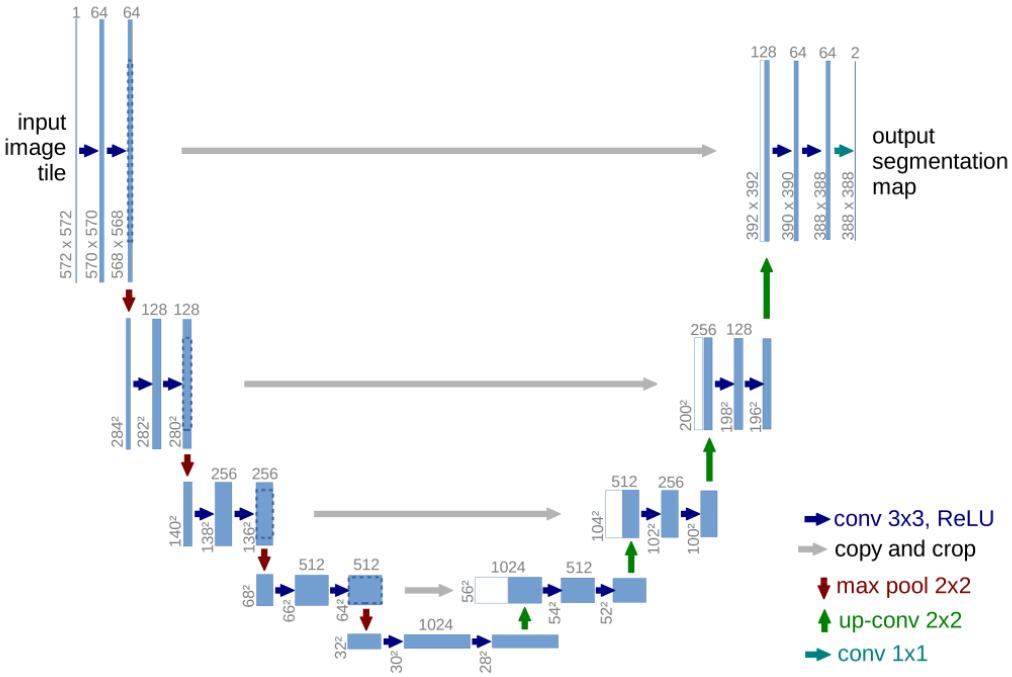

Figure 5.6 – **U-Net Architecture.** U-Net modified FCNet (LONG *et al.*, 2014) by adding more convolutions and increasing the number of channels in the decoder network, improving the network's capacity to fuse feature maps and, ultimately, enhancing the reconstruction of fine-grained details in the up-sampling path. Figure extracted from (RONNEBERGER *et al.*, 2015)

Although there are other minor architectural changes proposed by U-Net, such as the usage of VALID convolutions and feature concatenation in the skip-connections, these are of lesser importance and, therefore, will not be covered in detail in this work.

[5] U-Net originally employed transposed convolutions (DUMOULIN; VISIN, 2018) as the up-sampling operation. Nonetheless, recent works (LIN *et al.*, 2016; LI *et al.*, 2018; CHEN *et al.*, 2018) moved towards parameter-free operations, such as nearest neighbour and bi-linear interpolation, given their similar performance whilst being conceptually simpler.

5.4 Improved Decoder Designs: FPN and PAN

Following U-Net (RONNEBERGER *et al.*, 2015), several alternate decoder designs have been proposed. Similarly to U-Net (RONNEBERGER *et al.*, 2015), Feature Pyramid Network (FPN) (LIN *et al.*, 2016) proposed a top-down fusing strategy. However, instead of a symmetrical decoder design, the latter employed a light-weight decoder based on a single convolution followed by an element-wise summation as merging strategy. Consequently, by using summation instead of concatenation, FPN successfully avoid doubling the network's width at each decoder layer. Formally, for an encoder feature map $\mathrm{E} \in \mathbb{R}^{H \times W \times C}$ of an arbitrary layer $i \in [1, 7]$, the corresponding ith decoder layer's output is given by:

$$\mathrm{D}^i = \begin{cases} Conv(\mathrm{E}^i + Resize(\mathrm{D}^{i+1})), & \text{if } i < 7 \\ Conv(\mathrm{E}^i), & \text{otherwise} \end{cases}, \tag{5.1}$$

where *Resize* is an up-sample operation such as bi-linear interpolation.

PANet (LIU *et al.*, 2018) further extends FPN by adding a bottom-up path, which allows for information to also flow from high-resolution decoder layers to their low-resolution counterparts. Even though such design does not provide gains for semantic segmentation models given that only the highest-resolution layer of the decoder is used for prediction, PANet is useful for object detectors and top-down instance segmentation[6] approaches, which may use the output of all decoder layers selected based on the object size. The general idea of the FPN and PAN decoders, for a 7 layered[7] encoder backbone, is displayed in Figure 5.7 a) and b), respectively.

5.5 EfficientDet

EfficientDet (TAN *et al.*, 2019) was proposed in 2020 by the same authors as EfficientNet (TAN; LE, 2019). Accordingly, EfficientDet (TAN *et al.*, 2019) employs its precursor work as encoder network and applies the same compound scale factor to scale-up the decoder design. Additionally, the PAN decoder design is scrutinized and optimized for performance and efficiency. From Figure 5.7 b), the BiFPN decoder – sub-figure c) – is obtained by following four principles:

- Nodes connected to a single input edge are eliminated given that they do not contribute to feature fusing. Particularly, this is the case for the first D^7 and last decoder layers D^3 of Figure 5.7.

[6] Top-down instance segmentation approaches, i.e., first detect the instances and, then, perform segmentation, usually employ the same design as two-stage objector detectors.

[7] This encoder design with increased stride is based on the EfficientDet (TAN *et al.*, 2019) architecture.

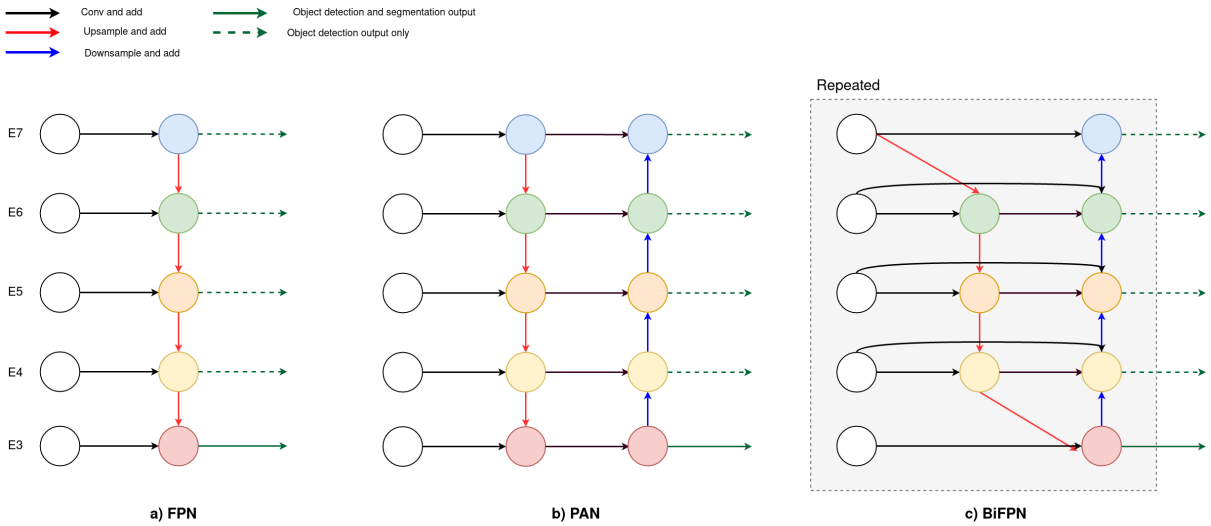

Figure 5.7 – **Decoder comparison.** Sub-figure a) shows the top-down FPN (LIN *et al.*, 2016) approach, whereas b) represents the bi-directional connections of PAN (LIU *et al.*, 2018). Finally, c) represents the BiFPN (TAN *et al.*, 2019) evincing its modifications upon the PAN design. Solid green arrows represent outputs used both in object detection and semantic segmentation tasks, whereas dashed arrows of the same color correspond to outputs used only in object detectors. Figure inspired in (TAN *et al.*, 2019).

- Convolutional layers are replaced by *depthwise separable convolutions* (VANHOUCKE, 2014; CHOLLET, 2017) to improve both FLOPs and parameter efficiencies.
- Residual connections (HE *et al.*, 2016) from input to output are added, enabling the decoder block to perform identity mapping and allowing for a better gradient flow.
- Instead of a single block, the whole decoder structure can be stacked multiple times depending on the encoder size. The compounding scale factor (TAN; LE, 2019) is employed to determine the number of blocks.

Additionally, based on the assumption that not all layers should contribute equally at each node, instead of performing an uniform element-wise summation, EfficientDet (TAN *et al.*, 2019) adds a weight parameter to each term, which is learned by the network via the optimization algorithm. Notably, to avoid numeric instabilities with unbounded learned parameters as weights, weighted-BiFPN adopts a weight normalization for each node as follows:

$$\mathrm{D}^i = \sum_i \frac{g(w_i)}{\epsilon + \sum_j g(w_j)} \cdot I_i, \tag{5.2}$$

where the function $g(\cdot)$ is the ReLU (NAIR; HINTON, 2010) activation function to guarantee that the weights w are non-negative, ϵ is a small value employed for numeric stability purposes and the inputs I_i can be either direct values of the same decoder level or product of *Resize* operations from different layers, as represented in Figure 5.7 c).

The complete EfficientDet (TAN *et al.*, 2019) architecture for object detection is displayed in Figure 5.8. Particularly, for semantic segmentation, the class and box prediction sub-networks are dropped and only the decoder layer with highest-resolution (pathway depicted in red in Figure 5.8) is used to predict the segmentation map.

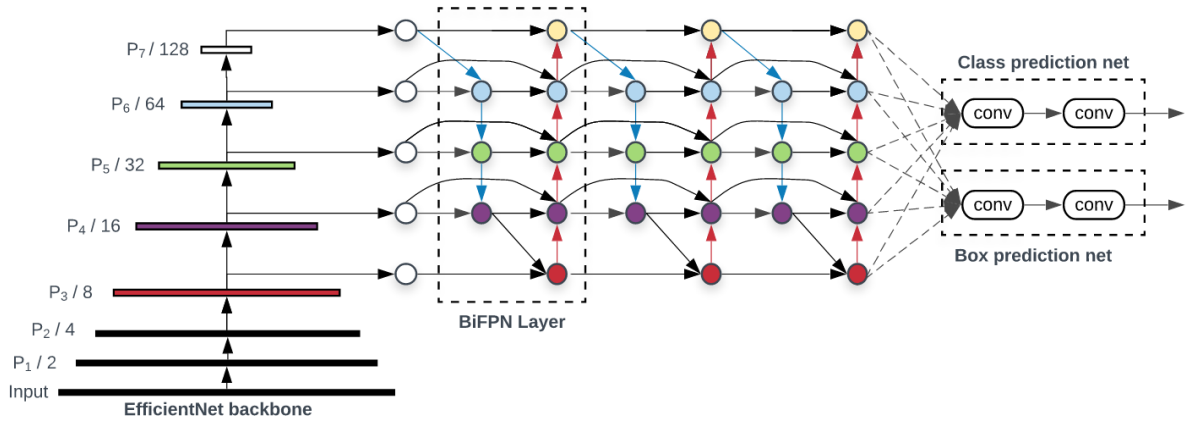

Figure 5.8 – **Complete EfficientDet architecture for object detection.** Figure extracted from (TAN *et al.*, 2019).

5.6 Feature-aligned Pyramid Network (FaPN)

In 2021, a novel decoder design was proposed to improve even further the recovery of high-frequency information lost throughout the encoder's pooling layers.

Feature-aligned Pyramid Network (FaPN) (HUANG *et al.*, 2021) extends FPN by introducing two additional blocks: Feature Alignment Module (FAM) and Feature Selection Module (FSM). The first is responsible for aligning the low-resolution feature maps, whereas the second is responsible for calibrating via attention mechanisms the skip connections before the feature fusion. An overview of FaPN and its main differences from FPN are depicted in Figure 5.9.

5.6.1 Feature Alignment Module (FAM)

In most deep learning architectures, the input image is donwsampled several times throughout the network. By the end of the encoder, feature maps are usually 32 to 64 times smaller than the original input. Huang *et al.* (2021) argued that, during this process, spatial misalignment may occur and the naive fusion between skip connection and the upsampled feature map, as performed in FPN (LIN *et al.*, 2016), may harm the prediction for fine borders.

In order to fix that, Huang *et al.* (2021) proposed to apply deformable convolutions (DAI *et al.*, 2017; ZHU *et al.*, 2019) after the upsampling of low-resolution

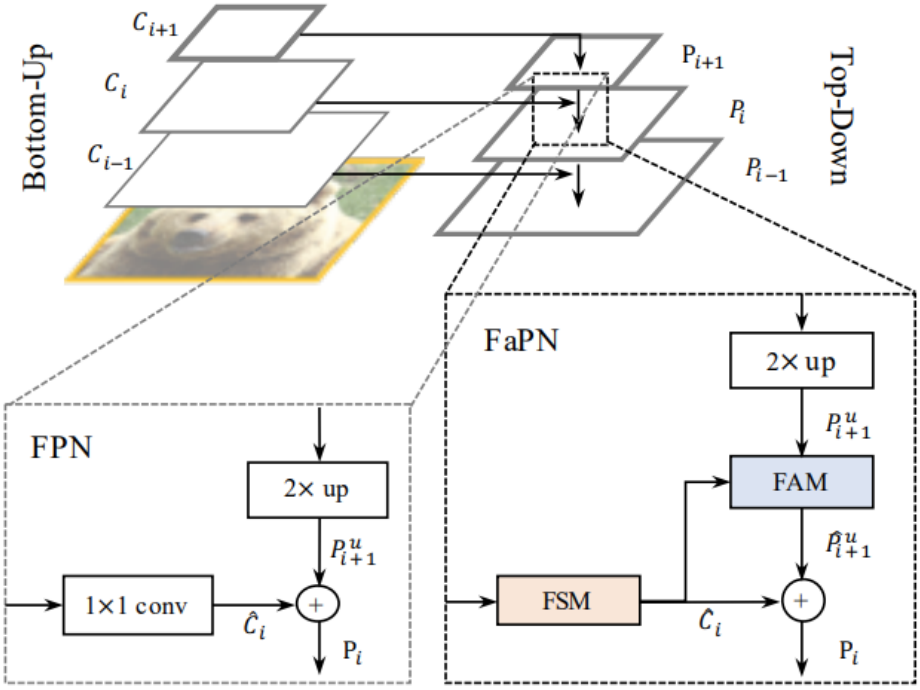

Figure 5.9 – **FaPN overview and comparison with FPN.** Figure extracted from (HUANG *et al.*, 2021).

feature maps. This special family of convolution enables the model to learn offset fields for each pixel in the convolution kernel, allowing the layer to perform its computation on a non-rigid grid, which is able to fix possible misalignments caused by the downsample operations. The FAM block can be visualized in Figure 5.10.

5.6.2 Feature Selection Module (FSM)

Inspired by attention mechanisms, Huang *et al.* (2021) proposed to apply a variation of the SE block (HU *et al.*, 2018) to the skip connection before the fusion in the feature pyramid. By doing so, the network is able to suppress channels of low semantic content, making sure that the reduction in channels for the fusion does not discard useful information.

The block is comprised by a global average pooling followed by a 1x1 convolution to extract the global statistics for each channel. In the sequence, a sigmoid function is applied to scale the results, which are, then, multiplied to the original feature map. Finally, similarly to a residual connection, the original feature map is added to the result. The complete block is shown in Figure 5.11.

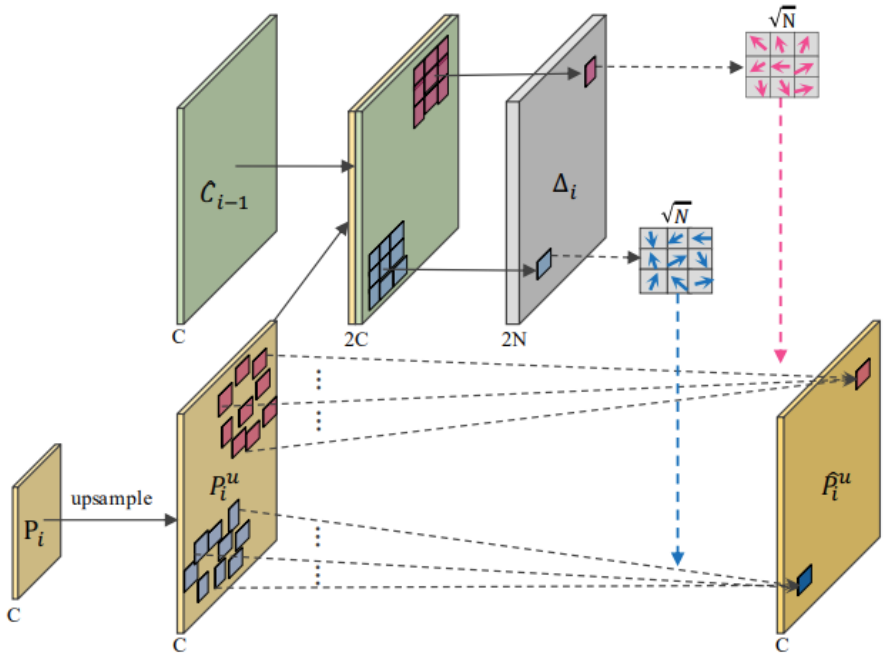

Figure 5.10 – **Overview of the Feature Alignment Module.** Figure extracted from (HUANG *et al.*, 2021).

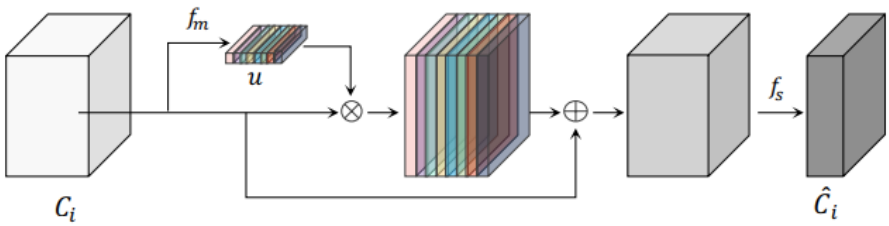

Figure 5.11 – **Overview of the Feature Selection Module.** Figure extracted from (HUANG *et al.*, 2021).

5.7 Loss Functions and Metrics

In this section, we briefly present the common loss functions used during the training phase of segmentation models, as well as the metrics used to evaluate their performance.

5.7.1 Metrics

An ideal metric for semantic segmentation should be able to correctly capture how well a prediction overlaps with the ground-truth object. However, at the same time, such metric should also be able to penalize excessive predictions lying outside the ground-truth area. Finally, invariance to scale is also paramount so that it is able to evaluate equally well objects of different sizes. The two most common metrics that satisfy these properties are explained in the sequence.

5.7.1.1 Jaccard Index

The Jaccard Index, also known as intersection over union (IoU), can be described for two discrete sets of points A and B as:

$$\mathcal{J}(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}, \tag{5.3}$$

where the operator $|\cdot|$ denotes the size of the set.

Alternatively, the Jaccard Index can be described in terms of *true-positives* (TP), *false-positives* (FP) and *false-negatives* (FN) for binary-classification setups as:

$$\mathcal{J} = \frac{TP}{TP + FN + FP}, \tag{5.4}$$

Equation (5.4) can be easily extended for multi-class setups by applying the *over-versus-rest* strategy, where, for each class $c \in \mathcal{C}$, a binary problem is set against all other classes and the final is result is taken as the average of all binary metrics.

5.7.1.2 Dice Score

The Dice score, also known as Sørensen–Dice coefficient, can be described as:

$$\mathcal{D}(A, B) = 2 \cdot \frac{|A \cap B|}{|A| + |B|} \tag{5.5}$$

Note that, despite very similar to the Jaccard Index, the Dice score uses the sum of the sizes in the denominator, which also includes $|A \cap B|$. Hence, a factor of 2 is used in the numerator to normalize the metric between 0 and 1. The Jaccard index and Dice score can be related by the expression $\mathcal{J} = \mathcal{D}/(2 - \mathcal{D})$.

5.7.2 Loss Functions

Differently from evaluation metrics, the loss functions used during the training phase of deep learning models must be differentiable. For semantic segmentation, such functions can be grouped into two categories: pixel-wise and IoU-based.

5.7.2.1 Cross-Entropy Loss

The cross-entropy loss is one of the most common options used in machine learning. Its application can be easily adapted to semantic segmentation by treating each pixel as an individual sample. As defined by Equation (5.6), it can be computed for an image $\mathcal{N} \in \mathbb{R}^{H \times W}$ and a set of classes $\mathcal{C}$:

$$\mathcal{L}_{ce}(\hat{y}, \overset{*}{y}) = -\frac{1}{|\mathcal{C}| \cdot |\mathcal{N}|} \sum_{c \in \mathcal{C}} \sum_{i \in \mathcal{N}} \overset{*}{y}_{i,c} \cdot log(\hat{y}_{i,c}), \tag{5.6}$$

where, for the pixel $i \in \mathcal{N}$ and class $c \in \mathcal{C}$, the network's prediction and the ground-truth label are represented by $\hat{y}_{i,c} \in [0, 1]$ and $\overset{*}{y}_{i,c} \in \{0, 1\}$, respectively. Additionally, the number of elements of a set is denoted by the operator $|\cdot|$.

5.7.2.2 Focal Loss

The Focal Loss (LIN *et al.*, 2017) is an extension of the cross-entropy loss, which was originally proposed to tackle the severe class imbalance experienced by single-stage object detection by reducing the weights of well classified samples. Similarly to the cross-entropy, it can also be applied at pixel-level for semantic segmentation tasks. Its binary form for a single sample can be described as follows:

$$FL(p) = \begin{cases} -\alpha \cdot (1-p)^{\gamma} \cdot \log(p) & y = 1 \\ -(1-\alpha) p^{\gamma} \cdot \log(1-p) & \text{otherwise} \end{cases}, \tag{5.7}$$

where γ is a hyperparameter controlling the reduction effect on well classified samples, α is another hyperparameter that directly controls the weights for the positive and negative classes and $p \in [0, 1]$ and $y \in \{0, 1\}$ are the model prediction and ground-truth label, respectively.

If no class weights are set, i.e., $\alpha = 0.5$, Equation (5.7) can be simplified to:

$$FL(p_t) = -(1-p_t)^{\gamma} \cdot \log(p_t), \tag{5.8}$$

where p_t is given by:

$$p_t = \begin{cases} p & y = 1 \\ 1-p & \text{otherwise} \end{cases} \tag{5.9}$$

Figure 5.12 shows the loss effect on well classified samples for different values of γ.

5.7.2.3 Soft-dice Loss

Even though the Cross-entropy and Focal Loss usually provide a good alternative to train semantic segmentation models, these cost functions optimize the model in a pixel-wise fashion, which diverge from how semantic segmentation metrics are usually computed (based on IoU). Thus, in an attempt to bring the optimization and evaluation of segmentation models closer, the soft-dice score (SUDRE *et al.*, 2017) was proposed as follows:

$$\mathcal{L}(\hat{y}, \overset{*}{y}) = 1 - \frac{2}{|\mathcal{C}| - 1} \sum_{c \in \mathcal{C}} \frac{||\overset{*}{y}_c \odot \hat{y}_c||}{||\overset{*}{y}_c||^2 + ||\hat{y}_c||^2}, \tag{5.10}$$

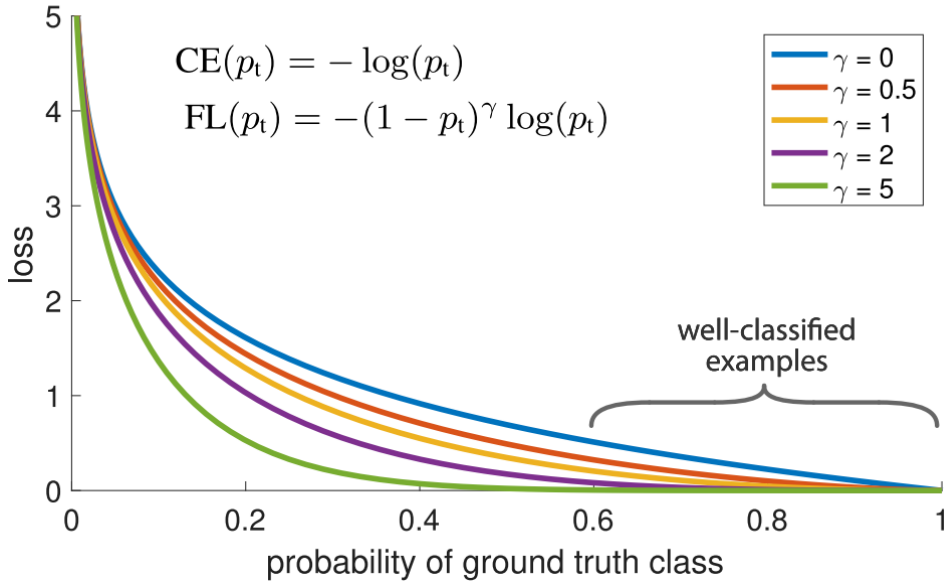

Figure 5.12 – **Focal loss effect on different γ values.** Figure extracted from (LIN *et al.*, 2017).

where the image vectors $\overset{*}{y}_c \in \mathbb{R}^{H \times W}$ and $\hat{y}_c \in \mathbb{R}^{H \times W}$ represent the ground-truth label and the model's prediction for a class $c \in \mathcal{C}$. Additionally, the operators $||\cdot||$ and $\odot$, respectively, refer to the L_2-norm and the Hadamard product (element-wise multiplication). Without any loss of performance, Equation (5.10) can be simplified by replacing the L_2 norm by a simple summation and dropping the square factors:

$$\mathcal{L}(\hat{y}, \overset{*}{y}) = 1 - \frac{2}{|\mathcal{C}| - 1} \sum_{c \in \mathcal{C}} \frac{\sum \overset{*}{y}_c \odot \hat{y}_c}{\sum \overset{*}{y}_c + \sum \hat{y}_c} \tag{5.11}$$

5.8 Final Remarks

In this chapter, we reviewed the first works on semantic segmentation in the context of deep learning and how image classification architectures can be modified in order to perform pixel-level predictions. Additionally, the most notorious follow-up designs of the precursor work of Long *et al.* (2014) were shown and discussed, highlighting the major improvements. Finally, we presented some novel architectures that are better able to reconstruct high-frequency details from the small resolution feature maps and we concluded the chapter by discussing common loss functions and metrics for semantic segmentation.

It should be noticed that despite the improvements achieved by the architectures presented so far, none of them were especially designed to cope with empty images nor very unbalanced datasets. Thus, the direct application of such models to the SSEI scenario may yield sub-optimal results, which, consequently, motivates us to address the

issue by proposing context-specific modifications to current encoder-decoder architectures, better adapting them for SSEI applications.

6 Proposed Method for Segmentation Under Extreme Imbalance Towards Full-background Images

In this chapter, we introduce a novel network architecture for semantic segmentation tasks, whose main focus is to deal with scenarios of severe imbalance towards full-background images. Such architecture is heavily based on: (1) Hypercolumns (HARIHARAN *et al.*, 2014a) to better incorporate context, essential to avoid predicting small false positive artifacts; and (2) attention mechanisms (HU *et al.*, 2018), which are helpful to mitigate the discrepancies in semantic content when merging feature maps from different layers.

Originally, the Hypercolumn block (HARIHARAN *et al.*, 2014a) was intended for networks without any decoder in their architecture. Thus, the representations of the shallow and intermediate layers injected by Hypercolumns into the classifier are helpful given their high-frequency information, such as precise localization, sharp edges, pose, illumination, etc. Here, we invert the original rationale and propose to utilize it along with a decoder network. Precisely, in our case, the Hypercolumns block operates on the outputs from the decoder block. In this context, it provides semantically rich feature maps from the low-resolution decoder layers to their high-resolution counterparts, incorporating semi-global context into the final network's representation.

Additionally, we extend the Hypercolumns block by adding a multi-level fuse block, whose goal is to improve the merging between feature maps originated from different network layers. As evinced by Li *et al.* (2018), such layers usually encode features of very different characteristics – both in terms of semantic content and high-frequency information – which can lead to sub-optimal results when naively combined together. In Chapter 8, we empirically demonstrate with ablation studies that our dedicated fuse module allows encoder-decoder networks to extract more gains from the context-enriched representation provided by Hypercolumns.

Aiming to mitigate imbalance issues, we also propose a novel auxiliary scale invariant segmentation loss based on the soft-dice score (SUDRE *et al.*, 2017) for non-empty images, on top of which we use a "smart" batch sampling scheme to reduce the discrepancy between classes and stabilize the training process.

6.1 Architecture

The architecture of our model is comprised of an encoder, which usually is the convolutional section of common image classification networks such as VGGs (SIMONYAN; ZISSERMAN, 2015), ResNets (HE *et al.*, 2016), SENets (HU *et al.*, 2018) and EfficientNets (TAN; LE, 2019); a U-Net-like decoder responsible for upsampling the encoder's output and reconstructing high-frequency information lost during pooling operations; the inverted hypercolumn block, whose objective is to enrich the high-resolution decoder's feature maps with additional context from layers that have larger receptive field; and a multi-level fuse block that mitigates the issue of combining feature maps of different semantic contents, improving the image's representation prior to the final segmentation classifier layer.

6.1.1 Decoder

For the decoder network design, U-Net (RONNEBERGER *et al.*, 2015) is adopted as main reference, upon which we make some modifications. Firstly, the transposed convolutions (DUMOULIN; VISIN, 2018) used originally for upsampling are replaced by common bi-linear upsampling kernels, as suggested by (ODENA *et al.*, 2016), to avoid generating checkerboard artifacts and reducing the number of parameters. Secondly, instead of concatenating the features in the skip-connections, we employ element-wise summation, which avoids doubling the number of channels at every connection, reducing the number of parameters and floating point operations without any loss of performance, as verified by recent works (LI *et al.*, 2018; LIN *et al.*, 2016; TAN *et al.*, 2019).

6.1.2 Inverted Hypercolumns

As already stated, we propose to utilize Hypercolumns at the end of the decoder network, which inverts its original rationale. At the low-resolution layers of the decoder, the descriptors are context- and semantically rich albeit lacking high-frequency information. As the decoder progressively increases its feature maps' resolution, precise localization is recovered, but the receptive field of the subsequent convolutions start getting smaller, which may undermine the network's capacity to propagate global and semi-global representation to the final classifier. Thus, by employing Hypercolumns, we assure that the final descriptor prior to classification contains representations not only from the last layer but also from all the intermediate ones, providing all levels of semantic content and high-frequency information. An overview of the proposed inverted Hypercolumns is depicted in Figure 6.1.

Formally, considering a feature map $\mathbf{F}_i \in \mathbb{R}^{C_i \times H_i \times W_i}$ of a decoder layer $i \in [1, L]$, the combined Hypercolumn representation $\mathbf{X} \in \mathbb{R}^{C \times H \times W}$ can be described by Equation (6.1):

$$\mathbf{X} = \sum_{i=1}^{L} \phi_i(\mathbf{F}_i), \tag{6.1}$$

where the transformation $\phi(\cdot)$ comprises an upsampling kernel[1] and a 1x1 convolution, in order to first upsample the intermediate input and, then, adapt its number of channels to a common dimension C, enabling the element-wise summation of feature maps from all decoder layers.

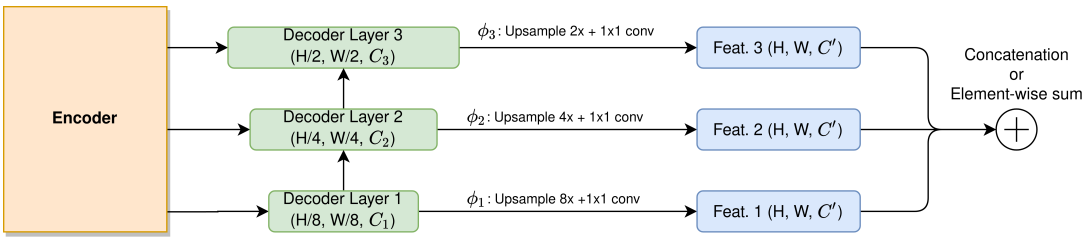

Figure 6.1 – **Hypercolumns applied to the decoder network.** The transformation $\phi(\cdot)$ converts each feature map into a common dimension, enabling the fusion by either channel concatenation or element-wise summation.

6.1.3 Multi-level Fuse Block

As evinced by Li *et al.* (2018), skip-connections have to merge feature maps with different characteristics: the upsampled descriptor from the decoder presents high semantic content, whereas the incoming skip-connection from the encoder carries fine-grained details but lower semantic context. Consequently, naively fusing these descriptors may lead to sub-optimal results. This phenomenon is further aggravated in the Hypercolums block where descriptors from several different layers are combined together. The color code in Figure 6.2 can aid in understanding such concept as it shows how feature maps of different natures are combined throughout the network. Thus, in order to maximize the joint representation of Hypercolumns we propose a multi-level fuse layer that relies on convolutions augmented with attention mechanisms to better combine descriptors of different characteristics both in channel and spatial dimensions.

Accordingly, the first step is to channel-wise calibrate the Hypercolumns descriptors $\mathbf{X}$ via the self-attention Squeeze-Excitation Block (HU *et al.*, 2018) ψ_1. Then, a block $\mathcal{F}$ comprised of three 3x3 convolutions accompanied by Batch Normalization

[1] Both nearest neighbour and bi-linear interpolations could be used. We observe that the latter yields better upsampling results at the expense of higher computational cost.

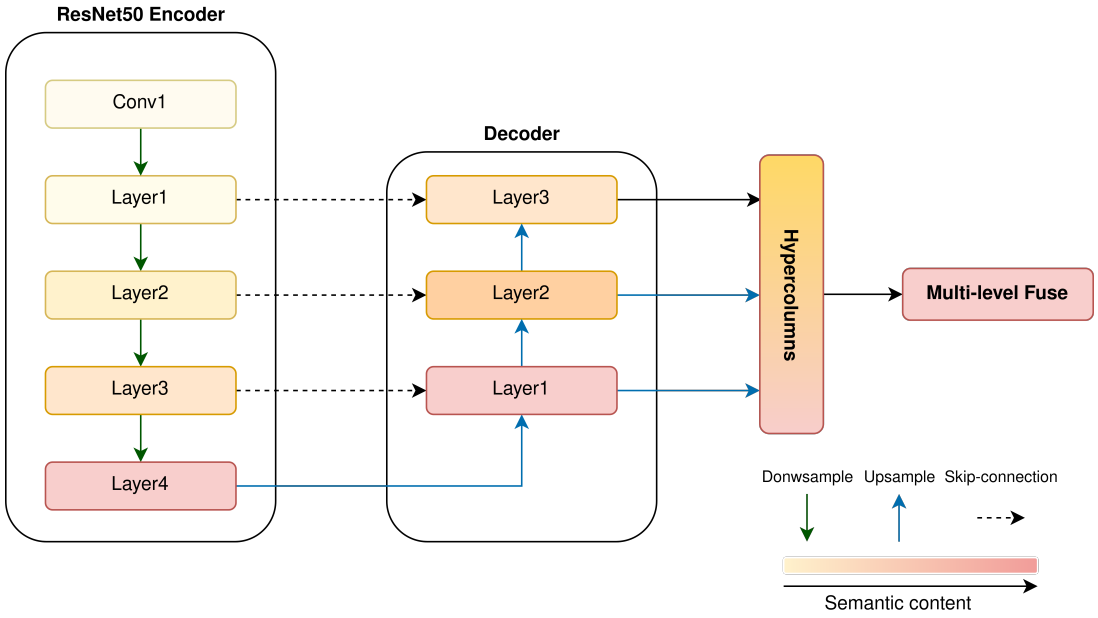

Figure 6.2 – **Complete Architecture Diagram.** Dashed, green and blue arrows denote skip-connections, down- and up-sampling operations, whereas the feature map's color represents its semantic content. Final segmentation classifier is added on top of the Multi-level Fuse block's output. Best viewed in color.

(IOFFE; SZEGEDY, 2015) and ReLU, whose receptive field is equivalent to a 7x7 convolution, is used to spatially fuse the common representation. Finally, the intermediate feature map $\mathbf{Z}$ undergoes a second SE Block ψ_2 to create the final calibrated representation.

Formally, the Multi-level Fuse Block $\mathcal{G}$ can be defined by Equation (6.2). Alternatively, such transformation can be interpreted as a non-linear mapping between an uncalibrated descriptor $\mathbf{X} \in \mathbb{R}^{C \times H \times W}$ and its calibrated counterpart. Figure 6.3 provides an overview on how Equation (6.2) can be translated into neural network's layers.

$$\mathbf{Z}^{'} = \mathcal{G}(\mathbf{X}) = \psi_2(\mathcal{F}(\psi_1(\mathbf{X}))) \tag{6.2}$$

Finally, Figure 6.2 depicts the complete architecture, where down- and up-sampling layers are denoted by green and blue arrows, respectively, and the color code represents the expected semantic content of each intermediate feature map. Notably, Figure 6.2 evinces how the Hypercolumns block has to merge descriptors of different characteristics, which is the intuition behind the Muli-level Fuse block.

6.2 Losses

In this section, we briefly introduce the main segmentation loss used to classify all images at pixel level. Additionally, the auxiliary segmentation loss employed for

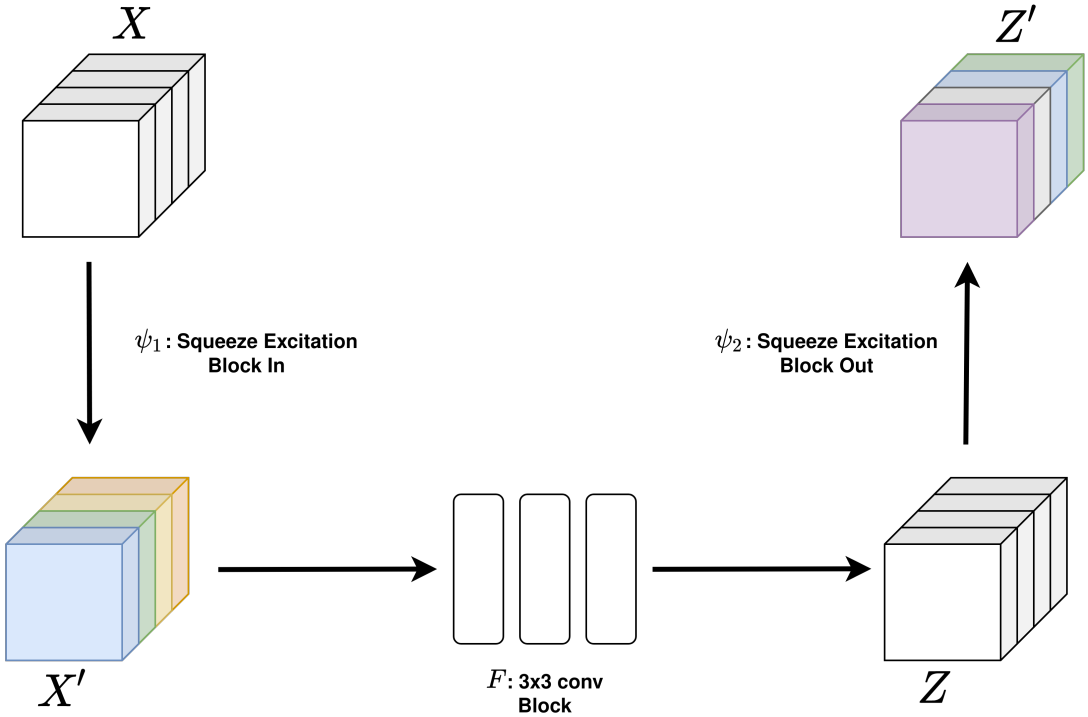

Figure 6.3 – **Multi-level Fuse Block.** The Hypercolumns' output feature map **X** first undergoes a channel-wise transform ψ_1 to reduce the impact of semantically bad and possibly co-occurring features. Then, a set of three 3x3 convolutions, followed by Batch Normalization (IOFFE; SZEGEDY, 2015) and ReLU are applied to also merge the features spatially, resulting in the feature map **Z**. Finally, another channel-wise transform ψ_2 is applied to re-calibrate the channels. Our segmentation classifier is applied on top of $\mathbf{Z}'$.

foreground images is also presented. Finally, we introduce the strategy used to combine both losses.

6.2.1 Main Segmentation Loss

As commonly adopted for semantic segmentation tasks, we employ a 1x1 convolution as final spatial classifier in order to convert the number of channels of the output descriptor to the total number of classes $\mathcal{C}$. Then, we employ the pixel-wise cross-entropy loss as defined in Equation (5.6).

6.2.2 Auxiliary Foreground Segmentation Loss

In addition to the main cross-entropy loss, an auxiliary segmentation loss based on the soft-dice score (SUDRE *et al.*, 2017), defined in Equation (5.11), is employed.

The rationale behind the adoption of such auxiliary loss lies on its properties of invariance to scale and class imbalance, which can be decisive to reduce the ambiguity between small objects and full-background images – very important cases given the scope

of this work. Nonetheless, in case of a correct prediction for empty images, the condition $\sum \overset{*}{y}_c + \sum \hat{y}_c \to 0$ can be easily met and, consequently, Equation (5.11) is not defined or can lead to inconsistencies. Although the convention of setting $0/0 := 1$ is a commonly used workaround, we argue that it should be avoided as it biases the learning towards false negatives since the loss incurred by correctly estimating an empty image is always 0, whereas by accurately predicting object pixels the loss is proportional to the IoU, which is usually significantly greater then 0 even for a reasonably good prediction.

In order to avoid such conundrum, we employ the auxiliary segmentation loss $\mathcal{L}_{fg}$ only for foreground images, i.e., samples of the training set that have at least one pixel belonging to one of the foreground classes $c \in \mathcal{C}, c \neq \mathcal{B}$. When used in conjunction with batch sampling schemes, e.g., over- and under-sampling, such approach assures that the denominator of Equation (5.11) is always greater than 0. Thus, conveniently, this loss is able to strictly focus on the semantic segmentation task without suffering of pixel-level class imbalance. The modified soft-dice loss for *foreground* images can be described as follows:

$$\mathcal{L}_{fg}(\hat{y}, \overset{*}{y}) = 1 - \frac{2}{|\mathcal{C}| - 1} \sum_{c \in \mathcal{C}, c \neq \mathcal{B}} \frac{\sum \overset{*}{y}_c \odot \hat{y}_c}{\sum \overset{*}{y}_c + \sum \hat{y}_c} \tag{6.3}$$

In practice, the two losses – cross-entropy $\mathcal{L}_{ce}$, which is computed on all images, and the auxiliary segmentation loss $\mathcal{L}_{fg}$, applied only on foreground images – are linearly combined, as defined in Equation (6.4), and the resultant tensor is back-propagated (RUMELHART *et al.*, 1986) to determine the network's gradients.

$$\mathcal{L} = \mathcal{L}_{ce} + \alpha \cdot \mathcal{L}_{fg} \tag{6.4}$$

The constant α is an hyperparameter responsible for controlling the weight of the foreground segmentation loss.

7 Experiments

Four types of experiments are performed in this work. Firstly, using a baseline single-stage CNN for semantic segmentation, we evaluate how "smart" image-level sampling schemes can be used to tackle the class imbalance problem. Secondly, some CNN architectures are trained for the problem of semantic segmentation with extreme imbalance towards empty images and their performances are compared. Additionally, two-stage pipelines, where an image classification network first filters the empty images and, then, another CNN segments the foreground images, are also benchmarked against their single-stage counterpart. Thirdly, we evaluate the compatibility of the proposed architecture and loss function with other pixel-wise class imbalance techniques, such as Focal Loss (LIN *et al.*, 2017), and modern encoder-decoder architectures, e.g., EfficientDet (TAN *et al.*, 2019) and Feature-aligned Pyramid Network (FaPN) (HUANG *et al.*, 2021). Lastly, ablation studies are performed in order to determine how each part of the proposed model, described in Section 6.1, contributes to the final result and, ultimately, to aid in comprehending the underlying mechanisms of the proposed method. In the remainder of this chapter, the chosen dataset and the methodology are presented, whilst the results are exhibited and discussed in detail in Chapter 8.

7.1 Datasets

In order to evaluate the performance of the proposed architecture, two datasets of distinct nature were chosen. The first, SIIM-ACR Pneumothorax[1], is related to a real-world application and represents a binary segmentation problem, whereas, the second, a MS-COCO (LIN *et al.*, 2015) subset, represents a multi-class setup of a famous dataset in the object detection and segmentation literature.

7.1.1 SIIM-ACR Pneumothorax

As one of our datasets to evaluate the models throughout this work, we chose the SIIM-ACR Pneumothorax Segmentation dataset, which comprises 10,675 lung X-ray images that are annotated at pixel-level for a binary classification setup. The positive class represents Pneumothorax, a lung pathology characterized by anomalous accretion of gases in the pulmonary pleural space, whose origin is usually attributed to trauma, spontaneous formation or byproduct of another pulmonary disease. Fast and accurate

[1] https://www.kaggle.com/c/siim-acr-pneumothorax-segmentation

automatic diagnosis is of paramount importance as it should be treated immediately either by decompression or drainage since it can be a life-threatening condition (Ni Fhlatharta; EATON, 2020; WEISSBERG; REFAELY, 2000; YARMUS; FELLER-KOPMAN, 2012).

This dataset is particularly pertinent to the scope of this work given that only 24.5% of the X-ray images have the pathology, i.e., 75.5% of images are empty, as can be observed in Table 7.1. Furthermore, when pixel-level imbalance is regarded, 99.68% of all dataset's pixels belong to the background (healthy) class. Finally, the original dataset is randomly divided – whilst respecting class stratification – into training, validation and test sets containing 8647 (81%), 961 (9%) and 1067 (10%) samples, respectively.

Table 7.1 – Class Statistics for the SIIM-ACR Pneumothorax Dataset

	Image-level %	**Pixel-level %**
Pneumothorax	24.25	0.32
Background	100	99.68

The columns show the representativity of each class in the dataset both on image- and pixel-level. The image-level statistics may add up to more than 100% since images usually contain pixels of more than one class.

7.1.2 MS-COCO: BikeCar

As a second dataset to evaluate our models and with reproducibility in mind, we opted for a dataset commonly used in the literature. It is important to remark, however, that almost all segmentation datasets, such as MS-COCO (LIN *et al.*, 2015), CityScapes (CORDTS *et al.*, 2016) and Pascal-VOC (EVERINGHAM *et al.*, 2010), do not have empty images in their composition, i.e., all the images in those datasets have at least one pixel belonging to one of the foreground classes. Thus, they are not directly compatible to the spirit of this work. With that in mind, we decided to adapt the MS-COCO(LIN *et al.*, 2015) dataset by using only a subset of classes whilst keeping all the images. By doing this, it is possible to control the percentage of empty images in the dataset based on the number of classes selected as foreground. Particularly, only the *Car* and *Bicycle* classes were selected yielding a multi-class dataset almost twice as unbalanced as the SIIM-ACR dataset. The statistics for MS-COCO: BikeCar are displayed in Table 7.2.

7.2 Evaluation Metrics

Following the literature, as semantic segmentation metric, the IoU (*intersection over union*) is employed. However, considering that the background class is usually much

Table 7.2 – Class Statistics for the COCO-BikeCar Dataset

	Image-level %	**Pixel-level %**
Bicycle	2.98	0.183
Car	10.70	0.617
Background	100	99.20

The columns show the representativity of each class in the dataset both on image- and pixel-level. Again, the image-level statistics may add up to more than 100% since images usually contain pixels of more than one class.

easier to estimate, we opted to compute and average only the IoU for the foreground classes in order to avoid biasing the evaluation process.

Additionally, an image-wise classification evaluation is performed according to the following criteria: if a single foreground pixel is predicted by the network, then, the sample is deemed as *positive*. Conversely, if all pixels of the segmentation mask are estimated as background, then, a *negative* prediction is regarded. For the image-wise classification task, the balanced accuracy metric (BRODERSEN *et al.*, 2010), defined in Equation (7.1), is used as it is most suited for unbalanced scenarios.

$$ACC_{bal} = \frac{1}{2} \cdot \left(\frac{TP}{P} + \frac{TN}{N} \right), \tag{7.1}$$

where the *true-positives*, *positives*, *true-negatives* and *negatives* are denoted by TP, P, TN and N, respectively.

7.3 Implementation Details

In this section, the settings and parameters used to train image classification and semantic segmentation models throughout this work are presented. The following can be regarded as the *default* setup for all models. When a model uses a setting or parameter different from those herein specified, it shall be elucidated in that model's own section.

7.3.1 Network

Given its good balance between size and performance, the ResNet-50 (HE *et al.*, 2016) is used as backbone in most experiments. Particularly, the backbone is initialized using the ImageNet (RUSSAKOVSKY *et al.*, 2015) pre-trained weights. As for semantic segmentation tasks, the decoder networks are based on U-Net (RONNEBERGER *et al.*, 2015) and follows the architecture presented in Section 6.1.1, even though the proposed

modifications of this work can be applied to any other modern decoder-based architecture, such as those from (LI *et al.*, 2018; CHEN *et al.*, 2018; PENG *et al.*, 2017). Additionally, as can be verified in Section 8.4, we also evaluate the elements of our proposal applied to modern segmentation architectures – EfficientDet-D2 (TAN *et al.*, 2019) and FaPN (HUANG *et al.*, 2021).

7.3.2 Optimizer

In all experiments, the RAdam (LIU *et al.*, 2019) optimizer, with default values of $\beta_1 = 0.9$ and $\beta_2 = 0.999$, is used along with a Cosine annealing (LOSHCHILOV; HUTTER, 2016) learning rate scheduler, whose start and end values are set to 10^{-3} and 10^{-6}, respectively. Additionally, in order to avoid overfitting, 10^{-4} is used as weight decay (GOODFELLOW *et al.*, 2016).

7.3.3 Loss

The cross-entropy (GOODFELLOW *et al.*, 2016) loss function is used for all image classification models. Also, its pixel-wise version is employed in segmentation tasks, unless otherwise specified[2]. Notably, the auxiliary foreground segmentation loss of Equation (6.3) is an exception and is only used in the proposed model, whose parameter α is set to 0.1, which was found empirically through grid-search.

7.3.4 Training Settings

All experiments were executed on a machine with the following specifications:

- OS: Arch Linux
- CPU: Intel i7-6800k, 3.4ghz,15mb, Hexa-core
- RAM: 32Gb
- GPU: Gigabyte Windforce NVIDIA RTX 2080ti

The PyTorch (PASZKE *et al.*, 2017) library with mixed-precision mode (NARANG *et al.*, 2018) was used to train the image classification and segmentation networks, for 30 and 37 epochs, for the SIIM-ACR and COCO: BikeCar datasets, respectively, which was found through grid-searching to be enough for full convergence.

[2] In some experiments we evaluate how Focal Loss (LIN *et al.*, 2017) can be used to mitigate the class imbalance. In such experiments, we explicitly mention the loss function used.

For the classification tasks, a batch size of 32 256×256 images was adopted. On the other hand, as bigger input sizes are helpful for segmentation models, the batch size is comprised of 16 512×512 images in such tasks. Notably, the batch size is halved due to memory constraints. Furthermore, the under-sampling (BRANCO *et al.*, 2015; He; Garcia, 2009; HE; MA, 2013) batch sampling strategy was adopted in order to reduce the image-level class imbalance between *foreground* and *background* classes. Finally, a simple horizontal flip was used as data augmentation in all experiments. Even though more elaborate augmentation schemes could result in better evaluation metrics, these image transformations are, in most cases, task- and dataset-specific and, thus, were avoided in this work.

8 Results

In this chapter, the computational experiments outlined in Chapter 7 are further explained and the obtained results are presented and analyzed.

8.1 Data Sampling and Batch Formulation

Before evaluating the results of the proposed architecture, we investigated if the batch sampling schemes described in Chapter 2, which are commonly used in image classifications tasks, can be useful for semantic segmentation with empty images. Such techniques are usually not employed for semantic segmentation since, in traditional segmentation datasets, at least one of the foreground classes is always present and the object size may vary widely from class to class, thus, making it very difficult to balance the dataset using image-level sampling strategies. However, in the scope of this work, those techniques can be employed to ensure that each batch is comprised of 50% of foreground images, reducing the overall class imbalance also at pixel-level.

For this experiment, a ResNet50-UNet segmentation network was trained in the SIIM-ACR dataset using the different batch sampling schemes. We compared the traditional random sampling against under- and over-sampling strategies, both targeting 50% foreground images in the training mini-batch. The results are displayed in Table 8.1.

It is possible to notice that the naive random sampling leads to significantly inferior results (-16% IoU), while under- and over-sampling yield the same IoU, but with a slight improvement in terms of image-level accuracy for the under-sampling strategy. Moreover, we observed during training that the under-sampling technique provides much more stable results over the training epochs. Therefore, for all the subsequent experiments, we adopted the under-sampling technique as the standard approach when building the training batches.

8.2 SIIM-ACR Pneumothorax Dataset

In this section, the semantic segmentation task under extreme imbalance towards full-background images is explored in the context of the SIIM-ACR Pneumothorax dataset. Accordingly, two types of pipelines, which are commonly found in practical applications, are investigated: single- and two-stages.

In the former, a single standalone network trained on all images is used to

Table 8.1 – Batch Sampling Study

	IoU	**Image-level Balanced Accuracy**
	Non-Empty only	
Random	0.352	0.794
Under-sampling	**0.418**	**0.835**
Over-sampling	**0.418**	0.818

The results for a simple U-Net on the SIIM-ACR dataset is shown for different image-level batch sampling schemes. Under-sampling not only performs best but we also observed significant improvements in stability during training.

predict segmentation masks on the entire testset. In this case, the model is expected to cope with empty images by only predicting the background class for all pixels in their segmentation mask. On the other hand, the latter is comprised of an image classification model, trained on all images and responsible for filtering out empty (healthy) samples, and a semantic segmentation network, which can either be trained only on foreground images or on the entire training set, should produce a segmentation mask for the sick patients as determined by the classifier.

The experimental results concerning the evaluation metrics of Section 7.2 – IoU and image-level balanced accuracy – for both single- and two-stage models are displayed in Table 8.2. As we can observe, the single-stage methods, in general, present considerably better results than two-stage pipelines. Additionally, for two-stage pipelines, we observe better results when the second stage, i.e., the segmentation model, is trained only on *foreground* images. On the other hand, if it is trained on all images similarly to a single-stage method, it tends to overcompensate towards the *background* class as both stages can amount to the *false negatives* count.

Table 8.2 also evinces that Hypercolumns (HARIHARAN *et al.*, 2014a) by itself does not improve the results. More specifically, when compared to U-Net (RONNEBERGER *et al.*, 2015), it achieves the same IoU but falls short on image-level accuracy by approximately 2%. We argue that, as similarly observed in (LI *et al.*, 2018), the simple merge strategy of the Hypercolumns block does not allow the network to fully exploit the descriptors from different layers. In fact, naively merging feature maps with low semantic content directly into the final representation, which is expected to be class-sensitive, can even be harmful as it may add too much class-agnostic features and, consequently, disrupt the pixel classification.

Table 8.2 – Results on the SIIM-ACR Pneumothorax Dataset

Classification	**Segmentation**			**IoU**	**Image-level Balanced Accuracy**
	Framework	Train on Empty Images	Loss	Non-empty Images	
Two-stage Approaches					
Yes	U-Net	No	Cross-Entropy	0.383	0.781
Yes	U-Net	Yes	Cross-Entropy	0.336	0.760
Single-stage Approaches					
No	U-Net	Yes	Cross-Entropy	0.418	0.835
No	Hypercolumns	Yes	Cross-Entropy	0.418	0.82
No	Proposed	Yes	Cross-Entropy	**0.481**	**0.84**
No	U-Net	Yes	Focal Loss	0.463	0.818

The first two rows show the results for the two-stage pipeline where a foreground classifier is first used to filter out empty images and, then, a segmentation network is used to segment the non-empty images. The remaining rows are single-stage pipelines and the segmentation network performs both tasks directly.

Alternatively, by employing the multi-level fuse block, our method is able to take better advantage of the joint representation from Hypercolumns, which directly accounts to a 15% improvement in IoU and 2.4% in balanced accuracy.

Another conclusion that can be drawn from Table 8.2 is that the proposed model not only tackles the class imbalance problem but goes beyond that. By exchanging the pixel-wise cross-entropy (HEATON, 2018) by Focal Loss (LIN *et al.*, 2017) as the loss function, the U-Net baseline model improves its IoU by 10.7% while experiencing a small drop of 0.6% in image-level accuracy. Nevertheless, even with Focal Loss, the U-Net still is significantly outperformed by the proposed model, which suggests that the proposed elements not only handle the class imbalance but also bring additional benefits to the segmentation model. In fact, by exploring the Focal Loss during the training of the proposed model, the results do not change significantly evincing its intrinsic capability of dealing with unbalanced datasets.

Finally, the predicted segmentation masks for three random images from the SIIM-ACR dataset, which were generated by the single-stage methods of Table 8.2, are exhibited in Figure 8.1. Notably, it can be observed how the proposed method is able to detect pneumothoraces instances in two X-ray images that were mostly missed by the other models.

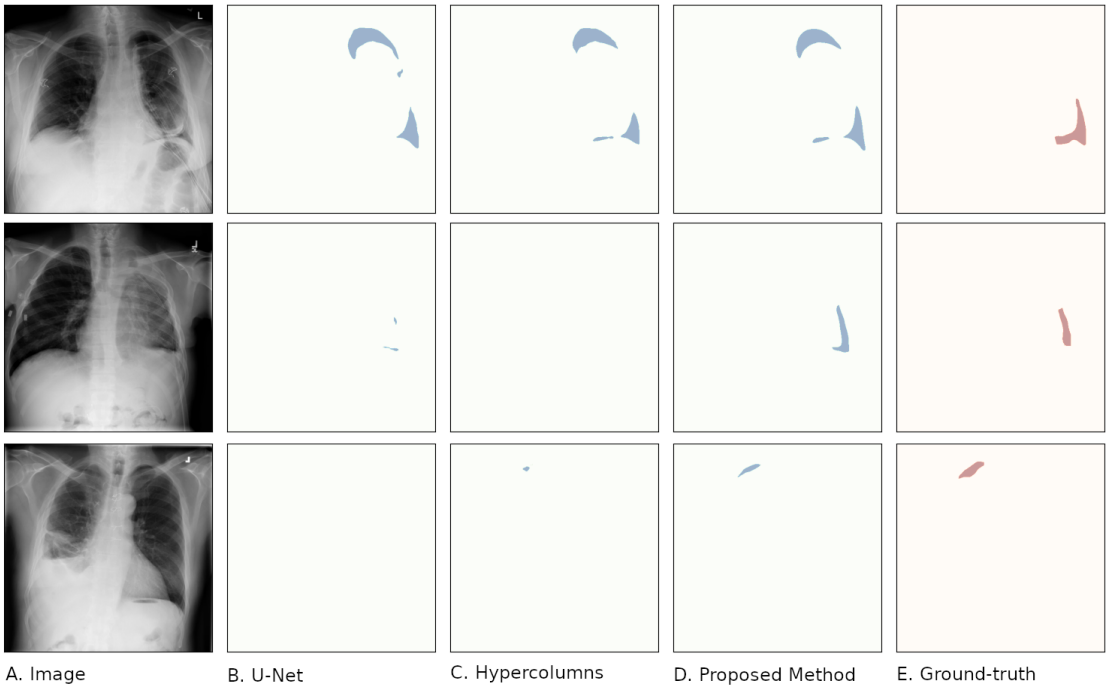

Figure 8.1 – **Some results on the SIIM-ACR dataset.** The first column shows the three random images used as input for the segmentation models, whilst the second, third and forth columns show the segmentation masks resultant from the U-Net (RONNEBERGER *et al.*, 2015), Hypercolumns (HARIHARAN *et al.*, 2014a) and the proposed model, respectively. The ground-truth segmentation masks are shown in the fifth column.

8.3 MS-COCO: BikeCar

Similarly to the experiments with SIIM-ACR Penumothorax Dataset, in this section the models and pipelines are evaluated in the MS-COCO: BikeCar dataset.

The results, which are displayed in Table 8.3, are in agreement with those of Table 8.2. Firstly, the single-stage pipelines outperform their two-stage counterparts. Secondly and, most importantly, the proposed model presents +13.3, 1.8 and 2% improvement in mIoU, multi-class and foreground accuracies, respectively, against its best competitors. Additionally, even with almost twice imbalance than the SIIM-ACR dataset, by itself, the original Hypercolumns is not sufficient to improve the results, which is consistent with the observation and rationale of Section 8.2.

8.4 Combining the Proposed Model with Modern Architectures

Given its simplicity and widespread adoption, U-Net (RONNEBERGER *et al.*, 2015) is definitively a well suited candidate to be used as baseline for this work's experiments. Nonetheless, due to the rapid growth of deep convolutional networks in

Table 8.3 – Results on the MS-COCO: BikeCar Dataset

Classification	**Segmentation**		**mIoU**	**Multi-class Accuracy**	**Image-level Balanced Accuracy**
	Framework	Train on Empty Images	Non-empty Images		
Two-stage Approaches					
Yes	U-Net	No	0.385	0.578	0.790
Yes	U-Net	Yes	0.397	0.572	0.785
Single-stage Approaches					
No	U-Net	Yes	0.456	0.619	0.836
No	Hypercolumns	Yes	0.448	0.622	0.848
No	Proposed	Yes	**0.517**	**0.630**	**0.853**

The first two rows show the results for the two-stage pipeline where a foreground classifier is first used to filter out empty images and, then, a segmentation network is used to segment the non-empty images. The remaining rows are single-stage pipelines and the segmentation network performs both tasks directly.

Table 8.4 – Results on the SIIM-ACR Pneumothorax Dataset Using EfficientDet and FaPN as Baseline

	IoU	
	Non-empty images	**Image-level Balanced Accuracy**
EfficientDet-D2	0.489	0.838
EfficientDet-D2 + *Proposed Method*	**0.506**	**0.851**
ResNet50-FaPN	0.471	**0.850**
ResNet50-FaPN + *Proposed Method*	**0.496**	0.841

In this experiment, EfficientDet-D2 (TAN et al.*, 2019) and ResNet50-FaPN (HUANG* et al.*, 2021) models were used as baseline to validate the add-in improvements of the proposed model on modern architectures. The experiment was performed using the SIIM-ACR Pneumothorax Dataset.*

computer vision tasks, its performance has been long surpassed by several recent works, such as PSPNet (ZHAO *et al.*, 2017), DeepLab variants (CHEN *et al.*, 2016; CHEN *et al.*, 2017; CHEN *et al.*, 2018), OCRNet (YUAN *et al.*, 2019), PANet (LI *et al.*, 2018), EfficientDet (TAN *et al.*, 2019) and Feature-aligned Pyramid Network (FaPN) (HUANG *et al.*, 2021). Thus, in this section, we incorporate the proposed method to the last two models (EfficientDet and FaPN) in order to analyze the impacts of such changes on more modern and stronger architectures.

Particularly, in the first experiment, the reference model employs EfficientNet-B2 (TAN; LE, 2019) and weighted-BiFPN (TAN *et al.*, 2019) as encoder and decoder,

respectively, whereas, in the second experiment, a ResNet-50 (HE *et al.*, 2016) was used as encoder for the recent Feature-aligned Pyramid Network (HUANG *et al.*, 2021) decoder.

The proposed modifications were applied according to the description in Section 6.1 and Figure 6.2. The sole difference, specifically for EfficientDet, is that we use the output of the last 5 decoder layers as inputs to the Hypercolumns block since the network has two extra pooling layers, leading to an increased output stride.

The results on the SIIM-ACR Pneumothorax dataset can be observed in Table 8.4. For EfficientDet, it is clear that the proposed modifications provide significative gains in both IoU (+3.5%) and balanced accuracy (1.5%), proving its versatility and usefulness even on top of a modern and very strong baseline. For FaPN, despite the small reduction of 1% in balanced accuracy, the IoU improved significantly (5%), which would justify its adoption in most applications.

8.5 Ablation Studies

In order to better comprehend the underlying mechanisms behind the improvements achieved by the proposed architecture, as observed in Tables 8.2, 8.3 and 8.4, we perform a series of ablation studies on the Multi-level Fuse block and the auxiliary foreground segmentation loss.

8.5.1 Multi-level Fuse Block

The proposed multi-level fuse block is comprised of three transformations: a convolution block $\mathcal{F}$, an input attention layer ψ_1 and a final attention ψ_2 operating at the output descriptor. Accordingly, the ablation process is performed by training several models on the SIIM-ACR dataset while incrementally adding each component to a base network. Naturally, the ResNet-50 (HE *et al.*, 2016) with modified U-Net (RONNEBERGER *et al.*, 2015) decoder and Hypercolumns (HARIHARAN *et al.*, 2014a) block – as specified in Section 6.1 – is a suitable candidate for such purpose and, therefore, was employed.

Table 8.5 displays the IoU and image-level balanced accuracy results obtained in the ablation studies. Even though, by itself, the spatial component $\mathcal{F}$ improved the IoU and accuracy by 2.1 and 1.5%, respectively, most gains occurred when the channel dimension was calibrated by the self-attention blocks ψ_1 and ψ_2. Furthermore, a remarkable IoU improvement of 12.1% was attained when ψ_1 calibrates the feature maps' channels prior to the spatial transformation $\mathcal{F}$, suggesting that the attention layers exploit global information to focus on certain previously disregarded channels, which likely contain small

objects, improving the segmentation quality. Finally, the second attention block ψ_2 operating at the output is able to further improve classification by 2.4% and IoU by 0.4%, which suggests that attention mechanisms not only can emphasize certain channels but also suppress less useful ones, reducing *false positives* – usually associated with small artifacts in empty images.

Table 8.5 – Ablation on Multi-level Fuse Block

	IoU	**Image-level Balanced Accuracy**
	Non-Empty only	
Hypercolumns	0.418	0.82
+ ConvFuse (F)	0.427	0.833
+ SE Block In (ψ_1)	0.479	0.819
+ SE Block Out (ψ_2)	**0.481**	**0.84**

The results for different MLF blocks on the SIIM-ACR Pneumothorax dataset are shown. The baseline, when only Hypercolumns is used, is displayed in the first row. For the next rows, the increments are added upon the previous rows until we have the complete architecture in the last row.

8.5.2 Auxiliary Foreground Segmentation Loss

The impacts of the auxiliary foreground segmentation loss of Equation (6.3) in the perceived gains of Table 8.3 were also investigated. This was achieved by studying the influence of the loss by modulating the hyperparameter α of Equation (6.4). The results for the MS-COCO: BikeCar dataset are displayed in the graph of Figure 8.2, where it is possible to observe that by increasing α, i.e., the auxiliary loss influence, the mIoU metric is significantly improved, saturating only around $\alpha = 0.3$. The accuracies, on the other hand, quickly reach their peak at $\alpha = 0.1$. Considering the trade-off between the metrics, we decided to use 0.1 as the default value for the α.

Surprisingly, for the SIIM-ACR Pneumothorax dataset, it was found that the usage of the auxiliary loss function does not significantly improve the results. It should be highlighted, however, that the aforementioned dataset is almost two times more balanced than the MS-COCO subset. Hence, considering that one of the main ideas behind the auxiliary loss is to tackle class imbalance, it is expected that its benefits should diminish as the dataset becomes more balanced.

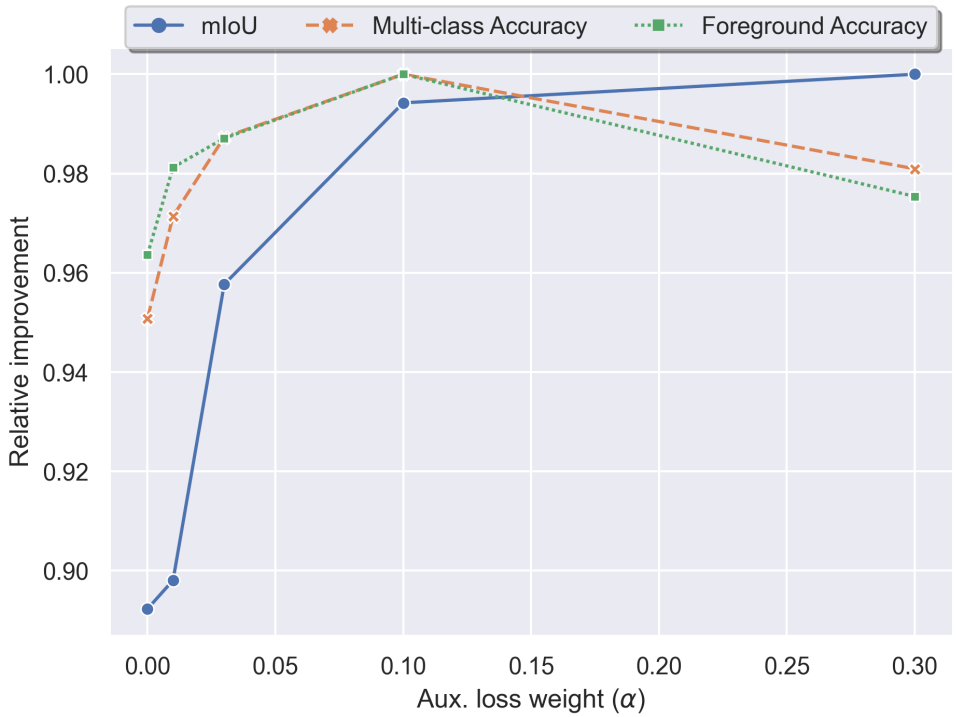

Figure 8.2 – **Ablation study on the auxiliary loss.** Although all metrics benefit from the additional loss, clearly the mIoU is the most sensitive since it suffers most from the class imbalance problem.

9 Conclusion

In this work, a careful study was performed to determine effective approaches to deal with semantic segmentation tasks with full background images. The two most common pipelines used in production – single- and two-stage – were presented and compared. Also, image-level batch sampling schemes often used in image classification tasks and that are neglected in semantic segmentation were studied. Interestingly, we verified that, in the context of empty images, such methods can be powerful allies to tackle the class imbalance towards empty images.

As our main contribution, a novel CNN extension for encoder-decoder architectures was presented for the context of semantic segmentation under extreme imbalance towards full-background images. The extension mainly relies on (1) a inverted version of Hypercolumns (HARIHARAN *et al.*, 2014a) to fuse the decoder's feature maps, aiming to exploit different levels of high-frequency information and semantic content; (2) a calibration layer based on the SE attention block (HU *et al.*, 2018); (3) an auxiliary loss function used to refine the segmentation quality on *foreground* images.

The proposed modifications achieved promising results, significantly outperforming its competitors in two different datasets. The first, SIIM-ACR Pneumothorax dataset, presents a challenging real-world application in the context of medical imaging. The second, on the other hand, is comprised by natural images and is built by modifying MS-COCO (LIN *et al.*, 2015) – a famous dataset in the literature – to adapt it to this work's context. The positive results of our method on both datasets evinces its general capability to improve encoder-decoder architectures when a significant number of empty images are present in the application context. Notoriously, by also applying our changes to two state-of-the-art models, namely EfficientDet (TAN *et al.*, 2019) and FaPN (HUANG *et al.*, 2021), we showed that our proposed extension is flexible and can provide gains even to strong modern segmentation architectures.

Additionally, enlightening ablation studies were performed on the Multi-level Fuse block and auxiliary foreground segmentation loss function in order to better comprehend the internal mechanisms of the proposed method. The first evinced that each element of the MLF block adds useful information for the semantic segmentation, whilst the input SE block ψ_1 brings most gains in terms of IoU. The latter has shown how the weight for the auxiliary loss α controls the trade-off between IoU and image-level accuracy. Additionally, it demonstrated that, for a well calibrated value of α, the auxiliary loss function brings significant gains both in IoU and accuracy.

9.1 Research Questions

In Section 1.4, the main questions that drove our investigation and research were presented. Now, we provide the clarification that emerged from our research.

1. What are the difficulties particularly imposed by the presence of *empty* images in the dataset?

    - Naturally, when a dataset is comprised by a large number of empty images, its class distribution gets skewed. This is specially true in semantic segmentation tasks where foreground images usually also have a significant number of background pixels. Thus, class imbalance is the main difficulty that must be addressed in the empty image context. Additionally, the detection of small objects also gets impaired when empty images are present, often mistaken by an empty mask. Hence, scale invariant loss functions and metrics are advised in this context.

2. Considering class imbalance, how single-stage approaches fare against their more complex two-stage counterparts?

    - It was observed that despite single-stage approaches having to deal with greater class imbalance, such class of models can exploit techniques to address the issue, e.g., Focal Loss (LIN *et al.*, 2017) and undersampling (BRANCO *et al.*, 2015; HE; MA, 2013), which tends to yield better performance than its two-stage counterpart.

3. Is it possible to employ "smart" image-level batch sampling schemes, commonly used in image classification tasks, to mitigate class imbalance in semantic segmentation with *empty* images?

    - It was found that batch sampling techniques not only can be employed with empty images but also yield significant improvements for single-stage approaches. When tested in the SIIM-ACR Pneumothorax dataset, improvements of over 18% in IoU and 5% in global accuracy were observed.

4. Can a especially-tailored architecture for the task at hand improve the results when compared to the *default* approaches?

    - By proposing an extension to encoder-decoder architectures carefully designed for the context at hand, we confirmed that specificity to the task can be a powerful aid when working with empty images. With simple modifications over the baseline, the proposed model is able to provide significant gains on both SIIM-ACR and MS-COCO: BikeCar datasets.

9.2 Future Research

At this point, we hope to have successfully managed to bring the – common, yet often overlooked – topic of SSEI to the spotlight. With that in mind and aiming to pave the way for further research on the theme, in this section we present some perspectives for future works.

- **Parametrized evaluation:** One possible follow-up experiment for this research would be to create a dataset where the amount of imbalance towards empty image is parametrized. In this setup, one should be able to evaluate the benefits of the proposed method as the imbalance gradually increases.
- **Application on networks without decoders:** Even though many state-of-the-art semantic segmentation networks still use decoders, e.g., (TAN *et al.*, 2019; HUANG *et al.*, 2021), others, like (CHEN *et al.*, 2018; YUAN *et al.*, 2019), decided to only remove pooling operations in the encoder and apply dilation to the last convolutional layers. Thus, under minor adaptations, one should be able to apply the proposed model on such network family.
- **Different attention mechanisms:** As observed in the ablation study of Chapter 8, the attention layer in the Multi-level Fuse Block plays a key role in the proposed method. However, in this work, only the pioneer and most famous Squeeze-and-Excitation (HU *et al.*, 2018) block was explored. Further research may try to employ different types of attention mechanisms such as CBAM (WOO *et al.*, 2018), scSE (ROY *et al.*, 2018) and ECA (WANG *et al.*, 2019).

Bibliography

Anand, R.; Mehrotra, K. G.; Mohan, C. K.; Ranka, S. An improved algorithm for neural network classification of imbalanced training sets. *IEEE Transactions on Neural Networks*, v. 4, n. 6, p. 962–969, 1993. Cited on page 32.

BA, J. L.; KIROS, J. R.; HINTON, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. Cited on page 47.

BAUDER, R.; KHOSHGOFTAAR, T. The effects of varying class distribution on learner behavior for medicare fraud detection with imbalanced big data. *Health Information Science and Systems*, v. 6, 12 2018. Cited on page 32.

Bauder, R. A.; Khoshgoftaar, T. M.; Hasanin, T. An empirical study on class rarity in big data. In: *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. [S.l.: s.n.], 2018. p. 785–790. Cited on page 32.

BJORCK, J.; GOMES, C.; SELMAN, B.; WEINBERGER, K. Q. *Understanding Batch Normalization*. 2018. Cited on page 47.

BRANCO, P.; TORGO, L.; RIBEIRO, R. *A Survey of Predictive Modelling under Imbalanced Distributions*. 2015. Cited 3 times on pages 32, 84, and 94.

BROCK, A.; DE, S.; SMITH, S. L.; SIMONYAN, K. High-performance large-scale image recognition without normalization. *arXiv preprint arXiv:2102.06171*, 2021. Cited on page 47.

BRODERSEN, K. H.; ONG, C. S.; STEPHAN, K. E.; BUHMANN, J. M. The balanced accuracy and its posterior distribution. In: *Proceedings - International Conference on Pattern Recognition*. [S.l.: s.n.], 2010. p. 3121–3124. ISBN 9780769541099. ISSN 10514651. Cited on page 82.

CHEN, L.; PAPANDREOU, G.; KOKKINOS, I.; MURPHY, K.; YUILLE, A. L. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, abs/1606.00915, 2016. Disponível em: <http://arxiv.org/abs/1606.00915>. Cited on page 89.

CHEN, L.-C.; PAPANDREOU, G.; SCHROFF, F.; ADAM, H. *Rethinking Atrous Convolution for Semantic Image Segmentation*. 2017. Cited on page 89.

CHEN, L.-C.; ZHU, Y.; PAPANDREOU, G.; SCHROFF, F.; ADAM, H. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. feb 2018. Disponível em: <http://arxiv.org/abs/1802.02611>. Cited 6 times on pages 9, 29, 64, 83, 89, and 95.

CHOLLET, F. Xception: Deep learning with depthwise separable convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2017. p. 1251–1258. Cited 2 times on pages 56 and 66.

CIREsAN, D. C.; GIUSTI, A.; GAMBARDELLA, L. M.; SCHMIDHUBER, J. Deep neural networks segment neuronal membranes in electron microscopy images. In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*. Red Hook, NY, USA: Curran Associates Inc., 2012. (NIPS'12), p. 2843–2851. Cited on page 61.

CLEVERT, D.-A.; UNTERTHINER, T.; HOCHREITER, S. Fast and accurate deep network learning by exponential linear units (elus). *arXiv: Learning*, 2016. Cited on page 37.

CORDTS, M.; OMRAN, M.; RAMOS, S.; REHFELD, T.; ENZWEILER, M.; BENENSON, R.; FRANKE, U.; ROTH, S.; SCHIELE, B.; R&D, D. A.; DARMSTADT, T. U. *The Cityscapes Dataset for Semantic Urban Scene Understanding*. [S.l.], 2016. Disponível em: <www.cityscapes-dataset.net>. Cited 3 times on pages 24, 29, and 81.

DAI, J.; QI, H.; XIONG, Y.; LI, Y.; ZHANG, G.; HU, H.; WEI, Y. Deformable convolutional networks. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2017. p. 764–773. Cited on page 67.

DE, S.; SMITH, S. L. *Batch Normalization Biases Residual Blocks Towards the Identity Function in Deep Networks*. 2020. Cited on page 47.

DOSOVITSKIY, A.; BEYER, L.; KOLESNIKOV, A.; WEISSENBORN, D.; ZHAI, X.; UNTERTHINER, T.; DEHGHANI, M.; MINDERER, M.; HEIGOLD, G.; GELLY, S.; USZKOREIT, J.; HOULSBY, N. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2020. Cited on page 31.

DUMOULIN, V.; VISIN, F. *A guide to convolution arithmetic for deep learning*. 2018. Cited 4 times on pages 39, 62, 64, and 75.

EVERINGHAM, M.; Van Gool, L.; I Williams, C. K.; WINN, J.; ZISSERMAN, A.; EVERINGHAM, M.; Van Gool Leuven, L. K.; CKI Williams, B.; WINN, J.; ZISSERMAN, A. The PASCAL Visual Object Classes (VOC) Challenge. *Int J Comput Vis*, v. 88, p. 303–338, 2010. Disponível em: <http://www.flickr.com/>. Cited 3 times on pages 24, 29, and 81.

Farabet, C.; Couprie, C.; Najman, L.; LeCun, Y. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 35, n. 8, p. 1915–1929, 2013. Cited on page 61.

FELDMAN, J. A.; FELDMAN, G. M.; FALK, G.; GRAPE, G.; PEARLMAN, J.; SOBEL, I.; TENENBAUM, J. M. The stanford hand-eye project. In: WALKER, D. E.; NORTON, L. M. (Ed.). *Proceedings of the 1st International Joint Conference on Artificial Intelligence, Washington, DC, USA, May 7-9, 1969*. William Kaufmann, 1969. p. 521–526. Disponível em: <http://ijcai.org/Proceedings/69/Papers/046A.pdf>. Cited on page 43.

Feng Ning; Delhomme, D.; LeCun, Y.; Piano, F.; Bottou, L.; Barbano, P. E. Toward automatic phenotyping of developing embryos from videos. *IEEE Transactions on Image Processing*, v. 14, n. 9, p. 1360–1371, 2005. Cited on page 61.

GANIN, Y.; LEMPITSKY, V. *N^4-Fields: Neural Network Nearest Neighbor Fields for Image Transforms*. 2014. Cited on page 61.

GLOROT, X.; BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. In: JMLR WORKSHOP AND CONFERENCE PROCEEDINGS. *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. [S.l.], 2010. p. 249–256. Cited on page 46.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>. Cited 10 times on pages 9, 34, 36, 39, 40, 41, 42, 43, 44, and 83.

HARIHARAN, B.; ARBELÁEZ, P.; GIRSHICK, R.; MALIK, J. Hypercolumns for Object Segmentation and Fine-grained Localization. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, v. 07-12-June-2015, p. 447–456, nov 2014. Disponível em: <http://arxiv.org/abs/1411.5752>. Cited 10 times on pages 10, 12, 59, 60, 61, 74, 86, 88, 90, and 93.

HARIHARAN, B.; ARBELáEZ, P.; GIRSHICK, R.; MALIK, J. *Simultaneous Detection and Segmentation*. 2014. Cited on page 61.

He, H.; Garcia, E. A. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, v. 21, n. 9, p. 1263–1284, 2009. Cited on page 84.

HE, H.; MA, Y. *Imbalanced learning: foundations, algorithms, and applications*. [S.l.]: John Wiley & Sons, 2013. Cited 3 times on pages 32, 84, and 94.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2015. p. 1026–1034. Cited 3 times on pages 9, 37, and 38.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. [S.l.]: IEEE Computer Society, 2016. v. 2016-December, p. 770–778. ISBN 9781467388504. ISSN 10636919. Cited 15 times on pages 10, 28, 45, 48, 49, 50, 51, 52, 53, 54, 56, 66, 75, 82, and 90.

HEATON, J. Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning. *Genetic Programming and Evolvable Machines*, Springer Nature, v. 19, n. 1-2, p. 305–307, jun 2018. ISSN 1389-2576. Cited on page 87.

HOFFER, E.; HUBARA, I.; SOUDRY, D. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. 05 2017. Cited on page 47.

HOSSEINI-ASL, E.; GIMEL'FARB, G. L.; EL-BAZ, A. Alzheimer's disease diagnostics by a deeply supervised adaptable 3d convolutional network. *CoRR*, abs/1607.00556, 2016. Disponível em: <http://arxiv.org/abs/1607.00556>. Cited on page 27.

HU, J.; SHEN, L.; SUN, G. Squeeze-and-Excitation Networks. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2018. p. 7132–7141. ISBN 9781538664209. ISSN 10636919. Disponível em: <http://arxiv.org/abs/1709.01507>. Cited 13 times on pages 10, 28, 51, 53, 54, 55, 56, 68, 74, 75, 76, 93, and 95.

HUANG, S.; LU, Z.; CHENG, R.; HE, C. FaPN: Feature-aligned pyramid network for dense image prediction. In: *International Conference on Computer Vision (ICCV)*. [S.l.: s.n.], 2021. Cited 10 times on pages 11, 67, 68, 69, 80, 83, 89, 90, 93, and 95.

IOFFE, S. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. *arXiv preprint arXiv:1702.03275*, 2017. Cited on page 47.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: *32nd International Conference on Machine Learning, ICML 2015*. International Machine Learning Society (IMLS), 2015. v. 1, p. 448–456. ISBN 9781510810587. Disponível em: <https://arxiv.org/abs/1502.03167v3>. Cited 7 times on pages 12, 46, 47, 49, 50, 77, and 78.

JOHNSON, J. M.; KHOSHGOFTAAR, T. M. Survey on deep learning with class imbalance. *Journal of Big Data*, SpringerOpen, v. 6, n. 1, dec 2019. ISSN 21961115. Cited on page 32.

Ker, J.; Wang, L.; Rao, J.; Lim, T. Deep learning applications in medical image analysis. *IEEE Access*, v. 6, p. 9375–9389, 2018. Cited on page 28.

KRIZHEVSKY, A.; HINTON, G. *et al.* Learning multiple layers of features from tiny images. Citeseer, 2009. Cited 2 times on pages 10 and 49.

LECUN, Y.; BOSER, B.; DENKER, J. S.; HENDERSON, D.; HOWARD, R. E.; HUBBARD, W.; JACKEL, L. D. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, v. 1, n. 4, p. 541–551, 1989. ISSN 0899-7667. Disponível em: <https://doi.org/10.1162/neco.1989.1.4.541>. Cited 2 times on pages 21 and 48.

Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, v. 86, n. 11, p. 2278–2324, 1998. Cited on page 51.

LI, H.; XIONG, P.; AN, J.; WANG, L. Pyramid Attention Network for Semantic Segmentation. may 2018. Disponível em: <http://arxiv.org/abs/1805.10180>. Cited 7 times on pages 64, 74, 75, 76, 83, 86, and 89.

LIN, T.-Y.; DOLLÁR, P.; GIRSHICK, R.; HE, K.; HARIHARAN, B.; BELONGIE, S. Feature Pyramid Networks for Object Detection. dec 2016. Disponível em: <http://arxiv.org/abs/1612.03144>. Cited 7 times on pages 11, 30, 64, 65, 66, 67, and 75.

LIN, T.-Y.; GOYAL, P.; GIRSHICK, R.; HE, K.; DOLLÁR, P. Focal Loss for Dense Object Detection. aug 2017. Disponível em: <http://arxiv.org/abs/1708.02002>. Cited 8 times on pages 11, 33, 71, 72, 80, 83, 87, and 94.

LIN, T.-Y.; MAIRE, M.; BELONGIE, S.; BOURDEV, L.; GIRSHICK, R.; HAYS, J.; PERONA, P.; RAMANAN, D.; ZITNICK, C. L.; DOLÍ, P. *Microsoft COCO: Common Objects in Context*. [S.l.], 2015. Cited 5 times on pages 24, 29, 80, 81, and 93.

LIU, L.; JIANG, H.; HE, P.; CHEN, W.; LIU, X.; GAO, J.; HAN, J. On the Variance of the Adaptive Learning Rate and Beyond. aug 2019. Disponível em: <http://arxiv.org/abs/1908.03265>. Cited on page 83.

LIU, S.; QI, L.; QIN, H.; SHI, J.; JIA, J. Path aggregation network for instance segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2018. p. 8759–8768. Cited 3 times on pages 11, 65, and 66.

LONG, J.; SHELHAMER, E.; DARRELL, T. Fully Convolutional Networks for Semantic Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE Computer Society, v. 39, n. 4, p. 640–651, nov 2014. Disponível em: <http://arxiv.org/abs/1411.4038>. Cited 7 times on pages 11, 59, 61, 62, 63, 64, and 72.

LOSHCHILOV, I.; HUTTER, F. SGDR: Stochastic Gradient Descent with Warm Restarts. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, International Conference on Learning Representations, ICLR, aug 2016. Disponível em: <http://arxiv.org/abs/1608.03983>. Cited on page 83.

LUO, P.; WANG, X.; SHAO, W.; PENG, Z. Towards understanding regularization in batch normalization. *ArXiv*, abs/1809.00846, 2019. Cited on page 47.

MAAS, A. L.; HANNUN, A. Y.; NG, A. Y. Rectifier nonlinearities improve neural network acoustic models. In: CITESEER. *Proc. icml*. [S.l.], 2013. v. 30, n. 1, p. 3. Cited 3 times on pages 9, 37, and 38.

NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: *Icml*. [S.l.: s.n.], 2010. Cited 7 times on pages 35, 36, 46, 49, 50, 53, and 66.

NARANG, S.; DIAMOS, G.; ELSEN, E.; MICIKEVICIUS, P.; ALBEN, J.; GARCIA, D.; GINSBURG, B.; HOUSTON, M.; KUCHAIEV, O.; VENKATESH, G.; WU, H. Mixed precision training. In: *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*. [S.l.]: International Conference on Learning Representations, ICLR, 2018. Cited on page 83.

Ni Fhlatharta, M.; EATON, D. A. Pneumothorax and chest drain insertion. *Surgery (Oxford)*, Elsevier Ltd, v. 38, n. 5, p. 275–279, may 2020. ISSN 02639319. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0263931920300454>. Cited on page 81.

ODENA, A.; DUMOULIN, V.; OLAH, C. Deconvolution and checkerboard artifacts. *Distill*, v. 1, n. 10, p. e3, 2016. Cited on page 75.

PASZKE, A.; GROSS, S.; CHINTALA, S.; CHANAN, G.; YANG, E.; FACEBOOK, Z. D.; RESEARCH, A. I.; LIN, Z.; DESMAISON, A.; ANTIGA, L.; SRL, O.; LERER, A. *Automatic differentiation in PyTorch*. [S.l.], 2017. Cited 2 times on pages 45 and 83.

PENG, C.; ZHANG, X.; YU, G.; LUO, G.; SUN, J. Large kernel matters — improve semantic segmentation by global convolutional network. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jul 2017. Disponível em: <http://dx.doi.org/10.1109/CVPR.2017.189>. Cited on page 83.

Pereira, S.; Pinto, A.; Alves, V.; Silva, C. A. Brain tumor segmentation using convolutional neural networks in mri images. *IEEE Transactions on Medical Imaging*, v. 35, n. 5, p. 1240–1251, 2016. Cited on page 28.

PHAM, H.; XIE, Q.; DAI, Z.; LE, Q. V. Meta Pseudo Labels. mar 2020. Disponível em: <http://arxiv.org/abs/2003.10580>. Cited on page 28.

PHAM, H.; XIE, Q.; DAI, Z.; LE, Q. V. Meta Pseudo Labels. mar 2020. Disponível em: <http://arxiv.org/abs/2003.10580>. Cited on page 28.

PINHEIRO, P. H. O.; COLLOBERT, R. *Recurrent Convolutional Neural Networks for Scene Parsing*. 2013. Cited on page 61.

PRATT, H.; COENEN, F.; BROADBENT, D. M.; HARDING, S. P.; ZHENG, Y. Convolutional Neural Networks for Diabetic Retinopathy. In: *Procedia Computer Science*. Elsevier B.V., 2016. v. 90, p. 200–205. ISSN 18770509. Disponível em: <www.sciencedirect.com>. Cited on page 27.

RAJPURKAR, P.; IRVIN, J.; ZHU, K.; YANG, B.; MEHTA, H.; DUAN, T.; DING, D. Y.; BAGUL, A.; LANGLOTZ, C.; SHPANSKAYA, K. S.; LUNGREN, M. P.; NG, A. Y. Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. *CoRR*, abs/1711.05225, 2017. Disponível em: <http://arxiv.org/abs/1711.05225>. Cited on page 27.

RAMACHANDRAN, P.; ZOPH, B.; LE, Q. Swish: a self-gated activation function. 10 2017. Cited 3 times on pages 38, 46, and 56.

RAMACHANDRAN, P.; ZOPH, B.; LE, Q. V. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017. Cited 3 times on pages 38, 46, and 56.

RONNEBERGER, O.; FISCHER, P.; BROX, T. U-net: Convolutional networks for biomedical image segmentation. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. [S.l.]: Springer Verlag, 2015. v. 9351, p. 234–241. ISBN 9783319245737. ISSN 16113349. Cited 11 times on pages 11, 12, 30, 63, 64, 65, 75, 82, 86, 88, and 90.

ROY, A. G.; NAVAB, N.; WACHINGER, C. Concurrent spatial and channel 'squeeze & excitation'in fully convolutional networks. In: SPRINGER. *International conference on medical image computing and computer-assisted intervention*. [S.l.], 2018. p. 421–429. Cited on page 95.

RUMELHART, D.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Nature*, v. 323, p. 533–536, 1986. Cited 3 times on pages 32, 62, and 79.

RUSSAKOVSKY, O.; DENG, J.; SU, H.; KRAUSE, J.; SATHEESH, S.; MA, S.; HUANG, Z.; KARPATHY, A.; KHOSLA, A.; BERNSTEIN, M.; AL. et. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, Springer Science and Business Media LLC, v. 115, n. 3, p. 211–252, Apr 2015. ISSN 1573-1405. Disponível em: <http://dx.doi.org/10.1007/s11263-015-0816-y>. Cited 6 times on pages 21, 27, 31, 50, 59, and 82.

SANCHEZ, J.; PERRONNIN, F. High-dimensional signature compression for large-scale image classification. In: . [S.l.: s.n.], 2011. p. 1665–1672. Cited on page 28.

SANDLER, M.; HOWARD, A.; ZHU, M.; ZHMOGINOV, A.; CHEN, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2018. p. 4510–4520. Cited on page 56.

SANTURKAR, S.; TSIPRAS, D.; ILYAS, A.; MADRY, A. *How Does Batch Normalization Help Optimization?* 2019. Cited on page 47.

SIMONYAN, K.; ZISSERMAN, A. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. Cited on page 75.

SORENSEN, T. J. *A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on Danish commons*. [S.l.]: I kommission hos E. Munksgaard, 1948. Cited 2 times on pages 24 and 28.

SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, JMLR. org, v. 15, n. 1, p. 1929–1958, 2014. Cited on page 56.

SUDRE, C. H.; LI, W.; VERCAUTEREN, T.; OURSELIN, S.; CARDOSO, M. J. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In: *Deep learning in medical image analysis and multimodal learning for clinical decision support*. [S.l.]: Springer, 2017. p. 240–248. Cited 3 times on pages 71, 74, and 78.

SZEGEDY, C.; VANHOUCKE, V.; IOFFE, S.; SHLENS, J.; WOJNA, Z. *Rethinking the Inception Architecture for Computer Vision*. 2015. Cited on page 51.

TAN, M.; CHEN, B.; PANG, R.; VASUDEVAN, V.; SANDLER, M.; HOWARD, A.; LE, Q. V. Mnasnet: Platform-aware neural architecture search for mobile. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2019. p. 2820–2828. Cited on page 56.

TAN, M.; LE, Q. V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. may 2019. Disponível em: <http://arxiv.org/abs/1905.11946>. Cited 13 times on pages 10, 13, 28, 51, 54, 55, 56, 57, 58, 65, 66, 75, and 89.

TAN, M.; PANG, R.; LE, Q. V. EfficientDet: Scalable and Efficient Object Detection. nov 2019. Disponível em: <http://arxiv.org/abs/1911.09070>. Cited 10 times on pages 11, 65, 66, 67, 75, 80, 83, 89, 93, and 95.

ULYANOV, D.; VEDALDI, A.; LEMPITSKY, V. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. Cited on page 47.

VANHOUCKE, V. Learning visual representations at scale. *ICLR invited talk*, v. 1, p. 2, 2014. Cited 2 times on pages 56 and 66.

WANG, Q.; WU, B.; ZHU, P.; LI, P.; ZUO, W.; HU, Q. Eca-net: Efficient channel attention for deep convolutional neural networks. *CoRR*, abs/1910.03151, 2019. Disponível em: <http://arxiv.org/abs/1910.03151>. Cited on page 95.

Wang, S.; Yao, X. Multiclass imbalance problems: Analysis and potential solutions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, v. 42, n. 4, p. 1119–1130, 2012. Cited on page 32.

WEISSBERG, D.; REFAELY, Y. Pneumothorax: Experience with 1,199 patients. *Chest*, American College of Chest Physicians, v. 117, n. 5, p. 1279–1285, may 2000. ISSN 00123692. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0012369215350856>. Cited on page 81.

WOO, S.; PARK, J.; LEE, J.-Y.; KWEON, I. S. Cbam: Convolutional block attention module. In: *Proceedings of the European conference on computer vision (ECCV)*. [S.l.: s.n.], 2018. p. 3–19. Cited on page 95.

WU, Y.; HE, K. Group normalization. In: *Proceedings of the European conference on computer vision (ECCV)*. [S.l.: s.n.], 2018. p. 3–19. Cited on page 47.

XIE, Q.; LUONG, M.-T.; HOVY, E.; LE, Q. V. *Self-training with Noisy Student improves ImageNet classification*. 2020. Cited on page 31.

XIE, S.; GIRSHICK, R.; DOLLáR, P.; TU, Z.; HE, K. *Aggregated Residual Transformations for Deep Neural Networks*. 2017. Cited on page 28.

YARMUS, L.; FELLER-KOPMAN, D. Pneumothorax in the critically ill patient. *Chest*, American College of Chest Physicians, v. 141, n. 4, p. 1098–1105, apr 2012. ISSN 19313543. Cited on page 81.

YU, F.; KOLTUN, V. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. Cited 2 times on pages 29 and 30.

YUAN, Y.; CHEN, X.; WANG, J. Object-Contextual Representations for Semantic Segmentation. sep 2019. Disponível em: <http://arxiv.org/abs/1909.11065>. Cited 2 times on pages 89 and 95.

ZHAO, H.; SHI, J.; QI, X.; WANG, X.; JIA, J. PSPNet. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, v. 2017-Janua, p. 6230–6239, dec 2017. Cited on page 89.

ZHU, X.; HU, H.; LIN, S.; DAI, J. Deformable convnets v2: More deformable, better results. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2019. p. 9308–9316. Cited on page 67.