Contribution to the development of an intelligent system of quantification of nutrients in meals from subsaharan Africa.

Thierry Roland Baban a Erep

RÉPUBLIQUE FRANÇAISE
Liberté Égalité Fraternité

Université de Toulouse

Doctorat de l'Université de Toulouse

délivré en co-tutelle avec Université de Yaoundé I

préparé à Toulouse INP

Contribution au développement d'un système intelligent de quantification des nutriments dans les repas d'Afrique subsaharienne.

Thèse présentée et soutenue, le 13 décembre 2024 par

Thierry BABAN A EREP

École doctorale
EDMITT - Ecole Doctorale Mathématiques, Informatique et Télécommunications de Toulouse

Spécialité
Informatique et Télécommunications

Unité de recherche
IRIT : Institut de Recherche en Informatique de Toulouse

Thèse dirigée par
Lotfi CHAARI et Eugène SOBNGWI

Composition du jury
Mme Sylvie CHAMBON, Présidente, Toulouse INP
M. Mounir KAANICHE, Rapporteur, Université Sorbonne Paris Nord
M. Thomas Florent KANAA, Rapporteur, Université d'Ebolowa
M. Zagrouba EZZEDDINE, Examinateur, Université de Tunis El Manar
M. Lotfi CHAARI, Directeur de thèse, Toulouse INP
M. Eugene SOBNGWI, Co-directeur de thèse, Université de Yaoundé 1

Membres invités
M. Pierre ELE, Ecole Nationale Supérieure Polytechnique de Yaoundé

Abstract

Malnutrition, including under- and overnutrition, is a global health challenge affecting billions of people. It impacts all organ systems and is a significant risk factor for noncommunicable diseases such as cardiovascular diseases, diabetes, and some cancers. Assessing food intake is crucial for preventing malnutrition but remains challenging. Traditional methods for dietary assessment are labor-intensive, time consuming and prone to underreporting bias. Advancements in AI have made Vision-Based Dietary Assessment (VBDA) a promising solution for automatically analyzing food images to estimate portions and nutrition. However, food image segmentation, a key task in VBDA, faces many challenges such as *food's non-rigid structure*, *high intra-class variation* (where the same dish can look very different), *inter-class resemblance* (where different foods appear similar) and the limited availability of public datasets.

Almost all food segmentation research has focused on Asian and Western foods, with no datasets currently available that include images of African food. African dishes often involve mixed food classes, making accurate segmentation challenging. Additionally, research has largely focus on RGB images, which provides color and texture but may lack geometric detail. To address this, RGB-D segmentation combines depth data with RGB images. Depth images provide crucial geometric details that enhance RGB data, improve object discrimination, and are robust to factors like illumination and fog. Despite its success in other fields, RGB-D segmentation for food is underexplored due to difficulties in collecting food depth images.

This thesis makes key contributions by developing new deep learning models for RGB (mid-DeepLabv3+) and RGB-D (ESeNet-D) image segmentation and introducing the first food segmentation datasets focused on African food images. **mid-DeepLabv3+** is based on DeepLabv3+, featuring a simplified ResNet backbone with and added skip layer (*middle layer*) in the decoder and SimAM attention mechanism. This model achieves the performance of the reference CNN models, while carrying half the computational load,

providing an optimum compromise between performance and computational efficiency. **ESeNet-D** consists on two encoder branches using EfficientNetV2 as backbone, with a fusion block for multi-scale integration and a decoder employing self-calibrated convolution and learned interpolation for precise segmentation. ESeNet-D outperforms many RGB and RGB-D benchmark models while having fewer parameters and FLOPs. Our experiments show that, when properly integrated, depth information can significantly improves food segmentation accuracy. We also present two new datasets : **AfricaFoodSeg** for *food/non-food* segmentation with 3,067 images (2,525 for training, 542 for validation), and CamerFood focusing on Cameroonian cuisine. CamerFood datasets include **CamerFood10** with 1,422 images from ten food classes, and **CamerFood15**, an enhanced version with 15 food classes, 1,684 training images, and 514 validation images. Finally, we address the challenge of scarce depth data in RGB-D food segmentation by demonstrating that Monocular Depth Estimation (MDE) models can aid in generating effective depth maps for RGB-D datasets.

Keywords : *Semantic Segmentation, Food Image, RGB-Depth Image, CNNs, Deep learning*

Résumé

La malnutrition, qu'elle soit liée à un apport insuffisant ou excessif en nutriments, représente un défi mondial de santé publique touchant des milliards de personnes. Elle affecte tous les systèmes organiques en étant un facteur majeur de risque pour les maladies non transmissibles telles que les maladies cardiovasculaires, le diabète et certains cancers. Évaluer l'apport alimentaire est crucial pour prévenir la malnutrition, mais cela reste un défi. Les méthodes traditionnelles d'évaluation de l'alimentation laborieuses, demandent beaucoup de temps et sont sujettes à des biais de sous-déclaration. Les progrès de l'IA ont permis la conception de systèmes VBDA, une solution prometteuse pour l'analyse automatique des images d'aliments afin d'estimer les portions et la composition nutritionnelle des aliments. Cependant, la segmentation des images d'aliments, une tâche clé des systèmes VBDA, est confrontée à de nombreux défis tels que la structure non rigide des aliments, de la *variation intra-classe élevée* (où le même type d'aliment peut apparaître très différent), de la *ressemblance inter-classe* (où différents types d'aliments semblent visuellement très similaires) et de la rareté des ensembles de données disponibles publiquement.

Presque toutes les recherches sur la segmentation des aliments se sont concentrées sur les aliments asiatiques et occidentaux, et il n'existe actuellement aucun ensemble de données comprenant des images d'aliments africains. Les plats africains comportent souvent des classes d'aliments mélangés, ce qui rend difficile une segmentation précise. En outre, la recherche s'est largement concentrée sur les images RGB, qui fournissent des couleurs et des textures mais peuvent manquer de détails géométriques. Pour remédier à ce problème, la segmentation RGB-D combine les données de profondeur avec les images RGB. Les images de profondeur fournissent des détails géométriques cruciaux qui améliorent les données RGB, améliorent la discrimination des objets et sont résistantes à des facteurs tels que l'éclairage et le brouillard. Malgré son succès dans d'autres domaines, la segmentation RGB-D pour les aliments est peu explorée en raison des difficultés à collecter des images de profondeur des aliments.

Cette thèse apporte des contributions clés en développant de nouveaux modèles d'apprentissage profond pour la segmentation d'images RGB (mid-DeepLabv3+) et RGB-D (ESeNet-D) et en introduisant les premiers ensembles de données axés sur les images alimentaires africaines. **mid-DeepLabv3+** est basé sur DeepLabv3+, avec un backbone ResNet simplifié et une couche de saut (*middle layer*) ajoutée dans le décodeur, ainsi que des couches mécanisme d'attention SimAM. Ce modèle atteint les performances des modèles CNN de référence tout en ayant la moitié de leur charge de calcul, ce qui permet un compromis optimal entre performance et efficacité de calcul. **ESeNet-D** est composé de deux branches d'encodeurs utilisant EfficientNetV2 comme backbone, avec un bloc de fusion pour l'intégration multi-échelle et un décodeur employant des convolutions auto-calibrée et interpolations entrainées pour une segmentation précise. ESeNet-D surpasse de nombreux modèles de référence RGB et RGB-D tout en ayant une charge computationnelle plus faible. Nos expériences ont montré que, lorsqu'elles sont correctement intégrées, les informations relatives à la profondeur peuvent améliorer de manière significative la précision de la segmentation des images alimentaires. Nous présentons également deux nouvelles bases de données : **AfricaFoodSeg** pour la segmentation *aliment/non-aliment* avec 3067 images (2525 pour l'entraînement, 542 pour la validation), et CamerFood, axée sur la cuisine camerounaise. Les ensembles de données CamerFood comprennent **CamerFood10** avec 1422 images et dix classes alimentaires, et **CamerFood15**, une version améliorée avec 15 classes alimentaires, 1684 images d'entraînement et 514 images de validation. Enfin, nous abordons le défi des données de profondeur rares dans la segmentation RGB-D des aliments en démontrant que les modèles MDE peuvent aider à générer des cartes de profondeur efficaces pour les ensembles de données RGB-D.

Mots-clés : *Segmentation Sémantique, Image Alimentaire, Image RGB-D, CNNs, Apprentissage profond*

Acknowledgements

As I write these acknowledgements, I find myself reflecting on an extraordinary journey that has spanned more than three years. While the path to completing this PhD has been challenging and often demanding, it has ultimately proven to be an incredibly enriching experience. This academic voyage would not have been possible without the remarkable individuals who have supported me throughout, and to them, I dedicate these words of gratitude.

I am profoundly indebted to my supervisory team : Professor **Lotfi CHAARI** in France, and Professors **Eugene SOBNGWI** and **Pierre ELE** in Cameroon. Your mentorship has been transformative, extending far beyond academic guidance. Throughout these years, you have consistently provided invaluable insights, encouraged innovative thinking, and demonstrated unfailing support for my research work. Your commitment to my success, has significantly shaped both my academic capabilities and personal growth. I would also like to thank the entire team of the RSD Institute research group in Cameroon, where the general idea for my thesis project was born.

My sincere thanks to the distinguished members of my thesis committee : Professors **Sylvie CHAMBON**, **Mounir KAANICHE**, **Thomas Florent KANAA**, and **Zagrouba EZZEDDINE**. Your willingness to evaluate my research work and provide thoughtful, constructive feedback has greatly enhanced the quality of my thesis. I am particularly grateful to my thesis reviewers, Professors Mounir KAANICHE and Thomas Florent KANAA, for their meticulous review and insightful comments that have substantially improved my manuscript.

I extend my sincere gratitude to Professor Boris TEABE, whose expertise and invaluable guidance enabled me to confidently adapt to this new environment. Your insightful advice and support have been invaluable to my academic journey. I am also particularly thankful to WAFAE Labriji, my office colleague during my final year of

doctoral studies. Your daily presence brought warmth and joy to the workplace, and I deeply value the friendship we have built. Our conversations and shared experiences made the challenging days of research much more enjoyable.

The IRIT - ENSEEIHT Laboratory became my second home thanks to the warm welcome and continuous administrative support from Vanessa ADJEROUD and Sandrine ANTUNES. Your assistance in managing the countless administrative aspects of my PhD journey has been invaluable. To all my laboratory colleagues, both named and unnamed in these acknowledgements, thank you for creating such a supportive and stimulating research environment. A special mention goes to Mohamed FAKHFAKH and Armel JEATSA TOULEPI, who embarked on this doctoral journey alongside me.

Beyond the academic sphere, I have been blessed with an incredible support network of friends. While I am grateful to everyone who has encouraged me along the way, I must particularly acknowledge Jafercine POKAM, Serifatu LATIFU, Cédric NGUEGANG, and my entire circle of friends in Toulouse. Your constant support, contagious humour and unwavering friendship were essential in helping me tackle the challenges of this journey.

Words cannot fully express my gratitude to Nicolas MBELE NDZANA - my classmate, friend, and brother. You have been my pillar of support throughout this journey, especially in handling all the university procedures in Cameroon. Your help has been truly immeasurable, and I am forever grateful for your dedication and friendship.

Above all, I am profoundly grateful to my family, whose love and support have carried me through this journey. Your constant encouragement and understanding have been my greatest source of strength. To my parents especially : your boundless faith in me and your tireless support have made this achievement possible. Your sacrifices and dedication have made this milestone possible, and my gratitude runs deeper than words can express.

This thesis represents not just my individual effort, but the collective support, guidance, and encouragement of all these wonderful people who have been part of my journey.

Table of Contents

List of Figures

List of Tables

Acronyms

VBDA	Vision-Based Dietary Assessment
DL	Deep Learning
CNN	Convolution Neural Network
ReLU	Rectified Linear Unit
WHO	World Health Organization
NCDs	Non Communicable Diseases
MDE	Monocular Depth Estimation
SVM	Support Vector Machine
FLOPs	Floating-point Operations
PA	Overall Pixel Accuracy
mIoU	Mean Intersection-over-Union

CHAPTER 1

Introduction

1.1 Context and Motivations

According to the World Health Organization (WHO) [10], malnutrition refers to deficiencies, excesses, or imbalances in a person's intake of energy and/or nutrients. The term malnutrition addresses three broad groups of conditions : **undernutrition**, which includes wasting (low weight-for-height), stunting (low height-for-age) and underweight (low weight-for-age) ; **micronutrient-related malnutrition**, which includes micronutrient deficiencies (a lack of important vitamins and minerals) or micronutrient excess ; and **overweight, obesity and diet-related noncommunicable diseases** (such as heart disease, stroke, diabetes and some cancers). Malnutrition, in one form or another, affects every country worldwide, making its eradication one of the most significant global health challenges [10]. In 2022, 2.5 billion adults were overweight, including 890 million who

were living with obesity, while 390 million were underweight [11]. Malnutrition has a tremendous negative impact on the normal functioning of every organ system [12]. It is the main risk factor of a number of Non Communicable Diseases (NCDs) such as cardiovascular diseases, diabetes and some cancers[13, 14]. As stated in a report from the WHO [15], in 2022, NCDs were responsible for 41 million deaths, accounting for 74% of all global deaths, with 40% of these occurring prematurely before the age of 70. NCDs epidemic not only has devastating health consequences for individuals, families, and communities, but also poses a significant burden on healthcare systems worldwide. This burden makes their prevention and control a crucial priority for the 21st century [13]. Moreover, the triple burden of malnutrition, which is the coexistence of overnutrition, undernutrition and micronutrient deficiencies, is increasing in low-income and middle-income countries (LMICs) [16].

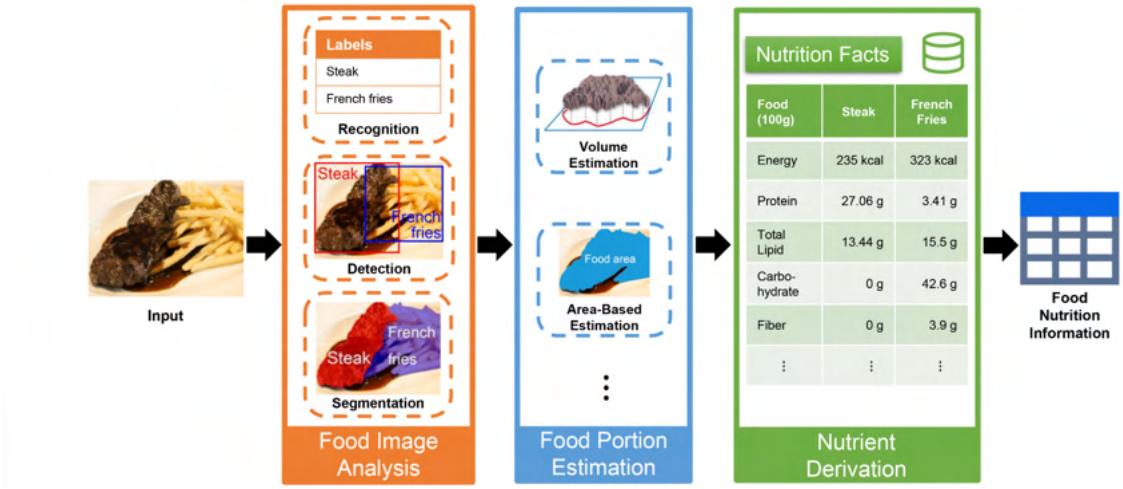
The etiology of malnutrition is diverse and complex, with dietary intake, dietary quality, and dietary habits playing major roles [17]. Therefore, accurate assessment of habitual food and nutrient intake has become essential for individuals willing to follow a healthy habits and life style [18]. However, accurately estimating dietary intake remains challenging. To leverage this challenge, dietary assessment has become the focus of widespread attention in various fields of computer vision, medicine and public health [19, 20, 21, 22, 23, 24]. Over the years, researchers have explored various methods for dietary assessment [25], such as the food frequency questionnaire (FFQ), dietary record (DR), and 24-hours-dietary recall (24-HDR). Among them, FFQ can be regarded as a long-term diet evaluation method, while 24-HDR and food records are the primary subjective approaches used in short-term evaluations. The implementation of these methods mainly involves self-reported information or structured interviews conducted under the supervision of dietitians. These traditional manual recording methods are labor-intensive, expensive [26], time-consuming, often do not result in accurate assessment of nutritional intake due to their dependence on self-report (e.g., energy intake under-reporting [27, 28]), and high-demanding for a certain level of literacy and communication skills [29]. Specifically, these methods are highly subjective in nature and require respondents to recall all foods and drinks consumed on a previous day for 24HR or (most often), leading to recall bias and erroneous conclusions [26]. Furthermore, when keeping food records (weighted or estimated), individuals can react to the assessment by underrating or underrecording [28, 30] food intake and therefore the records might not reflect habitual intake. As a result, traditional methods make the dietary assessment process more difficult, if not impossible, for individuals of a certain age, such as children, teenagers and elderly. Traditional methods often demand significant human resources

and are susceptible to subjective bias and human error, limiting their applicability in large-scale epidemiological studies. These methods therefore make food data collection unreliable and difficult to assess on a large scale.

However, recent advancements in artificial intelligence (AI) have significantly reshaped the methodologies employed by dietitians for assessing dietary intake and influenced how the general public manages their dietary habits. AI techniques open new possibilities for automatic dietary assessment by directly analyzing food images [19]. The widespread use of portable devices with camera (smartphones, wearable devices,...) with enhanced capabilities together with the advancements in computer vision enabled the development of **VBDA**. This consists of taking an image of a meal as input and automatically process relevant dietary information as an output [23]. Compared with traditional methods, VBDA can provide a solution to eliminate subjectivity, get rid of time and space constraints, and enhance the comprehensiveness and accuracy of dietary intake assessment. It can not only reduce the burden of keeping food intake journal, but also provide immediate dietary assessments, demonstrating great potentials in effective diet monitoring and control[19]. For these reasons, many research in the field of computer vision has shown great interest on VBDA systems [23, 19, 31, 32, 33, 34, 35, 36]. They utilize computer vision models to directly identify food items categories, evaluate their volume and estimate nutrient content from camera pictures.

VBDA systems typically involve three stages as illustrated in Fig. 1.1 : **food image analysis**, **portion estimation**, and **nutrient derivation** [19]. **Food image analysis** entails segmenting food regions from the background and recognizing each type of food item present in the image. This stage partitions food image into multiple food items at the pixel level, assigning a food label to each pixel. Food image analysis can be resumed to the semantic segmentation of the food image. Such precise localization of food areas is important for the subsequent estimation of portion sizes. **Food portion estimation** entails evaluating the volume or weight of each recognized food item. In the final **nutrient derivation** stage, converting food portions into meaningful nutritional informations relies on referring to a food nutrient database (e.g., USDA database [37], FAO/INFOODS database for West Africa [38]). The performance of the first two stages heavily relies on the effectiveness of artificial intelligence algorithms and the availability of relevant food datasets, while the final stage depends on a nutritional composition database.

Food image semantic segmentation is a fundamental task for VBDA system. This task involves assigning a label to each pixel in the food image, effectively segmenting the image into different food items or categories. However food semantic segmentation is challenging due to various factors [33]. One of the primary challenges is the **non-**



Source : Wei Wang et al. [19]

Fig. 1.1 VBDA system multi-stage architecture.

rigid structure of food, which differs from common objects. This characteristic makes it difficult to utilize shape as a reliable feature for machine learning models. Additionally, foods usually have high **intra-class variation**, meaning that the visual characteristics of the same food can differ significantly from one cook to another. The same food may have different morphological and color characteristics, which increases the difficulty of network learning. This variation is particularly pronounced in African foods, further complicating accurate food recognition. Furthermore, **inter-class resemblance** is another source of potential recognition issues, as different food items can appear very similar, as illustrated in Fig. 1.2. Some examples of generic food with such resemblances include brownie and chocolate cake, margarine and butter, peach and nectarine fruits. Moreover, certain dishes may contain various ingredients, resulting in the same dish with distinct visual aspects. Another significant challenge in food image semantic segmentation is the **scarcity of publicly available datasets** which hinders the development of accurate segmentation models. Additionally, overlapping and blending of foods on plates further hinder segmentation. Current research on food image segmentation and recognition focuses mainly on images of Asian and Western foods [33]. Unfortunately, there are only few number of publicly available datasets for food image segmentation, and they don't incorporate images of African foods. At best of our knowledge, at the time of this study there is no publicly available dataset for food image segmentation based on Africa food. However, African foods, including Cameroonian foods, are very different from asian/western foods and present their own unique challenges. African dishes often consist of multiple mixed classes of food, as depicted in Fig. 1.1. This complexity adds significant

difficulty when attempting to segment and recognize individual food items. The more food classes are mixed together on a plate, the more challenging it is to accurately detect the contours of each food component in the dish.

Fig. 1.2 African food images : some examples of different Camerounian meals with a very similar yellow texture.

Although there are several achievements in semantic segmentation, most of the studies only focus on RGB images. RGB information provides models with distinct color and texture but not sufficient geometric information. It is therefore difficult to distinguish instances and context sharing similar colors and textures [39]. In order to deal with such difficulties, researchers begin to use depth information to assist RGB semantic segmentation. The combination of RGB and depth information, called RGB-D, is interesting for several reasons mentioned in [39, 40]. For example, depth images are able to offer geometric information, and hence it is possible to enrich the representation of RGB images and to better distinguish various objects. In addition, depth images are less sensitive to environmental disturbances such as illumination, fog, etc. However, some studies [41, 40] demonstrated that directly applying the complementary depth information into existing RGB frameworks or simply ensemble results of two modalities may lead to inferior performance. Thus, researchers continuously come up with various methods in recent years in order to improve the efficiency of RGB-D semantic segmentation. While depth features have been extensively studied for segmenting indoor and outdoor scenes [39, 42, 43], their potential for food segmentation remains largely unexplored. Most existing research in food segmentation relies solely on RGB images [44, 33], primarily due

to the scarcity of public datasets containing RGB-D images and the challenge to collect huge amount of food depth images.

1.2 Key Contributions

Following the context and issues developed above, the objective of this thesis is to proposed new efficient deep learning models for RGB and RGB-D image segmentation in application to African food images. The main contributions of this thesis are described below.

- We propose a novel segmentation model called **mid-DeepLabv3+** inspired by the well-known semantic segmentation architecture DeepLabv3+ [45], with three key modifications. Firstly, our backbone is a reduced version of ResNet[46] in which we have excluded the 5th convolutionnal stage. This modification reduces the number of parameters of the model but also affect the performance. To recover the loss of performance we introduced secondly an additional skip layer in the decoder path and thirdly a SimAM [47] attention mechanism. The new skip layer which we called *middle layer*, reintroduces more general extracted features that have potentially been lost in the encoder's path. These enhancements are designed to improve the model's segmentation performance while significantly reducing its complexity. Our experimental results demonstrate that mid-DeepLabv3+ is an optimal choice, providing an excellent balance between performance and computational efficiency.
- We introduced a new efficient RGB-D segmentation method called **ESeNet-D**, that outperforms several benchmark RGB and RGB-D models while keeping a relatively small computational load. ESeNet-D comprises an encoder with two branches RGB and depth data, using EfficientNetV2 [8] as backbone. An efficient fusion block is used to merge extracted RGB and depth features extracted at three different scales. For our model, we have built a simple but efficient decoder that takes advantage of self-calibrated convolution and learned interpolation to process the outputs of the fusion blocks in order to obtain the final segmentation mask. We also release **ESeNet**, the RGB version of our model which outperform many popular RGB segmentation models while having lower size, parameters and floating-point operations. Our experiments show that, when properly integrated, depth information can significantly improves food segmentation accuracy.
- Public food segmentation datasets are rare, and the construction of a new dataset remains a tedious task, but one that enables us to advance research in the field. We

present the firsts dataset for food image segmentation focusing on African cuisine : **AfricaFoodSeg** dataset for *food/non-food* segmentation and the **CamerFood** dataset for multiclass semantic segmentation. CamerFood dataset includes images of the most commonly consumed Cameroonian dishes[48]. AfricaFoodSeg has two classes (Food and non-Food) and contain 3067 images divide into 2525 for training class and 542 for validation. In our work, we developed two versions of the CamerFood dataset : **CamerFood10** and **CamerFood15**. CamerFood10 consists of images from ten food classes, with a total of 1,422 images—1,032 in the training set and 209 in the validation set. CamerFood15 is an enhanced version of CamerFood10, featuring 15 food classes, 1,684 training images, and 514 validation images. In CamerFood15, we increased the number of images per class and improved the annotations compared to its predecessor. Lastly, the AfricaFoodSeg, CamerFood10, and CamerFood15 datasets are publicly available.

- One of the most challenging tasks in RGB-D food image semantic segmentation, is depth data collection [39, 33] : RGB-D datasets for food segmentation are rare. In this work, we demonstrate that MDE models can facilitate the generation of RGB-D datasets for food segmentation. In addition to making evidence that depth maps are useful for semantic segmentation, we also investigate how depth maps quality may impact segmentation performance.
- All datasets and code utilized in this study are publicly accessible at the following link : https://github.com/babanthierry94/ESeNet-D .

1.3 Organization of the Manuscript

This thesis consists of six chapters including introduction, technical background, literature review, first contribution, second contribution and conclusion.

Chapter 1 is the introduction, it provides a concise overview of the context and challenges that led to this research. We outline the aims and objectives of the thesis, concluding with a summary of the key contributions made and an overview of the manuscript's organization.

Chapter 2 covers the foundational knowledge required for image semantic segmentation. It introduces core deep learning concepts and the basic components of Convolution Neural Network (CNN), highlighting state-of-the-art solutions for each

component, along with their advantages and drawbacks. This technical background sets the stage for a better understanding of our contributions.

Chapter 3 offers a literature review focused on food image segmentation. It begins with an overview of existing food image segmentation datasets and an analysis of various dataset construction processes in the literature. We then discuss the three majors categories of food image segmentation approaches, including traditional machine learning with handcrafted features, semi-automatic approaches, and fully automatic deep learning techniques. The challenges, benefits, and limitations of each approach are discussed.

Chapter 4 and **5** detail our contributions, experimental evaluations, and result analyses. **Chapter 4** begins by explaining the construction of our **AfricaFoodSeg** and **CamerFood** datasets, followed by an in-depth presentation of the mid-DeepLabv3+ model and the rationale behind its architectural choices. **Chapter 5** introduces our second contribution, reviewing RGB-D image segmentation approaches and explaining the principles of MDE. We then provide a detailed presentation of our **ESeNet-D** model, emphasizing how it differs from existing state-of-the-art methods.

Chapter 6 concludes the thesis by summarizing the research and findings. It discusses the limitations of the current work and outlines potential future directions to address these challenges.

1.4 Publications

Baban A Erep Thierry Roland, and Lotfi Chaari. **"mid-DeepLabv3+ : A Novel Approach for Image Semantic Segmentation Applied to African Food Dietary Assessments."** Sensors 24.1 (2023) : 209. **[Published]**

Baban A Erep Thierry Roland, Chaari Lotfi, Sobngwi Eugene, Ele Pierre. (2024, September). **"ESENET-D : EFFICIENT SEMANTIC SEGMENTATION FOR RGB-DEPTH FOOD IMAGES."** In 2024 IEEE 34rd International Workshop on Machine Learning for Signal Processing (MLSP) (pp. x-x). **[Published]**

Baban A Erep Thierry Roland, Chaari Lotfi, Sobngwi Eugene, Ele Pierre. **"Self-Calibrated Convolutions Towards Multimodal RGB-Depth Food Image Semantic Segmentation."** IEEE Transactions on Pattern Analysis and Machine Intelligence., 2024. **[Submitted]**

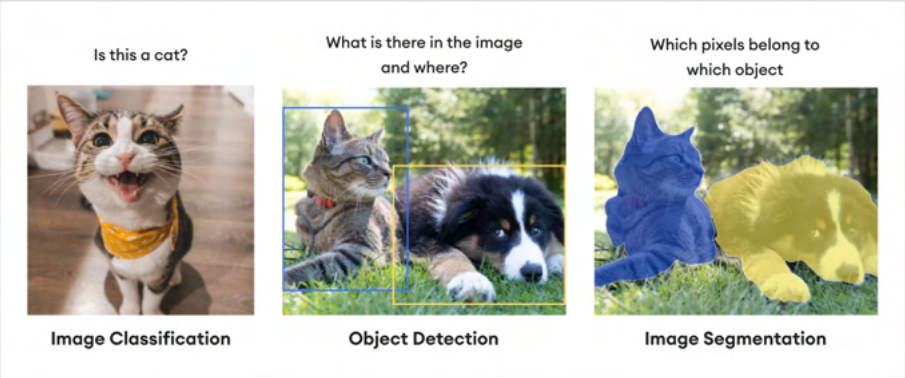
CHAPTER 2

Conceptual Basis

2.1 Introduction

Image segmentation is one of the main areas of computer vision and digital image processing that goes beyond simple image classification and object detection to provide a more granular understanding of image content. While image classification assigns a single

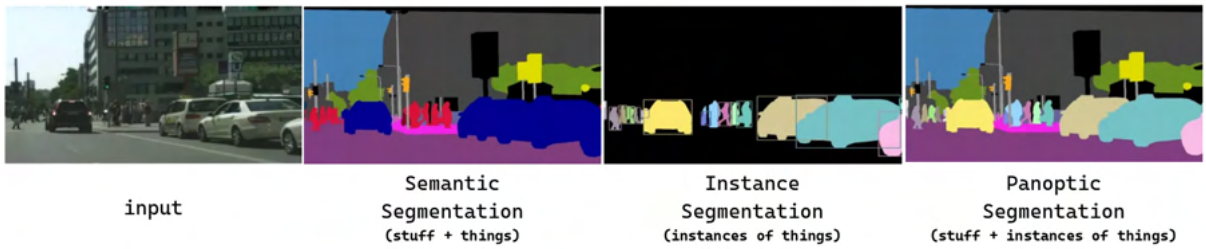
label to an entire image, and object detection identifies objects and their locations within images, image segmentation takes it a step further by partitioning images into distinct segments or regions at the pixel level (see. Fig. 2.1).



Source : https://www.superannotate.com

Fig. 2.1 Illustration of the output of image classification, object detection and image segmentation tasks.

Depending on the purpose, we distinguish three types of image segmentation within the domain of computer vision (see. Fig. 2.2) : semantic segmentation, instance segmentation, and panoptic segmentation. **Semantic segmentation** entails the classification of each pixel in an image into predefined semantic classes, offering a broad understanding of scene content devoid of individual object instances. **Instance segmentation** detects and delineates each object instance in an image, assigning a class and a unique ID to each instance. Therefore, it combines elements from object detection, which aims to classify and localize individual objects using bounding boxes, and semantic segmentation, which aims to classify each pixel into a fixed set of categories without distinguishing between different instances of the same class [49]. This type of segmentation is useful in critical applications where identifying individual objects is important, such as object tracking in videos or counting the number of specific items. **Panoptic segmentation** introduced by Kirillov et al. [50], combines both semantic and instance segmentation tasks in a unified framework. It aims to provide a segmented image where each pixel is assigned either a class label (for stuff categories) or both a class label and an instance ID (for thing categories), where *'stuff'* refers to amorphous regions like sky, grass, road (no distinct instances), and *'things'* refer to countable objects like people, cars, animals (distinct instances). Such an approach offers a holistic perspective, integrating semantic context with instance-level details, proving invaluable for tasks demanding comprehensive scene understanding, such as scene parsing and autonomous navigation systems.



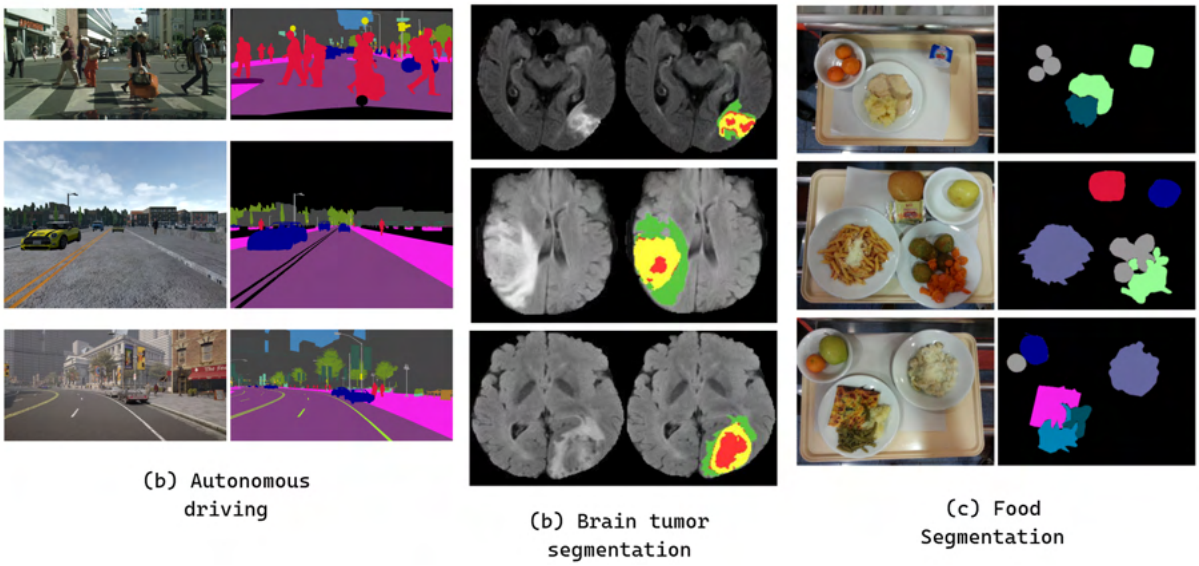
Source : https://www.labellerr.com

Fig. 2.2 Output of the different type of image segmentation : semantic, instance and panoptic.

Semantic segmentation finds wide-ranging applications across diverse domains where the distinction between individual objects is not necessary. Some example of semantic segmentation application are present in Fig. 2.3. In the context of autonomous driving, where it is used for recognizing road, vehicles and pedestrians to assist in vehicle navigation and obstacle detection [51]. In medical imaging, semantic segmentation facilitates the precise identification of organs, tissues, and abnormalities from medical images, assisting clinicians in diagnosis, surgical interventions, and treatment planning [52]. Detection of melanoma [53], tumors [54], glaucoma [55], lung infection [56] are some of real-world application of semantic segmentation in medical domain. Semantic segmentation is also utilized in satellite and aerial imagery analysis [57] for land cover classification, urban planning, and environmental monitoring. In VBDA System [58, 19], semantic segmentation is a critical task for identifying and delimiting each food item present in an image. Portion estimation is carried out only after the segmentation stage have identified each food item. Overall, semantic segmentation serves as a fundamental tool in various applications, contributing to advancements in fields such as transportation, healthcare, agriculture, and environmental monitoring.

2.2 Unsupervised, Supervised and Weakly-Supervised Learning

The principle of semantic image segmentation involves classifying each pixel in a given image into a specific category. The performance of existing approaches is influenced by the used models and the quality of the employed databases for training. Popular models like Fully Convolutional Network (FCN)[60], U-Net[45] and DeepLab[61] trained to classify each pixel in an image into predefined categories using labeled data. This



Source : (a) Ivanovs et al.[51], (b) Zhang et al.[54], (c) Sharma et al.[59]

Fig. 2.3 Few sample of image semantic segmentation application.

approach, named **supervised learning segmentation**, involves annotated datasets, where each image has corresponding labels for every pixel. The model learns to map pixels to their respective classes by minimizing a loss function that captures the difference between its predictions and the ground truth during training. While supervised learning offers high accuracy and detailed image comprehension, it requires extensive labeled datasets. The effectiveness of supervised learning heavily depends on the quality and quantity of labeled data, often requiring significant human effort for annotation. However, the collection of large-scale annotations is labor-intensive and it is difficult to obtain. **Unsupervised Semantic Segmentation** involves partitioning an image into semantically meaningful regions or categories without relying on labeled training data. Unlike supervised methods that depend on extensive annotated datasets, unsupervised approaches exploit the inherent structures and patterns in the data, such as visual features, texture, color, or spatial relationships, to cluster similar pixels into segments. Common techniques include clustering, self-supervised feature learning, and contrastive learning to derive meaningful representations. This approach is particularly valuable in domains like medical imaging and remote sensing, where obtaining labeled data can be costly or impractical [62]. However, unsupervised semantic segmentation faces several challenges that impact its practicality. A significant limitation is the lack of interpretability, as the results often fail to align with human-defined semantic categories, making validation and interpretation difficult. Additionally, without ground truth labels, evaluating the segmentation quality

objectively poses a challenge. Finally, in complex scenarios, unsupervised methods generally underperform compared to their supervised counterparts. **Weakly supervised learning segmentation** aims to exploit strength of supervised and unsupervised methods. It aims to perform image segmentation using limited or imprecise labels. Unlike fully supervised learning, which relies on detailed pixel-level annotations, weakly supervised learning utilizes more readily available and less expensive forms of supervision, such as image-level labels indicating the presence or absence of certain objects without specifying their exact locations, bounding boxes providing rough localization of objects but not detailed pixel-wise boundaries, scribbles consisting of simple lines or strokes indicating approximate regions of interest, and individual points marking the presence of objects without detailed contours. The advantages of this approach include reduced annotation effort, significantly lowering the cost and time required for data annotation ; scalability, making it feasible to use larger datasets since detailed annotations are not necessary ; and flexibility, allowing the use of various forms of supervision that are easier to obtain. Weakly supervised semantic segmentation has several notable limitations. A significant drawback is that it often involves complex training frameworks, requiring advanced techniques to bridge the gap between weak supervision and accurate pixel-level predictions. These frameworks typically depend on multi-stage pipelines or intricate regularization strategies, increasing the overall implementation complexity. Moreover, as with unsupervised approaches, the lack of high-quality ground truth annotations poses a challenge in reliably evaluating the quality of semantic segmentation predictions.

Finally, while supervised segmentation faces the challenge of requiring large amounts of annotated data, it remains the most effective approach for achieving accurate, human-interpretable segmentation. In contrast, unsupervised and weakly supervised methods generally exhibit lower accuracy, as the absence of precise, pixel-level annotations provides less guidance for the model, resulting in performance gaps.

2.3 Semantic Image Segmentation Approaches

Semantic segmentation methods can be categorized into three main groups : traditional automatic methods, deep learning-based automatic methods, and interactive methods. **Traditional automatic methods** employ unsupervised techniques to extract regions of interest and then use machine learning classification methods like Gradient Boosting, Random Forests, or Support Vector Machines, leveraging hand-crafted features such as color, texture, and shape. Informative image descriptors are obtained from manual feature extraction methods such as SIFT, Gabor Filters, and HOG. The use of handcrafted feature

is the principal characteristic of traditional methods. These traditional methods still offer advantages when processing small-sample data or images with minimal noise. However, their performance is often limited by the expressive power of hand-crafted features. With the advancement of deep learning techniques, **deep learning-based automatic methods** have gradually become the mainstream. These methods automatically learn image feature representations, eliminating the need for manually designing feature extractors. Deep learning models significantly enhance segmentation accuracy and robustness by leveraging large datasets and powerful computational resources. They can capture complex patterns in images, leading to superior performance compared to traditional methods. **Interactive methods**, on the other hand, incorporate user input to guide and refine the segmentation process, combining the strengths of both automatic and manual approaches to achieve high accuracy with reduced annotation effort. In application such as dietary assessment, interactive segmentation methods could face significant challenges. The primary issue is the user burden, as these methods require manual input, such as clicks or strokes, to refine segmentation, making the process time-consuming and user-dependent. Real-time performance is another concern, as interactive methods demand quick feedback to be practical, but high computational requirements can slow down the process, resulting in latency and user frustration. Usability is also a challenge, as non-expert users may struggle with the tool's complexity, leading to a steep learning curve. For these reasons, deep learning-based automatic approaches remain the mainstream choice in many application domains.

According to the network architecture, deep learning models for image semantic segmentation can be broadly categorized into four main families [63] : CNN-based, transformer-based, MLP-based, and others hybrid architectures. **CNN-based models** have been foundational in the field, setting benchmarks for high accuracy by effectively capturing spatial hierarchies in images. In the field of Deep Learning (DL), the CNN is one of the most famous and commonly employed architectures [64]. Through multiple layers of feature transformation, the underlying feature representation of the original data is gradually transformed into a higher-level feature representation, and the processed data is fed into a prediction function to settle the final output of purpose task [65].

With the success of **Transformer-based models** in natural language processing (NLP) [66], many researchers have begun exploring their use as stand-alone architectures for vision tasks. In NLP, transformers operate on a group of tokens, which can be words, subwords, or groups of words. However, since images lack natural tokens similar to words, in computer vision, tokens are created by dividing an image into a sequence of **patches**, typically of fixed size [67]. Moreover, transformers in computer vision leverage

the **self-attention** mechanism, enabling the model to focus on different parts of the input image and capture long-range dependencies and contextual information that traditional convolutional neural networks (CNNs) might miss. This approach contrasts with CNNs, which rely on localized receptive fields and gradually aggregate information through stacked layers. However, transformers also come with challenges. They typically require a large amount of data and computational resources to train effectively, due to their vast number of parameters and the need for extensive computations during the self-attention process. **Multi-layer perceptron (MLP)** models in semantic segmentation represent a novel approach that leverages the simplicity and computational efficiency of multi-layer perceptrons (MLPs) to achieve high performance in image analysis tasks. Unlike traditional CNNs and transformer-based models, MLP-based architectures, such as the MLP-Mixer [68], utilize MLPs for both spatial and channel-wise information mixing. This involves dividing the input image into patches and using MLPs to process these patches by mixing information across spatial locations and feature channels. This approach reduces complexity while maintaining the ability to capture important image features. Despite their simplicity, MLP models have shown competitive performance in semantic segmentation, efficiently handling tasks like object boundary delineation and region classification. Their ability to provide strong results with fewer computational resources makes MLP models an attractive alternative for various applications in computer vision.

As the reported accuracy on image classification benchmarks continues to increase by new network designs from various camps, no conclusion can be made as which structure among CNN, Transformer, and MLP performs the best or is most suitable for vision tasks. This is partly due to the pursuit of high scores that leads to multifarious tricks and exhaustive parameter tuning [63, 69]. In this work, our study focuses on CNN-based models for fully supervised learning. Therefore, in the following sections, we will present the main components that make up the architecture of CNN models and those that influence learning performance.

2.4 Convolutional Neural Network (CNN)

CNNs are inspired by the neural mechanisms of the visual cortex whose architecture was inspired by the neurobiological experiments conducted on cat visual cortex cells in 1962 by Hubel and Wiesel[70]. More specifically, a CNN is build to simulate the sequence of cells which forms the visual cortex. Since their appearance, CNNs have been extensively applied in a range of different fields, including image classification, object detection, semantic segmentation, speech processing, medical image analysis, sentiment analysis,

etc. The main characteristics of CNN architectures are summarized below :

— **Sparse Connections** : in Fully Connected neural networks, each neuron of a layer is linked with all neurons in the following layer. By contrast, in CNNs, only a few weights are available between two adjacent layers. This sparse interaction makes CNNs computationally efficient compared to fully connected networks, especially for large input data like images.

— **Weight Sharing** : CNNs take advantage of data spatial correlation present in 2D input-data structures like images. Instead of learning separate parameters for each location, the same set of parameters (weights) is used across different parts of the input image. This parameter sharing significantly reduces the number of parameters, making CNNs more efficient and easier to train, especially when dealing with high-dimensional inputs like images.

— **Translation Invariance** : meaning CNNs can recognize patterns regardless to their location in the input image. This is achieved through the use of convolutional layers, which slide filters (also known as kernels) over the input to detect features irrespective of their position.

— **Feature Hierarchies** : CNNs are capable of automatically learning hierarchical representations of features. Lower layers tend to learn simple features like edges and textures, while deeper layers learn more complex features and eventually whole object representations.

Although there are numerous CNN models, their architecture essentially consists of the same basic components such as Convolution Layer, Pooling Layer, Activation Function, Batch Normalisation Layer and Fully connected layer.

2.4.1 Convolution layer

Convolutional layer is the fundamental building block of a CNN. It consists of a collection of convolutional filters. This layer computes a dot product between the filter value and the image pixel values. The output matrix called *feature map* is obtained by sliding the filter over the image. An example of convolution operation process is shown in Fig. 2.4. Filters (or Kernels) are a set matrices of learnable parameters. A Convolution layer is characterized by the following main hyperparameters :

— **filter or Kernel size** : It determines the sliding window's dimensions, which should be smaller than the image. Typically, smaller sizes like 1, 3, 5, and rarely 7 are used. Odd-sized kernels provide a clear center pixel for symmetric convolution, simplifying

padding and output size consistency. A kernel is described by a grid of values, each called a weight. Initially, random weights are assigned, which are adjusted during training to extract significant features. Various methods can initialize these weights.

— **Stride** : This parameter controls the number of pixels the kernel moves with each step of convolution.

— **Padding** : Padding adds zeros to the image borders, allowing the kernel to fully filter every position, ensuring proper processing of the edges.

— **Number of filters /Depth** : The number of filters in a convolutional layer dictates how many patterns or features the layer will identify, determining the distinct characteristics it focuses on.

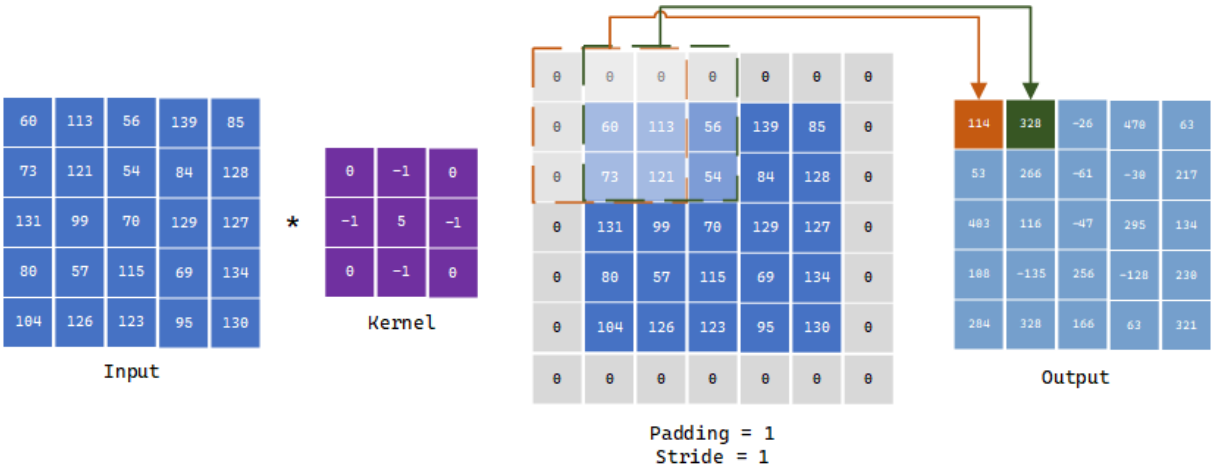


Fig. 2.4 An example of Convolution operation.

The output size of the convoluted layer is determined by several factors, including the input size $(H_{\text{in}}, W_{\text{in}}, C_{\text{in}})$, kernel size $(K_{\text{h}}, K_{\text{w}})$, number of filters N, stride S, and padding P. The formula to calculate the output size $(H_{\text{out}}, W_{\text{out}}, C_{\text{out}})$ is as follows :

$$(H_{\text{out}}, W_{\text{out}}, C_{\text{out}}) = \left(\left\lfloor \frac{H_{\text{in}} - K_{\text{h}} + 2P}{S} \right\rfloor + 1, \left\lfloor \frac{W_{\text{in}} - K_{\text{w}} + 2P}{S} \right\rfloor + 1, N \right) \tag{2.1}$$

The term $\lfloor \cdot \rfloor$ represents the floor function, which rounds down to the nearest integer.

2.4.2 Grouped Convolutions

Grouped convolutions is a technique in CNNs where input and output channels are divided into groups, and convolution operations applied independently within each group. This reduces parameters and computations significantly, making it ideal for

resource-constrained environments, unlike traditional convolution layers that process all channels and demand higher computational resources. Through modular filter groups, grouped convolutions facilitate model and data parallelism during training. Data parallelism involves splitting the dataset into chunks and training them sequentially, while model parallelism distributes the model across computational resources. By minimizing filter correlation within groups, grouped convolutions allow each group to learn unique representations. Fig. 2.5 illustrates the difference between standard and grouped convolutions.

To illustrate how grouped convolution reduce the number of parameters, let us consider a standard convolution with C_{in} input channels, C_{out} output channels, and a kernel size of $k \times k$. The number of parameters in a standard convolution is $C_{in} \times C_{out} \times k \times k$. In a grouped convolution with N groups, the number of parameters is reduced to $\frac{C_{in}}{N} \times \frac{C_{out}}{N} \times k \times k \times N$. However, when the number of parameters in the group convolution is reduced, the performance of a CNN using grouped convolution layers may decrease compared to a CNN using standard convolution layers [71, 72]. Here's a breakdown of how grouped convolution works :

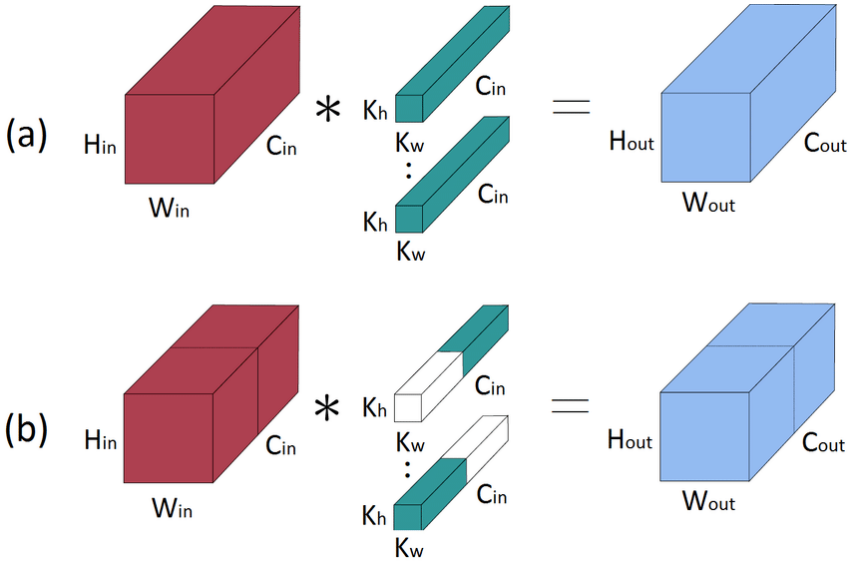
— **Division of Channels** : The input channels are divided into N groups. Each group is treated as a separate set of channels.

— **Independent Convolutions** : Convolutions are applied independently to each group. For instance, if the input has C_{in} channels and is divided into N groups, each group will have $\frac{C_{in}}{N}$ channels. Separate filters are applied to each of these groups.

— **Concatenation of Outputs** : The outputs of the group-wise convolutions are concatenated to form the final output.

Grouped convolution has been integrated into various state-of-the-art architectures, demonstrating its effectiveness in balancing computational efficiency with model accuracy. Techniques like grouped convolution play a crucial role in enabling the development of lightweight and efficient neural networks. The grouped convolution method was originally proposed in 2012 by AlexNet[74] to distribute the model on two GPUs for the lack of graphics card memory. Grouped convolution has been utilized in several influential neural network architectures, such as ResNext[75], ShuffleNet [76, 72], Deep Roots[77], Condensenet[78, 79, 80], Balanced Group Convolution [81], ResGANet [82]. On the basis of grouped convolution, some researchers have proposed depthwise convolution [83, 84, 85].

Depthwise convolution is a more extreme case of grouped convolution, which refers to



Source : Gibson et al.[73]

Fig. 2.5 Standard vs. Grouped Convolutions : (a) In standard convolution, each filter processes all input channels; (b) In grouped convolution with two groups (N=2), the input is split into two halves, and half of the filters are applied to each half of the input.

a grouped convolution scheme with the number of groups equal to the quantity of input feature maps.

2.4.3 Separable Convolution

Separable Convolution (SepConv) is a variant of standard convolution which splits the computation into two steps : depthwise convolution applies a filter to each input channel individually, and pointwise convolution combines these outputs linearly. By doing so, SepConv significantly reduces the number of parameters and computational cost while maintaining comparable performance to traditional convolution. This make it suitable for resource-constrained environments. SepConvis widely adopted in various deep learning models such as Xception, MobileNet, DeepLabv3+ [45], EAR-Net [86], and EfficientNet [8], where it replaces standard convolution to improve efficiency.

— **Depthwise Convolution** : applies an individual filter to each input channel, enabling independent capture of spatial information for each channel, unlike traditional convolution, which uses a single filter for the entire input feature map.

— **Pointwise Convolution** : uses a 1x1 kernel on the output of depthwise convolution to mix information across channels and create new features through linear combinations.

Considering a standard convolution with C_{in} input channels, C_{out} output channels and a kernel size of $k \times k$. The number of parameters is given by :

$$\boldsymbol{\mathcal{P}}_{Conv} = C_{in} \times C_{out} \times k \times k. \tag{2.2}$$

Separable convolution involves two stages : depthwise convolution and pointwise convolution. In depthwise convolution each input channel is convolved with its own set of $k \times k$ filters. The pointwise convolution is a 1×1 convolution that mixes the channels from the depthwise convolution to produce C_{out} output channels. The number of parameters is given by :

$$\mathcal{P}_{depthwise} = C_{in} \times 1 \times k \times k \tag{2.3}$$

$$\mathcal{P}_{pointwise} = C_{in} \times C_{out} \times 1 \times 1 \tag{2.4}$$

$$\boldsymbol{\mathcal{P}}_{SepConv} = (C_{in} \times k \times k) + (C_{in} \times C_{out}). \tag{2.5}$$

By using a separable convolution instead of standard convolution we reduction the computation load by

$$\textbf{Gain} = \frac{\mathcal{P}_{Conv}}{\mathcal{P}_{SepConv}} = \frac{C_{out} \times k^2}{k^2 + C_{out}}. \tag{2.6}$$

2.4.4 Dilated Convolution

Dilated convolution, also known as **Atrous Convolution**, introduced by Holschneider et al.[87], is a variant of standard convolution. It introduces a dilation factor to the convolution kernel, allowing it to sample input values with a specific stride across the input feature map. Unlike standard convolution, where the kernel slides with a fixed stride, dilated convolution inserts gaps between kernel elements, controlled by the dilation factor, enabling it to capture wider context without increasing the number of parameters. The dilation factor in dilated convolution increases the receptive field without adding parameters by inserting gaps between kernel values. An illustration of the dilated convolution operation is provided in Fig. 2.6. Dilated convolution expands the receptive field, captures multi-scale features, and reduces spatial resolution loss compared to larger filters in regular convolution. However, it can reduce spatial resolution in output feature maps, potentially impacting the network's ability to retain fine-grained spatial information and leading to higher computational costs compared to regular convolutions with equivalent filter size and stride. Dilated convolutions have been used successfully in various applications, such as semantic segmentation [87, 88], where a larger context is helpful.

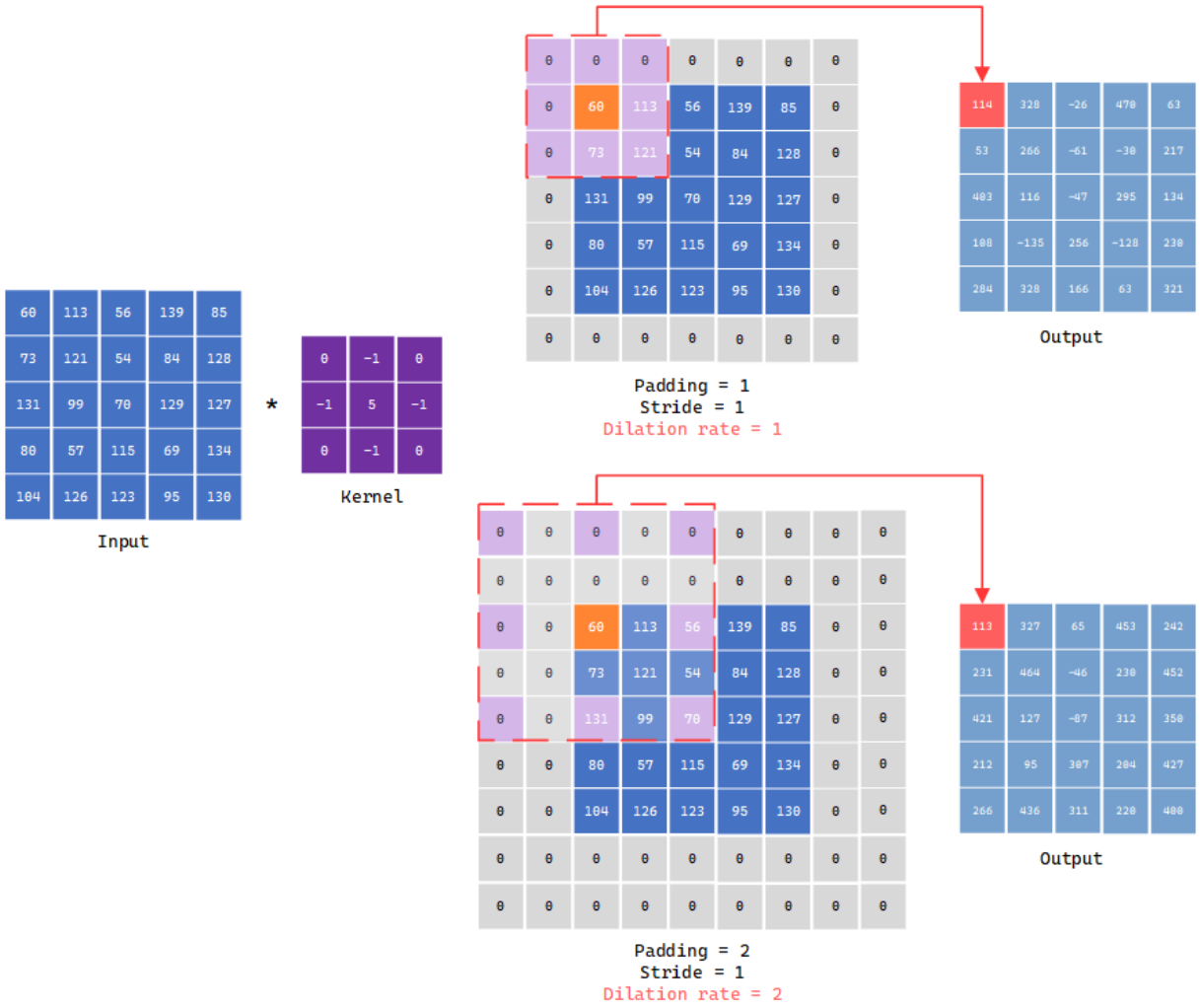


Fig. 2.6 Illustration of dilated convolution operation.

2.4.5 Pooling

Pooling in convolutional networks is designed to filter noisy activations by abstracting a receptive field into a single representative value. While this aids the CNN to retain robust activations in upper layers, it also results in the loss of spatial information, which may be crucial for tasks like semantic segmentation which need precise pixel-wise classification [89]. Pooling reduces the spatial dimensions (width and height) of the input feature maps by pooling over non-overlapping rectangular regions of size $(K_{\mathrm{h}}, K_{\mathrm{w}})$. The most popular pooling strategies are : max-pooling and average-pooling. **Max pooling** selects the maximum value from each local patch in the feature map, highlighting pronounced features like edges, while **average pooling** takes the mean value, resulting in smoother feature extraction. **Global Pooling** uses a slicing window size equal to the input feature

map size, producing a single pooled output value for each channel. This operation collapses the spatial dimensions into a single value per channel of the feature map. Global Pooling can be implemented using either max pooling or average pooling strategies. The Fig. 2.7 illustrates the principle of the various pooling operations we have mentioned. Pooling, unlike standard convolution with a stride greater than one, does not require learning parameters. Pooling is invariant to translation, meaning small translations of the input image do not significantly affect the values of the pooled feature map [90]. However, important details can be lost during pooling, which may be a potential drawback.

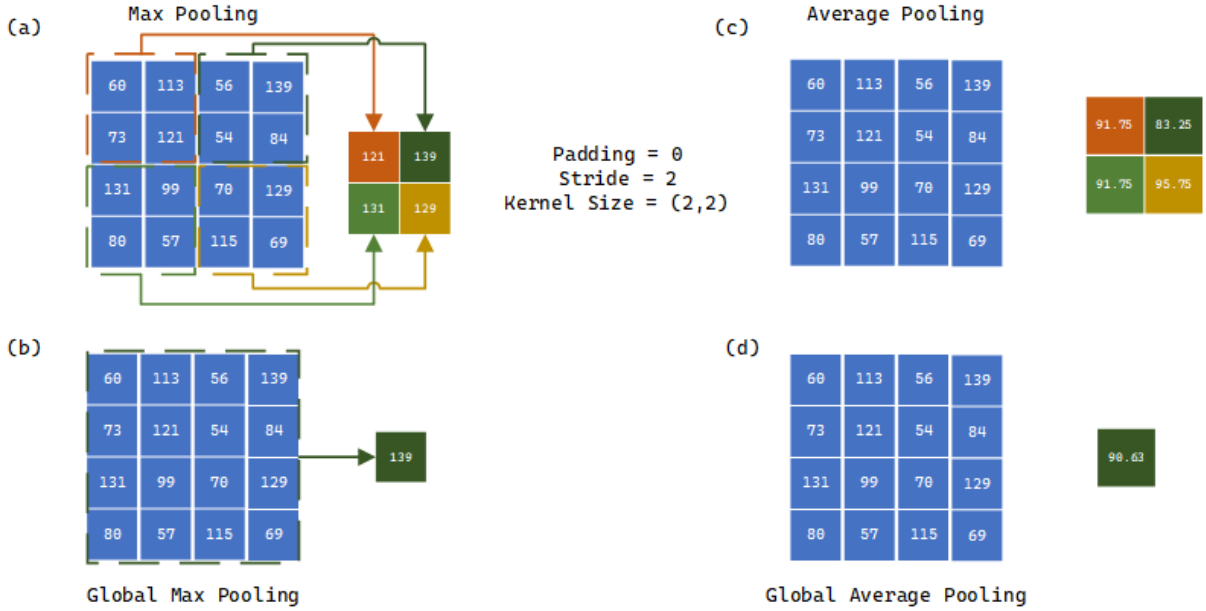


Fig. 2.7 Principle of different pooling operations.

The output size of the pooling operation is determined by several factors, including the input size $(H_{\text{in}}, W_{\text{in}}, C_{\text{in}})$, slicing window size $(K_{\text{h}}, K_{\text{w}})$, number of filters N, stride S, and padding P. The formula to calculate the output size $(H_{\text{out}}, W_{\text{out}}, C_{\text{out}})$ is as follows :

$$(H_{\text{out}}, W_{\text{out}}, C_{\text{out}}) = \left(\left\lfloor \frac{H_{\text{in}} - K_{\text{h}} + 2P}{S} \right\rfloor + 1, \left\lfloor \frac{W_{\text{in}} - K_{\text{w}} + 2P}{S} \right\rfloor + 1, C_{\text{in}} \right). \tag{2.7}$$

2.4.6 Upsampling Methods

While pooling layers reduce spatial dimensions and extract dominant features from feature maps, upsampling methods increase spatial resolution, thereby restoring or enhancing the resolution of feature maps. Various techniques can be employed for this purpose, each with its own advantages and applications. Here are some common methods :

Transpose convolution, also known as deconvolution, uses learned filters to upsample input feature maps, effectively reversing the dimensional reduction of traditional convolution. By distributing pixel values over a larger area using a learned kernel, it increases the dimensions of the feature maps. This technique is used in architectures like Fully Convolutional Network (FCN) and U-Net but can be computationally intensive and may introduce checkerboard artifacts.

2D interpolation is a classical technique used in image processing to increase image spatial dimension by estimating pixel values at new locations based on surrounding known pixels. The most common interpolation methods are bilinear, bicubic and nearest interpolation. **Bilinear interpolation** considers the closest 2x2 neighborhood of known pixel values and performs a linear interpolation to estimate the new pixel value, resulting in smooth but sometimes slightly blurred images. **Bicubic interpolation**, on the other hand, considers a larger 4x4 neighborhood and performs cubic interpolation, which typically produces sharper and more visually appealing results than bilinear interpolation. **Nearest neighbor interpolation** is the most simple method, it assigns the value of the nearest pixel, leading to a more pixelated and blocky image. When comparing these methods, bilinear interpolation is more computationally efficient than bicubic interpolation due to its reliance on fewer neighboring pixels and simpler calculations. However, bicubic interpolation, although more computationally intensive, provides superior image quality and is preferred in scenarios where visual fidelity is paramount. Nearest neighbor interpolation, being the simplest, has the lowest computational load but produces the least smooth results, often resulting in noticeable artifacts. While interpolation techniques are efficient and straightforward to implement, they do not add new information to the image, as they only estimate values based on existing pixels. This limitation means that while interpolation is useful for tasks like resizing images or filling in missing data, it may not always match the quality of results obtained through more complex methods like *learned upsampling* or *convolutional* techniques, which can leverage additional data throw learning.

Unpooling, also known as *inverse pooling*, is a technique used in CNNs to reverse the effects of pooling operations and restore the spatial dimensions of feature maps. Its principle is illustrated in Fig. 2.8. Unlike pooling, which reduces the resolution by summarizing local regions, unpooling aims to expand the feature maps back to their original size.The most common unpooling method is max unpooling. It restores the maximum values to their original locations based on indices from the max pooling step,

with the remaining positions filled with zeros or another value. This preserves the locations of important features but often produces sparse feature maps, necessitating additional convolutional layers to refine the output. Another drawback is the increased complexity due to the need to record the locations of maximum pixels during pooling. Unpooling have been used in the following papers [91, 89].

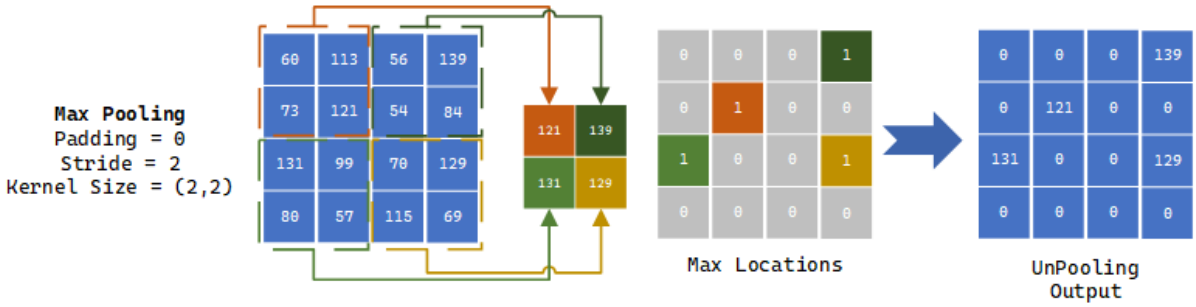


Fig. 2.8 Principle of unpooling operation.

2.4.7 Activation functions

Activation function introduces non-linearity into CNNs model, allowing it to learn and represent complex relationships in data that may not be linear[92]. Some desirable properties of activation functions include :

— The most fundamental property of an activation function is introducing non-linearity. Without it, each layer's output would just be a linear transformation of the previous layer's input, limiting the network's ability to model complex relationships [65].

— **Computational Expense** : Activation functions are used after every layer and need to be computed millions of times in deep networks, so they should be computationally inexpensive.

— **Differentiable** : Most optimization algorithms for training neural networks depend on derivatives to update weights during backpropagation, so activation functions should be differentiable, or at least almost everywhere differentiable, to facilitate the training process.

— **Monotonicity** : It help preserve the order of input values, potentially aiding convergence during optimization and enabling the network to learn meaningful representations. However, this property is not always essential, as shown by the success of non-monotonic functions like Rectified Linear Unit (ReLU)[93].

There are many activation functions proposed in the literature in the last three decades, some more computationally complex or with higher performance than others [92, 94]. Nevertheless, there are a few very popular ones like : Sigmoid, Tanh, ReLU, Swish and Softmax.

Sigmoid or *logistic function*, maps input values to the $(0,1)$ interval, normalizing neuron outputs. While it ensures smooth gradients and bounded outputs, it suffers from vanishing gradients, non-zero-centered output, and saturation for large inputs. Additionally, its exponential nature increases computation time.

$$\textbf{Sigmoid}(x) = \frac{1}{1+e^{-x}} \tag{2.8}$$

Hyperbolic tangent (Tanh) function maps outputs to the range $[-1,+1]$, making it effective for data with mean-centered features and capturing both positive and negative correlations. While it addresses the non-zero-centered issue of the Sigmoid function, it requires more computational time due to the need to compute two exponential functions.

$$\textbf{Tanh}(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{2.9}$$

Rectified Linear Unit (ReLU) [93] is a simple function which is the identity function for positive input and zero for negative input. Hence, the range of ReLU is $[0,+\infty[$. It has a gradient of one for positive inputs and zero for negative inputs. ReLU addresses the computational complexity of Sigmoid and Tanh functions and does not saturate for positive inputs, reducing the vanishing gradient problem. This makes ReLU easier to optimize, leading it to become the default activation function used across the deep learning community.

$$\textbf{ReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} \tag{2.10}$$

A common issue with ReLU is the *"dying ReLU"* problem, where neurons can become inactive and output zero consistently, hindering learning. To address this, variants like **Leaky-ReLU** [95], **Parametric ReLU (PReLU)** [96], and **Exponential Linear Unit (ELU)** [97] have been developed. These variants allow small, non-zero outputs for negative inputs. Leaky ReLU uses a fixed small slope for negative inputs, while PReLU uses a trainable slope. ELU adds robustness to noise by providing a negative saturation regime.

$$\textbf{Leaky-ReLU}(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0.01 \times x & \text{if } x < 0 \end{cases} \tag{2.11}$$

$$\textbf{PReLU}(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha x & \text{if } x < 0 \end{cases} \tag{2.12}$$

$$\textbf{ELU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{if } x \leq 0 \end{cases} \tag{2.13}$$

Swish [98] is an activation function introduced by Google in 2017. It performs slightly better than ReLU due to its smoother transition at $x = 0$, which aids convergence during training. However, Swish is computationally more computationally expensive. To address this, **HSwish** [99] was introduced as a more efficient variant.

$$\textbf{Swish}(x) = \frac{x}{1 + e^{-\alpha x}} \tag{2.14}$$

$$\textbf{HSwish}(x) = x \cdot \frac{\min(\max(0, x+3), 6)}{6} \tag{2.15}$$

Despite these new proposed activation functions, the standard ReLU remains a popular choice as an activation function in many neural network architectures [94]. Many practitioners have favored the simplicity and reliability of ReLU because the performance improvements of the other activation functions tend to be inconsistent across different models and datasets.

Softmax is an activation function used for multi-class classification problems. It converts a real vector of length K into another vector of the same length, with values in the range $[0,1]$ that sum to 1. This transformation represents the predicted probabilities for each class, with larger values indicating higher probabilities. The Softmax function is defined as follows :

$$\textbf{Softmax}(x) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \tag{2.16}$$

2.5 Underfit and Overfit

Machine learning has achieved significant success in practical applications, but it often encounters challenges such as overfitting and underfitting. Concepts like variance and bias help to understand theses issues, and regularization techniques provide methods to address them.

2.5.1 Variance and Bias

Variance in machine learning refers to a model's sensitivity to fluctuations in the training data. A model with ***high variance*** fits the training data too closely, including its noise and outliers, leading to high performance on the training set but poor generalization to new data. This phenomenon, known as **overfitting**, happens when the model is too complex to learn for the amount of data, capturing details that do not generalize beyond the training set. High variance is marked by a large gap between low training error and high validation error. To address this, techniques such as simplifying the model, reducing features, applying regularization, using cross-validation, and employing ensemble methods (e.g., bagging or boosting) can be effective. These strategies help create a more robust model that captures essential patterns and improves generalization to new data.

Bias in machine learning refers to the error from using a simplified model to approximate a complex problem. A model with ***high bias*** makes strong assumptions about the data, often resulting in **underfitting** and high errors on both training and validation datasets. High bias is linked to overly simplistic models, such as using a linear model for a non-linear problem. To reduce high bias, one can increase model complexity, add relevant features, reduce regularization, and improve data quality.

The goal machine learning is to achieve a balance between bias and variance, ensuring the model is neither too simple to underfit nor too complex to overfit, for optimal performance.

2.5.2 Regularization techniques

Regularization techniques are essential in machine learning for mitigating overfitting, enhancing generalization, and improving performance on unseen data. They introduce constraints or penalties to prevent models from becoming overly complex and fitting noise in the training data. Several regularization techniques exists such as normalization, early stopping, dropout, and data augmentation. When applied effectively, these methods help

to balance between model complexity and generalization, ensuring effective performance in diverse applications.

Batch Normalization (BN) [100] is a technique used in neural networks to enhance training stability and accelerate convergence. It normalizes each layer's inputs to have zero mean and unit variance, typically applied over mini-batches during training. By reducing internal covariate shift, which refers to variations in network activations due to weights updates, BN allows layers to learn more independently and enhances overall network performance. BN mitigates vanishing and exploding gradient issues by consistently normalizing weights to zero-mean and unit standard deviation. This *"safety precaution"* allows training with large learning rates, as weights cannot grow uncontrollably since their means and variances are normalized [101]. Overall, batch normalization enhances model performance, speeds up convergence, and improves generalization to new data

$$\mathbf{BN}(x) = \gamma \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta, \tag{2.17}$$

Where: x = is the input to the batch normalization layer
μ = is the mean of the mini-batch
σ^2 = is the variance of the mini-batch
γ = is the scale parameter (learnable)
β = is the shift parameter (learnable)
ϵ = is a small constant added to the denominator for numerical stability.

σ, μ are non-learnable parameters which are saved as part of the *'state'* of the Batch Norm layer.

Dropout is a regularization technique used in neural networks to prevent overfitting and improve generalization. During training, dropout randomly deactivates a proportion of neurons in the network for each forward and backward pass, forcing the network to learn redundant representations. This process makes the network more robust and less sensitive to specific weights or neurons, reducing the risk of overfitting to the training data. Dropout effectively simulates training a large ensemble of models by varying architectures at each iteration, enhancing the network's generalization ability [102]. However, during inference or evaluation, dropout is turned off, and the full network is used to make predictions.

Early Stopping is a technique used to prevent overfitting by halting the training process when the model's performance on a validation set starts to degrade. During training, a

model might continue to improve its accuracy on the training data while its performance on the validation data worsens, indicating overfitting. By monitoring validation loss or accuracy, early stopping detects this point and stops the training before overfitting occurs. This ensures the model retains its ability to generalize to new, unseen data. Early stopping is particularly useful in training deep neural networks, where prolonged training can lead to significant overfitting. It also saves computational resources by avoiding unnecessary training epochs once optimal performance is reached.

Cross-validation involves dividing data into multiple subsets, or "folds", and repeatedly training and validating the model, each time using a different fold as the validation set and the rest as the training set. A common method is *k-fold cross-validation*, where data is split into k folds, and the model is trained and validated k times, with each fold used once as the validation set. The results are averaged to provide a more reliable performance estimate. This method mitigates overfitting and selection bias, ensuring each data item is used for validation. Cross-validation maximizes data use, aiding in hyperparameter tuning and model selection, especially with limited data.

Transfer Learning [103, 104] inspired by human learning, involves applying knowledge from one problem to another. It reduces training time and resources, enhances model performance, and addresses limited data issues. This approach includes three main strategies tailored to different tasks and data availability :

— *Feature Extraction*, uses a pre-trained model to extract relevant features from new data, applying the learned representations to identify important patterns in the related dataset. This method, often applied in image segmentation with encoder-decoder architectures, accelerates training and improves performance, especially with limited datasets or constrained resources. Pre-trained models like ResNet or VGG16, typically trained on large datasets such as ImageNet, are commonly used in the encoder path for this purpose.

— *Fine-tuning* involves unfreezing some or all of the layers of the pre-trained model and retraining them on the new dataset. This allows the model to adapt the pre-learned features to a specific new task. This is specially useful with small or specialized datasets.

— *Multi-task Learning* consist in training a model on multiple related tasks simultaneously, allowing it to share knowledge across tasks and enhance performance on each one.

2.6 Data Augmentation

Progress in the field of AI is driven by the availability of large amounts of training data. However, in many areas, such as medical image analysis or food analysis, it is difficult to have such large amounts of data. [105]. Data Augmentation, a data-space solution to the problem of limited data. It involves techniques that enhance the quality and size of a training dataset through modifications or transformations of existing data, allowing expansion of the dataset even when new data is hard to obtain [105]. This process enhances training data quality by introducing variability, which reduces overfitting and increases model robustness. Data augmentation techniques for image analysis can be divided into **deep learning-based** and **traditional** methods. Traditional methods include geometric transformations, photometric transformations, kernel filters, and image mixing [105, 106]. However, not all data augmentation techniques are suitable for every task. Below, we outline those commonly used for semantic segmentation.

1. **Image Transformation-based data augmentation :**

   Geometric Transformations

   — *Rotation* : Rotating images by a certain angle (e.g., 90°, 180°, or a random angle within a specified range) can help the model become invariant to the orientation of objects.

   — *Translation* : Shifting images horizontally or vertically. This helps the model learn to recognize objects regardless of their position in the image.

   — *Scaling* : Resizing images up or down. This helps the model become invariant to the size of objects.

   — *Flipping* : Horizontally or vertically flipping images. Horizontal flipping is more common and can help the model recognize mirrored versions of objects.

   — *Cropping* : Randomly or systematically cropping out parts of the given image and then resizes the cropped part back to a certain size. This can help the model focus on different parts of the image and learn to handle occlusions.

   — *Affine Transformations* : Applying a combination of linear transformations like scaling, rotation, translation, and shearing to introduce more variability.

   Photometric Transformations : Is a type of traditional transformation that changes pixels' values rather than their positions.

   — *Brightness Adjustment* : Varying the brightness of images to help the model learn under different lighting conditions.

— *Contrast Adjustment* : Changing the contrast to simulate different imaging conditions.

— *Saturation Adjustment* : Altering the saturation to make the model robust to variations in color intensity.

— *Hue Adjustment* : Shifting the hue of the image to change colors slightly, making the model more color-invariant.

— *Blurring :* modifies an image by averaging the pixel values within a certain neighborhood around each pixel, effectively reducing the sharpness and details of the image. Common methods to achieve blurring include (gaussian, average, median and motion blur). This can help the model deal with varying degrees of image sharpness.

— *Colour jittering* : Randomly changing the brightness, contrast, saturation, and hue in a single operation.

Noise Injection improves model robustness by adding different types of noise, such as Gaussian, salt and pepper, and speckle, to the training data. This technique helps models perform better on real-world data by making them more resilient to noise and enhancing generalization [107].

Elastic Deformation introduces random, localized distortions to images, simulating non-rigid transformations. This technique helps models learn to segment objects that may be deformed, such as varying anatomical shapes in medical imaging.

Mixing images techniques involves techniques like *MixUp* and *CutMix*. MixUp combines two images and their corresponding masks by taking a weighted sum, helping to smooth out the decision boundary between different classes. CutMix combines two images by cutting a patch from one image and pasting it into another image, with the corresponding masks combined accordingly.

Patch-based Augmentation extracting random patches from images and their corresponding masks to create new training samples [108]. This helps the model focus on different parts of the image and can be particularly useful for training with limited data.

2. **Deep Learning-based Data Augmentation**

GAN-based Data Augmentation involves generating realistic synthetic data using Generative Adversarial Networks (GANs) [105, 106].

Neural Style Transfer applies the style of one image to the content of another, generating new variations of the original image.

2.7 Loss Functions

The optimal performance of deep-learning segmentation models hinges on selecting the right network structure and the appropriate objective function [109]. Model training involves adjusting parameters to minimize the loss function, which is the mathematical expression of the learning objective. To ensure accurate and efficient learning, the loss function must be able to cover edge cases [110]. A loss function measures the discrepancy between the predicted output and the ground truth labels [109].

Based on their focus and objectives, loss functions in semantic segmentation can be regrouped into three main groups [109, 110] : **Pixel-level loss functions** operate at the individual pixel level, aiming to ensure accurate classification of each pixel within segmented regions. These functions calculate the difference between predicted pixel values and corresponding ground truth labels independently for each pixel. In contrast, **region-level loss functions** focus on the overall class segmentation by maximizing the alignment between the predicted segmentation mask and the ground truth mask, emphasizing overlap and prioritizing accurate object segmentation over pixel-wise details. **Boundary-level loss functions** specifically address the precision of object boundaries in the segmentation task to effectively separate overlapping objects. These functions work to minimize the distance or dissimilarity between the predicted boundary and the ground truth boundary, promoting fine-grained alignment of segmented regions.

Cross-Entropy Loss Cross-Entropy (CE) loss evaluates how well the model's predicted probability distribution matches the true distribution of class labels for each pixel location across the entire image. It is calculated independently for each pixel, treating each pixel as a separate classification problem. this loss function widely employed in image semantic segmentation, penalizes incorrect predictions and encourages high probabilities for correct classes at each pixel :

$$L_{CE}(Y, \hat{Y}) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{C} y_{ik} \log(\hat{p}_{ik}), \tag{2.18}$$

Where: $\mathbf{Y}$ = Groundtruth mask (H,W,C)
$\hat{\mathbf{Y}}$ = Predicted mask (H,W,C)
$\mathbf{y}_{ik}$ = The true label value. Taking 1 if pixel i belongs to class k and 0 else.
$\hat{\mathbf{p}}_{ik}$ = The predicted probability that the pixel i belongs to class k.
$\mathbf{C}$ = The number of channels
$\mathbf{N}$ = The number of pixels. N = H × W.

Cross-entropy loss assigns equal importance to each pixel by evaluating class predictions individually and averaging over all pixels. This can lead to poor performance in cases of class imbalance, where more prevalent classes in the image or dataset dominate the training process. To address this issue, weights can be assigned to each class to balance their influence on the overall loss. One common approach used by Long et al. [60], is to use inverse class frequency, where weights are inversely proportional to the number of samples in each class. Thus, classes with fewer samples receive higher weights, while those with more samples receive lower weights. This adjustment helps improve model performance on underrepresented classes. Other weighting strategies may be used depending on the specific task. For instance, Ronneberger et al. [61] propose a weighting scheme that gives higher weights to pixels at object borders. This approach improved their U-Net model's ability to segment cells in biomedical images, making individual cells more identifiable in binary segmentation maps. When assigning weights to each class, the loss function is expressed as follows :

$$L_{WCE}(Y,\hat{Y}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{C}\alpha_k y_{ik}\log(\hat{p}_{ik}), \tag{2.19}$$

where α_k is the weight assigned to class k to balance its contribution in the loss calculation.

Focal Loss , introduced by Lin et al. in 2017 [111], is a modified version of cross-entropy loss designed to address class imbalance. It adjusts weights for easy and hard samples, with hard samples being misclassified with high probability and easy samples being correctly classified with high probability. By incorporating a modulating factor, Focal Loss reduces the weight of well-classified examples, thereby focusing the model's attention on difficult, misclassified examples. This adjustment is particularly effective for tasks with significant class imbalance, such as object detection, where background samples significantly outnumber foreground samples. Focal Loss is express as follow :

$$L_{Focal}(Y,\hat{Y}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{C}\alpha_k\,(1-\hat{p}_{ik})^{\gamma}\,y_{ik}\log(\hat{p}_{ik}), \tag{2.20}$$

where γ is non-negative focusing parameter that adjusts the rate at which easy examples are down-weighted. $\alpha_k \in [0,1]$ is the weighting factor for class k. It can be a constant or a class-specific weight to address class imbalance.

Dice Loss [112] is a loss function used in image segmentation to measure the overlap between predicted and ground truth masks. Based on the Sørensen-Dice coefficient, it aims to maximize this overlap by minimizing *1 - Dice Coefficient*. This loss function effectively handles class imbalance by giving equal importance to both foreground and background classes. Particularly useful in medical imaging, Dice Loss helps achieve accurate segmentation of small or irregular structures. The Dice coefficient is calculated for each class individually (as show in Eq.2.21), aand the average of these coefficients is subtracted from one to obtain the Dice Loss (as presented in Eq.2.22) :

$$Dice_k = \frac{2|Y \cap \hat{Y}|}{|Y| + |\hat{Y}|} = \frac{\sum_{i=1}^{N} y_{ik}\hat{y}_{ik}}{\sum_{i=1}^{N} y_{ik} + \sum_{i=1}^{N} \hat{y}_{ik}} \tag{2.21}$$

$$L_{Dice}(Y, \hat{Y}) = 1 - \frac{1}{C}\sum_{c=1}^{C} Dice_k, \tag{2.22}$$

Where: $\mathbf{y}_i$ = ground-truth label is 1 if pixel i belongs to class k and 0 otherwise;
$\hat{\mathbf{y}}_i$ = predicted label, is 1 if pixel i belongs to class k and 0 otherwise;
N = The number of pixels.

Tversky Loss was designed by Salehi et al. [113] to address the limitations of Dice Loss, particularly in cases with significant class imbalances. Based on the Tversky index, it introduces adjustable parameters to balance false positives and false negatives. Tversky Loss uses parameters α and β to weigh false positives and false negatives, offering flexible control over the segmentation process. This makes it especially valuable in medical imaging, where precise detection of small or irregular objects is crucial and false negatives can be more problematic than false positives :

$$\text{Tversky}_k = \frac{\sum_{i=1}^{N} y_{ik}\hat{y}_{ik}}{\sum_{i=1}^{N} y_{ik}\hat{y}_{ik} + \boldsymbol{\alpha}\sum_{i=1}^{N} y_{ik}(1-\hat{y}_{ik}) + \boldsymbol{\beta}\sum_{i=1}^{N} \hat{y}_{ik}(1-y_{ik})} \tag{2.23}$$

$$L_{\text{Tversky}}(Y, \hat{Y}) = 1 - \frac{1}{C}\sum_{k=1}^{C} \text{Tversky}_k, \tag{2.24}$$

Where: $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ = are parameters that control the weighting of false positives and false negatives, respectively.

Combining different loss functions is common practice to leverage their advantages and mitigate their drawbacks. One popular approach is to combine Dice Loss and Weighted Cross-Entropy Loss, known as Combo Loss.

Combo Loss, introduced by Taghanaki et al. [114], Combo Loss integrates the strengths of Dice Loss and Cross-Entropy Loss for semantic segmentation tasks. Weighted Cross-Entropy Loss addresses data imbalance by giving more weight to underrepresented classes, while Dice Loss enhances the segmentation of smaller objects. Additionally, Weighted Cross-Entropy Loss provides smooth gradients, and Dice Loss helps avoid local minima. The combined loss is formulated by adding the Cross-Entropy Loss and Dice Loss with a modulating term to control their contributions, defined as :

$$L_{\text{Combo}}(Y,\hat{Y}) = \boldsymbol{\lambda} \cdot L_{\text{WCE}}(Y,\hat{Y}) + \mathbf{(1-\lambda)} \cdot L_{\text{Dice}}(Y,\hat{Y}), \tag{2.25}$$

where λ is a balancing parameter that adjusts the contribution of each loss component, typically ranging from 0 to 1.

Loss functions play an essential role in model performance. However, there is no universally *"best"* loss function for all applications. Each loss function has its strengths and drawbacks, and their efficiency depends highly on data characteristics [110] and the choice of function hyperparameters [109]. Therefore, experimentation and validation are often necessary to determine the most effective loss function for a given task.

2.8 Optimizer

In the context of deep learning, an optimizer is an algorithm used to adjust model weights in order to minimizing an objective function. Through iterative process, it adjusts the model parameters in small steps, aims to converge to a set of parameter values that yield minimal loss on the training data. The optimizer guides how the network's learnable parameters are updated based on the gradients of the loss function. Gradient Descent is a common optimizer in deep learning, updating weights in the direction of the negative gradient of the loss function. There are various gradient descent variants, differing in how much data they use to compute the gradient of the objective function. **Traditional Gradient Descent** (or Batch Gradient Descent) processes the gradient using the entire training dataset, which can be slow and impractical for large datasets. **Stochastic Gradient Descent (SGD)** computes the gradient using each randomly picked training example,

offering speed but introducing variability due to frequent updates. Mini-Batch Gradient Descent combines both approaches, performing updates on small random sets of instances called mini-batches, balancing speed and stability. **Mini-Batch Gradient Descent** uses small subsets of the training data. It balances the stability of full-batch gradient descent with the efficiency of stochastic gradient descent, accelerating training and improving convergence. The term SGD is often used even when mini-batches are employed. The update rule for **SGD** can be expressed as follows :

$$\theta_{t+1} = \theta_t - \eta \nabla J(\theta_t), \tag{2.26}$$

Where: θ = as the vector of parameters (weights) of the neural network layer,
$J(\theta)$ = as the loss function,
η = as the learning rate,
$\nabla J(\theta)$ = as the gradient of the loss function with respect to the parameters.

Some optimization techniques such as Momentum [115] and Nesterov Accelerated Gradient (NAG) descent [116] have been proposed to enhance performance of gradient descent algorithms. Both methods aim to overcome the limitations of basic gradient descent, such as slow convergence and oscillations [117]. **Momentum** [115] accelerates gradient descent algorithm by adding a fraction of the previous update to the current gradient, helping to smooth out updates and navigate through local minima. This approach speeds up convergence and reduces oscillations in the optimization process :

$$\begin{aligned} v_0 &= 0 \\ v_{t+1} &= \gamma v_t + \eta \nabla J(\theta_t) \\ \theta_{t+1} &= \theta_t - v_{t+1}. \end{aligned} \tag{2.27}$$

Nesterov Accelerated Gradient (NAG) improves gradient descent by computing gradients at a future "lookahead" position, incorporating momentum from previous updates.

$$\begin{aligned} v_0 &= 0 \\ v_t &= \gamma v_{t-1} + \eta \nabla J(\theta_{t-1} - \gamma v_{t-1}) \\ \theta_{t+1} &= \theta_t - v_t. \end{aligned} \tag{2.28}$$

In gradient-based optimization, it is often more effective to adaptively adjust the step size for each parameter based on the gradient of the loss function, rather than using a fixed step size [118]. To achieve this, parameter-wise adaptive learning rate techniques have been

developed, such as Root Mean Square Propagation (**RMSProp**) [119], Adaptive Gradient (**AdaGrad**) [120], Adaptive Moment Estimation (**Adam**) [121], and Adam with decoupled weight decay (**AdamW**) [122]. **Adam** is an optimization algorithm that combines the advantages of two other methods : **AdaGrad** and **RMSProp**. In other words, it takes into account both the exponential decay average of past gradients and the exponential decay average of past squared gradients. Adam dynamically adjust learning rate for each parameter by estimating first and second moments of the gradients. By incorporating both the mean m_t and the uncentered variance v_t of the gradients. With these characteristics, Adam is suitable for handling sparse gradients on complex problems with complex data and a large number of features. The strategy for updating the weights of **Adam** [121] is presented in the following equations.

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1)\nabla J(\theta_t) \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2)(\nabla J(\theta_t))^2 \\ \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t} \\ \theta_{t+1} &= \theta_t - \frac{\eta}{\sqrt{\hat{v}_t}+\epsilon}\hat{m}_t, \end{aligned} \tag{2.29}$$

Where: m_t , v_t = first and second moment estimates of the gradients, respectively,
$\hat{m}_t$, $\hat{v}_t$ = bias-corrected estimates of the moments,
β_1 , β_2 = exponential decay rates for the moment estimates,
η = learning rate,
ϵ = small constant added for numerical stability,
λ = weight decay coefficient,
$\nabla J(\theta_t)$ = loss function gradient with respect to the parameters θ at iteration t.

AdamW[122], also known as *"Decoupled Weight Decay Regularization"*, is a variant of the Adam optimizer that incorporates weight decay directly into the optimization process. It decouples weight decay from the gradient update step, addressing a limitation in the original Adam algorithm. While Adam applies L2 regularization directly to the gradient, which can lead to undesired interactions between the learning rate and the regularization term, AdamW separates these concerns. In AdamW, weight decay is applied directly to the weights after the gradient update, ensuring a more consistent and effective regularization. This modification leads to better generalization and performance, particularly in training

deep neural networks, as it prevents the distortion of the gradient magnitudes caused by the regularization term in Adam. The update rule for **AdamW** is similar to Adam, with the addition of a term for weight decay regularization :

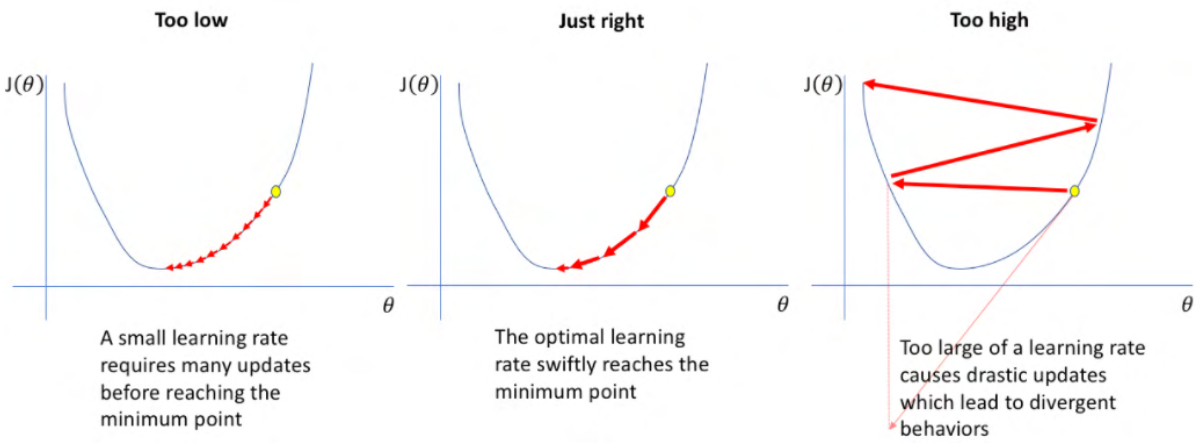
$$
\begin{aligned}
m_t &= \beta_1 m_{t-1} + (1-\beta_1)\nabla J(\theta_t) \\
v_t &= \beta_2 v_{t-1} + (1-\beta_2)(\nabla J(\theta_t))^2 \\
\hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\
\hat{v}_t &= \frac{v_t}{1-\beta_2^t} \\
\theta_{t+1} &= \theta_t - \frac{\eta}{\sqrt{\hat{v}_t}+\epsilon}(\hat{m}_t + \lambda\theta_t),
\end{aligned}
\tag{2.30}
$$

Where: λ = weight decay coefficient,
$\lambda\theta_t$ = the weight decay term.

2.9 Learning Rate Scheduler

The learning rate controls how much optimization algorithms adjust model parameters during training. The Fig. 2.9 illustrates the training progress with different level of learning rate. If the learning rate is too high, the model updates its parameters quickly, which can speed up learning but also risks missing the best solution by jumping around too much. If the learning rate is too low, the model updates its parameters slowly, making learning more stable but also much slower and possibly getting stuck without finding the best solution. The best learning rate is in the middle : fast enough to learn quickly but not so fast that it overshoots the goal. Finding the right learning rate is crucial but can be complex, as the optimal rate depends on various factors like model architecture, dataset complexity, and the optimization algorithm used. Techniques like learning rate schedules, adaptive learning rate methods, and learning rate annealing are often used to adjust the learning rate dynamically during training. These methods help balance convergence speed with stability.

Decay Learning Rate addresses the limitations of fixed learning rates by gradually reducing the learning rate during training. Initially, a higher learning rate speeds up training and allows the model to learn quickly. As training progresses, the learning rate decreases, which helps in fine-tuning the model and achieving better convergence.



Source : https://www.jeremyjordan.me/nn-learning-rate/

Fig. 2.9 Impact of learning rate in training progress.

This approach balances rapid learning with stable convergence. Decay Learning Rate is expressed in Eq.2.31 and different decay functions are presented in Table.2.1.

$$\eta(t) = \eta_0 g(t) \tag{2.31}$$

Where: $g(t)$ = decay function to adjust the learning rate at iterations t
η_0 = initial learning rate

Cyclic Learning Rates (CLR) adjust the learning rate cyclically within a predefined range during training, rather than using a fixed value or decay schedule. Recent studies show that decaying learning rates, when set too low initially, can lead to slow convergence and missed opportunities for faster training, while higher initial values can cause issues similar to those with fixed learning rates. CLR addresses these problems by varying the learning rate periodically, which can help achieve target accuracy more quickly and efficiently, often in fewer epochs despite potential negative effects. However, CLR can sometimes lead to oscillations or instability in the training process, as frequent changes in the learning rate might disrupt the convergence. Cyclic Learning Rates schedule is expressed as :

$$\eta(t) = |(\eta_{max} - \eta_{min})|g(t) + \eta_{min} \tag{2.32}$$

Complex network design can face unstable optimization issues. To address this, starting with a small learning rate leads to better convergence but slows progress, while a large learning rate can cause divergence. Partial warm restarts are introduced as solution to enhance convergence rates and manage ill-conditioned functions [125]. Warm-up strategies gradually increase the learning rate from a low value to a target value over

Tab. 2.1 Learning Rate Scheduling Methods. Where $l \quad (l > 1)$ step size, γ decay rate, N number of epoch

LR Schedule	g(t)
Fixed Step Size	$\gamma^{floor(t/l)}$
Exponential Decay [123]	$\gamma^{t/l}$
Inverse Time Decay	$\frac{1}{(1+\gamma t)}$
Polynomial Decay	$(\eta_0 - \eta_N)(1 - \frac{t}{N})^p + \eta_N$
Cosine	$\frac{1}{2}\left(1 + \cos\left(\frac{2\pi t}{l}\pi\right)\right)$
Triangular [124]	$\max\left(0, 1 - \left\lvert\frac{t}{l} - 2 \cdot floor\left(1 + \frac{t}{2l}\right) + 1\right\rvert\right)$
Triangular 2 [124]	$\max\left(0, 1 - \left\lvert\frac{t}{l} - 2 \cdot floor\left(1 + \frac{t}{2l}\right) + 1\right\rvert\right) \times \frac{1}{2^{floor\left(\frac{t}{2l}\right)}}$

a specified period. This stabilizes optimization and can be applied to various learning rate schedulers, including linear, exponential, and cosine warm-ups. A common approach involves a linear increase to a maximum value followed by a decrease until training ends. Warm-up is particularly effective for training large and deep networks or using large batch sizes.

2.10 Performance Evaluation

Performance evaluation of semantic segmentation is based on two main criteria : accuracy and computation complexity. The accuracy of the model reflects its ability to make good predictions, while its computation complexity induces model speed and memory requirements. In what follows, we analyze these two criteria separately.

2.10.1 Accuracy Metrics

Semantic segmentation models predict the class of each pixel in an image. Various metrics assess their performance, and the choice of metric depends on the application's specific needs. However, a model may excels under one metric may perform poorly under another. As reported in many review papers on semantic segmentation [126, 127, 128, 129, 130, 131], the main evaluation metrics for multiclass segmentation task are Pixel Accuracy (PA) and Intersection over Union (IoU). These metrics are normalized and range between 0 and 1, where a value of 0 indicates poor performance.

Note that **True Positives** pixels represent pixels labeled as class i and correctly classified as belonging to the given class (according to the target mask), whereas a **True Negative** represents a pixel that is correctly identified as not belonging to the given class.

False Positives (false alarms) for a given class i are pixels that are not labeled as class i, but classified as class i. Similarly, pixels labeled as class i but classified in another class are **False Negatives**. In Equations below, **k**is the total number of food classes and $\mathbf{TP}_i$, $\mathbf{TN}_i$, $\mathbf{FP}_i$, $\mathbf{FN}_i$ are respectively the total number of True Positive, True Negative, False Positive, and False Negative pixels for the class i. $\mathbf{N}_{\text{total}}$ is the total number of pixels in the dataset and $\mathbf{n}_i$ is the number of pixels of class i in the ground truth.

Pixel Accuracy (PA) : also known as *Global Accuracy* is defined as the proportion of correctly classified pixels among the total pixels of the dataset. PA is straightforward to understand and calculate and provides a quick snapshot of how well the model is performing. Although, high accuracy does not necessary imply superior segmentation ability because it does not fit for imbalanced dataset. In fact, in cases of imbalanced datasets, where one class is much more frequent than the other, overall accuracy can be misleading. For example, if over 95% of the pixels belong to class A and the rest belongs to class B in the ground truth image, and all the pixels in the predicted segmentation are assigned to class A, then the pixel accuracy would be over 95%. However, this model is practically useless as it does not show any ability to distinguish B from A.

In mathematical terms, it is given by :

$$\mathbf{PA} = \frac{\sum_{i=1}^{k}(TP_i + TN_i)}{\sum_{i=1}^{k}(TP_i + FP_i + FN_i + TN_i)}. \tag{2.33}$$

Mean Pixel Accuracy (mPA) is the extended version of PA, in which the ratio of correct pixels is computed in a per-class manner and then averaged over the total number of classes, as :

$$\mathbf{mPA} = \frac{1}{k}\sum_{i=1}^{k}\frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i}. \tag{2.34}$$

Intersection-Over-Union (IoU) or Jaccard Index is a statistic used for comparing the similarity and diversity of sample sets. In semantic segmentation, it is the ratio of the intersection of the pixel-wise classification results with the ground truth, to their union. It penalizes false positives and false negatives, giving a clearer indication of the quality of segmentation and therefore less sensitive to unbalanced dataset than PA.

$$\mathbf{IoU} = \frac{\sum_{i=1}^{k} TP_i}{\sum_{i=1}^{k}(TP_i + FP_i + FN_i)}. \tag{2.35}$$

Mean Intersection-Over-Union (mIoU) Mean Intersection-over-Union (mIoU) is widely used for multi-class semantic segmentation and is calculated by averaging the IoUs calculated for each class. It is much more effective measure than the pixel accuracy and

does not suffer from the class imbalance issue. It measures the average overlap between the predicted segmentation masks and the ground truth masks across all classes.

$$\textbf{mIoU} = \frac{1}{k}\sum_{i=1}^{k}\frac{TP_i}{TP_i + FP_i + FN_i}. \tag{2.36}$$

Frequency-weighted Intersection over Union (FwIoU) is a variation of the IoU metric that takes into account the relative frequency of each class in the dataset. This metric is particularly useful in scenarios where there is a significant class imbalance, ensuring that the performance on more frequent classes has a proportionate impact on the overall score. By weighting classes according to their frequency, FwIoU mitigates the issue of class imbalance, giving a fairer overall performance measure compared to standard IoU or pixel accuracy.

$$\textbf{FwIoU} = \frac{1}{N_{\text{total}}}\sum_{i=1}^{k}\frac{n_i \cdot TP_i}{TP_i + FP_i + FN_i}. \tag{2.37}$$

Dice Coefficient or F1-Score, is also a commonly used metric for evaluating semantic segmentation models. It is defined as the ratio of twice the intersection of two sets to the sum of the cardinalities of the two sets. In the context of segmentation, it measures the overlap between the predicted and ground truth segmentation masks. The Dice coefficient is very similar to the IoU. They are positively correlated, meaning if one says model A is better than model B at segmenting an image, then the other will say the same.

$$\textbf{Dice} = \frac{\sum_{i=1}^{k} TP_i}{\sum_{i=1}^{k}(2 \cdot TP_i + FP_i + FN_i)} = \frac{2 \cdot IoU}{IoU + 1}. \tag{2.38}$$

2.10.2 Computational Complexity

These metrics focus on different aspects such as model complexity, computational requirements, and training efficiency. Some of the key metrics include :

- **Number of Parameters** : This metric counts the total number of parameters (weights and biases) in the model. Generally, fewer parameters indicate a simpler model, which may be less prone to overfitting and require less memory.
- **Floating-point Operations (FLOPs)** : FLOPs measure the number of floating point operations required to execute the model inference on a single input. It provides an estimate of the computational complexity of the model.
- **Memory Usage** : This metric quantifies the amount of memory required to store the model parameters and intermediate activations during inference. It is important for resource-constrained environments such as mobile devices or edge devices.

— **Inference Time and Training Time** : *Inference time* measures the time taken by the model to process a single input and produce an output. It reflects the computational efficiency of the model and is crucial for real-time applications where low latency is required. *Training time* measures the time taken to train the model on a training dataset until convergence. It depends on factors such as the model architecture, dataset size, and hardware resources. Shorter training times are desirable for faster experimentation and deployment. The performance of these metric significantly depends on the hardware utilized. Thus, for an algorithm, any execution time metric should be accompanied by a thorough description of the hardware used [128].

— **Model Size** refers to the size of the model file or memory footprint when stored on disk or in memory. It includes both the parameters and additional metadata required to represent the model architecture. Smaller model sizes are advantageous for deployment, especially in resource-constrained environments.

— **Energy Efficiency** measures the amount of energy consumed by the model during inference or training. It is important for mobile devices, IoT devices, and other battery-powered systems where energy consumption is a critical factor.

2.11 Conclusion

In this chapter, we reviewed the fundamentals of semantic image segmentation. We introduced learning techniques used in AI : supervised, unsupervised, and semi-supervised approaches, highlighting the advantages and disadvantages of each. Supervised approaches typically deliver the best results, especially for segmenting diverse objects in complex scenes like food images. However, they are difficult to implement because they require large amounts of annotated data, which is time-consuming and labor-intensive to obtain.

We then focused on a specific type of deep learning model, CNN. We presented the basic components of these models, explaining their functionality, strengths, and weaknesses. This included discussing different types of convolution, pooling operations, upsampling methods, and activation functions. Next, we explored factors that can impact model training and techniques to address underfitting or overfitting issues. Finally, we presented the various metrics used to evaluate semantic segmentation models, including precision metrics and those assessing the model's computational complexity.

Understanding the foundations of semantic segmentation models is essential for analyzing existing solutions and proposing more efficient architectures. Additionally,

understanding evaluation metrics is crucial for comparing the performance of our solutions with existing ones.

CHAPTER 3

Food Image Segmentation

3.1 Introduction

NCDs and obesity can be prevented through dietary assessment, which monitors daily food intake and guides healthier food choices [19]. Recent advancements in AI,

especially in computer vision, have enabled the creation of VBDA systems. They involve capturing images of meals and using computer vision model to automatically extract dietary information. A fundamental task in VBDA is food segmentation, which identifies and isolates food items in images by assigning labels to each pixel through semantic segmentation. Over the years, food image segmentation has advanced significantly with the progress in deep learning and computer vision. The evolution has been driven primarily by food image datasets and segmentation approaches. These datasets can be categorized based on various characteristics, such as the type of cuisine, the method of image collection, or the type of annotation. However, food image datasets specific to segmentation are quite rare. Most of the existing datasets in the literature are based on Asian or Western cuisine, despite the fact that each type of cuisine presents its own challenges. Furthermore, there is no dedicated database for African cuisine. Regarding segmentation approaches, several methods can be identified, notably automatic approaches with handcrafted feature extraction, semi-automatic approaches, and automatic approaches with deep learning.

In this chapter, we provide a review of the existing work in the field of food segmentation. The chapter is organized as follows : first, we will give an overview of the existing food image databases, highlighting their different characteristics. Then, we will present the main segmentation approaches in the VBDA field, discussing the advantages and disadvantages of each approach.

3.2 Food Images Datasets

Nowadays the development of deep learning has highly improved accuracy of various computer vision tasks such as image classification, image generation and semantic segmentation. In supervised semantic segmentation by deep learning, a large-scale mask image dataset annotated for each pixel is required to train segmentation models. Collecting such datasets is a labor-intensive task, and the quality of annotations directly affects the performance of the models. PASCAL VOC 2012[132] and MS COCO [133] are widely used as large-scale segmentation datasets, in which the annotations correspond to generic objects, animals and vehicles. Out of the 80 classes that make up the MS COCO dataset, only 10 classes represent common foods (apple, banana, broccoli, cake, carrot, donut, hot dog, orange, pizza, sandwich). Therefore, they are not suitable a training dataset for food image segmentation models [134]. Furthermore, they are not useful to train segmentation models for traditional food images from a given region. Although there exists many food image datasets, only a few of them are publicly available and have pixel-

wise annotation necessary supervised semantic segmentation task [127, 135, 33, 134]. This is mainly because the image pixel annotation process is particularly tedious and sensitive. Existing databases can be distinguished by the different characteristics, such as the number of images, the number of food classes, the type of annotation, the images acquisition technique and cuisine type.

3.2.1 Type of Annotation

Food image datasets can be classified based on their usage, such as food recognition, detection, or semantic segmentation. The type of annotation determines the dataset's purpose. For classification, datasets like **Food-101** provide labeled images of different food categories, enabling the training of models to identify and categorize food items. Detection datasets, like **UECFOOD-100** [136] and **UECFOOD-256**[137], include annotations with bounding boxes indicating the location and class of food items, supporting the development of models that can localize and identify multiple food items within an image. Semantic segmentation datasets, such as **FoodSeg103** and **UECFoodPix-Complete**, offer pixel-wise annotations that allow models to delineate the precise boundaries of food items, essential for tasks requiring detailed segmentation. In this work, we are interested in datasets suitable for full segmentation tasks with deep learning. As such, we will not delve into image-labeled datasets that are limited to food recognition only.

3.2.2 Single/Multi food item per image

Datasets also vary based on the number of food items in the images, which directly affects their complexity and use cases. Some datasets consist of images with a single food item, while others feature multiple food items. Single-item datasets are easier to label and often have clean backgrounds, making them ideal for food recognition tasks. In contrast, multi-food item datasets are more complex, requiring more intensive annotation and are used for tasks like food detection or segmentation. Datasets such as MedGRFood-Segmented [138], PFID [139], Food201-Segmented [140], and UECFOOD100 [136] are primarily composed of single-item images, often representing simple scenes, which may limit the performance of models trained on them when handling more complex images. However, many food segmentation datasets include both single and multi-food images, offering flexibility for models to handle both simple and complex scenarios.

3.2.3 Food origin

Most datasets contain images of foods that are specific to given regions such as Food50Seg and **ChineseDiabetesFood187** [141] that are composed of Chinese food images. **UECFoodPix** [142], **UECFoodPixComplete** [134] and **SUEC Food** Dataset [143] contain popular food in Japan. **MyFood** [1] is build with the 10 most consumed foods in Brazil. Also **Food201-Segmented** [140] is composed of images from USA restaurants and MedGRFood-Segmented [138] contain Greek and Mediterranean food. To the state of our knowledge, all publicly available datasets are composed of Western foods (USA, Italia, Brazil, Europe) and Asian (China, Japan...) foods. There is currently no available dataset for image segmentation which is composed for African foods images.

3.2.4 Annotation techniques

Pixel-wise annotation of food image is a very tedious task. Over the literature we distinguish pixel-wise, manual and assisted annotation methods. *Manual annotation* involves human annotators meticulously labeling each pixel in an image to indicate to which class it belongs. It is the approach used for **ChineseDiabetesFood187** [141], **MyFood**[1] and **FoodSeg103** [144] datasets. This method is highly accurate but labor-intensive and time-consuming, often requiring significant expertise and effort, especially for complex images. On the other hand, *automatic pixel-wise annotation* leverages algorithms and machine learning models to perform segmentation with minimal human intervention. Techniques like GrabCut [145], deep learning models, and other computer vision algorithms can automatically distinguish between different regions in an image. Authors of **UECFoodPix** [142] and **SUEC Food** Dataset [143] used Fast-RCNN and GrabCut [145] to automatically generate pixel-wise annotations using respectively the bounding box annotation of UECFood100 [136] and UECFood256 [137]. While automatic methods are faster and scalable, they may require initial training data and can sometimes produce less accurate results compared to manual annotation [134]. Combining both approaches can enhance accuracy and efficiency, where automatic annotation methods handle large datasets and manual intervention refines critical areas. This solution is adopted for **Food201-Segmented** [140] and **UECFoodPixComplete** [134] dataset. Where authors use GrabCut[145] to generated annotation mask and then manually refine them to correct wrong annotations.

3.2.5 Collecting images

Mainly, there are four methods for collecting images for food image datasets. First, images can be captured in a standardized laboratory environment (such as **UNIMIB2016** [146] dataset), which ensures high resolution and good quality images. However, this method typically limits the number of collected images. Second, images can be downloaded from the internet, either from social networks [147, 148] or search engines (**MyFood** [1]). This approach allows collection of large number of images, but can also result in a large number of non-food images that need to be sorted. Downloaded images may vary in quality, including blurry images, images with text, low-resolution images, or retouched images. Third, images can be collected directly from users, which provides a realistic representation of real-life scenarios. this method is adopted by authors of **ChineseDiabetesFood187** [141], which collected daily diet images of Chinese diabetics. [149] collected images from *myFoodRepo* app users during two years to build MyFoodRepo-273 [149] dataset. However, implementing this method can be challenging, as it requires a large number of users and an extended period to collect a substantial amount of images. Finally, some datasets are build with images from other existing datasets. For instance **UECFoodPixComplete** [134] dataset was build by pixel-wise annotating UECFOOD100 [136] images. Likewise, **Food201-Segmented** is made with a subset of ETH Food-101 [148] segmented images and **SUEC Food** [143] with UECFOOD256 [137] dataset (see Table. 3.1). While ETH Food-101 dataset is a dataset for image classification which consists of 101 food categories with 750 training and 250 test images per category, making a total of 101k images. The labels for the test images have been manually cleaned, while the training set contains some noise.

Table 3.1 lists the publicly available food image datasets for segmentation tasks at the current stage of our investigation. The datasets emphasized in bold are our contributions and will be discussed in detail in the following chapter of this thesis. In Tab. 3.1, we include only those datasets with a publicly accessible download repository. Consequently, datasets like SWVie-Food [150], Food50Seg [151], ChineseDiabetesFood187 [141], and MedGRFood-Segmented [138] are excluded due to the lack of a direct download link or the requirement for access requests. These datasets are categorized based on key attributes, including the *year* of publication, the *number of classes*, the total *number of images*, the *method of image collection*, and the *origin* of the dishes represented in the images. Figure 3.1 shows some sample images from **UNIMIB2016** [146], **UECFoodPixComplete** [134] and **MyFood** [1] datasets.

Tab. 3.1 Summary of publicly available food images datasets for segmentation. **Origin** : As→Asian, Eu→European, **Usage** : S→Segmentation (Pixel-wise annotation), D→Detection (Bounding box annotation).

Year	Name	Images [Classes]	Collect method	Annotation	Origin	Usage
2012	UECFOOD100[136]	12,740 [100]	Web	Manual	Japan	D
2015	UECFOOD256 [137]	28,897 [256]	Web	Manual	As	D
2015	Food201-Segmented [140]	12,525 [201]	Food-101 [148]	Semi Auto	USA	S
2017	UNIMIB2016 [146]	1,027 [73]	Normalized	Manual	Italia	S,D
2019	SUEC Food [143]	31,995 [256]	UECFOOD256	Auto	Japan, As	S
2019	UECFoodPix [142]	10.000 [102]	UECFOOD100	Auto	Japan, As	S
2020	MyFood [1]	1,250 [9]	Web	Manual	Brazil	S
2020	VIPER-FoodNet [152]	14,991 [82]	Web	Manual	USA	D
2021	MyFoodRepo-273 [149]	24,119 [273]	Users	Manual	Eu	S
2021	FoodSeg103 [144]	9,490 [103]	Recipe1M	Manual	As, Eu	S
2021	UECFoodPixComplete [134]	10,000 [102]	UECFOOD100	Semi Auto	Japan, As	S
2022	UECFOOD100-Seg [153]	12,740 [100]	UECFOOD100	Manual	Japan, As	S
2023	**CamerFood10** [154]	**1,241 [10]**	**Web, Users**	**Manual**	**Africa**	**S**
2024	**CamerFood15**	**2,198 [15]**	**Web, Users**	**Manual**	**Africa**	**S**
2024	**AfricaFoodSeg**	**3,067 [2]**	**Web, Users**	**Manual**	**Africa**	**S**

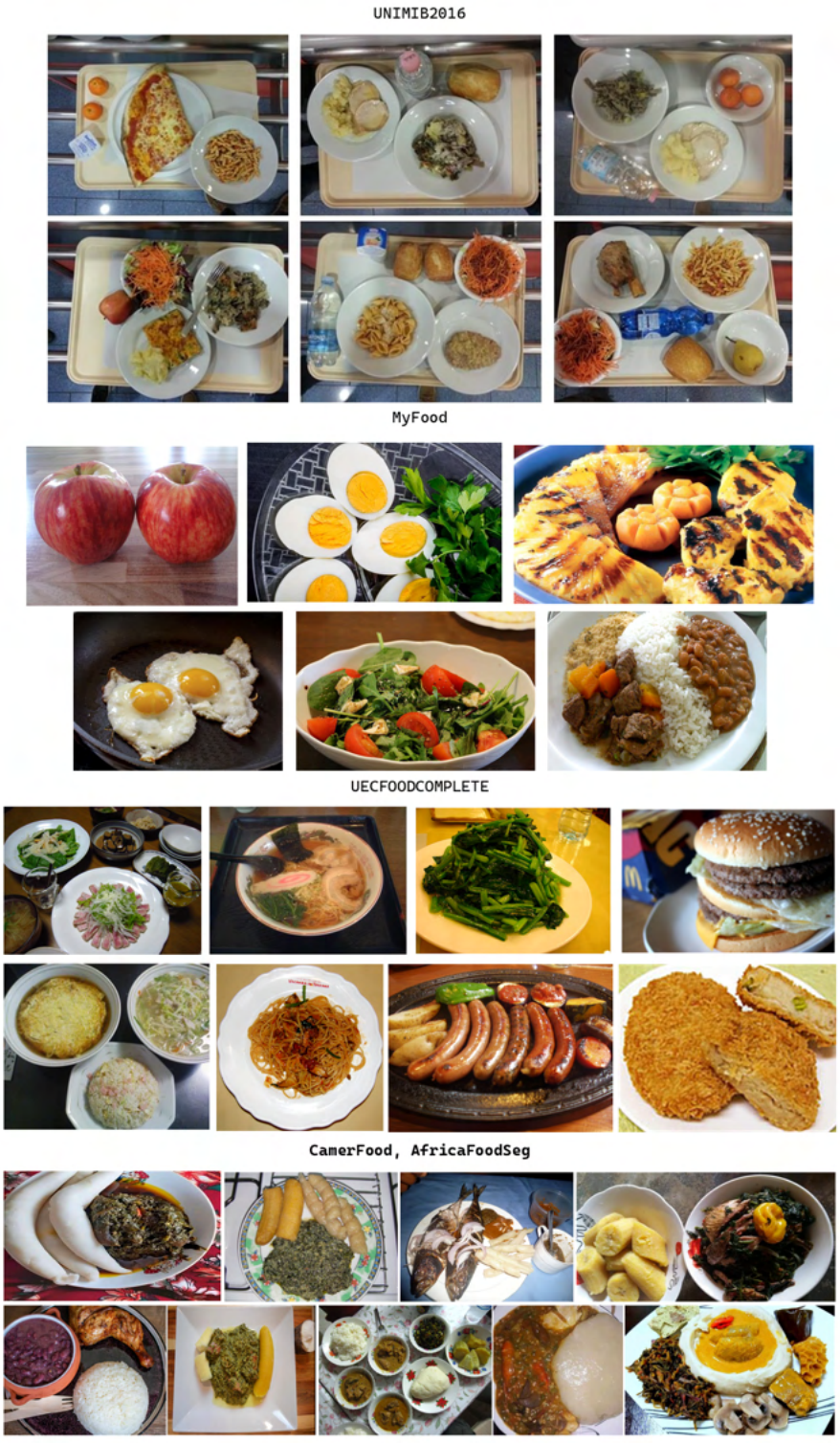


Fig. 3.1 Few image of UNIMIB2016, UECFoodPixComplete, MyFood and our proposed datasets (CamerFood, AfricaFoodSeg).

3.3 Food segmentation methods : State-of-Art

Food segmentation task involves partitioning an image into distinct regions that represent different food items present in the image. This process classifies each pixel into a specific category, ensuring that pixels with the same label share common characteristics. The task becomes particularly challenging when multiple food items overlap or there is a strong lack in terms of visual features for differentiation. To address these challenges, various methods have been published, categorized into three main types [127] : (i) **automatic approaches using machine learning (ML) with handcrafted feature extraction** (ii) **semi-automatic approaches**, and (iii) **automatic approaches with deep learning feature extraction**.

3.3.1 Automatic approaches with handcrafted features

Early food segmentation approaches initially relied on image segmentation methods utilizing handcrafted features [131, 127]. These methods typically follow three main steps : **food region identification, feature extraction, and classification of food regions**. Food **region identification** involves separating the foreground from the background in an image, identifying regions of interest that may contain food items, and distinguishing each food item from anothers. This task relies on traditional image processing techniques such as Normalized Cuts , Graph Cut, Simple Linear Iterative Clustering (SLIC), Deformable Part Model (DPM), the J-measure and Segmentation (JSEG) algorithm, K-means, region growing and merging, and GrabCut. **Feature extraction** involves computing a descriptor vector that best captures the underlying visual information. The most commonly used visual features include color, texture, shape, and size. Techniques for extracting these features include Scale Invariant Feature Transform (SIFT), Histogram of Oriented Gradients (HOG), Gabor filters, MR8 filters, and Local Binary Patterns (LBP). To **assign a label to each identified region** based on its extracted features, traditional machine learning techniques such as Support Vector Machine (SVM) and linear regression are popular methods. Image processing for region identification in complex images with non-uniform backgrounds can be challenging. Early methods simplified this by assuming certain conditions, such as non-overlapping food items, specific plate colors or shapes, and known visual properties of the background [19, 155, 127].

Mariappan et al. [156] assumed that food items were distinctly separate, placed on a white plate, and situated on a uniform tablecloth, either black and white checkered or solid black, enhancing contrast between food and background (as shown in Fig. 3.2). They used threshold segmentation to isolate food regions and then classified them using SVM

based on color and texture features. The process involves two steps : first, converting the image to grayscale, thresholding it to create a binary image, and segmenting using 8-point connectivity while ignoring small segments. Second, converting image to YCbCr color space to determine the plate's color mean. Non-segmented pixels during the first step are compared with the mean value of the color space histogram of the plate to identify plate pixels which were labeled as food. Segmentation was refined through an 8-point connected neighbor search. Color features were derived from pixel intensity in the YCbCr space, and texture features were extracted using Gabor filters. The database used in this work, they use a dataset composed of 50 (17 training and 33 test) replica food images and 256 (11 training, 245 test) real food images. These images where acquired using specific conditions.

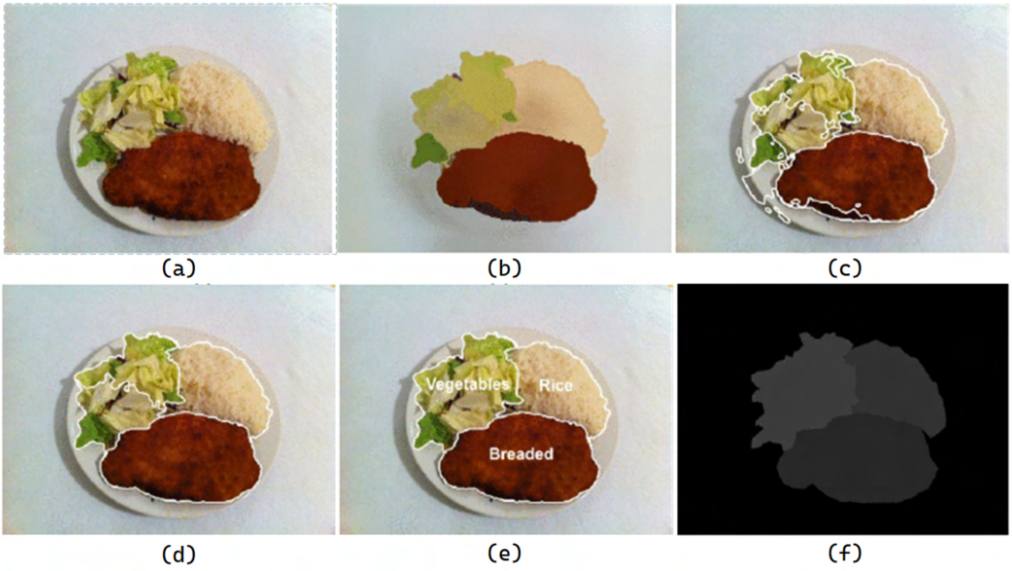
(a) Example of Successfully Classified Food Items. (b) Example of Misclassified Food Items.

Source : Mariappan et al. [156].

Fig. 3.2 Examples of classified food items from Mariappan et al.

Eskin and Mihailidis [157] assumed a white, elliptical plate and stand that food items are more colorful than the plate (see Fig. 3.3). Their segmentation method, similar to [158, 156], used thresholding in HSV color space and morphological operations. They classified food regions with Logistic Regression, utilizing color histograms as color features, Gray Level Co-occurrence Matrix (GLCM) for texture, and region area for shape feature. Their work used a dataset of 676 images featuring 49 food classes (e.g., apples, bananas, strawberries, rice, steak, peas), taken in a controlled environment with two types of backgrounds. Each image had one plate of food with one to four items per plate, touching but not overlapping.

Some approaches use graph-theoretic segmentation instead of threshold segmentation. For instance, Zhu et al. [159] applied normalized cut, treating each pixel as a graph node and segmentation as a graph partitioning problem. Pouladzadeh et al. [160] combined graph cut with texture segmentation, using Gabor filters for texture and SVM for

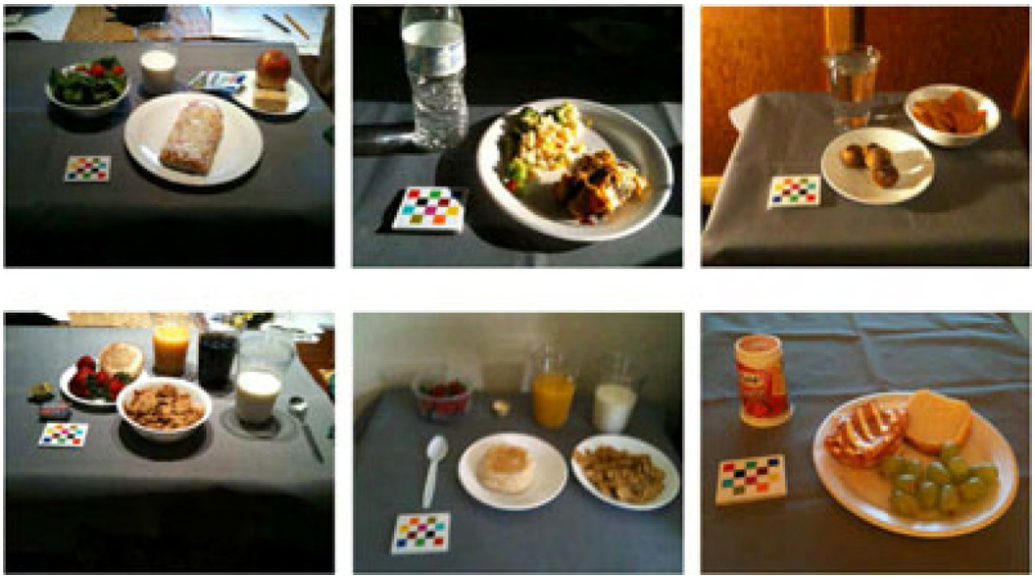


Source : Eskin et al. [157]

Fig. 3.3 The processing stages of the system proposed by Eskin et al. : (a) input image, (b) mean-shift filtering result, (c) region growing result, (d) region merging result, (e) recognition result and (f) desirable output.

classifying food items based on texture, color, shape, and size. Some methods iteratively refine segmentation using classification results. He et al. [161] segmented food based on local variation, examining variability in adjacent regions. Zhu et al. [162] proposed multiple segmentation hypotheses, using normalized cut to dynamically select the number of segments based on classification results. Some examples of images used in their work are presented in Fig. 3.4, showing a table mat of a uniform colour, an official marker and well-separated foodstuffs and drinks.

Anthimopoulos et al. [163] proposed a method where they first convert the image to the CIELAB color space, then apply pyramidal mean-shift filtering, region growing, and merging to segment the image in different region of interest. Before proceeding to the recognition stage the system should specify which of the produced segments correspond to food items by discarding the background segments. To this end, they locate the plate in the image by using an ellipse detector. Local Binary Pattern (LBP) served as Texture features. As for color features, the histogram of most dominant food colors was used. To this end, a hierarchical version of the k-means algorithm is applied to cluster the color space created by the training set of food images. Non-linear SVM with a Radial Basis Function (RBF) kernel where then used to classify identified food items. Their dataset includes 65 manually annotated images and over 5000 web-gathered, annotated food images for training and testing.



Source : Zhu et al. [162]

Fig. 3.4 Some examples of images used by Zhu et al.

One of the most popular works using theses techniques has been presented by Matsuda and Yanai [136] and takes into consideration the problem of images with multiple foods without simplification assumption. They detect several candidate food regions by fusing outputs of several region detectors (DPM, circle detector and JSEG). Then, they recognize each candidate region independently using various feature descriptors (SIFT bag, HoG, Gabor textures) and SVM with multiple-kernel learning. An example of segmentation of a food image is shown in Fig. 3.5, which illustrates the output of region segmentation with JSEG and the result of region integration.

Source : Matsuda and Yanai[136]

Fig. 3.5 Example of candidate region detection by region segmentation proposed by Matsuda and Yanai : (a) input image ; (b) result of region segmentation with JSEG ; (c) Result of region integration.

Traditional image processing techniques like thresholding, GrabCut, or k-means struggle with food image segmentation when there is low contrast between food items

and the background, or when the background is not uniform. To improve segmentation accuracy, automatic approaches with manual feature extraction impose strict assumptions on food images, such as specific plate colors or shapes, uniform backgrounds, non-overlapping food items, consistent lighting, and a fixed number of food items and plates. These methods are very sensitive to variations, which makes them impractical for real-world use where images are taken under diverse and uncontrolled conditions. Consequently, methods relying on handcrafted features face significant robustness issues and perform poorly on real-world images. To enhance accuracy, some research suggests semi-automatic approaches where users interactively indicate the locations of background and foreground regions, helping the model in areas where it encounters difficulties.

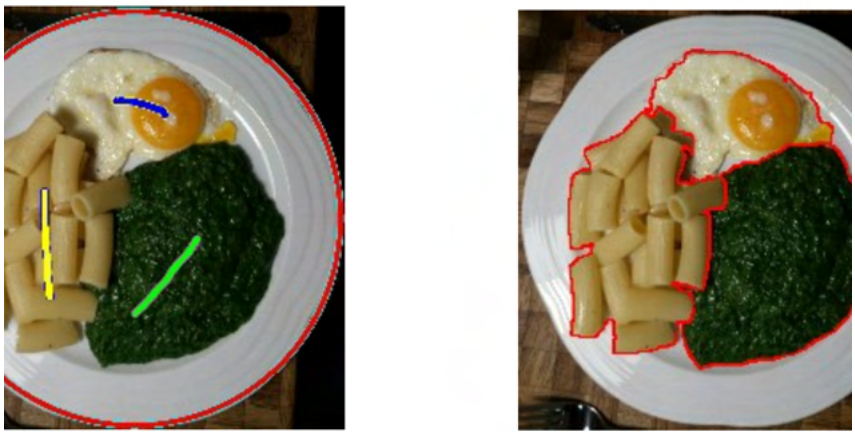
3.3.2 Semi-Automatic approaches with handcrafted features

Semi-automatic and automatic approaches with handcrafted features both follow a similar framework involving food region identification, feature extraction, and classification of food regions. The primary difference is that semi-automatic approaches incorporate user interaction to improve the process of food region identification. In semi-automatic food segmentation techniques, users delimit regions of interest or mark some pixels as food or background. By knowing the exact boundaries of food items, users can guide the segmentation algorithm for more accurate results. While semi-automatic approaches are more precise than automatic methods using handcrafted features, they are also more tedious, requiring extra actions from the user, unlike fully automatic methods where users simply capture the food image.

Kawano and Yanai [164] proposed a mobile food recognition system implemented as an Android smartphone application. Users draw bounding boxes on the screen, and the system recognizes food items within these boxes. Their approach uses the GrabCut algorithm to refine the bounding boxes and extract the food items. GrabCut requires initial foreground and background regions as seeds, so the system designates the regions within the bounding boxes as foreground and the areas outside the doubly-extended boxes as background. The extracted food regions are then classified using a linear SVM with a fast χ^2 kernel, utilizing HOG and color patches with Fisher Vector coding as image features.

Dehais et al. [155] proposed methods for segmenting multiple food items in an already detected dish. An automatic segmentation method is presented able to detect and segment an arbitrary number of different food items in a dish. This method uses a CNN to automatically detect food borders that guide a region growing/merging technique. A

semi-automatic version of the method is also proposed that improves the results by using minimal input from the user. The proposed methods are trained and tested on non-overlapping subsets of a food image database including 821 images, taken under challenging conditions and annotated manually. The automatic and semi-automatic food segmentation methods reached average accuracy of 88% and 92%, respectively. The steps involved in semi-automatic segmentation are illustrated in Fig. 3.6, showing the lines drawn by the user to indicate the areas of interest to the system.



Source : Dehais et al. [155]

Fig. 3.6 Example of semi-automatic segmentation proposed by Dehais et al. : (a) usergiven seeds ; (b) grown regions.

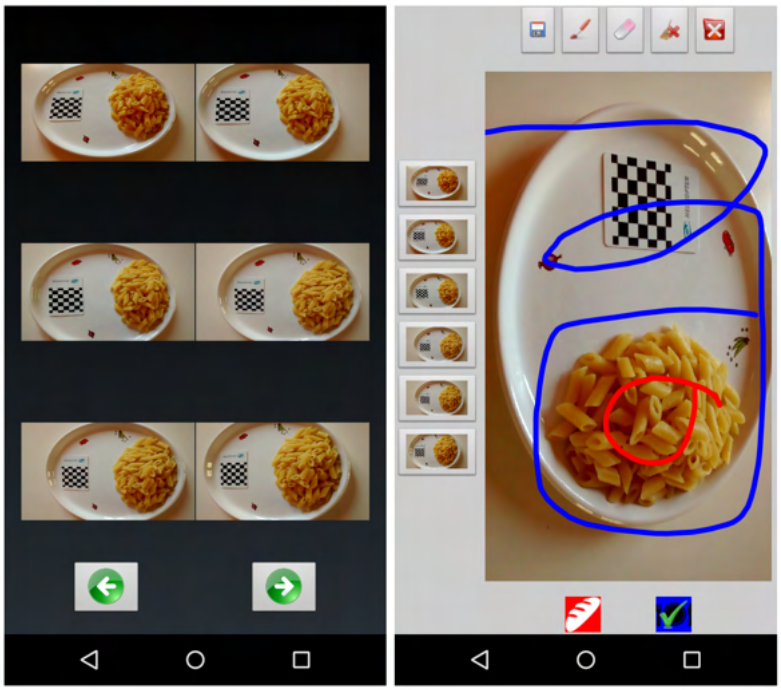
Inunganbi et al.[165] proposed an interactive food item segmentation algorithm that extracts food regions based on user inputs. A Random Forest (RF) classifier is used to classify each pixel as either food or background using RGB components as features. The RF training process is conducted in real-time based on portions of the image selected by the user. To address holes in the segmented food parts due to improper lighting, Boundary Detection and Gappy Principal Component Analysis methods are applied to restore missing information. Local Binary Pattern (LBP) and Non-Redundant Local Binary Pattern (NRLBP) are used to extract features from the restored food parts, which are then fed into a SVM classifier to differentiate between food images. All experiments were performed using the ETH Food101 database, considering only 50 classes.

Hassannejad et al. [166] proposed a segmentation approach to tackle the challenge of food image segmentation with complex backgrounds or multiple food items. They developed a mobile app that allows users to easily mark parts of the image as food or non-food regions. A customized iterative graph cut algorithm is used for complete image segmentation. By marking some pixels, the user imposes hard constraints on the

segmentation. The system uses a Gaussian Mixture Model (GMM) and K-Means to generate image clusters and initialize the graph. Then, the iterative graph cut algorithm segments the entire image automatically. The app is user-friendly and imposes no specific limitations on food types or the number of food items in the image. Tested on various food images taken by mobile phones in different situations, familiar users achieved up to 93.1% accuracy on their first try (with less than 5% of falsely segmented pixels), while unfamiliar users achieved 88% accuracy.

Similar method is used in [167] for a new approach for Image-Based Food Volume Estimation. The method consists of three steps : firstly, a short video of the food is taken by the user's smartphone. From such a video, six frames are selected based on the pictures' viewpoints as determined by the smartphone's orientation sensors. Secondly, the user marks one of the frames to seed an interactive segmentation algorithm (see Fig. 3.7). Food items are then segmented using a Gaussian Mixture Model alongside the graph-cut algorithm. A similar method is used in [167] for a new approach to image-based food volume estimation. This method involves three steps : first, the user takes a short video of the food with their smartphone. Six frames are selected from the video based on the smartphone's orientation sensors. Second, the user marks one of these frames to seed an interactive segmentation algorithm. Finally, food items are segmented using a Gaussian Mixture Model (GMM) alongside a graph-cut algorithm.

Semi-automatic segmentation methods have been introduced to address the limitations of automatic methods using handcrafted features. These approaches allow user intervention, enabling corrections of automatic segmentation errors, thus ensuring higher accuracy. With user interaction, semi-automatic methods are not constrained by images with complex backgrounds or multiple food items. However, they may still be slower than fully automatic methods and can be time-consuming for complex or ambiguous images. Additionally, people with disabilities might find it challenging to use dietary assessment systems based on semi-automatic approaches. Both semi-automatic and automatic methods with handcrafted features share drawbacks related to the selected features. While using a wide range of features can enhance pattern understanding, it also increases computational load. A significant drawback of handcrafted feature approaches is their limited adaptability, requiring features to be tailored to specific types of images, which demands deep domain expertise and substantial effort. Even then, the resulting segmentation may fail to capture the true complexity of the image content. Handcrafted features also struggle with variability in lighting, scale, and noise, reducing their robustness and accuracy in real-world applications. Deep learning feature extraction has been proposed to address these issues, improving segmentation performance by automatically



(a) selected frames (b) manual marking

Source : Hassannejad et al. [167]

Fig. 3.7 Snapshots of the application proposed by Hassanejad et al. After the user takes a short video, six frames are selected automatically (a) and marked by the user to seed segmentation (b).

learning and extracting complex features from the dataset over several training iterations.

3.3.3 Automatic approaches with deep learning feature extraction

Evolution of deep learning and CNNs approaches has remarkably reduced the use of handcrafted features for food image semantic segmentation. Deep learning methods automatically extract food image features and perform better than methods using traditional image processing techniques [33, 127]. In the literature, we have identified two types of food segmentation approaches that use deep learning feature extraction. Some works leverage image processing techniques to identify food regions and then use deep learning models to classify these regions. Other papers use deep learning models to directly assign a label to each pixel in the image. The advantage of the first method is that it does not require a pixel-wise annotated dataset ; instead, it can work with datasets that have image-level labels, which are much easier to obtain. However, the fully deep learning method is more robust to variations, as it can capture complex patterns and high-level semantics present in images with various kinds of backgrounds.

Pouladzadeh et al. [168] is one of the first studies to utilize deep learning feature extraction by combining image processing techniques with deep learning models. They employ graph cut segmentation to identify food regions and use a CNN model for classification. Their experiments demonstrate that this combination significantly improves the accuracy of food recognition and classification compared to a method using color-texture segmentation and the one using graph cut followed by color-texture segmentation. They achieve an accuracy of 99% for single food items, while the other methods achieve 92.21% and 95%, respectively. The dataset used in this study comprises 30 different categories of food and fruits, with each category containing more than 100 images. Although this work does not include mixed food images, it highlights the potential of deep learning methods to achieve better performance than traditional segmentation methods. Okamoto and Yanai [169] estimate rough position of dishes based on edge detection results, and apply color-pixel-based k-means clustering for estimating bounding box of food regions. Then GrabCut were apply with the detected bounding box to extract food item. Figure 3.8 presents an example of food region extraction, showing the result of the different processing steps. Finally, Network in Network (NIN) [170] model was used to classify food items. However, if these previous cited works tackle the problem of handcrafted features selection, they only use images with single food item and uniform background image while in real world food image will have multiple mixed foods items and non-uniform background.

Source : Okamoto and Yanai [169]

Fig. 3.8 The processing steps of food region extraction proposed by Okamoto et al.. (a) Provide a meal photo, (b) Detect a bounding box of a food dish region based on edges, (c) Detect a bounding box of food region by k-means, (e) Detect food region by Grabcut.

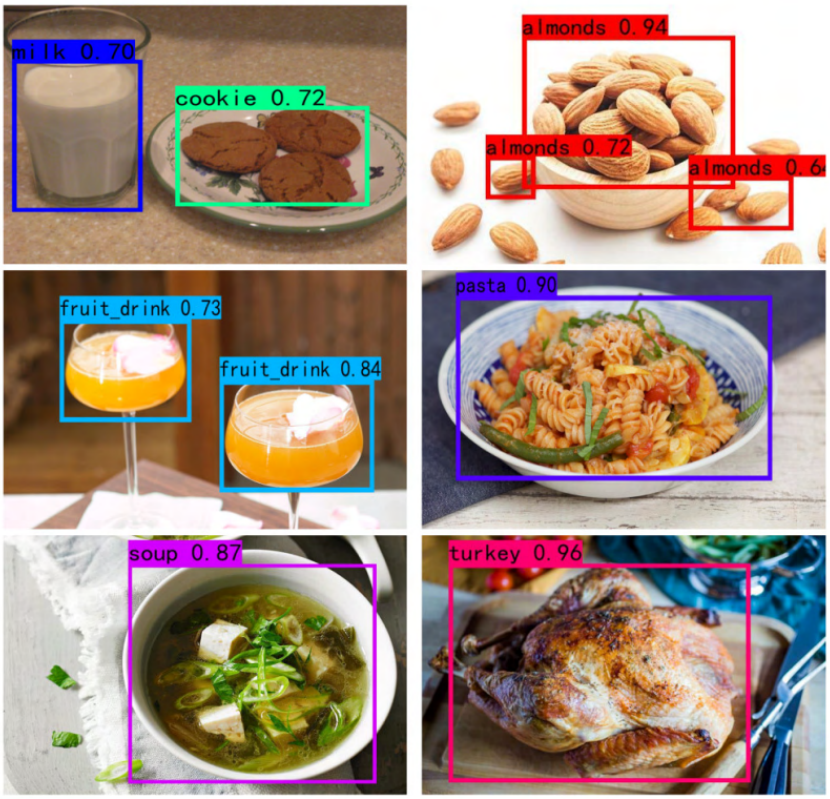
Im2Calories [140] was the pioneering work that utilized CNN models for full semantic segmentation of real-world food images. Their pipeline begins by using the GoogLeNet[171] model, a CNN-based architecture, for binary classification to determine

if an image contains food. At this step, the image is classified into two categories : *"food"* and *"non-food"*. If the image is food-related, they employ the DeepLab model for semantic segmentation, ie pixel-wise classification of the image. The DeepLab[172] architecture involves a series of dilated convolutions, producing an output that is bilinearly interpolated. This output is then refined using a fully connected Conditional Random Field (CRF) for edge-sensitive label smoothing and fine-tuning the final predictions. The DeepLab model was pretrained on ImageNet and subsequently finetuned on the Food201-segmented dataset.

Some studies have focused on simultaneous localization and recognition of foods[173, 152] using object detection models (see. Fig. 3.9). Chiang et al. [174] which proposes a model based on Mask Region-based Convolutional Neural Network (Mask-RCNN) with a union post-processing technique. Nhut Lam et al. [150] used YOLOv7 for the recognition of southwestern Vietnamese food, utilizing the SWVie-Food dataset. Poply and Jothi [175] proposed a method which combines the outputs of both Food detection and Segmentation. An image first passes through Faster R-CNN to predict bounding boxes of food items. Each bounding box is then passed to RefineNet to output semantic segmentation masks associated with the detected food items. Models in this work are trained on UNIMIB2016 food database.

However, for dietary assessment and food calorie evaluation applications, food image semantic segmentation is more suitable than simple localization and recognition of food items. This is because semantic segmentation provides precise delimitation of food items, which is necessary for accurate volume estimation, whereas food detection only provides bounding boxes. Although image detection requires less tedious bounding box annotations compared to the pixel-wise annotations needed for semantic segmentation, most dietary assessment work incorporates semantic segmentation methods due to their superior precision.

In last few years, several works addressing food image semantic segmentation have been published, leveraging increasingly powerful deep learning models. Semantic segmentation involves classifying each pixel in an image into a predefined category (an example is shown in Fig.3.10). Freitas et al. [1] compare the FCN, ENet, SegNet, and DeepLabV3+ model on a Brazillian food dataset (MyFood). Okamoto and Yanai [134] used DeepLabv3+ model on their UECFoodPixComplete dataset and obtained 55.5% of mIoU. Wu et al.[144] proposed a novel fully automatic semantic segmentation method consisting of a recipe learning module and an image segmentation module (ReLeM). They used a Long short-term memory (LSTM) network as the encoder and the vision transformer architecture as the decoder. They achieved 43.9% mIoU in the FoodSeg103 database. Sharma et al.[59]



Source : Chen et al. [173]

Fig. 3.9 Some detection examples with NCNN proposed by Chen et al..

proposed a novel architecture named GourmetNet which incorporates both channel and spatial attention informations in an expanded multi-scale feature representation using advanced Waterfall Atrous Spatial Pooling (WASPv2) [176] module with channel and spatial attention mechanisms. GourmetNet achieves state-of-the-art performance on the UNIMIB2016 and UECFoodPix datasets. Achieving on these datasets an mIoU of 71.79% and 65.13% respectively. Liang et al. [141] introduced a model called ChineseFoodSeg to address challenges specific to Chinese food images, such as blurred outlines, rich colors, and varied appearances. Their model outperformed DeepLabv3+, U-Net, and Mask-RCNN on the ChinesseDiabetesFood187 dataset, achieving an accuracy of 94% and mIoU of 79%. However, their proposed method is more complex and less time-efficient compared to DeepLabV3+. [138] uses Pyramid Scene Parsing Network (PSPNet) with ResNet-101 backbone pretrained on ImageNet2012 and achieves an mIoU score 0.758 in 50 classes of the Mediterranean Greek Food (MedGRFood-Segmented) image dataset. [177] proposes a SegNet + MobileNet semantic segmentation network which consists of MobileNet as encoder and SegNet decoder type. In their experiment, the new system

achieved an impressive 97.82% accuracy on their own dataset based on images acquired in a controlled environment with single food item and uniform background. [153] trained YOLACT [178] and DeepLabv3+ models on a new dataset Segmented-UECFood100 obtained expanding the original UECFood-100 database with segmentation masks. In this paper, YOLACT outperformed DeeplabV3+ reaching 64.63% mIoU. YOLACT (You Only Look At Coefficients) is a simple, fully convolutional model for real-time instance segmentation.

Source : Sharma et al. [59]

Fig. 3.10 Examples of food image semantic segmentation.

More recently, Lan et al. [179] explored the zero-shot capability of the Segment Anything Model (SAM) for food image segmentation. To address the lack of class-specific information in SAM-generated masks, they proposed a novel framework called FoodSAM. This innovative approach integrates coarse semantic masks with SAM-generated masks to enhance segmentation quality. FoodSAM outperforms state-of-the-art methods on both the FoodSeg103 and UECFoodPix Complete datasets. Additionally, Nguyen et al. [180] proposed a multi-task network called FoodMask for clustering-based food instance counting, segmentation, and recognition. FoodMask features three branches for counting, semantic segmentation, and contour map generation. Unfortunately, these approaches lack of sufficient details for reproducibility, and no public code is available.

3.4 Conclusion

In this chapter, we presented a state-of-the-art review on food image segmentation, highlighting the available datasets and major segmentation approaches we have identified. Our research revealed that most studies rely on the same datasets, while some others have not made their databases publicly available. We have gathered all the datasets for food image segmentation with available download links. Notably, public databases are scarce due to the labor-intensive and time-consuming nature of collecting and annotating a large number of food images. We have classified the food image datasets we found in the literature on the origin of the dishes in the images, the annotation method, and how the images were collected. Automatic annotation methods are fast but prone to annotation errors. In contrast, manual annotation is time-consuming and resource-intensive but provides higher-quality annotations. A compromise is found with semi-automatic annotation methods, where a model is used for pre-annotations that are then refined manually. However, these methods require an initial model capable of making the first annotations, which can be challenging if the dataset is entirely different from existing ones or if the images are too complex to make assumptions about the background or food arrangement.

In the second part of this chapter, we present the main types of approaches for food image segmentation identified in the literature : **automatic approaches using machine learning with handcrafted feature extraction**, **semi-automatic approaches**, and **automatic approaches using deep learning**. Approaches based on manual feature extraction are less effective than deep learning-based methods, though the latter requires a large amount of data to train models. Semi-automatic approaches are appealing as they allow for refining segmentation results with user input or guidance, but they require additional user effort and can be slower than fully automatic methods, particularly for complex or ambiguous images. Additionally, people with disabilities may find it difficult to use dietary assessment systems based on semi-automatic approaches.

From this state-of-the-art review, we observe that research on food image segmentation and VBDA are fields that garner significant interest from the scientific community. However, studies do not address African food images and are almost exclusively based on RGB informations. In the following sections of this manuscript, we will present our contributions to the segmentation of African food and RGB-D images. Our proposed methods adopt an automatic segmentation approach based on deep learning, optimizing the trade-off between model performance and computational load.

CHAPTER 4

African Food image Segmentation

4.1 Introduction

Recent advancements in Artificial Intelligence (AI), particularly in computer vision and machine learning, have enabled the development of VBDA systems. These systems involve capturing images of meals and using computer vision techniques to automatically extract relevant dietary information. VBDA typically includes three main stages : vision-based food analysis, portion estimation, and nutrition calculation. Image segmentation plays a key role in the food analysis stage, where individual food items are identified and separated. Given VBDA's potential to improve eating habits and help prevent non-communicable diseases, it is garnering growing attention from researchers. However a literature review [35, 19, 33] show that, research on food image segmentation and recognition has largely concentrated on Asian and Western cuisine. Despite the extensive research in this field, publicly available datasets for food segmentation remain limited, as creating new datasets is a labor-intensive. Although datasets are a crucial component for advancing research in this field. Unfortunately, none of the existing datasets include African dishes, even though African cuisine, particularly Cameroonian dishes, presents unique challenges. African dishes often combine multiple food types, making the segmentation and identification of individual items more complex. The more food item are mixed together on a plate, the harder it becomes to accurately detect the contours of each component.

Our work specifically addresses these challenges by focusing on sub-Saharan African food, with an emphasis on Cameroonian cuisine. In this chapter, we present the first datasets for food image segmentation that focus on African cuisine : the **AfricaFoodSeg** dataset for *food/non-food* segmentation and the **CamerFood** dataset for multiclass semantic segmentation. The CamerFood dataset includes images of the most commonly consumed Cameroonian dishes.

In the second part of this chapter, we introduce **mid-DeepLabv3+**, a new segmentation model inspired by DeepLabv3+[45] that aims to highly reduce parameters and FLOPs while maintaining similar performance. We made three key modifications : using a reduced ResNet[46] (ResNet50, ResNet101) backbone by excluding the last convolution stage, adding a new skip layer in the decoder to recover lost features, incorporating a SimAM[47] attention mechanism. These modifications result in a model that achieves comparable segmentation performance while reducing its size to half that of standard DeepLabv3+[45] with ResNet backbone.

4.2 African Images Datasets

In this section, we present newly created datasets specifically designed for African food segmentation, built which images collected from the web. By gathering images from various online platforms, we have captured a wide array of real-world scenarios, making these datasets a valuable resource for advancing research in dietary assessment. Constructing a dataset from web-sourced images is a challenging process, and the following sections outline the steps taken to develop our AfricaFoodSeg and CamerFood datasets.

4.2.1 Dataset Building Steps

4.2.1.1 Define Food Categories

Our primary objective is to create the first African food dataset for image segmentation. In this work we begin with a focus on Cameroonian cuisine. Although we begin with a focus on Cameroonian cuisine, our datasets implicitly include dishes from several other countries. This is because several Cameroonian dishes also exist in other sub-Saharan African countries under different names. To build the dataset, we began by compiling a comprehensive list of food items, specifically selecting the 42 most consumed foods in Cameroon as identified in [48].

4.2.1.2 Build Search Queries

Once our food list is ready, the next step was to create search queries to retrieve relevant images from web search engines. For each food item, we generated various search terms to capture as many images as possible. Dishes known by local names are translated into English. For example, *"kpwem"* is also known as *"cassava leaves"*. In Cameroon, where both French and English are official languages, some dishes have names in both languages, such as *"sauce jaune"* and *"yellow soup"*, or *"manioc"* and *"cassava"*. Additionally, certain Cameroonian dishes are known by different names in other countries. For instance, *"kpwem"* in Cameroon is the same dish called *"saka-saka"* in Gabon or *"Pondu"* in Congo and the DRC. Although the recipe might vary slightly by region, the dish's visual appearance remains consistent. Therefore, for each dish, we prepare a query that includes its name in the local language, French, English, and any synonyms used in other countries. This approach ensures that we download images that might be published online under different names. Finally we enhance the queries with relevant keywords such as *"dish"*, *"meal"*, *"cuisine"* or *"recipe"* to gather context-rich images and refine the search. For example, we add the keyword *"dish"* for English queries, *"plat"* for French queries, and

include the country name, such as *"Cameroon"* or *"Cameroun"* if using the Cameroonian name for the dish, or the appropriate country name if using another name.

For example :

— **Poulet DG** : This dish doesn't have a translation in the local language or English and doesn't exist in other countries. Therefore, it produces two search queries : *"Poulet DG dish Cameroon"* and *"Poulet DG plat Cameroun."*

— **Kpwem** : This dish is also called "Feuilles de manioc" in French and "Cassava leaves" in English. Additionally, it is known as "Saka-Saka" in Gabon and "Pondu" in Congo. Thus, this dish generates five queries : *"Kpwem Cameroun"*, *"Feuilles de manioc plat Cameroun"*, *"Cassava leaves dish Cameroon"*, *"Pondu Congo"*, and *"Saka-Saka Gabon"*.

4.2.1.3 Web Scraping

After compiling the list of queries, we manually tested each one to assess the quality of the search results. We then used a Python script to automate the image downloading process. This script automates querying search engines and downloading images, utilizing libraries such as *BeautifulSoup*[1] and *Selenium*[2] which are key tools for web scraping. Selenium automates web interactions, such as clicking and form filling, and is ideal for dynamic sites with JavaScript. BeautifulSoup, a Python library, parses HTML and XML to extract data easily. When used together, Selenium can fetch the dynamic content, and BeautifulSoup can then process and analyze the HTML to extract specific information. The images were primarily downloaded from *Google Search*[3], with each query's results stored in specific folders. Additionally, we supplemented the dataset with images taken before meals by ourselves and friends using our phones.

4.2.1.4 Data Cleaning

Web scraping yielded a large number of images, many of which were not useful. Therefore, the next step was to clean the data. After collecting the images, we manually reviewed each folder, removing non-food-related and poor-quality images. The remaining images were then renamed and merged into a single folder. To further refine the dataset, we used a Python script to filter out small images with a height or width of less than 400 pixels, as they often lacked sufficient detail. The script also employed the Laplacian

1. https ://www.crummy.com/software/BeautifulSoup/
2. https ://www.selenium.dev/
3. https ://images.google.com/

operator to detect and remove blurry images by measuring the variance of the image's Laplacian ; images with variance below a certain threshold were considered blurry and discarded. Finally, we used image average hashing to detect and eliminate duplicate or closely similar images, ensuring the dataset's diversity and quality.

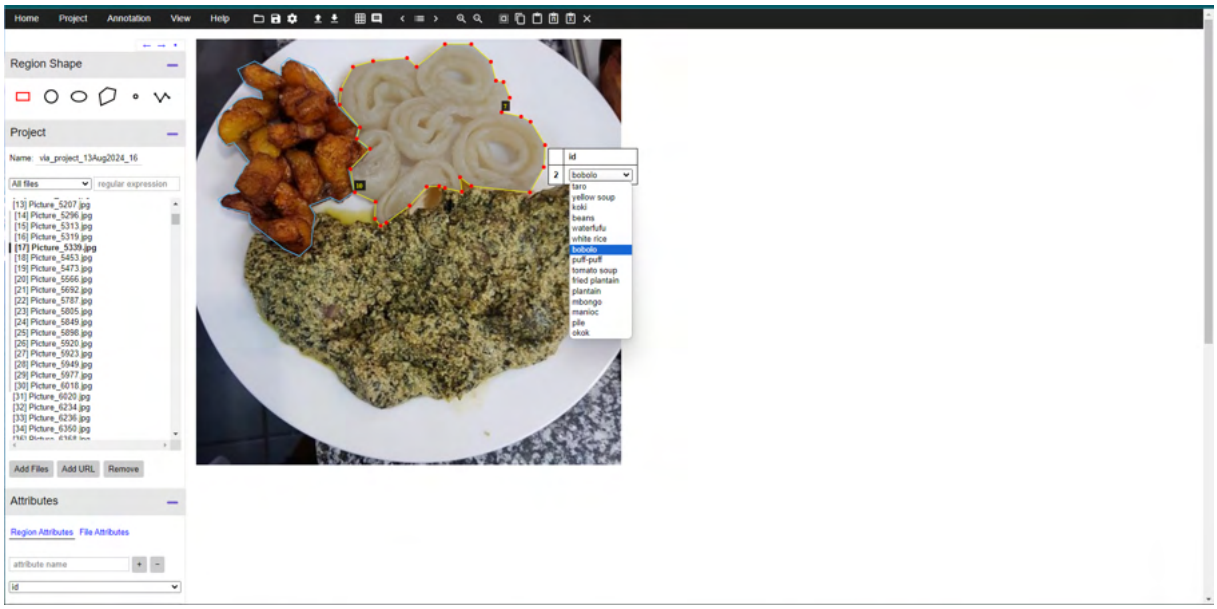


Fig. 4.1 Overview of VIA annotator with an image of CamerFood dataset.

4.2.1.5 Image Annotation

After data cleaning process, we were left with about three thousand images. We then used the VGG Image Annotator (VIA) tool [181] to annotate these images with polygon annotations and generate a JSON file. VGG Image Annotator (VIA) is a lightweight, open-source, web-based tool for annotating images, audio, and video, requiring no installation. Developed by the Visual Geometry Group (VGG), it fits into a single HTML page under 400 KB and is open-source under the BSD-2 Clause license, suitable for both academic and commercial use. VIA supports various annotation types (e.g., points, polygons), making it versatile for tasks like object detection and image segmentation. Annotations can be imported/exported in formats like CSV, JSON, and PASCAL VOC. VIA can be run locally or on a web server and customized for specific tasks. The figure Fig. 4.1 present a screenshot of our interface, showing annotation of an image of our dataset. Each food item was labeled at the pixel level according to its corresponding category on our food list.

4.2.1.6 Dataset Refinement

After the annotation step we organize images in class specific folder. This help to to efficiently create training and validation sets. Training and validation sets are created by taking 10-30% of images per class, depending of the overall number of images and level of intra-variation. For example if a class have a very high intra-class variation, it is important to represent it in the validation set. It is important to do it manually to avoid case where validation set is too simple or complex than the training set. At this set we output two folder containing training and validation images.

4.2.2 AfricaFoodSeg Dataset : *Food/Non-Food* Segmentation

Using the previously built training and validation folders, we create a dataset named **AfricaFoodSeg**. It has two classes (Food and non-Food) and contain 3067 images divide into 2525 for training class and 542 for validation. To the best of our knowledge, AfricaFoodSeg is the first dataset for food/non-food segmentation focus on African food. In semantic segmentation, distinguishing between food and non-food objects can be significantly helpful in various ways. A model pre-trained on food/non-food segmentation serves as a robust foundation for more complex tasks like food classification, enabling faster adaptation and improved accuracy. By segmenting food from non-food objects, especially in applications like food recognition or dietary tracking, the model can focus its computational resources on relevant areas, resulting in more accurate and efficient recognition. Additionally, this approach reduces noise by excluding non-food items such as plates, utensils, and backgrounds, which might otherwise confuse the model and degrade its performance. We present in Fig. 4.2 few samples of AfricaFoodSeg dataset and their groundtruth masks.

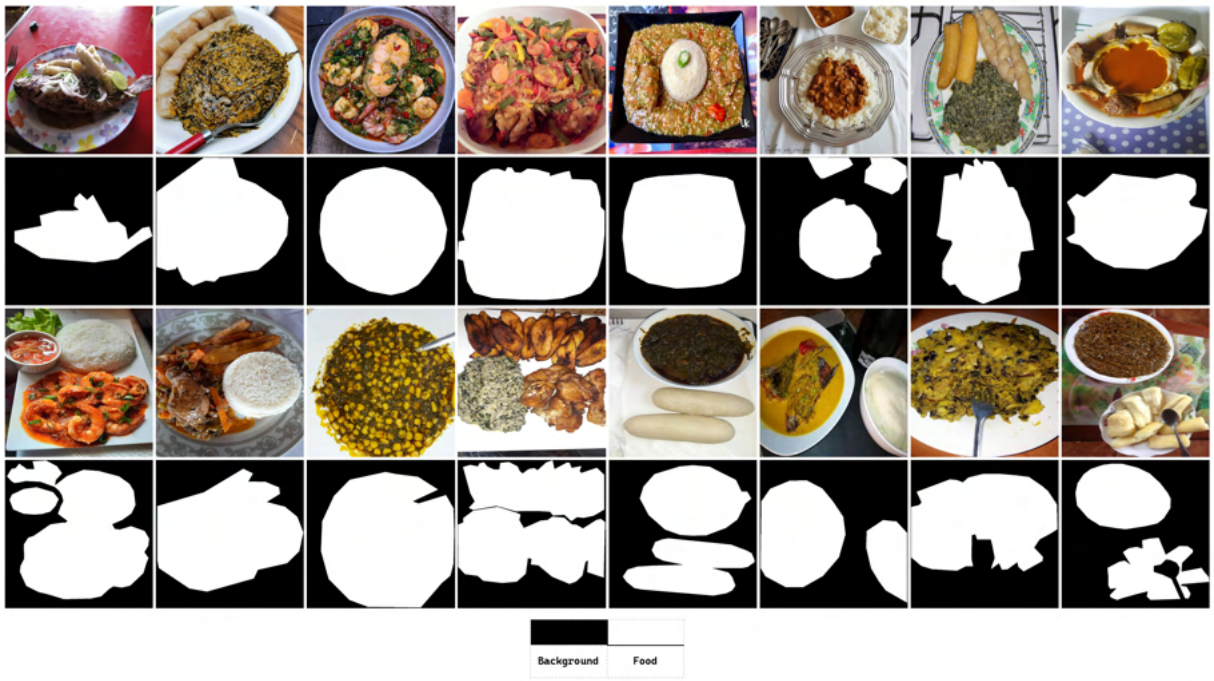


Fig. 4.2 Few samples of AfricaFoodSeg (*Food/non-Food*) dataset and their corresponding groundtruth mask.

4.2.3 CamerFood Dataset

After dataset refinement, we analyse the number of occurrences of each class. Then write a script to remove in the json annotation files, occurrences of all classes with few occurrences to avoid a highly unbalanced dataset. We initially selected 10 classes to form the CamerFood10 dataset. Later, after collecting additional images, we expanded and created an updated version, CamerFood15, which includes 15 classes. The two versions of CamerFood dataset are presented below :

— **CamerFood10** dataset, with 10 classes, consists of 1,241 images and 1,513 annotated food items. The dataset is divided into a training set of 1,032 images and a validation set with 209 images.

— **CamerFood15** dataset is an enhanced version of the CamerFood10. It adds five more classes, increases images per class, and improves annotations compared to its predecessor. CamerFood15 has a total of 2198 images divide into 1684 for training and 514 for validation. Fig. 4.3 summarizes image counts per class for both dataset versions.

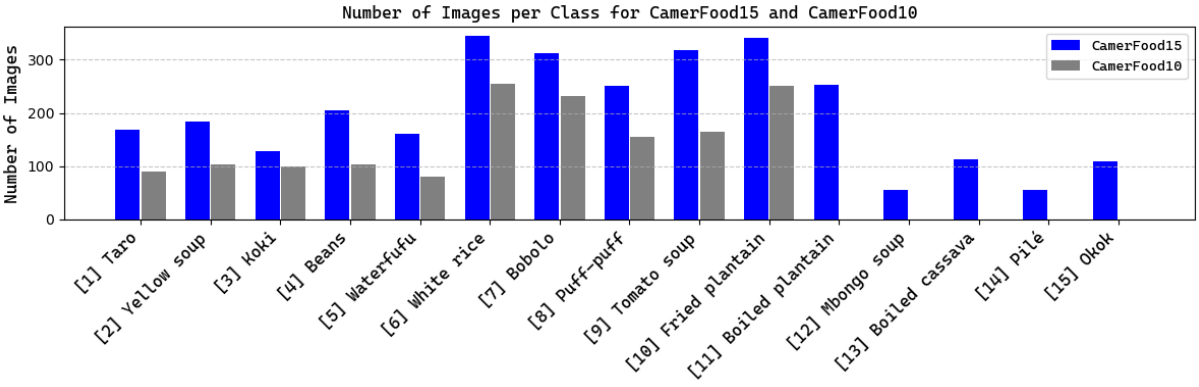


Fig. 4.3 Number of images per class in CamerFood15 and CamerFood10 datasets.

In Fig. 4.4, we present some image from the CamerFood10 and CamerFood15 datasets with their groundtruth masks, highlighting mixed-food presentation in African food and showing that images may contain objects from one or multiple classes.

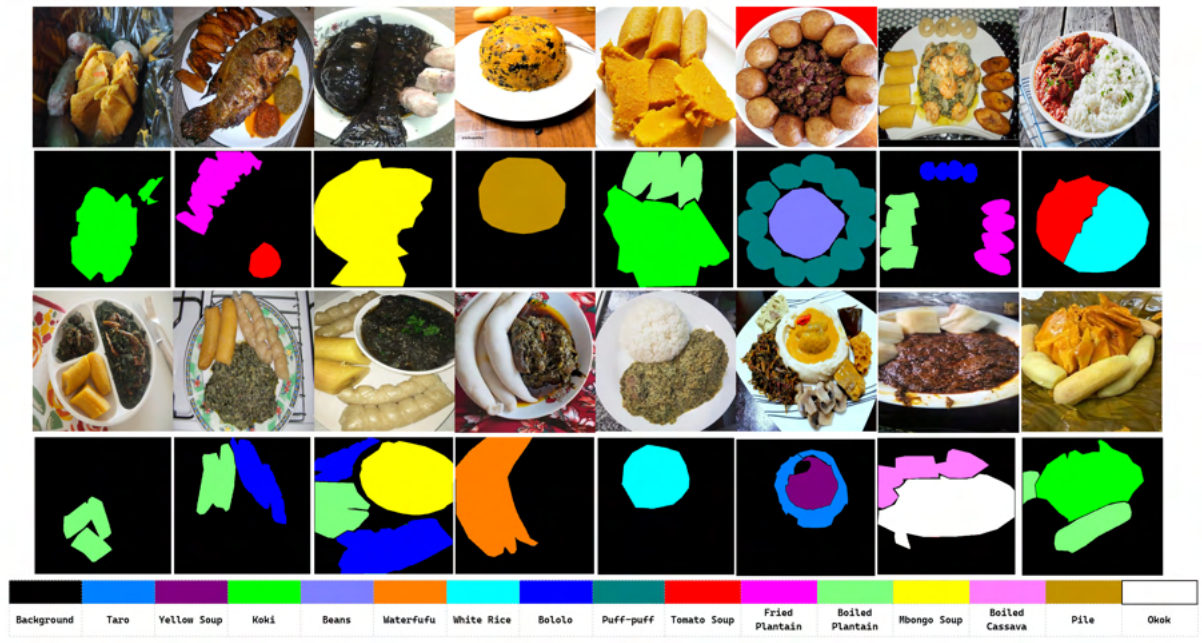


Fig. 4.4 Few samples of CamerFood10 and CamerFood15 datasets.

The table Tab. 4.1 reports the percentage of the number of classes per image. CamerFood15 have 66.86% of image with one class, 30.93% images with two classes, 2.07% with three classes and 0.14% with four classes. The number of food classes per image induces the complexity of the dataset with reflected real world situation. This distribution is respected in training and validation sets. It should be noted that the number of classes per image can be even higher because after annotation we had to ignore certain classes

because they did not have a sufficient occurrences in the dataset.

Tab. 4.1 Number of different classes per image in CamerFood15, excluding the background class.

Set	1 class	2 classes	3 classes	4 classes
Training	66.77%	30.99%	2.07%	0.18%
Validation	66.54%	31.70%	2.14%	0.0%

We also carried out an exploratory analysis of CamerFood15 dataset, producing the distribution of relative food item size, statistics on the number of classes per image and the number of occurrences per class. The Fig. 4.5 illustrates the number of images and the number of occurrences in each class of the CamerFood15 dataset in the training and validation sets. CamerFood15 dataset exhibits some variations in the number of images per class, ranging from a minimum of 55 images in Classes 12 and 14 to a maximum of 344 images in Class 6. The mean number of images per class is approximately 200, with a standard deviation of about 98, highlighting the substantial variability across classes. This discrepancy indicates that some classes, such as Classes 6, 7, 9, and 10, have significantly more images than others, like Classes 12, 14, and 15, resulting in an imbalanced dataset. This imbalance could impact learning model's performance, potentially leading to bias towards classes with more images. To mitigate this, it may be necessary to employ techniques that address the imbalance, depending on the specific requirements of your model and use case.

Our dataset is mainly composed of medium- and large-sized objects distributed as follows : Training set (18.49% small objects, 43.08% medium objects and 38.44% large objects), Validation set (15.81% small objects, 44.54% medium objects and 39.65% large objects). We define small objects as those whose relative size is less than 5% (length x height) of the entire image, medium objects as those whose size is between 5% and 20% of the image, and large objects as those whose size is greater than 20%. Fig. 4.6 shows the distribution of CamerFood15 relative object size by class. Object size in a dataset can significantly impact segmentation performance. When objects are too small, they may lack sufficient detail for accurate identification, making it challenging for the model to distinguish boundaries and features. This can lead to poor segmentation, especially in complex scenes. Conversely, larger objects tend to have more distinguishable features, which can improve segmentation accuracy. However, only an analysis of segmentation performance for each class of CamerFood15 will allow us to objectively assess whether the distribution of the number of images per class or the size of the objects in our dataset affects performance.

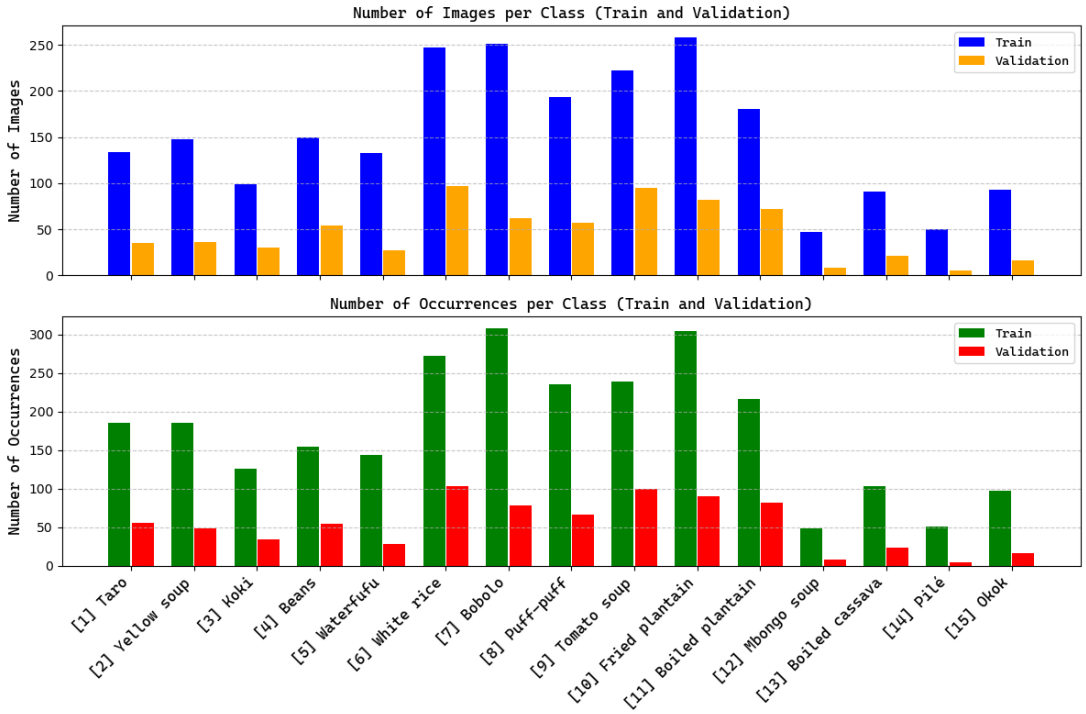


Fig. 4.5 **CamerFood15 class distribution** : number of images per class and occurrence frequency. Number of images show how many images include a given food class. Occurrences indicate how often that food class appears across the dataset.

Fig. 4.6 **CamerFood15** object size distribution : count of small, medium, and large items per class.

4.3 mid-DeepLabv3+

In this section, we present the second contribution of this chapter. **mid-DeepLabv3+** stand for *middle-DeepLabv3+*, is inspired by the well-known semantic segmentation architecture DeepLabv3+ [45], with adding of a middle layer in the decoder path and attention mechanism modules. All design choices made in constructing our model's architecture were driven by two main objectives : reducing the model's size and computational load while maintaining competitive segmentation accuracy compared to other CNN benchmark models.

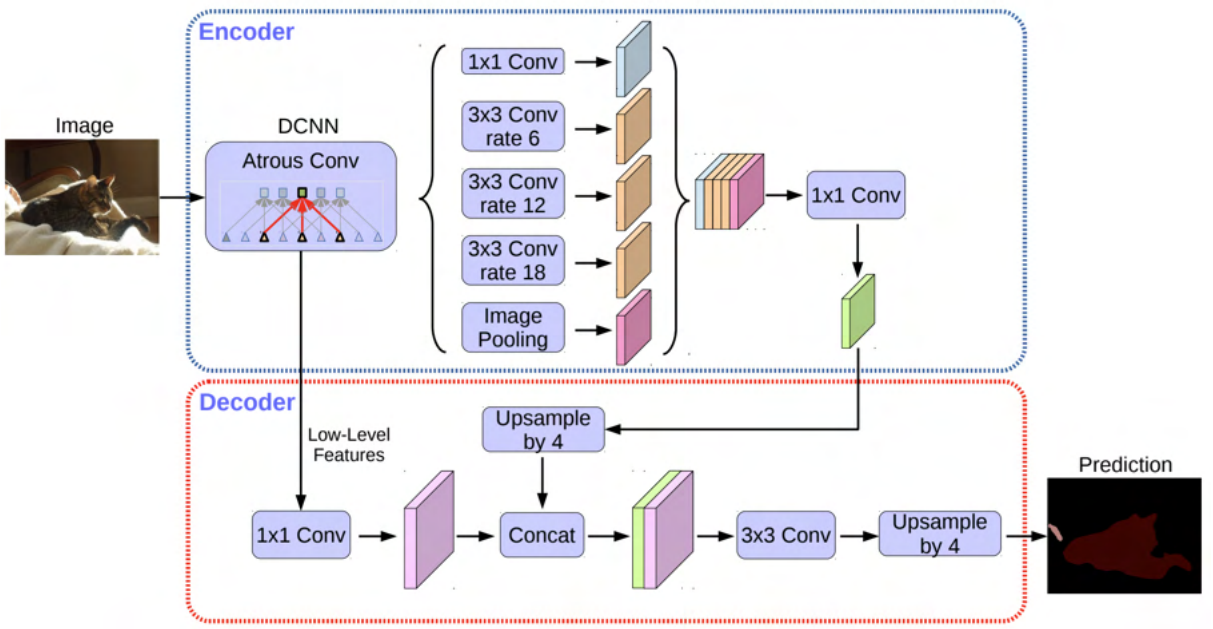
4.3.1 Related Work

4.3.1.1 DeepLabv3+

DeepLab models family are among some of the most popular image segmentation approaches. The main characteristic of DeepLab familly is the *Atrous Spatial Pyramid Pooling* (ASPP) module with its ability to capture long range context and multi-scale information. DeepLabv2[88] initially introduced ASPP as four parallel atrous convolutions with different dilation rates applied in the input feature map, which are then fused together, thus capturing objects as well as image context at multiple scales to robustly segment objects at multiple scales. Concatenating the outputs of multiple parallel atrous convolutions aggregates multi-scale context with different receptive field resolutions. However, applying extremely large dilation rates inhibits capturing long range context due to image boundary effects.

DeepLabv3[182] brings several key enhancements to leverage drawbacks of DeepLabv2. The ASPP module was improved by incorporating image-level features to encode global context, further boosting performance. The resulting ASPP known as ASPPv2, consists of five parallel branches : one 1×1 convolution and three 3×3 convolutions with different dilation rates. Additionally, image-level features are introduced by applying global average pooling on the input feature map, followed by a 1×1 convolution and bilinear upsampling to yield an output with the same dimensions as the input feature map. Finally, the result from each of the parallel branches are concatenated and passed through another 1×1 convolution. To control the resolution of output feature map of the feature encoder backbone without adding extra learning parameters, atrous convolutions were used. Specifically, in the ResNet backbone used, the standard convolutional layers in the last stage were replaced with atrous convolutional layers with a dilation rate of 2.

DeepLabv3+[45] which architecture is presented in Fig. 4.7, enhances its predecessor



Source : Chen et al.[45]

Fig. 4.7 **DeepLabv3+** architecture.

DeepLabv3 by adding a lightweight yet effective decoder module. While the same ASPP module extracts rich contextual information, the decoder refines the segmentation results, particularly at object boundaries. This is achieved by upsampling the low-resolution feature maps produced by ASPP and combining them with corresponding high-resolution features from earlier stages of the network. The final output is a high-resolution segmentation map that accurately delineates object boundaries. DeepLabv3+ uses an enhanced Xception model as its backbone for feature extraction, balancing computational efficiency and performance, but it can also utilize a ResNet backbone. DeepLabv3+ have been a popular choice for many applications such as autonomous driving, medical imaging, and scene understanding.

4.3.1.2 Attention Mechanism

Attention mechanism is a technique in neural networks, particularly in natural language processing and computer vision, that helps models focus on the most relevant parts of input data by assigning different weights to different elements. It improves the model's performance by prioritizing important information, such as words in a sentence or regions in an image, for a given task. Attention mechanisms have provided benefits in many visual tasks [183] such as image classification, object detection, semantic segmentation or

image generation.

In image processing, these mechanisms can be broadly divided into three categories : including channel attention, spatial attention, and hybrid categories combining channel & spatial attention [183]. Considering that different channels in feature maps usually represent distinct objects [184], **Channel attention** adaptively recalibrates the weight of each channel. This process can be seen as a form of object selection, determining *which channels to focus on*. **Spatial attention** can be seen as an adaptive spatial region selection mechanism thus determining *where in the channel to pay attention*. **Channel & Spatial attention** combines the advantages of channel attention and spatial attention. It adaptively selects both important objects and regions [184].

The **Squeeze-and-Excitation (SE)** block [7] was one of the pioneering attention modules, designed to perform channel-wise feature recalibration. It selectively emphasizes informative features while suppressing less relevant ones. SE attention achieves this by first capturing global spatial information from the feature map, followed by using two fully connected layers to model interactions between channels. The output of these layers is then used to refine the feature map at the channel level, enhancing the network's focus on the most significant features. Building on this, the **Bottleneck Attention Module (BAM)** [185] and **Convolutional Block Attention Module (CBAM)** [186] both integrate channel and spatial attention mechanisms, but they do so differently. BAM sequentially combines channel and spatial attention, while CBAM refines this by applying them in parallel. **CBAM** features two sequential sub-modules : channel and spatial attention. The channel attention sub-module operates similarly to an SE block but incorporates max and average pooling to aggregate global information. After channel attention, the spatial attention mechanism generates a 2D attention map by compressing the refined feature map along the channel axis using both average and max pooling. These two resultant 2D feature maps are concatenated and passed through a convolutional layer followed by a sigmoid activation to produce the final spatial attention map. This attention map is then multiplied element-wise with the input feature map, highlighting key regions and suppressing irrelevant ones, thereby sharpening the model's focus on critical spatial information. **BAM**, on the other hand, employs dilated convolutions to enlarge the receptive field of its spatial attention sub-module and constructs a bottleneck structure. It infers channel and spatial attention in two parallel streams, subsequently summing the resized attention maps from each branch. The channel attention branch operates similarly to an SE block, while the spatial attention branch incorporates a bottleneck structure with dilated convolutions. However, a limitation of both CBAM and BAM is that channel and spatial attention are computed independently, neglecting any potential interactions between them [187].

This could limit their ability to learn more discriminative features. **Triplet Attention** [187] addresses this by considering cross-dimensional interactions, thereby enhancing the model's ability to capture richer discriminative features. It employs three branches to capture interactions across Height, Width, and Channel dimensions. Each branch starts by rotating the input along different axes, followed by a Z-Pool layer that aggregates information by combining max-pooling and average pooling along the zeroth dimension. A standard convolutional layer is then used to model relationships between the remaining two dimensions, improving the model's capacity to capture complex dependencies across spatial and channel dimensions.

Attention mechanisms have become increasingly important in computer vision with the rise of deep learning [183]. However, no universal solution exists, as the performance of different attention modules varies depending on the task, dataset, model, and where the module is integrated in the model architecture. It is common practice to experiment with various attention mechanisms to find the best fit for a specific objective.

4.3.2 Model Architecture

In the following we present in detail each block of our proposed mid-DeepLabv3+ model. The Fig. 4.8 present the overall architecture of **mid-DeepLabv3+** model. Our proposed model follows an Encoder-Decoder structure, with the backbone for feature extraction and the ASPP module forming the core of the encoder. We introduce several modifications to the standard DeepLabv3+ architecture, including enhancements in feature extraction, improvements to the decoder, and the incorporation of attention mechanisms.

4.3.2.1 Encoder

Backbone and features extraction : Inspired by [188] and its work on road boundaries estimation, we employ a reduced ResNet[46] network (without the 5th stage) for features extraction. ResNet (Residual Network)[46], have been introduced by He et al. to address the degradation problem, where deeper networks suffer from vanishing gradients and diminishing performance. The core building block of ResNet is the residual block, which typically consists of two or three convolutional layers, batch normalization, and ReLU activation, followed by a shortcut or skip connection. This connection ensures that gradients can flow more effectively through the network. As show at the Fig. 4.9, in ResNet-50, there are 50 layers in total, structured into 16 residual blocks, while ResNet-101 has 101 layers, organized into 33 residual blocks. These residual blocks are grouped into

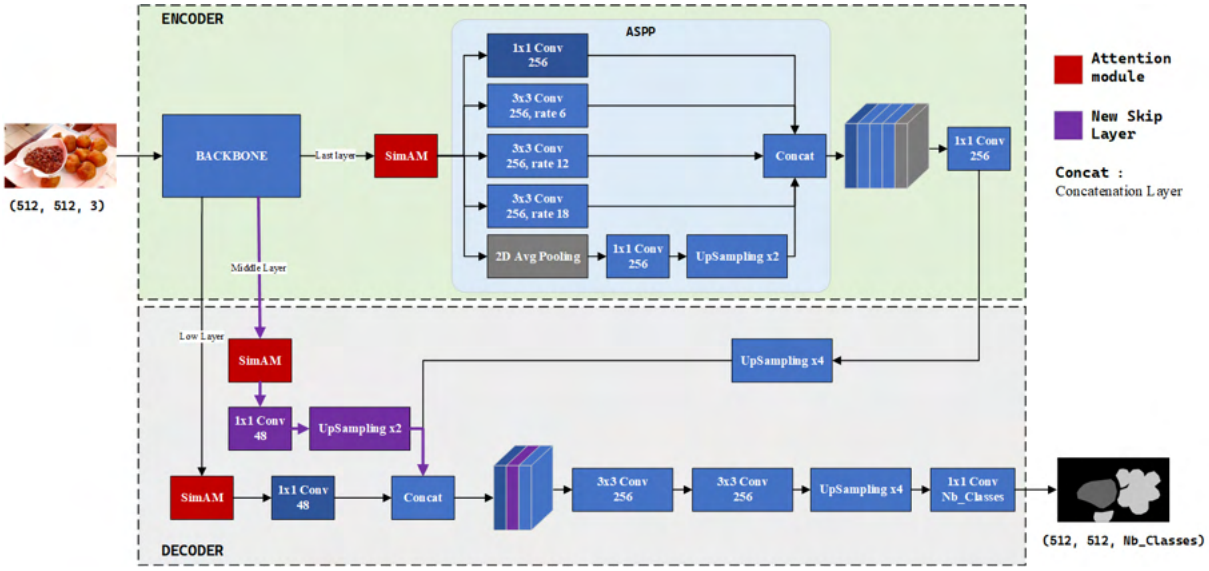


Fig. 4.8 Architecture of our proposed model : mid-Deeplabv3+. In red, the blocks of the attention mechanism and in purple the new skip layer we are adding. The term 3×3 *Conv* 256 refers to a convolution operation with a 3x3 kernel size and 256 filters. Each convolution is followed by Batch Normalization and ReLU activation layers..

four stages, with each stage having a different number of blocks depending on the network depth. Specifically, ResNet-50 has 3, 4, 6, and 3 blocks in its four stages, respectively, whereas ResNet-101 has 3, 4, 23, and 3 blocks. ResNet[46] models have been used as backbone in computer vision tasks such as instance segmentation (eg. Mask-RCNN[49]) and image semantic segmentation (eg. DeepLabv3 [182]) .

In our proposed mid-DeepLabv3+ model architecture, we used ResNet50 and ResNet101 as backbone for feature extraction. The feature extraction module then output three features maps, named in this work as low, middle and deep features. The shapes of the low, middle, and deep features map are $(128 \times 128 \times 64)$, $(64 \times 64 \times 128)$, and $(32 \times 32 \times 256)$, respectively. In the ResNet50[46] model, these layers correspond to *conv*2_*block*3_2_*relu*, *conv*3_*block*4_2_*relu*, *conv*4_*block*6_2_*relu* and in ResNet101 the deep feature correspond to *conv*4_*block*23_2_*relu*. We present in Fig. 4.10 the backbone architecture with ResNet101 and the different extracted features layers. By incorporating the middle layer, we effectively introduce additional levels of detail and context to improve the segmentation performance of our model. We denote *output stride* as the ratio of input image spatial resolution to the final output resolution (last features layer). In this work we adopt an *output stride* of 16 as Chen et al. [45] show that it is the best trade-off between speed and accuracy.

	Output Size	ResNet50	ResNet101
Conv1	256 × 256	7 × 7, 64, stride 2	
	128 × 128	3 × 3 maxPool, stride 2	
Conv2_x	128 × 128	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
Conv3_x	64 × 64	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$
Conv4_x	32 × 32	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 23$
Conv5_x	16 × 16	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1 × 1	avgPool, 1000-d FC, Softmax	

Source : He et al. [46]

Fig. 4.9 Architectures of ResNet50 and ResNet101 and output size of each stageor an input image size of 512 × 512. Building blocks are shown in brackets, with the numbers of blocks stacked. Downsampling is performed by first convolution of stage 3, 4 and 5 corresponding to *Conv3_1*, *Conv4_1*, and *Conv5_1* with a stride of 2.

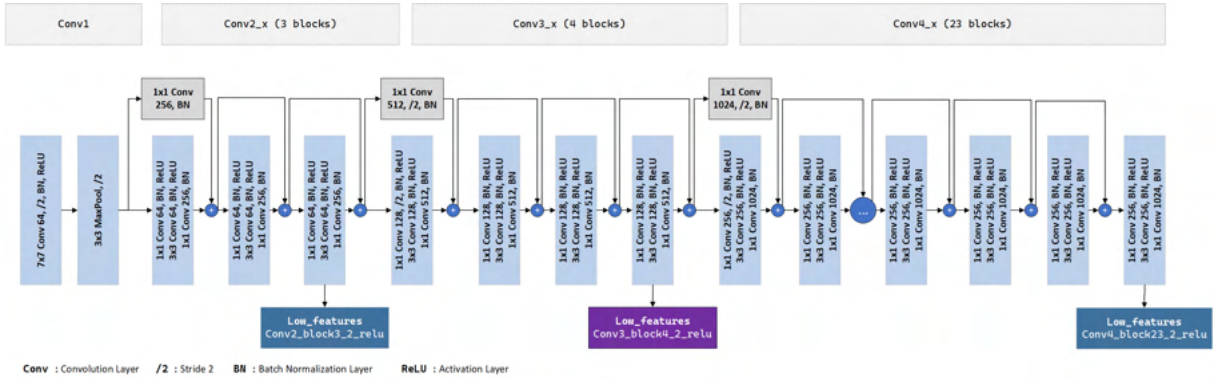


Fig. 4.10 **mid-DeepLabv3+** features extraction backbone based on a scaled-down version of the ResNet101 architecture. This is the ResNet101 model without its fifth convolution stage (*Conv5*)

ASPP : In mid-DeepLabv3+ encoder path, the depth-level features map extracted from the backbone is passed trough the ASPP module. ASPP[182] module, captures multi-scale information with atrous convolutions at different rates. As shown in Fig. 4.8, ASPP includes (a) one 1×1 convolution and three 3×3 dilated convolutions with dilation rates of 6, 12, and 18, each followed by Batch Normalization and ReLU layers, and (b) image-level features obtained via global average pooling of the input feature map, followed by a 1×1 convolution and bilinear upsampling to match the input feature map's dimensions. The

outputs from these parallel branches are concatenated and passed through another 1×1 convolution, followed by Batch Normalization and ReLU layers. All convolutions in the ASPP module use 256 filters.

4.3.2.2 Decoder

One of the challenges with encoder-decoder based segmentation models such as DeepLabv3+ [45] is the loss of detailed features during the downsampling process in the encoder and upsampling in the decoder. To address this, our proposed model architecture incorporates an additional extracted middle layer from the encoder backbone, as illustrated in Fig. 4.8. By extracting an *intermediate layer*, we introduce into the decoder some important features which might be lost in the encoder path and enhances the reconstruction of the final semantic mask. The decoder path of mid-DeepLabv3+, is simple, but effective. First, the low-level features extracted from the encoder backbone (see *Low Layer* in Fig. 4.8) undergo a 1x1 convolution to reduce the number of channel. We adopt a convolution with 48 filters like in DeepLabv3+ [45]. The mid-level features is passed through a 1x1 convolution and bilinearly upsampled by a factor of 2 to match the size of other reduced features layers of the decoder. These reduced features are then concatenated with the ASPP module output, which is bilinearly upsampled by a factor of 4. The concatenated features are further refined using two 3x3 convolution layers each followed by Batch-Normalization and ReLU layers. Following this, the features are bilinearly upsampled again by a factor of 4. Finally, 1x1 convolution, where the number of filters matches the number of classes, is applied to generate the final segmentation mask.

4.3.2.3 Attention module

Our mid-DeepLabv3+ model introduces an attention mechanism added after each feature extraction layer. Attention mechanisms have proven to be effective in computer vision, allowing models to focus on relevant parts of the input by weighting features according to the their importance in the input [183]. Several works have demonstrated that incorporating attention can enhance the performance of semantic segmentation models, including DeepLabv3+ [189, 190, 191]. In our work, we have chosen to use the **SimAM (Simple Attention Module)** [47] attention model. SimAM is a lightweight attention module that does not introduce additional parameters. It directly estimates 3D weights, instead of expanding 1D or 2D weights as in some other spatial and channel attention mechanisms. Despite its simplicity, SimAM performs comparably to popular attention mechanisms, while having no-additional parameters.

The design of SimAM is based on well-known neuroscience theory. In visual neuroscience, the most informative neurons often show unique firing patterns from nearby neurons. Additionally, an active neuron may suppress the activity of its neighbors, a phenomenon known as spatial suppression [192]. Neurons with strong spatial suppression should be prioritized in visual processing. The simplest way to identify them is by measuring the linear separability between a target neuron and others. Based on these neuroscience findings, they defined an energy function for each neuron. By minimizing this function we obtain the linear separability between a target neuron t and all other neurons in the same channel. Consequently, the energy minimisation function for each neuron is as follows :

$$e_t^* = \frac{4(\hat{\sigma}^2 + \lambda)}{(t - \hat{\mu})^2 + 2\hat{\sigma}^2 + 2\lambda} \tag{4.1}$$

Where : $\hat{\mu}$ and $\hat{\sigma}$ are mean and variance of neurons in the channel while λ is a regularization parameter. The lower energy e_t^*, the more neuron t is distinctive from surround neuron. Therefore the, the importance of each neuron is obtained by $\frac{1}{e_t^*}$.

Additionally, the *Sigmoid* function is applied to scale the attention vector between 0 and 1. Then an element-wise multiplication with the input feature map resulted to a recalibrated final feature map. Sigmoid is used to restrict value of inverted energy. It do not influence the relative importance of each neuron because *Sigmoid* is a monotonic function. The whole weights refinement is expressed as :

$$X^{'} = \textbf{Sigmoid}(\frac{1}{E}) \odot X \tag{4.2}$$

Where : X is the input feature map, $X^{'}$ the recalibated feature map and E groups all e_t^* across channel and spatial dimensions. Except the calculations of channel mean $\hat{\mu}$ and variance $\hat{\sigma}$, all computing of SimAM module are element wise operations.

Following equations 4.1 and 4.2, the implementation code of SimAM with TensorFlow machine learning library is shown in Algorithm.1.

Algorithm 1 SimAM TensorFlow Implementation

```
class SimAM(tf.keras.layers.Layer) :
    def ___init___(self, lambda=1e-7, **kwargs) :
        super(SimAM, self).___init___(**kwargs)
        self.lambda = lambda

    def call(self, X, **kwargs) :
        # X Input feature [N, H, W, C]
        # compute the mean (μ)
        mu = tf.math.reduce_mean(X, axis=(1, 2), keepdims=True)
        # compute square variance (σ^2)
        v = tf.math.square(X - mu)
        sigma2 = tf.math.reduce_mean(v, axis=(1, 2), keepdims=True)
        # compute inverted energy E_inv
        A = 4 * (sigma2 + self.lambda)
        B = tf.math.square(X - mu) + 2 * (sigma2 + self.lambda)
        E_inv = B / A
        # return re-calibrated features
        Y = X * tf.keras.activations.sigmoid(E_inv)
        return Y
```

In this work, we tested multiple attention mechanisms such as SE (Squeeze-and-Excitation) [7], CBAM [186], BAM [185], ECA [193], Triplet Attention [187] and Coordinate Attention [194]. As shown in Fig. 4.8, in our model architecture, attention module (represented by red-colored blocks) is added after each extracted backbone layers. It enable the model to focus on important features and improve segmentation performance. The results presented in the remainder of this chapter will justify our architecture design choices.

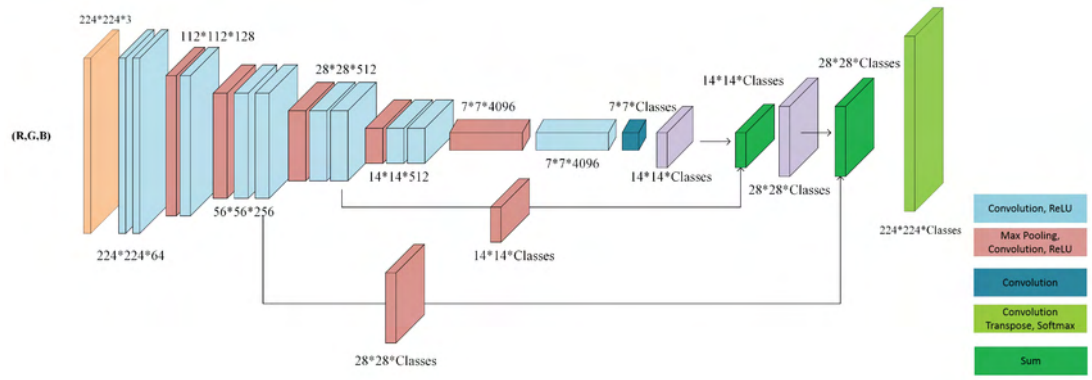
4.4 Evaluation Experiments and Results

4.4.1 Compared Approaches

To evaluate our model mid-DeepLabv3+ we compared its performance with some popular CNN semantic image segmentation models, including FCN [60], U-Net [61], DANet [195], EANet [196], GourmetNet [59], and DeepLabv3+ [45].

4.4.1.1 FCN-8

Introduced by Long et al., Fully Convolutional Network (FCN) [60] is a pioneer work to train a network end-to-end for semantic segmentation. As presented in Fig. 4.11, FCN operates by progressively downsampling the input image through a series of convolutional and pooling layers, creating feature maps that capture hierarchical information. To produce the final segmentation map, FCN use upsampling layers to gradually restore the spatial resolution of the feature maps to match the original input size. Additionally, skip connections are employed to combine high-level semantic information with finer details from earlier layers.

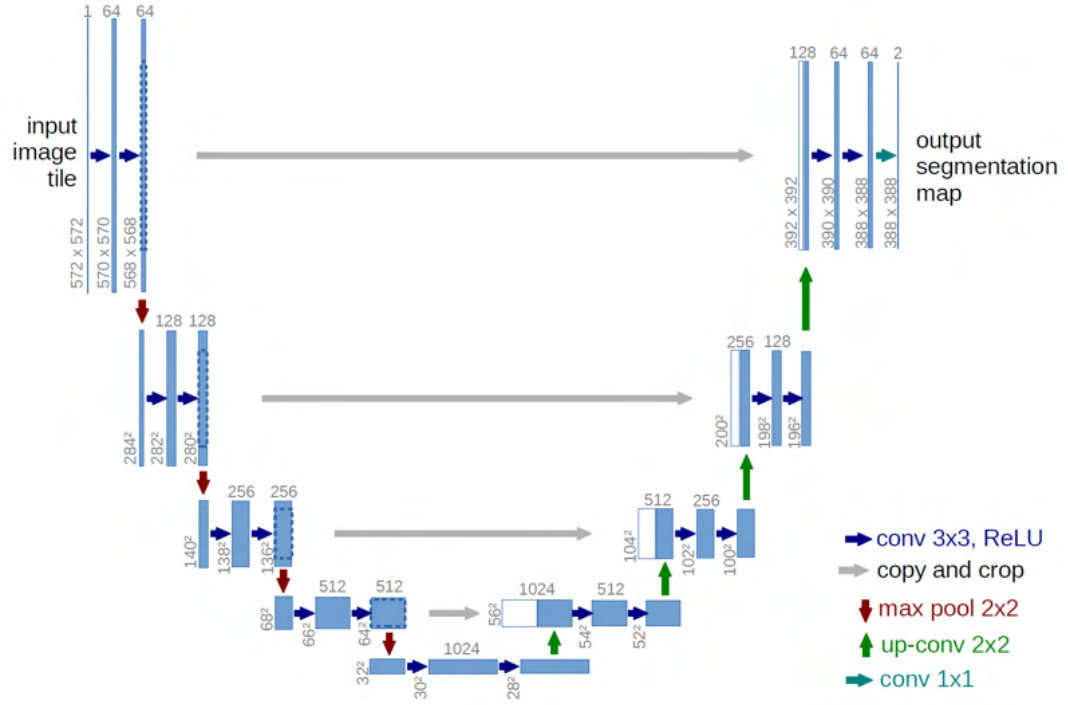


Source : Piramanayagam et al. [197]

Fig. 4.11 FCN-8 architecture.

4.4.1.2 U-Net

U-Net [61] has been primarily designed for biomedical image segmentation but has proven effective in various other segmentation tasks. It is known for its U-shaped architecture (presented in 4.12), which consists of a contracting path (encoder) and an expansive path (decoder). The encoder captures context and compresses the spatial dimensions through a series of convolution layers, followed by max-pooling operation,



Source : Ronneberger et al. [61]

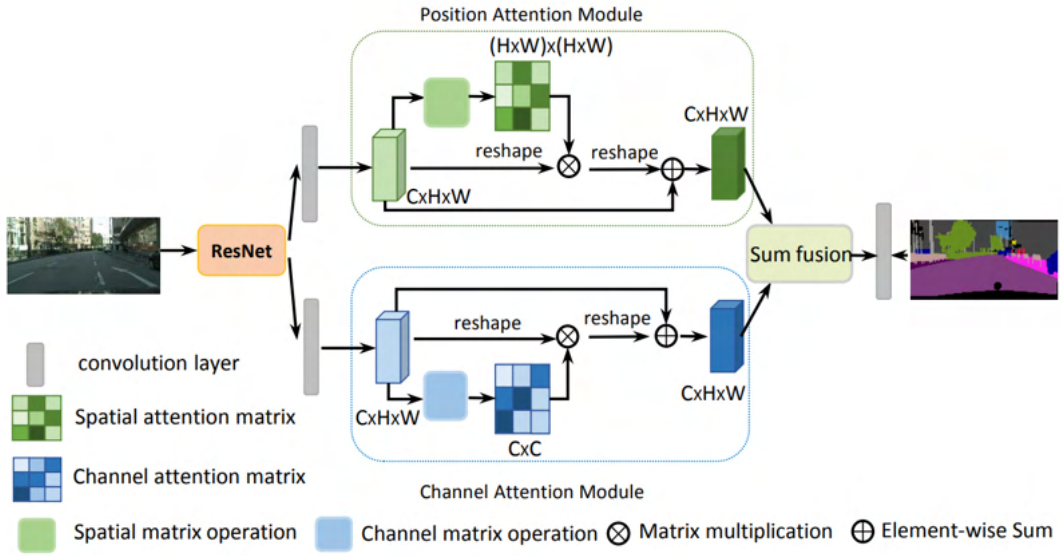
Fig. 4.12 U-Net architecture. Each blue box represents a multi-channel feature map, with the number of channels indicated above the box. The x-y dimensions are shown at the lower left corner of the box. White boxes depict copied feature maps, while arrows indicate the various operations performed.

progressively reducing the spatial resolution while increasing the depth of the feature maps. Conversely, the decoder path gradually restores the spatial resolution by performing upsampling operations using up-convolution (Transposed Convolution), complemented by series of convolution layers.

4.4.1.3 DANet

Introduced by Fu et al. in 2019, the *Dual Attention Network* (DANet) [195] tackles semantic segmentation by leveraging the self-attention mechanism to capture rich contextual dependencies. Unlike earlier approaches that rely on multi-scale feature fusion to capture context, DANet adaptively integrates local features with their global dependencies. As shown in its architecture is presented at Fig. 4.13, DANet processes the backbone's output feature map, through two parallel branches : position attention and channel attention, focusing on spatial and channel domains, respectively. The *position attention module* aggregates features across all positions using weighted sums based on feature similarity, while the *channel attention module* models cross-channel relations. The

outputs from both branches are then combined to generate the final feature representation.



Source : Fu et al. [195]

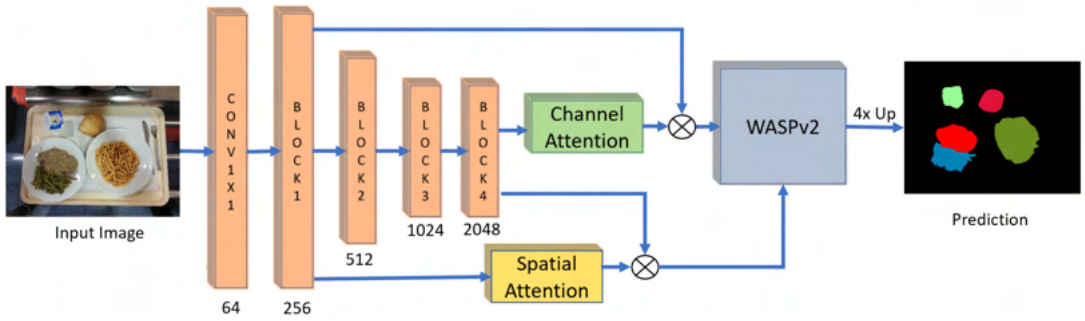
Fig. 4.13 An overview of the Dual Attention Network.

4.4.1.4 GourmetNet

GourmetNet [59] which architecture is presented in Fig. 4.14, was proposed by Sharma et al. in 2021 for food image semantic segmentation. The GourmetNet model starts with a ResNet101 backbone (output stride 16) to extract features. These features are refined by merging multi-level features from the backbone through spatial and channel attention modules. Then, refined features are then processed by an improved Waterfall Atrous Spatial Pooling (WASPv2) module [176], which combines the advantages of cascade filtering and pyramid representations.

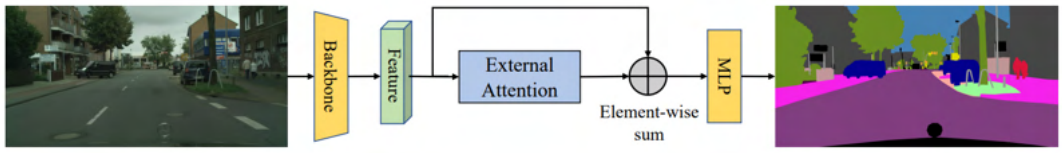
4.4.1.5 EANet

The *External Attention Network* (EANet) has been proposed by Guo et al.[196] in 2022. It is a very simple architecture (see Fig. 4.15) which consist of a backbone for feature encoding follow by an External Attention and a Multi-Layer Perceptron (MLP). They designed External Attention to not only improves the model's ability to capture broader context but also reduces computational complexity compared to self-attention mechanisms, making EANet both powerful and efficient.



Source : Sharma et al. [59]

Fig. 4.14 GourmetNet architecture with ResNet101 backbone (output stride 16) to extract features. The numbers below each block indicate feature channel counts of it output.



Source : Guo et al. [196]

Fig. 4.15 EANet architecture for semantic segmentation.

4.4.2 Experimental Environment

Our experiments were conducted on the Grid'5000 [198] infrastructure using a node equipped with an NVIDIA A40 GPU (with CUDA 12.2) and AMD EPYC 7413 processor. We implemented our model in TensorFlow, initializing backbone of all models with ImageNet pretrained weights from the Keras API [9]. Additionally, other convolutional layers were initialized using the HeNormal initializer [96]. The training environment used TensorFlow 2.12 and Python 3.10.

All training was done with same hyperparameters : input image size of $512px \times 512px$, batch size of 4, *AdamW* optimizer with *Weight Decay*$= 5 \times 10^{-4}$, and *Categorical Cross-Entropy* loss function. We used an *Exponential Decay* learning rate schedule with an initial rate of 10^{-5}, and saving the best iteration for the validation set. Data augmentation, including flips (up, down, left, right), brightness (max=0.3), contrast (min=0.3, max=1.3), and random cropping, was applied randomly to all images, tripling the number of training samples. For experiments with the CamerFood15 and MyFood datasets, all models were trained for $N = 200$ epochs. In contrast, we used $N = 60$ epochs for the AfricaFoodSeg dataset. This choice was based on preliminary experiments showing that models typically

achieved their best performance well before N epochs and tended to overfit after N epochs. We standardized the number of epochs across all models for objective comparison and saved only the weights of the best results on the validation set.

Finally, all datasets and code used in our work are publicly available in the repositories listed below. We have also shared the weight files for our model.

4.4.3 Evaluation Results and Discussions

In the following sections, we present the results from various experiments conducted to validate our contributions.

4.4.3.1 mid-DeepLabv3+ Performance Analysis

For our first experiment, using the CamerFood15 dataset, we examined how the different contributions affect model performance. We began by evaluating performance metrics for DeepLabv3+ with a reduced ResNet101 backbone (as described in Section 4.3.2.1), which we consider our *baseline model*. Next, we tested various versions of the proposed model by incrementally adding the SimAM attention module, the *middle layer*, and both to the baseline network. Our final model, named mid-DeepLabv3+, is the version that incorporates both the SimAM module and the middle layer (See Fig. 5.17). Moreover, we compared our model with the standard DeepLabv3+ using ResNet101 backbone, as our mid-DeepLabv3+ model is inspired by it.

The table Tab.4.2 illustrates improvement achieved by each contribution over the baseline model. This table provides a comparison based on evaluation metrics like the **number of parameters**, number of floating-point operations (**FLOPs**), **model size**, **Overall Pixel Accuracy (PA)**, and **mean Intersection Over Union** (**mIoU**). These metrics have been described in Section 2.10. From these results, we can observe that reducing the size of the feature extraction backbone significantly decreases the computational load, but it also leads to a considerable drop in performance. The inclusion of a new skip layer (*middle layer*) plays an important role for enhancing performance by capturing additional contextual information and refining segmentation. Additionally, the attention mechanism improves performance in both the baseline model and the baseline model with the proposed *middle layer*. Finally, the combination of the reduced backbone, *middle layer*, and SimAM module achieves performance similar to DeepLabv3+, while significantly reducing the computational load by half.

Tab. 4.2 Results of mid-DeepLabv3+ ablation experiments for various configurations trained on CamerFood15 dataset.

Architecture	Params(M)	FLOPs(B)	Size (MB)	PA(%)	mIoU(%)
DeepLabv3+ (ResNet101)[45]	59.506	92.707	226.999	95.44	82.88
Baseline (reduced ResNet101)	30.927	60.457	117.979	94.88	81.95
Baseline & SimAM	30.927	60.462	117.979	95.26	82.11
Baseline & Mid-Layer	31.044	62.294	118.425	95.42	82.55
Baseline & Mid-Layer & SimAM	**31.044**	**62.301**	**118.425**	**95.53**	**82.72**

4.4.3.2 mid-DeepLabv3+ Performance with Different Attention Mechanisms

In this experiment we tested some benchmark attention mechanism on mid-DeepLabv3+ model. We tested multiple attention mechanism such as ECA [193], Coordinate Attention [194], Triplet Attention [187], SE (Squeeze-and-Excitation) [7], BAM [185], CBAM [186], and SimAM[47]. The table Tab.4.3 present results obtained in this experiment. For the metrics of the number of parameters, FLOPs, and model size, we reported relative with respect to mid-DeepLabv3+ model with SimAM[47] attention mechanism. Model with the SimAM[47] outperforms other benchmark attention mechanisms, offering better performance with the lowest number of parameters and smallest size. Although SimAM does not have the lowest FLOPs, it still provides the best ratio of computational load to performance.

Tab. 4.3 Results of mid-DeepLabv3+ with some popular attention mechanisms trained on CamerFood15 dataset.

Architecture	Params(M)	FLOPs(B)	Size (MB)	PA(%)	mIoU(%)
ECA [193]	+2.11e-3	-5.50e-3	+8.06e-3	94.89	81.38
CoordAtt [194]	+11.77e-3	-2.85e-3	+44.89e-3	94.88	81.25
TripletAtt [187]	+0.89e-3	+6.04e-3	+3.40e-3	94.92	80.83
SE [7]	+11.23e-3	-5.49e-3	+42.83e-3	95.31	82.69
BAM [185]	+33.18e-3	+85.35e-3	+126.58e-3	95.10	81.64
CBAM [186]	+11.52e-3	-1.54e-3	+43.95e-3	94.31	79.32
mid-DeepLav3+ with SimAM [47]	**31.044**	**62.301**	**118.425**	**95.53**	**82.72**

4.4.3.3 mid-DeepLabv3+ Comparison with Other CNN Benchmark Models

This third experiment is conducted to access how our model perform in comparison to other popular CNN models for image semantic segmentation on the CamerFood15 dataset. We compare our model to DeepLabv3+ [45], GourmetNet [59], DANet [195], EANet [196], and U-Net [61]. In order to make an objective comparison of mid-DeepLabv3+ with DeepLabv3+ [45] we tested two different backbones, ResNet101[46] and ResNet50 [46] as describe in [182]. All the models were built using an *output stride* = 16.

Results of this experiment are presented in table Tab.4.4. We observe that, FCN-8 model with a VGG16 [199] backbone delivered poor results while being too heavy, with a high number of parameters. Although U-Net is a classic model in many studies dealing with medical image segmentation, it performed poorly on our dataset. Additionally, UNet has a very high number of FLOPs, leading to longer time for training iteration. DANet[195] outperforms U-Net and FCN but remains a heavy model. In this experiment, the best mIoU was achieved using GourmetNet [59] and DeepLabv3+ [45] with a ResNet101 [46] backbone. Models with a VGG16 backbone performed poorly, while the same model with a ResNet101 [46] backbone yielded better accuracy compared to one with a ResNet50 [46] backbone. This is primarily due to the strong correlation between segmentation performance and the accuracy of the backbone. Since ResNet101 [46] outperforms in image classification tasks compared to both ResNet50 [46] and VGG16 [199], it enhances the performance of models using it as the backbone. Additionally, ResNet101 [46] is deeper than ResNet50 [46], with more parameters and floating-point operations (FLOPs), resulting in higher computational demands for models based on it compared to those using ResNet50.

Our model mid-DeepLabv3+ stands out as the best in terms of the number of parameters, FLOPs, and model size, while still delivering performance close to the top-performing model, even reaching the highest PA. This makes mid-DeepLabv3+ an optimal choice, offering an excellent balance between performance and computational efficiency. Our experiments show that, with the same backbone, mid-DeepLabv3+ achieves performance comparable to the original DeepLabv3+ [45] and GourmetNet [59] but is significantly lighter. Moreover, our model with ResNet101 [46] is also lighter and more powerful than a DeepLabv3+ [45] model with ResNet50 [46]. This underscores the effectiveness of our enhancements to DeepLabv3+ [45], achieving a more efficient model without compromising performance.

The results presented in Tab. 4.4 are illustrated in Fig. 4.16, where mIoU is plotted against the number of parameters, model size, and FLOPs. The top-left corner represents high mIoU (y-axis) and low x-axis value (Parameters, Size, or FLOPs, depending on the subplot). Therefore the best ratio of performance to computational load is observed when models are closest to the top-left corner of the plot. By analyzing the distance of each model from the top-left corner for the different plot, we found that our mid-DeepLabv3+ models with ResNet50 and ResNet101 backbone are the most efficient in terms of performance relative to computational load.

Tab. 4.4 mid-DeepLabv3+ results and comparison with state-of-the-art models on CamerFood15 dataset. Results for our model are shown in bold, and the best result for each metric is highlighted in green. The symbol, ↓ means that, less is better and ↑ greater is better.

Model	Backbone	Params(M) ↓	FLOPs(B) ↓	Size (MB) ↓	PA(%) ↑	mIoU(%) ↑
FCN-8 [60]	VGG16	134.34	110.88	537.40	93.92	76.44
U-Net [61]	VGG16	37.46	223.09	149.90	93.53	75.46
GourmetNet [59]	ResNet101	47.36	104.85	180.68	95.31	82.95
DANet [195]	ResNet101	67.06	78.19	255.80	95.09	80.71
EANet [196]	ResNet101	53.88	62.55	205.52	95.22	81.35
DeepLabv3+ [45]	ResNet101	59.51	92.71	227.00	95.44	82.88
mid-DeepLabv3+	ResNet101	**31.04**	**62.30**	**118.43**	**95.53**	**82.72**

Model	Backbone	Params(M) ↓	FLOPs(B) ↓	Size (MB) ↓	PA(%) ↑	mIoU(%) ↑
DeepLabv3+ [45]	ResNet50	40.44	73.30	154.25	95.14	81.60
mid-DeepLabv3+	ResNet50	**11.97**	**42.89**	**45.68**	**95.24**	**81.04**
DeepLabv3+ [45]	ResNet101	59.51	92.71	227.00	95.44	82.88
mid-DeepLabv3+	ResNet101	**31.04**	**62.30**	**118.43**	**95.53**	**82.72**

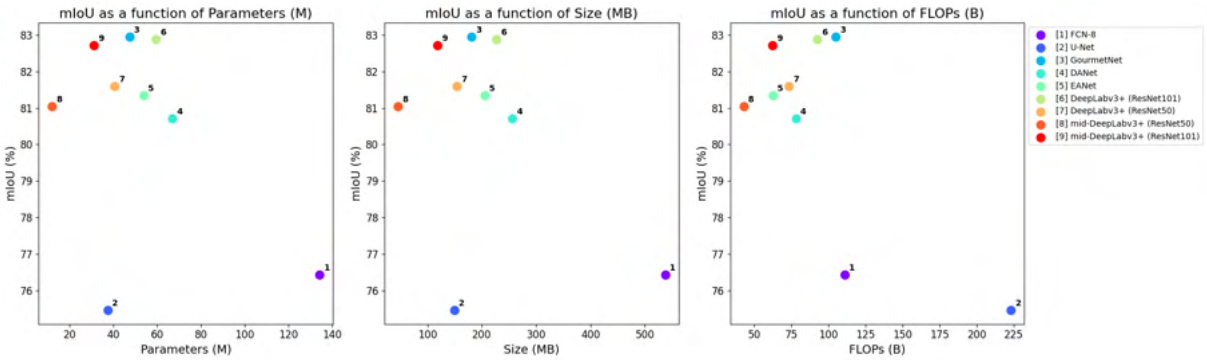


Fig. 4.16 Analysis of models performance and computational load. The best models are those closest to the top left-hand corner. These are the ones with the best ratio of performance to computing load.

4.4.3.4 mid-DeepLabv3+ Evaluation on MyFood [1] Dataset

Here, the performance of the mid-DeepLabv3+ model is evaluated on another dataset. For this purpose, we choose MyFood dataset published by Freitas et al. [1], made up of images of most consumed Brazilian food. MyFood dataset has 10 classes and contains 1,250 images (see Tab.3.1).

With MyFood dataset, we trained the same models as those used in the CamerFood15 experiment, except of FCN and U-Net, which had already shown very poor results in the previous experiment. The results obtained are reported in Tab. 4.5. In this experiment, the MyFood dataset, GourmetNet[59] and EANet[196] provided better results, although GourmetNet[59] has a high number of FLOPs. Compared to other

models, mid-DeepLabv3+ has a lower computational load while still achieving satisfactory results. Specifically, we observe a mIoU absolute difference of only 0.43 compared to the best model, EANet[196], while our mid-DeepLabv3+ (ResNet101) model is 42.4% lighter. Similarly, when comparing to GourmetNet [59], our model is 34.45% lighter with only a 0.30 difference in mIoU. Additionally, our model with ResNet101 as the backbone performs better than DeepLabv3+[45] with both ResNet101 and ResNet50 backbone. Although the performance difference is not substantial, our model remains twice as lightweight. These results validate our contributions to mid-DeepLabv3+, leading to an improved version of DeepLabv3+[45] that is both lighter and delivers closed performance.

Tab. 4.5 mid-DeepLabv3+ results and comparison with state-of-the-art models on MyFood [1] dataset.

Model	Backbone	Params(M) ↓	FLOPs(B) ↓	Size (MB) ↓	PA(%) ↑	mIoU(%) ↑
GourmetNet [59]	ResNet101	47.36	104.51	180.67	93.85	81.99
DANet [195]	ResNet101	67.05	78.19	255.79	93.57	80.47
EANet [196]	ResNet101	53.87	62.55	205.51	93.85	82.12
DeepLabv3+ [45]	ResNet101	59.51	92.37	226.99	93.82	81.32
mid-DeepLabv3+	ResNet101	**31.04**	**61.96**	**118.42**	**93.86**	**81.69**

4.4.3.5 mid-DeepLabv3+ Evaluation on AfricaFoodSeg Dataset

We also trained our mid-DeepLabv3+ model on the AfricaFoodSeg dataset. In this experiment, we compared the performance of our model with the same CNN models used in the previous experiment. The results of this experiment are presented in Table Tab.4.6. In this experiment we observed best performance with GourmetNet [59] model. Although our model, mid-DeepLabv3+, did not achieve the best results on this dataset, its performance it still competitive with other CNN architectures while being approximately twice as lightweight.

Tab. 4.6 mid-DeepLabv3+ results and comparison with state-of-the-art models on AfricaFoodSeg dataset.

Model	Backbone	Params(M) ↓	FLOPs(B) ↓	Size (MB) ↓	PA(%) ↑	mIoU(%) ↑
GourmetNet [59]	ResNet101	47.36	103.91	180.66	96.85	93.80
DANet [195]	ResNet101	67.05	78.19	255.77	96.66	93.50
EANet [196]	ResNet101	53.87	62.54	205.50	96.78	93.72
DeepLabv3+ [45]	ResNet101	59.50	91.77	226.98	96.77	93.70
mid-DeepLabv3+	ResNet101	**31.04**	**61.36**	**118.41**	96.68	93.52

4.4.3.6 CamerFood15 Class-wise Performance Analysis

Finally, we computed the IoU for each class in our CamerFood15 dataset to further assess the proposed model's performance across different food classes. The table Tab.4.7 presents the classe-wise IoU for DeepLabv3+ [45], GourmetNet [59], DANet [195], EANet [196], and mid-DeepLabv3+. We selected models with a ResNet101 backbone that achieved the highest overall mIoU on the CamerFood15 dataset.

After analysis, we observed lower performance for classes with few occurrences, such as *"12-Mbongo soup"*, *"13-Boiled cassava"*, and *"15-Okok"* (as shown in Fig. 4.5). On the other hand, higher mIoU scores were achieved for classes with high number of occurrences, such as *"6-White rice"*, *"8-Puff-puff"*, and *"10-Fried plantains"*. Notably, high performance in classes like *"3-Koki"* and *"14-Pilé"*, despite their few occurrences, can be attributed to their low intra-class variation. Computing performance for each class allows a detailed analysis of the segmentation results. We can observe that despite our model mid-DeepLab (ResNet101) did not achieve the highest overall mIoU, it outperformed other models in most individual classes. These findings further validate the effectiveness of the improvements introduced in this work.

Tab. 4.7 Analysis of class-wise segmentation performance of the mid-Deeplabv3+ model for the CamerFood15 dataset.

Class Id	Class Name	mid-DeepLabv3+	DeepLabv3+	EANet	GourmetNet	DANet
0	*Background*	**94.55**	94.36	94.07	94.13	93.97
1	*Taro*	**80.61**	78.92	80.47	78.51	79.14
2	*Yellow soup*	**81.89**	79.15	78.02	80.65	77.10
3	*Koki*	**86.27**	89.13	85.05	88.66	85.44
4	*Beans*	**80.11**	79.87	80.29	73.7	77.83
5	*Waterfufu*	**80.85**	78.78	77.91	79.96	73.28
6	*White rice*	**89.28**	89.16	88.40	89.29	89.34
7	*Bobolo*	**81.36**	80.36	79.34	78.96	78.64
8	*Puff-puff*	**91.70**	90.32	91.27	91.23	90.89
9	*Tomato soup*	**83.27**	81.58	82.50	81.85	83.95
10	*Fried plantain*	**91.59**	91.50	90.03	91.08	90.10
11	*Boiled plantain*	**83.79**	84.39	83.29	82.84	81.51
12	*Mbongo soup*	**64.41**	67.63	42.73	79.07	49.53
13	*Boiled cassava*	**68.86**	72.47	69.78	69.74	77.51
14	*Pilé*	**91.49**	95.23	95.68	91.31	91.82
15	*Okok*	**73.48**	73.28	74.32	76.22	71.29
	mIoU	**82.72**	82.88	81.35	82.95	80.71

4.4.3.7 Segmentation Qualitative Analysis with CamerFood15

In this section, we analyze the resulting segmentation masks of a few sample images from the CamerFood15 dataset. Fig. 4.17 illustrates the resized images, their ground truth masks, and the predicted masks obtained using mid-DeepLabv3+, DeepLabv3+, EANet, DANet, and GourmetNet with ResNet101 backbone. By analyzing the predicted masks, we can say that although our mid-DeepLabv3+ model doesn't always provide the best results, it remains generally competitive compared to other CNN models.

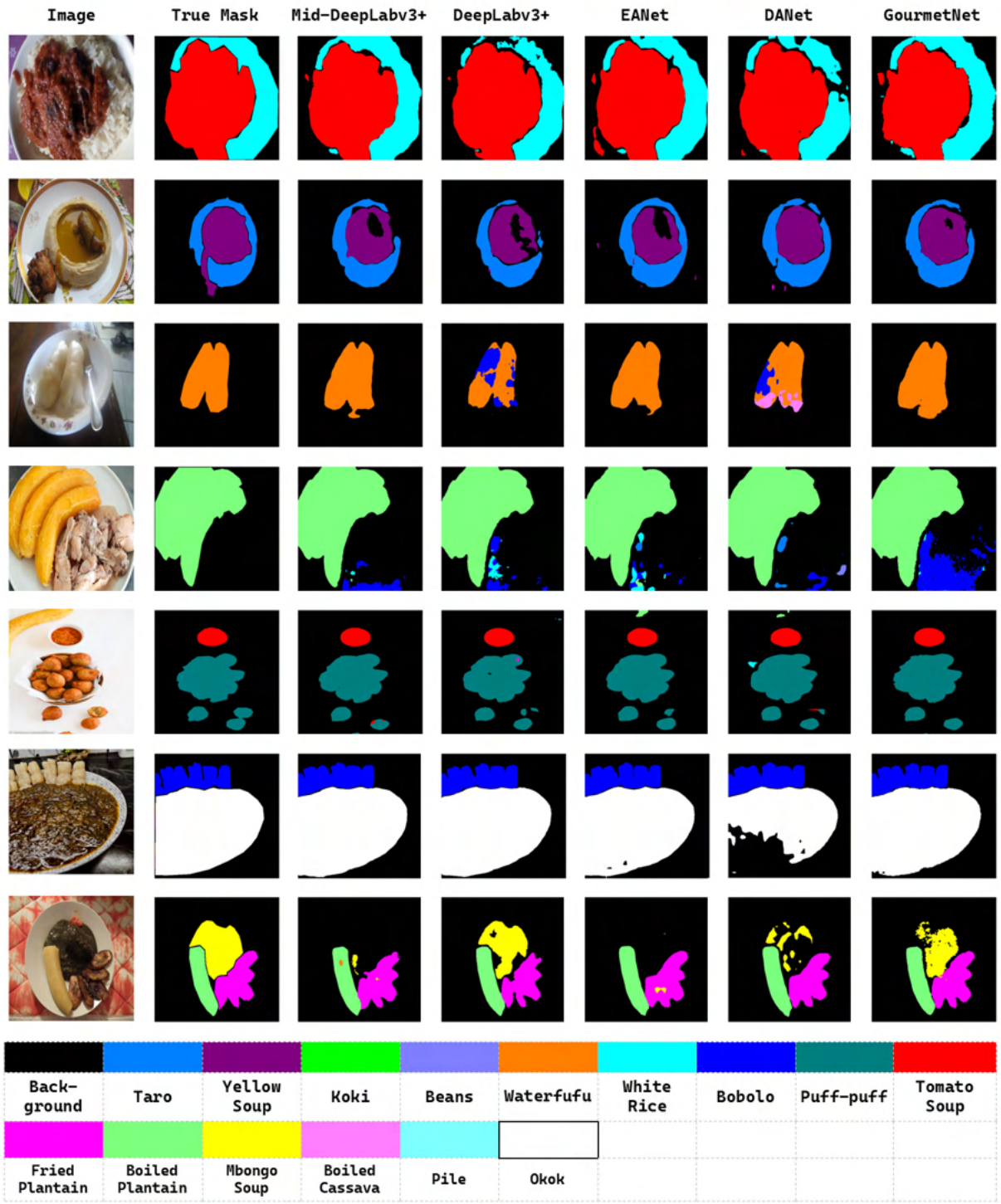


Fig. 4.17 mid-DeepLabv3+ few predictions with CamerFood15.

4.5 Conclusion

In this chapter we address the issue of developing a VBDA system for African foods. Focusing on the image analysis stage, we have made two main contributions. With publicly available datasets on food segmentation being scarce (as shown in table Tab.3.1), we build two food image segmentation datasets **CamerFood** and **AfricaFoodSeg** which are the first publicly available dataset for African food. As African foods have their own specificity, as we have shown earlier in this work, we are convinced that this dataset will be a valuable asset for researchers in the field of automatic dietary assessment. Next, we proposed a new model **mid-DeepLabv3+** for image segmentation. This model is based on DeepLabv3+ and uses the SimAM attention mechanism after backbone feature extraction, combined with a new middle layer in the decoder path. Mid-DeepLabv3+ also uses a reduced version of ResNet (ResNet50 or ResNet101 without the 5th stage) as feature extraction backbone, allowing it to have fewer parameters and be lighter. Our model outperformed many popular CNNs benchmark model for image segmentation, achieving a mIoU value of 82.72% on CamerFood15 dataset. Compared to other benchmark, mid-DeepLabv3+ achieved similar performance while having two time less computational load.

The experiments in this chapter validate our dataset and establish a foundation for developing efficient and lightweight segmentation models. However, mid-DeepLabv3+ still falls short in segmentation performance, highlighting the challenge of maintaining high accuracy while reducing computational load. Furthermore, our model may not be competitive with newer transformer-based models such as SegFormer. Therefore, our next step involves exploring new approaches to further improve segmentation performance while maintaining a relatively low computational load. In the following chapter, we introduce a new model that leverages multimodal images, self-calibrated convolution and employs multistage feature extraction, building on the middle layer concept demonstrated in mid-DeepLabv3+.

CHAPTER 5

Depth map to Improve Food Image Segmentation

5.1 Introduction

Food image semantic segmentation is challenging due to factors like food's flexible structure, diverse visual appearance within classes, similarities between classes, and limited public datasets [33, 200, 201]. Specifically for African foods, intra-class variation is significantly strong, making accurate segmentation more difficult. Additionally, overlapping and blending of foods on plates further hinder segmentation. Although there are several excellent achievements in image semantic segmentation, most of the studies only focus on RGB images. RGB images provide models with information about the colour and texture of objects, but not enough about their spatial geometry. It is therefore difficult to distinguish between instances that share similar colours and textures [39]. To meet these challenges, the researchers began combining depth information with RGB images to improve the accuracy of semantic segmentation. The combination of RGB and depth information, called RGB-D, is interesting for several reasons. Depth data reveals the structure and geometric information of objects in a scene, and hence enrich features found in RGB images. Also, depth images are resistant to environmental disturbances such as lighting, fog, etc [39, 42]. This help to better distinguish various objects, thereby enhancing image segmentation accuracy. Numerous studies have demonstrated that spatial information significantly improves the accuracy of semantic segmentation and have confirmed the effectiveness of learning from complementary patterns. However, some studies [41] indicate that directly integrating complementary depth information into existing RGB frameworks or simply combining the results of both modalities may lead to inferior performance. Thus, researchers continuously come up with various methods in recent years in order to improve the efficiency of RGB-D semantic segmentation. While depth features have been extensively studied for segmenting indoor and outdoor scenes [39], their potential for food segmentation remains largely unexplored. Most existing research in food segmentation relies solely on RGB images, primarily due to the scarcity of public datasets containing RGB-D images.

In this chapter, we present a new architecture for RGB-D image semantic segmentation named ESeNet-D that efficiently leverages both RGB and depth features modalities to achieve higher performance. Thus, demonstrating the efficacy of depth information in enhancing food image segmentation. We also explore MDE as a practical alternative in the absence of true depth images obtained from depth sensors. The main contributions presented in this chapter are described below :

— We introduced an efficient RGB-D segmentation method called **ESeNet-D**, that outperforms several benchmark models while keeping a relatively small weight in

comparison to other benchmark model. ESeNet-D comprise an encoder with two branches using EfficientNetV2 [8] as backbone to process RGB and depth data as input and an efficient fusion block to merges extracted RGB and depth features extracted at three different scales. Then the decoder path takes benefit of self-calibrated convolution and learned interpolation to process outputs of fusion blocks to segmentation mask.

— We also release **ESeNet**, the RGB version of our model. Its outperform many popular models while having lower size, parameters and floating-point operations.

— For the RGB-D semantic segmentation, one of the most challenging tasks is dataset collection [39, 33] : RGB-D datasets for food segmentation are rare. In this work, we demonstrate that MDE models can facilitate the generation of RGB-D datasets for food segmentation. In addition to making evidence that depth maps are useful for semantic segmentation, we also investigate how depth maps quality (resolution, accuracy,...) may impact segmentation performance.

5.2 Related Works

5.2.1 Multi-modal Image Segmentation : State-of-Art

Generally, in computer vision, object are segmented and recognized based on their color and texture attributes. However, the inclusion of depth data can help reduce uncertainty when segmenting objects with similar appearance characteristics. Incorporating depth information alongside appearance data (i.e., RGB) can enhance the performance of semantic segmentation, as the depth channel provides complementary information to the RGB channels and encodes the structural details of a scene [3]. Couprie et al. [202] observed that segmentation performance improves for classes with similar depth, appearance, and location when depth information is utilized. Conversely, relying solely on RGB data is more effective for recognizing object classes with high variability in depth [202]. As a result, the optimal approach to fusing RGB and depth information remains an open question. By effectively integrating visual cues with detailed geometric structures provided by depth information, the performance of semantic segmentation for indoor and outdoor scenes can be significantly improved [39, 42]. How to better combine RGB images with depth information is the key difficulty of RGB-D semantic segmentation. Therefore, over the last decade many approaches have been have proposed. An early approach known as **early fusion** consist of fusing RGB and depth images channels before

extracting features (encoder) and then input the result into the network for learning, as shown in figure Fig. 5.1(a).

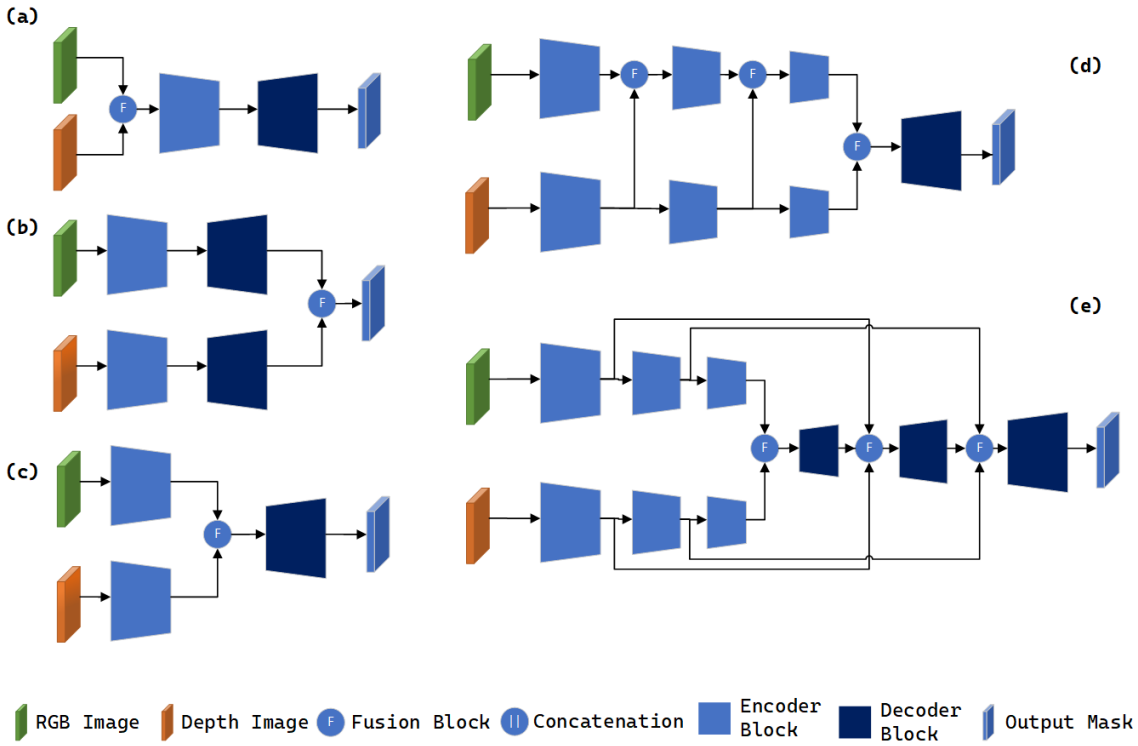
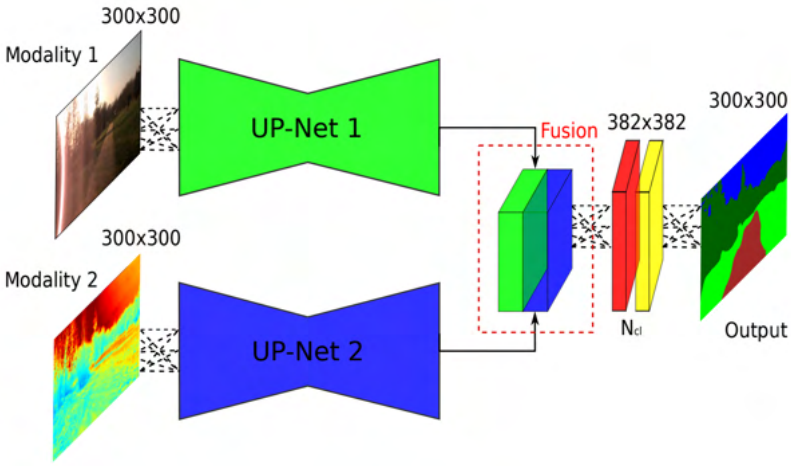


Fig. 5.1 Different existing architectures for RGB-D semantic segmentation. The Fusion Block can be concatenation, element wise addition or more complex transformation. **(a)** Early fusion, **(b)(c)** Late fusion, **(d)(e)** Multistage fusion.

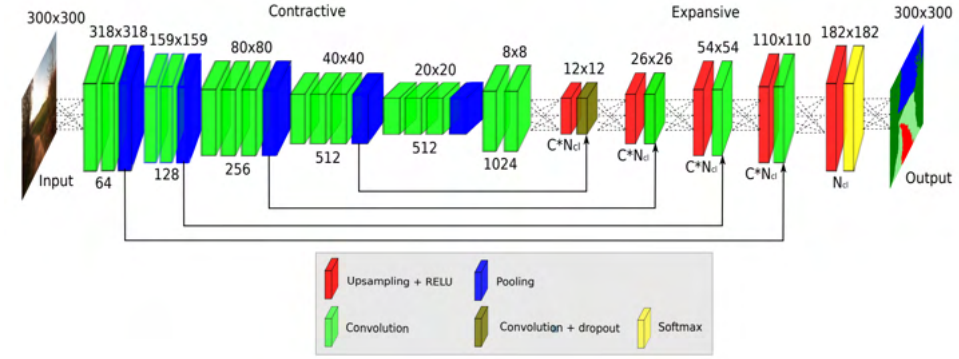
This approach is used by [202] which extended a multi-scale RGB CNN architecture to accommodate RGB-D data. They simply concatenated color and depth images channels to build a four dimension input image for the model. However, such simple stack of RGB and depth images is not efficient as it ignores the complementarity between RGB and depth information [42, 39].

In the **late fusion** approach depicted in Fig. 5.1(b), the RGB and depth images are processed separately through their respective encoder and decoder branches. The two predictions are then fused to produce the final segmentation mask. This method was utilized by [2], where a network with two distinct streams was constructed to individually learn the segmentation of RGB and depth images. The predictions from each branch were subsequently fused to generate the final semantic mask, as illustrated in Fig. 5.2. Where UpNet architecture shown in Fig. 5.3 is build on a contraction and expansion principle. VGG16 architecture is used as basis on the contraction side and the expansion side



Source : Vavalda et al. [2]

Fig. 5.2 Late-Fusion Convolution architecture proposed by Valada et al. [2]



Source : Guo et al. [2]

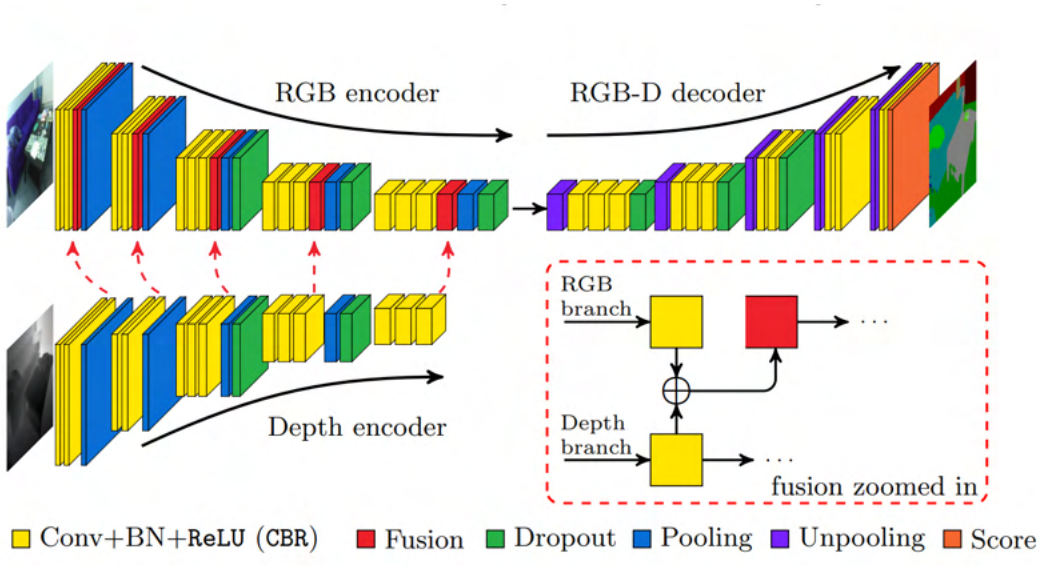
Fig. 5.3 UpNet architecture with up-convolutional layers of size $C \times N_{cl}$, where N_{cl} is the number of classes and C is a scalar factor of filter augmentations.

consists of five up-convolutional refinement segments. Each up-convolutional refinement is composed of one up-sampling layer followed by a convolution layer and a ReLU activation layer. They replace the last fully connected layer by a 3x3 convolutions with 1024 filters.

In another variation of the late fusion approach, the two branches for RGB and depth features share a common decoder. The RGB and depth features are extracted by their respective encoders, fused together, and then passed to the shared decoder for the final segmentation result, as shown in Fig. 5.1(c). The **late fusion** strategy allows each branch to specialize in extracting modality-specific features : color and texture from RGB images, and geometric, illumination-independent features from depth images. However the main drawback of the late fusion method is that it primarily fuses high-level features, potentially overlooking the importance of low-level features, which may contain crucial information

that can improve the segmentation process. Typically, low-level features include edges, textures, and simple patterns, while high-level features capture more complex patterns such as shapes, objects, and semantic concepts.

Hazirbas et al. [3] suggests that segmentation performance can be enhanced by fusing features at multiple stages rather than focusing solely on low-level (early fusion) or high-level features (late fusion). Typically, RGB and depth features are fused at various scales during the encoder or decoder stages. This **multistage fusion** strategy allows the model to capture fine-grained details as well as broader contextual information by merging features at different scales. Two main variants of multistage fusion are observed in the literature. The first approach involves integrating depth features into the RGB encoder by fusing feature maps at different stages, as shown in Fig. 5.1(d). This method is based on the concept that enriched semantic RGB features can be further enhanced with additional depth information and forms the basis of models such as FuseNet [3] and RedNet [203]. The second variant involves combining RGB and depth features extracted at different scales and then passing the fused result to a common decoder, as illustrated in Fig. 5.1(e). This approach is utilized in models such as RDFNet [4] and SSMA [204].

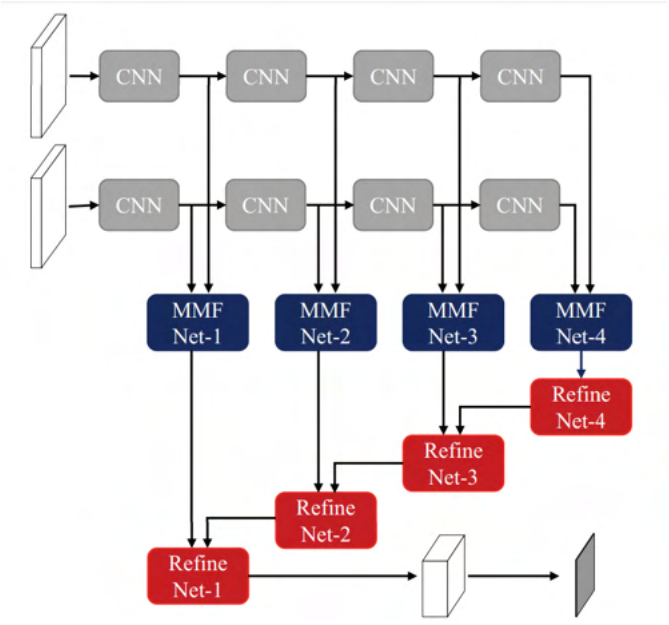


Source : Hazirbas et al. [3]

Fig. 5.4 The architecture of FuseNet model, proposed by Hazirbas et al [3]. Colors indicate the layer type. The network contains two branches to extract features from RGB and depth images, and the feature maps from depth is constantly fused into the RGB branch, denoted with the red arrows. In our architecture, the fusion layer is implemented as an element-wise summation, demonstrated in the dashed box.

The first variant of the multistage fusion approach is employed by FuseNet [3]

illustrated in Fig. 5.4. It is an encoder-decoder network that consists of two parallel branches, which simultaneously extract features from RGB and depth images. As the network goes deeper, it progressively fuses the depth features into the RGB feature maps, enhancing the representation at each stage. The encoder branches are based on the VGG16 network architecture, excluding the fully connected layers. Feature maps from the RGB and depth encoders are fused at different stages using summation, with the resulting score maps being fed back into the RGB branch, as illustrated in Fig. 5.4. The decoder is a counterpart of the encoder and employs memorized unpooling to upsample the feature maps. However, a limitation of this approach is that simple summation of RGB and depth features may not fully capture the complex correlations between RGB and depth information resulting in lower accuracy compared to the state-of-the-art RGB-only CNN architecture [4]. Additionally, the unpooling operation used in the decoder is less efficient compared to alternative methods like bilinear interpolation.

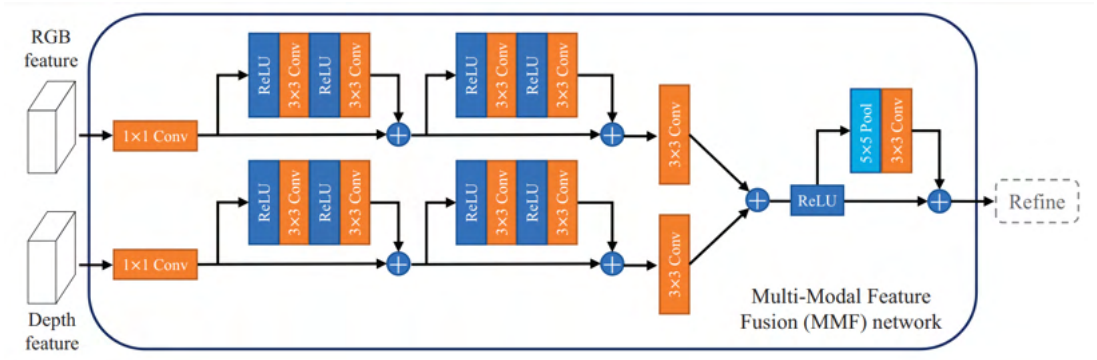


Source : Park et al. [4]

Fig. 5.5 The architecture of RDFNet model [4] for RGB-D semantic segmentation. The network initially fuses multimodal features using a block called MMFNet, and then enhances the fused features through a sequence of RefineNet blocks.

RDFNet proposed by Park et al.[4], adopted the second variant of multistage fusion, shown in Fig. 5.1(e). RDFNet which architecture is illustrated in Fig. 5.5, consists of two encoder branches for extracting RGB and depth features, both using a ResNet model as the backbone. It introduces a Multi-Modal Feature Fusion network (MMFNet) to fuse the

RGB and depth features output by each stage of the ResNet backbone. In MMFNet, each feature map undergoes a 1x1 convolution, followed by two residual convolution units, and a 3x3 convolution. The features from the two branches are then fused by summation and passed through a residual pooling operation, which incorporates contextual information into the fused feature. The outputs of MMFNet are subsequently refined using multi-level fusion through RefineNet blocks, as proposed by [205].

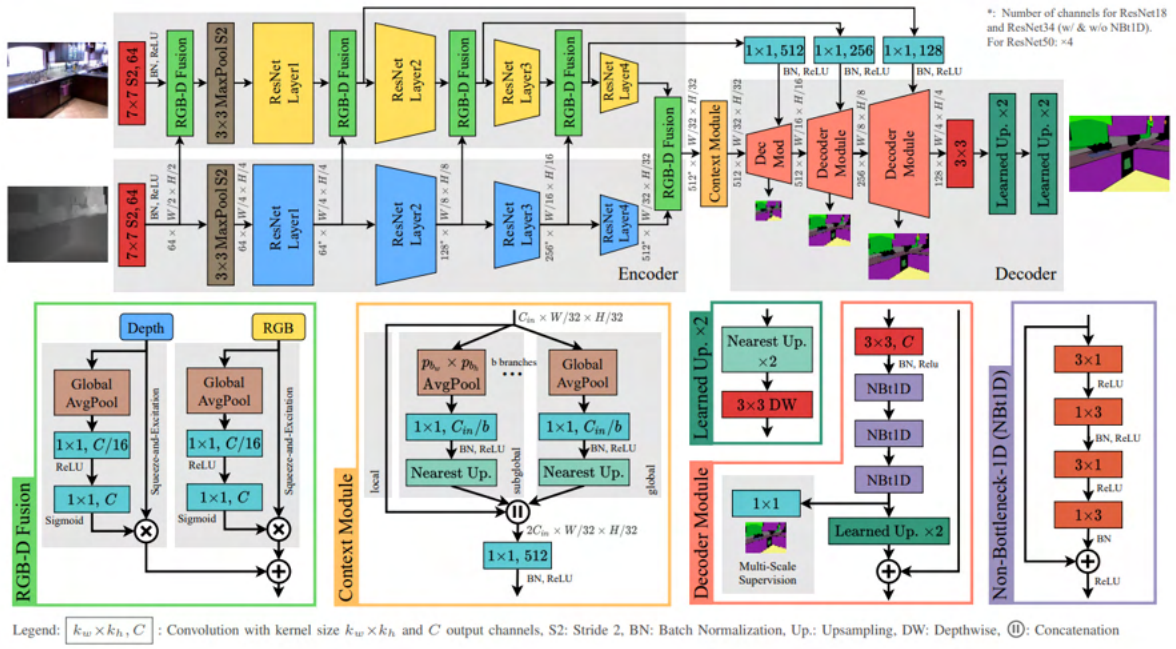


Source : Park et al. [4]

Fig. 5.6 Diagram of our multi-modal feature fusion network (MMNFNet) used in RDFNet model [4].

Other work such as ESANet [5] which architecture is shown in Fig. 5.7, adopted the two variants of **multistage fusion** at the same time, integrating depth feature in RGB encoder at different scale and refining fused feature multi-level fusion. Both the RGB and depth encoders use a ResNet architecture as their backbone. At each of the five resolution stages in the encoders, depth features are fused into the RGB encoder. Before fusion, the features from both modalities are re-weighted using a Squeeze and Excitation (SE) attention module and then combined through element-wise summation, as illustrated in Fig. 5.7(light green). The SE attention mechanism enables the model to learn which features from each modality to emphasize or suppress based on the input, thereby improving the effectiveness of the fusion process.

However, RGB-D semantic segmentation introduces significant parameters and computational complexity compared to its RGB counterpart. Although the most widespread existing models in the literature mainly use a heavy backbones architecture such as the ResNet and VGG models for the RGB and Depth branches. Consequently, our objective in this work is to propose an efficient network architecture tailored to reduce both the number of FLOPs and parameters enabling faster inference while maintaining robust segmentation performance.



Source : Seichter et al. [5]

Fig. 5.7 Overview of our proposed ESANet[5] for RGB-D segmentation (top) and specific network parts (bottom).

5.2.2 Monocular Depth Estimation

Depth plays a key role in improving the understanding of a scene's spatial structure across various applications. While depth cameras are becoming increasingly popular, depth estimation remains a highly valuable approach that continues to attract significant interest from researchers for several reasons. Using a depth estimation model instead of a depth camera offers several advantages, particularly in terms of cost, flexibility, and scalability. Depth cameras like LiDAR or Time-of-Flight sensors can be expensive, power-intensive, and limited operating conditions[206], while depth estimation models rely on standard RGB cameras, which are far more affordable and widely available. Additionally, depth estimation models are more adaptable in various environments, working well in challenging lighting conditions and avoiding the interference issues common with active depth sensors.

Monocular depth estimation (MDE), binocular depth estimation (BDE), and multi-view depth estimation (MVDE) are different methods for inferring depth information in computer vision [207]. BDE, uses two images from a stereo camera setup, mimicking human binocular vision. The disparity between the two images allows direct calculation of depth, but it requires specific hardware alignment. MVDE uses multiple images captured

from different viewpoints, such as from a moving camera or multiple cameras, to estimate depth. MDE on the other hand, uses a single RGB image to predict depth, relying entirely on learned features from the image to infer 3D structure. It is challenging because a single image lacks direct depth cues, but it is highly versatile and applicable in any scenario where only one camera is available. Despite its challenges, MDE is becoming increasingly popular, and achieved significant success across a wide range of applications, including robotics, assisted surgery, virtual reality, autonomous driving, and more [207, 208]. Modern supervised deep learning methods have the advantage to achieve high accuracy when trained on richly annotated datasets. Moreover with supervised method there is no-need of additional assumptions about the input data like with self-supervised methods[208]. However, acquiring large-scale labeled datasets with pixel-level depth annotations can be time-consuming and expensive.

Over the years, various datasets have been create for MDE, consisting of RGB images paired with corresponding depth annotations. These datasets vary in terms of the environments and objects they capture (e.g., indoor/outdoor scenes, dynamic objects), the type of depth annotations provided (sparse/dense, absolute/relative depth), the method of obtaining depth (laser, time-of-flight, SfM, stereo, human annotation, synthetic data), image quality, camera settings, and overall size [206]. Each dataset comes with its unique attributes, biases, and challenges. Training a model on a single dataset typically results in strong performance on that dataset's test split, due to consistent camera parameters, depth annotations, and environments, but may limit the model's ability to generalize to new data with different characteristics [206]. Therefore, despite the promising performance of early MDE deep learning models, they are hard to generalize to unseen domains [6].

To develop a robust model capable of generalizing across diverse scenarios, it is essential to have training data that reflects the diversity of the visual world. The main challenge lies in acquiring such data at a sufficient scale. Ranftl et al. [206] was one of the first to introduce an efficient solution by training models on a collection of diverse datasets, demonstrating that this approach significantly enhances generalization when tested on previously unseen datasets with different characteristics. Subsequent works [206, 209, 210, 6] have reinforced this approach, demonstrating that training MDE models on a wide range of diverse and mixed datasets with varying characteristics significantly enhances the models' ability to predict depth for any unseen RGB image—a capability known as *zero-shot* depth estimation.

Zero-shot cross-dataset transfer refers to the ability of a machine learning model, typically trained on a set of specific datasets, to generalize and perform well on new, unseen datasets without requiring additional training or fine-tuning. In the context of monocular

depth estimation, this means the model is trained on a diverse range of datasets, and it can then estimate depth from RGB images in entirely different datasets or environments that were not part of the training data. This capability hinges on the model's ability to capture general features and patterns from the training data that are applicable across a wide variety of scenes and conditions, enabling it to adapt to new visual inputs in a zero-shot manner—i.e., without any prior exposure to the new data during training.

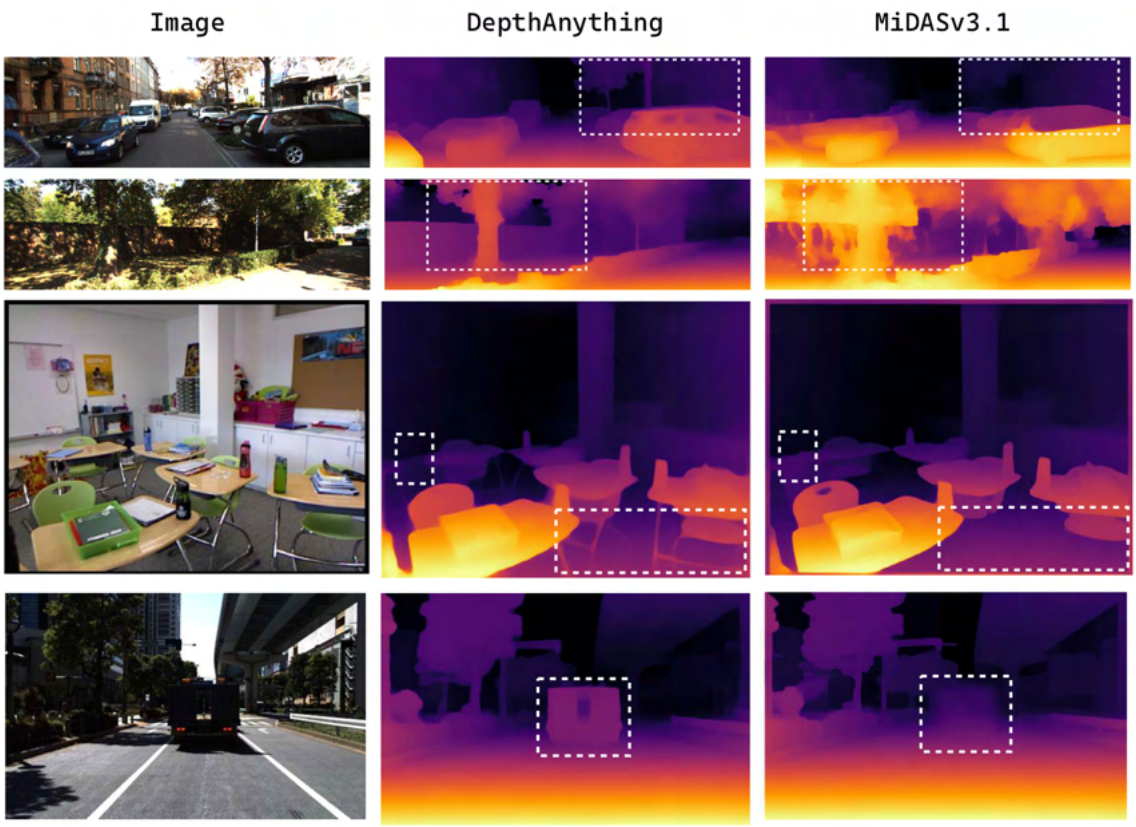
To enable effective multi-dataset joint training, a milestone work MiDaS utilizes an affine-invariant loss to ignore the potentially different depth scales and shifts across varying datasets. Thus, MiDaS v1[206] provides *relative depth* information which is a sense of which objects are closer or farther away without specific distances. In contrast *metric depth* refers to the actual, physical distance between the camera and objects in a scene, expressed in standard units of measurement, such as meters or centimeters. While MiDAS v1 relies on CNN backbones, MiDAS v3.1 [210] adopts a pretrained BEiT model, inspired by the success of transformers in computer vision. Moreover MiDAS v3.1 is trained in more data than previous version, mixing additional datasets to obtain 1.4M labelled images. More recently in 2024, Yang et al.[6] proposed DepthAnything model which unleash the power of large-scale unlabeled data. To this end, they scale up the dataset by designing a data engine to collect and automatically annotate large-scale unlabeled data, which significantly enlarges the data coverage and thus is able to reduce the generalization error. DepthAnything is trained with 1.5M labeled images from six public datasets and 62M unlabeled images from eight large-scale public datasets. This approach significantly expands data coverage, reducing generalization errors. DepthAnything's main advantage lies in its extensive and varied training set.

At the time of this research MiDASv3.1 [210] and DepthAnything [6] respectivelly support by Intel Labs and TikTok are the best MDE model with zero-shot relative depth estimation performance. Based on reported results by [6] show in Fig.5.8 and our experimentation on CamerFood15 dataset, DepthAnything demonstrate the best generalizability performance.

Method	Encoder	KITTI [18]		NYUv2 [54]		Sintel [7]		DDAD [20]		ETH3D [51]		DIODE [59]	
		AbsRel	δ_1	AbsRel	δ_1	AbsRel	δ_1	AbsRel	δ_1	AbsRel	δ_1	AbsRel	δ_1
MiDaS v3.1 [5]	ViT-L	0.127	0.850	0.048	0.980	0.587	0.699	0.251	0.766	0.139	0.867	0.075	0.942
Depth Anything	ViT-S	0.080	0.936	0.053	0.972	0.464	0.739	0.247	0.768	0.127	**0.885**	0.076	0.939
	ViT-B	0.080	0.939	0.046	0.979	**0.432**	0.756	0.232	0.786	**0.126**	0.884	0.069	0.946
	ViT-L	**0.076**	**0.947**	**0.043**	**0.981**	0.458	**0.760**	**0.230**	**0.789**	0.127	0.882	**0.066**	**0.952**

Source : Yang et al. [6]

Fig. 5.8 **Zero-shot relative depth estimation**. DepthAnything is compared to the best model from MiDaS v3.1. Note that MiDaS does not strictly follow the zero-shot evaluation on KITTI and NYUv2, because it uses their training images. Yang et al. [6] provide three model scales for different purposes, based on ViT-S (24.8M), ViT-B (97.5M), and ViT-L (335.3M), respectively. Better : AbsRel ↓, $\delta 1$ ↑. **Best**, second best results.



Source : Yang et al. [6]

Fig. 5.9 Qualitative comparison between the MiDASv3.1 and DepthAnything[6] models on zero-shot relative depth estimation of few sample image from various context. The brighter color denotes the closer distance.

5.3 Multi-modal Image Segmentation Applied to African Food Image

In this section, we introduce a novel model architecture (see Fig. 5.10), designed for RGB-D semantic image segmentation. Our model, referred to as **ESeNet-D** (**E**fficient **Se**gmentation **Net**work), features separate encoder branches for RGB and depth images, multiple fusion blocks, and a shared decoder.

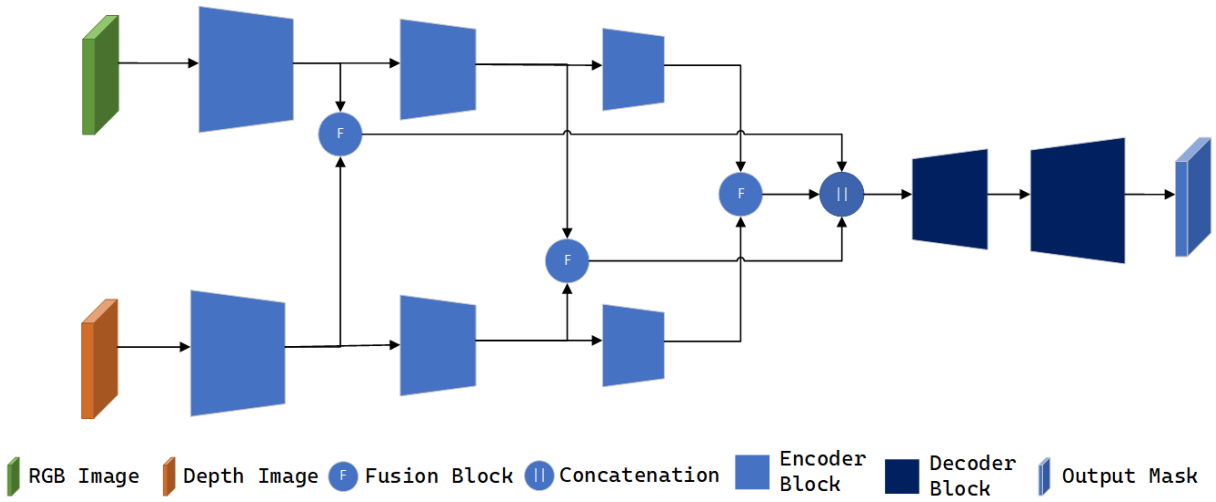


Fig. 5.10 ESeNet-D architecture overview.

5.3.1 Model Architecture

The detailed architecture of our ESeNet-D model is presented in Fig. 5.11. It can be divided into two parts : Encoder and Decoder. Our model is based on the principle of aggregating different feature map from low to high level. This principle has already proven its effectiveness with semantic segmentation models such as mid-DeepLabv3+ [154], DeepLabv3+[45], SegFormer[211], or AdapNet++[2], to name a few. Our model's encoder has two branches : one for RGB and one for depth image. Each branch extracts three layers of features at different resolutions, which are fused with the corresponding layers from the other branch. These fused layers are passed to the decoder to classify each pixel. A comprehensive description of the individual components of our model, along with the motivations behind each design choice, is provided in the following sections.

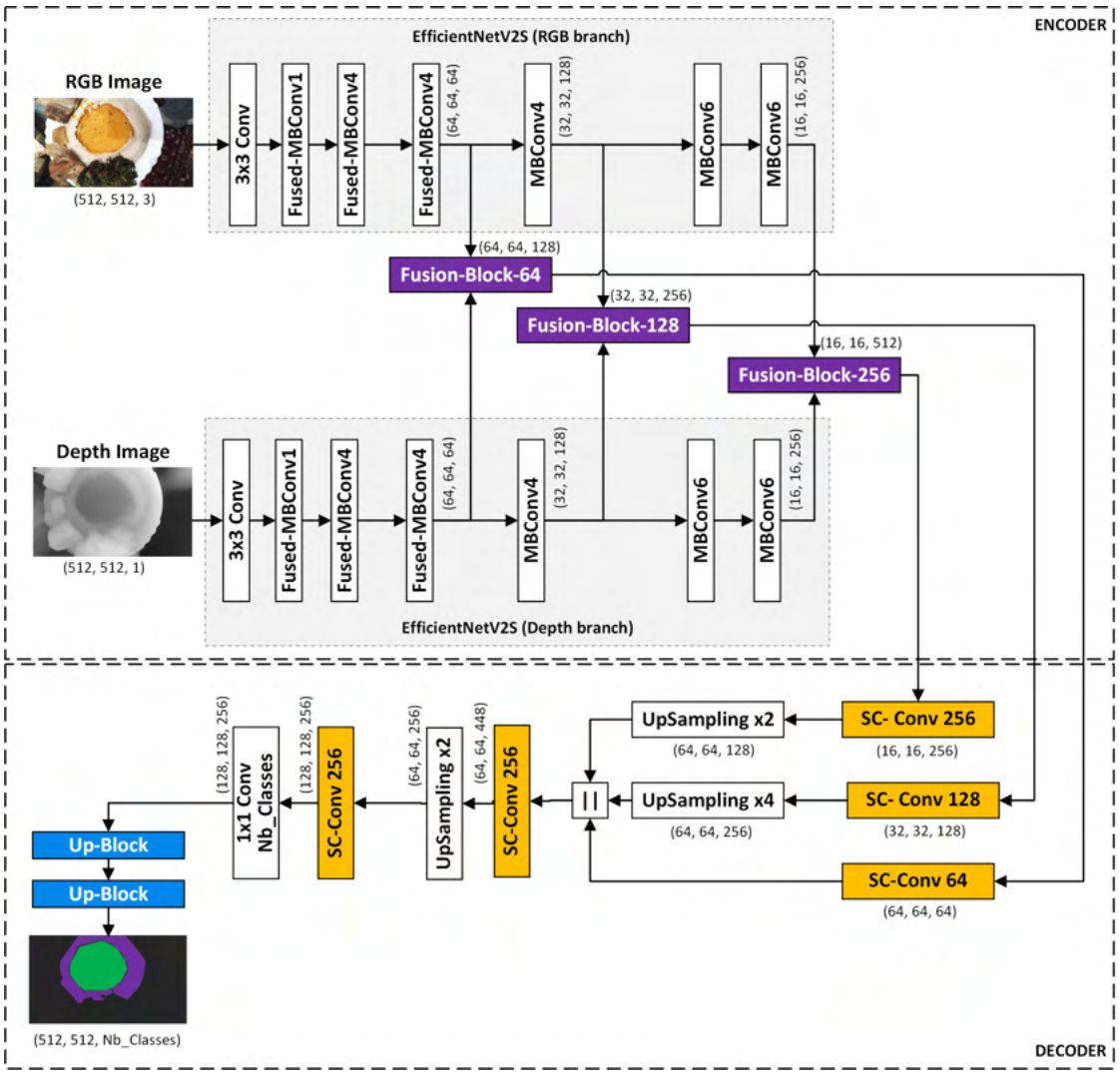


Fig. 5.11 Detailled architecture of our proposed model ESeNet-D for RGB-D segmentation. || stands for a Concatenation Layer.

5.3.1.1 Encoder

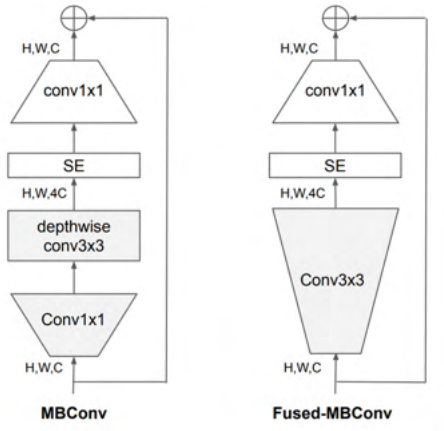
Like some benchmark models as FuseNet [3], RedNet [203], RdfNet [4], SSMA [204], we adopt an encoder with two branches using the same base model as backbone. Assuming efficient feature extraction is crucial for good segmentation, we prioritized optimizing performance and parameter trade-offs for our encoder path. Therefore we choose EfficientNetV2 [8] as backbone, recognized for its superior performance on ImageNet ILSVRC2012 compared to popular classification models such as VGG, ResNet, ResNeSt, and DeiT/ViT. The table Tab.5.1 show parameters and top1-accuracy comparison with some classification models. EfficientNetV2S achieves 83.9% top-1 accuracy on ImageNet, outperforming ViT by 2.0% and training faster with the same computing resources.

Compared to most image classification benchmark models, it performs better in learning speed, inference time, and accuracy, balancing efficiency with the number of parameters and FLOPs [8]. EfficientNetV2S features a 3x3 convolution, two Fused-MBConv, and three MBConv blocks. The architecture of EfficientNetV2S model and the different blocks Fused-MBConv and MBConv is presented in Fig.5.12.

Our encoder has two branches for RGB and Depth feature extraction using EfficientNetV2S models without the classifier block, ie., the last 1x1 convolution, the pooling and fully-connected layers. The RGB branch processes the RGB image, while the Depth branch uses a concatenated depth image for a three-channel input. Following models like [212, 204, 154, 86], we extract feature layers at different scales (64, 64, 64), (32, 32, 128) and (16, 16, 256). This correspond to EfficientNetV2S blocks Fused-MBConv4, MBConv4, and MBConv6. MBConv6 is the last block of the model before the top classification layer. These features are then fused and passed to the decoder.

Tab. 5.1 Comparison between EfficientNetV2S and other benchmark classification models [8, 9] in terms of top-1 accuracy and number of parameters (in million).

	EfficientNet2VS	**ResNet50V2**	**VGG16**	**DeiT/ViT**
Top1 Acc	83.9%	76.0%	71.3%	83.1%
Params	21.6M	25.6M	138.4M	86M



EfficientNetV2-S architecture

Stage	Operator	Stride	#Channels	#Layers
0	Conv3x3	2	24	1
1	Fused-MBConv1, k3x3	1	24	2
2	Fused-MBConv4, k3x3	2	48	4
3	Fused-MBConv4, k3x3	2	64	4
4	MBConv4, k3x3, SE0.25	2	128	6
5	MBConv6, k3x3, SE0.25	1	160	9
6	MBConv6, k3x3, SE0.25	2	256	15
7	Conv1x1 & Pooling & FC	-	1280	1

Source : Tan et al. [8]

Fig. 5.12 Description of EfficientNetV2-S and the architecture of MBConv and FusedMBConv blocks, where SE refers to the Squeeze-and-Excitation attention module [7].

5.3.1.2 Fusion-Block

In the encoder, we extract three RGB and depth feature layers of equal size. Each pair of feature maps is adaptively fused and recalibrated using a fusion block as shown in Fig. 1.2. The Fusion-Block concatenates the two modalities and uses an attention mechanism to re-weight features based on their informative importance. It aims to model the correlation between modality-specific feature maps before fusion, allowing the network to emphasize informative features from one modality and suppress less informative ones from the other.

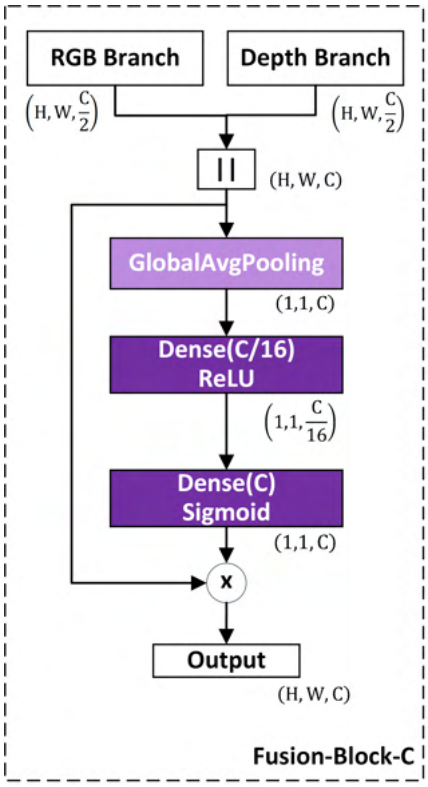


Fig. 5.13 Overview of our **Fusion block**. || is a Concatenation Layer, BN+ReLU implies a Batch-Normalization Layer follow by a ReLU activation Layer. N is a parameter representing the number of channels of input feature map.

The structure of the Fusion-Block is shown in Fig. 5.13. Let $X^{rgb} \in \mathbb{R}^{H \times W \times C/2}$ and $X^{depth} \in \mathbb{R}^{H \times W \times C/2}$ denote respectively features maps from RGB and $Depth$ modalities, where C is the number of features channels and $H \times W$ is the spatial dimension. First we concatenate the two modalities feature maps across the channels axis to yield $X \in \mathbb{R}^{H \times W \times C}$. We then adaptively recalibrate the fused feature map X by computing a scale map s. This scale map is used to recalibrate X via channel-wise multiplication, selectively enhancing important features and suppressing less useful ones. The scales map

s is evaluate through a two-step process. First, global average pooling (**GAP**) is applied to each channel, summarizing the global spatial information into a channel descriptor. This reduces the spatial dimensions, effectively compressing the information and capturing the global context of the feature map. Secondly, we utilizes two fully connected layers $\mathbf{W}_1$ and $\mathbf{W}_2$ with a ReLU (δ) activation in between, and then a sigmoid (σ) activation to generate the channel weights. This produces a vector $\mathbf{s} \in \mathbb{R}^{1\times1\times C}$, where each element is a value between 0 and 1 representing the importance of the corresponding channel. Finally, we recalibrate the feature map X by channel-wise multiplication with the vector $\mathbf{s}$. Each channel of X is scaled by the corresponding element of $\mathbf{s}$. We then obtain a feature map $Y \in \mathbb{R}^{H\times W\times C}$, where each channel is recalibrated according to its importance. As described above, the merged map Y is obtained according to the following equations :

$$X = [X^{rgb}, X^{depth}] \tag{5.1a}$$
$$z = \mathbf{GAP}(X) \tag{5.1b}$$
$$s = \sigma(\mathbf{W}_2 \odot \delta(\mathbf{W}_1 z)) \tag{5.1c}$$
$$Y = X \odot s \tag{5.1d}$$

where $\odot$ is the Hadamard operator.

5.3.1.3 Self-Calibrated Convolution Block

In the decoder path of our network architecture, we use self-calibrated convolution (SC-Conv) instead of a standard convolution layer. Introduced in [213], SC-Conv enhances feature extraction by enlarging the receptive field at each spatial location.

The fundamental idea behind SC-Conv is to introduce a self-calibration mechanism that allows the network to adjust its feature representations adaptively based on the input data. SC-Conv introduces several significant advantages over conventional convolutional layers. Firstly, it allows each spatial location to adaptively consider its surrounding context and model inter-channel dependencies, effectively enlarging the field-of-view. Secondly, by focusing on the context around each spatial location, SC-Conv reduces the risk of incorporating irrelevant information from distant regions, thus improving the accuracy of target object localization. Lastly, SC-Conv encodes multi-scale information, which is crucial for tasks like object detection that require recognition across various scales. This multi-scale capability significantly enhances the network's ability to detect and distinguish objects of different sizes, leading to improved performance in object detection, image segmentation and related tasks [213, 214, 215, 216, 217]. SC-Conv allows the

network to dynamically adjust its filters based on the input, making it more adaptable to various local patterns and contexts within an image. By recalibrating features based on their context, the network learns more nuanced and powerful representations. The self-calibration mechanism can be implemented in various ways, offering flexibility in feature adjustment and combination.

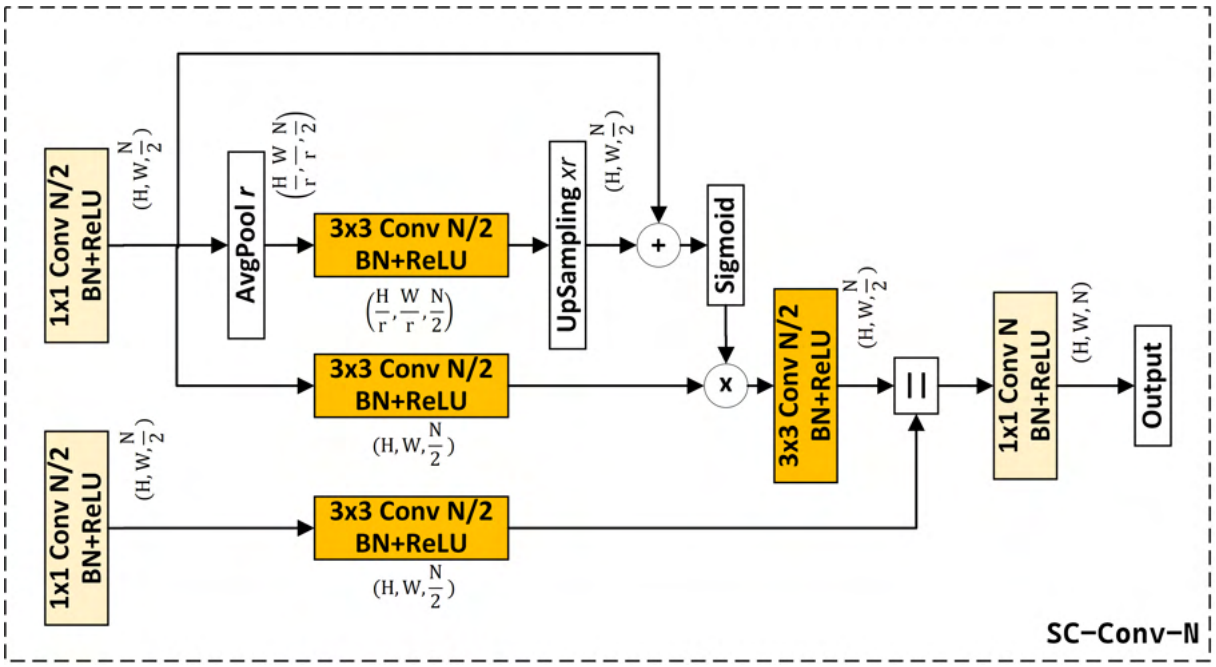


Fig. 5.14 Overview of our **Self-Calibrated Convolution (SC-Conv) block**. || is a Concatenation Layer, BN+ReLU implies a Batch-Normalization Layer follow by a ReLU activation Layer. N is a parameter representing the number of channels of input feature map.

The SC-Conv block used in this work is illustrated in Fig. 5.14. It has a dual-branch structure : the first branch that the calibration operation that generates a calibrated map and the second that performs standard convolution. Consider $X \in \mathbb{R}^{H \times W \times C}$ the input feature map $Y \in \mathbb{R}^{H \times W \times N}$ and the output of a a SC-Conv with N filters. The workflow of SC-Conv computation is the detailed below in Eq.5.2.

Note that for notation convenience, we omit the batch-normalization and the ReLU activation function after each convolution as present in the Fusion-Block architecture in Fig. 1.2.

At the beginning of the calibration branch, the feature map X is passed to a pointwise convolution to obtain a feature map $X_1 \in \mathbb{R}^{H \times W \times \frac{N}{2}}$. Then we use an average pooling operation $\mathbf{AvgPool}_r$ (where r is the filter size and stride) to reduce the input size and enlarge the receptive field. The output $T \in \mathbb{R}^{\frac{H}{r} \times \frac{W}{r} \times \frac{N}{2}}$ of the pooling operation

is then passed through a convolution to model the channel relationship and a 2D up-sampling bilinear interpolation operator $\mathbf{Up}_{\times r}$ to scale to the original feature space and obtain $K_1 \in \mathbb{R}^{H\times W\times \frac{N}{2}}$. Following the work of Liu et al. [213], we adopt $r = 4$. Next, calibration operation ouptut $K_2 \in \mathbb{R}^{H\times W\times \frac{N}{2}}$ as formulated in Eq.5.2e. K_2 is then passed to a convolution with $\frac{N}{2}$ filters to output the final feature map $Y_1 \in \mathbb{R}^{H\times W\times \frac{N}{2}}$ of the first pathway. For the second pathway, the output $Y_2 \in \mathbb{R}^{H\times W\times \frac{N}{2}}$ is calculated using a pointwise convolution follow by a 3×3 convolution each with $\frac{N}{2}$ filters. Finally the output $Y \in \mathbb{R}^{H\times W\times N}$ of the SC-Conv is compute by the concatenation of the outputs Y_1 and Y_2 of the two pathways follow by a pointwise convolution with N filters to fit with the output shape as if we use a standard convolution.

$$X_1 = \mathbf{Conv}_{1\times 1}(X) \tag{5.2a}$$
$$T = \mathbf{AvgPool}_r(X_1) \tag{5.2b}$$
$$K_1 = \mathbf{Up}_{\times r}\left(\mathbf{Conv}_{3\times 3}(T)\right) \tag{5.2c}$$
$$K_2 = \mathbf{Conv}_{3\times 3}(X_1) \odot \sigma(K_1 + X_1) \tag{5.2d}$$
$$Y_1 = \mathbf{Conv}_{3\times 3}(K_2) \tag{5.2e}$$
$$Y_2 = \mathbf{Conv}_{3\times 3}\left(\mathbf{Conv}_{1\times 1}(X)\right) \tag{5.2f}$$
$$Y = \mathbf{Conv}_{1\times 1}([Y_1, Y_2]). \tag{5.2g}$$

The Sc-Conv presented in [213, 217] follows the principle of grouped convolutions, the input feature map is split into multiple portions, yet differently, each portion is not equally treated but responsible for a specific functionality. In fact, the input feature map $X \in \mathbb{R}^{H\times W\times C}$ is split into two blocks $X_1 \in \mathbb{R}^{H\times W\times \frac{C}{2}}$ and $X_2 \in \mathbb{R}^{H\times W\times \frac{C}{2}}$. X_1 is used for calibration and X_2 for the standard convolution. The calibration branch then use only half of the input channels. Our architecture is slightly different : we use two 1×1 convolutions with $\frac{N}{2}$ filters, providing two advantages. Firstly, it creates a more flexible architecture that easily adapts to the desired number of convolution output channels. Secondly, recalibration is performed using the entire input feature map, which improves model performance.

5.3.1.4 Learned Up-sampling (Up-Block)

In our architecture we adopt a learned interpolation instead of simple interpolation operator.Learned upsampling, outperforms standard interpolation methods like nearest, bilinear or bicubic interpolation by learning optimal ways to reconstruct high-resolution features from low-resolution inputs. While standard interpolation methods apply fixed

mathematical formulas to resize images, learned interpolation adaptively refine details during training. This adaptability allows it to better preserve edges and textures, improving the overall quality of upsampled images in tasks like image segmentation.

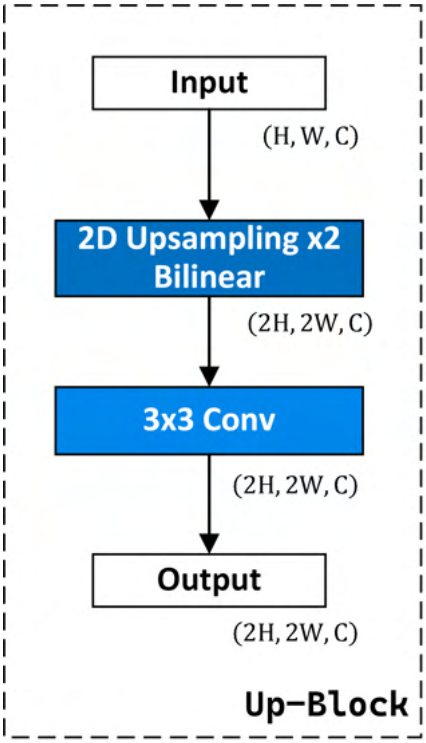


Fig. 5.15 Overview of our **Learned Up-sampling block (Up-Block)**.

Our learned interpolation block illustrated in Fig. 5.15 is implemented by combining $\mathbf{Up}_{\times 2}$ a standard bilinear interpolation operator with a factor of 2 and a 3×3 Convolution operation. Consider an input feature map $X \in \mathbb{R}^{H\times W\times C}$, the computational steps of the Up-Block follow (5.3). The initial interpolation provides a coarse up-sampled output with a height and width two times larger $(2H, 2W, C)$, which is then refined by a Convolution layer that learns the optimal way to adjust the up-sampled values. The convolution in the Up-Block matches the input's channel count, ensuring $Y \in \mathbb{R}^{2H\times 2W\times C}$ has the same channels as the input.

$$Y = \mathbf{Conv}_{3\times3}(\mathbf{Up}_{\times2}(X)). \tag{5.3}$$

5.3.1.5 Decoder

As depicted in Fig. 5.11, our encoder produces three branches from the outputs of various Fusion-Blocks, which are then passed to the decoder. Each encoder branch undergoes SC-Conv to halve the channel count, followed by bilinear interpolation to

maintain size consistency. These obtained feature map, representing fused features (RGB and Depth) at different spatial resolution are then concatenated. This multi-scale feature fusion ensures that the network can effectively utilize both fine-grained details and broader context information for accurate segmentation. The result goes through two consecutive SC-Conv of 256 filters with a bilinear upsampling operation in between. Finally, a 1×1 convolution is used for final prediction and two consecutive UP-Blocks are applied to obtain the same size as the input image. We use two upsampling blocks instead of one like many other popular models [45][196][211] to avoid abrupt increase in spatial resolution. Our learned interpolation blocks (Up-Block) will gradually refine the feature maps, capturing finer details and improving the quality of the final predicted mask.

Finally, our decoder architecture differs from other RGB-D architectures, as shown in Fig. 5.1 and Fig. 5.10. Unlike other architectures that rely on a cascade-based processing of fused features from the encoder, our approach adopts a simpler design. We implement a straightforward decoder, inspired by its proven effectiveness in well-known models like SegFormer[211] and DeepLabv3+[45]. This method involves simply concatenating features from different scales extracted in the encoder and processing them in a single step. This approach better leverages the correlations between channels at varying resolutions.

5.3.1.6 ESeNet (RGB model)

In this work, we introduce ESeNet, the RGB-only version of our ESeNet-D model. The architecture of ESeNet model is illustrated in Fig. 5.16. It is similar to ESeNet-D but excludes the Depth feature extraction branch, making the concatenation layer in the Fusion-Block no more useful. The primary purpose of ESeNet is to evaluate the influence of the Depth branch on the overall segmentation performance.

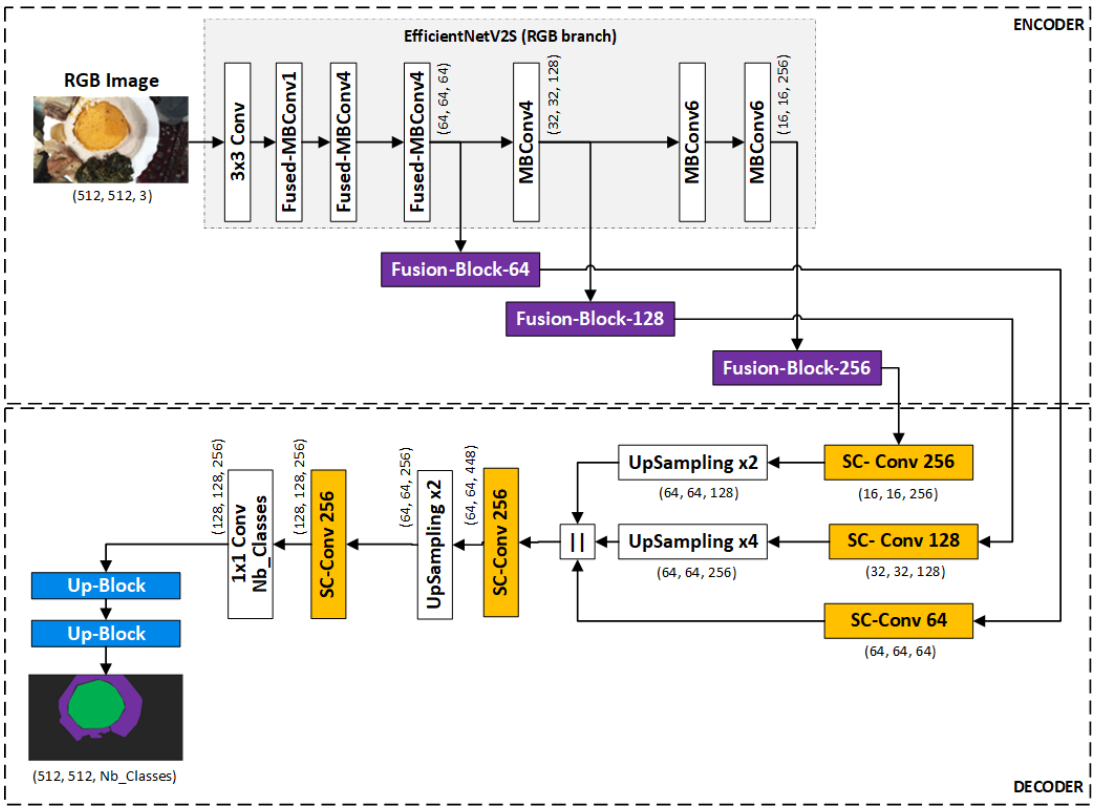


Fig. 5.16 ESeNet architecture. RGB version of ESeNet-D model. Here the concatenation layer (||) in the Fusion-Block is no longer needed.

5.3.2 Food Depth Image Acquisition

To address the challenge of obtaining depth maps for food image datasets, we propose leveraging recent advancements in MDE as a practical solution. We used two state-of-the-art models, MiDAS v3.1 [210] and DepthAnything [6], capable of accurately predicting depth maps from RGB images. These models trained on various RGB-D datasets perform particularly well for Zero-shot depth estimation and thus achieved pretty good generalization. We made experiments with depth maps generated by the two MDE models to investigate the influence of depth map quality on semantic segmentation

accuracy. Fig. 5.17 shows some samples of CamerFood15 and MyFood datasets with their estimated depth maps using DepthAnything.

Fig. 5.17 Few samples samples images of CamerFood15 and MyFood dataset with their generated depth map.

5.4 Evaluation Experimentation and Results

5.4.1 Experimental Environment

Our experiments were conducted on the Grid'5000 [198] infrastructure using a node equipped with an NVIDIA A40 GPU (with CUDA 12.2) and AMD EPYC 7413 processor. We implemented our model in TensorFlow, initializing model backbones with ImageNet pretrained weights from the Keras API [9] and other layers using the HeNormal initializer [96]. The training environment used TensorFlow 2.12 and Python 3.10.

We used HuggingFace 4.37 for SegFormer, MiDAS, and DepthAnyThing models. All training was done with same hyperparameters : input image size of $512px \times 512px$, batch size of 4, *AdamW* optimizer with *Weight Decay*$= 5 \times 10^{-4}$, and *Categorical Cross-Entropy* loss function. We used an *Exponential Decay* learning rate schedule with an initial rate of 10^{-4}, training all models for 200 epochs and saving the best iteration for the validation set. Data augmentation, including flips (up, down, left, right), brightness (max=0.3), contrast (min=0.3, max=1.3), and random cropping, was applied randomly to all images, tripling the number of training samples. All datasets and code used in this paper are publicly available here [1].

1. https ://github.com/babanthierry94/ESeNet-D

5.4.2 Evaluation Results and Discussions

In this section, we present the results from various experiments conducted to validate our contributions.

5.4.2.1 How Depth Map Quality Impacts the Results

In this experiment we investigate the impact of depth map accuracy on our model's results by training it with depth maps from MiDASv3.1[210] and DepthAnyThing[6]. As shown in [6] and confirmed by our experiments, DepthAnything is more accurate than MiDAS v3.1. Fig. 5.18 shows estimated relative depth maps of few samples from CamerFood15 dataset obtained with two different MDE models (DepthAnything and MiDAS v3.1). In this figure we have framed the areas where the difference between the two predictions is perceptible with dashed boxes. We can notice that the DepthAnything model provides a more detailed depth map than MiDAS v3.1. Thus, as expected, a more accurate depth map estimation leads to better performance as reported in Tab. 5.2.

Tab. 5.2 Performance using different depth estimation model.

Model	Depth Prediction	PA (%)	mIoU (%)
ESeNet-D	MiDAS v3.1	95.92	85.12
ESeNet-D	**DepthAnything**	**96.61**	**86.76**

5.4.2.2 Contribution of Depth Modality

We evaluate the impact of depth modality on segmentation accuracy by training ESeNet-D and ESeNet (the RGB version of our model) on the CamerFood15 dataset. Comparing ESeNet trained with only RGB images to ESeNet-D trained with RGB and Depth data reveals the importance of the depth data in the performance improvement. Results in Tab. 5.4 show that ESeNet achieves an mIoU of 84.83%, while ESeNet-D gives a +1.93 score improvement, demonstrating the significant role of the depth information in enhancing segmentation. We also analyze depth's impact on class-wise performance (see Tab. 5.7). Despite increasing computational load, depth modality enhances both overall and class-specific segmentation results by providing complementary information.

5.4.2.3 ESeNet-D Block Contribution with CamerFood15 Dataset

In this experiment, we analyzed the contribution of the different fundamental blocks of our model to the final result. We tested multiple combinations as shown in Table 5.3. Each

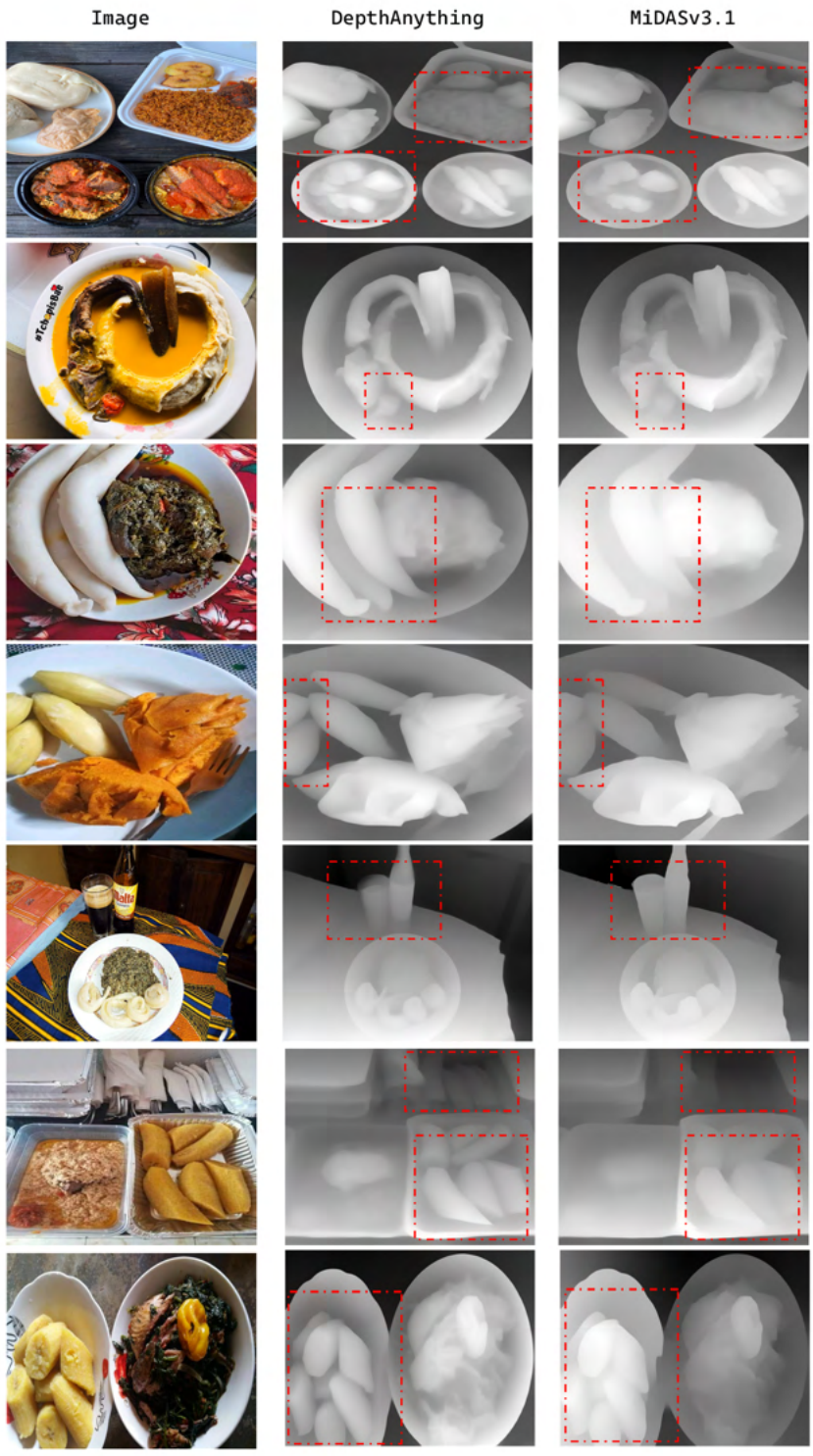


Fig. 5.18 Qualitative comparison between the MiDASv3.1 and DepthAnyThing models on zero-shot relative depth estimation of some image from the CamerFood15 dataset.

of our proposed blocks was replaced by its standard equivalent to investigate how each of our design choice improves the result. Fusion-Block, SC-Conv and Up-Block blocks have been respectively replaced by Concatenation, 3×3 Convolution and 2D bilinear UpSampling. From these results we can derive the conclusion that simply concatenating the features of RGB and Depth modalities is not efficient. It may even decrease model performance, as seen by comparing *model 4* and *model 6* in Table 5.3). The Fusion-Block and SC-Conv blocks significantly enhance the model's performance. Models with standard 3×3 convolutions are heavier and have higher computational loads but lower performance compared to those with SC-Conv, as seen by comparing *model 5* and *model 6*. Additionally, the Up-Block improves accuracy over conventional upsampling with a similar number of parameters and a slight increase in FLOPs. The best performance is achieved by combining the Fusion-Block, SC-Conv, and Up-Block (*model 6* in Table 5.3).

Tab. 5.3 Contribution of each fundamental parts in our proposed ESeNet-D model using CamerFood15 dataset.

				Params(M)	FLOPs(B)	Size(MB)	PA(%)	mIoU(%)
1	Concatenation	$\text{Conv}_{3\times3}$	UpSampling	43.18	44.59	164.70	95.77	84.77
2	Fusion-Block	$\text{Conv}_{3\times3}$	UpSampling	43.22	44.59	164.87	95.96	85.48
3	Fusion-Block	SC-Conv	UpSampling	42.58	42.53	162.44	96.25	86.25
4	Concatenation	SC-Conv	Up-Block	42.54	43.18	162.29	95.92	84.53
5	Fusion-Block	$\text{Conv}_{3\times3}$	Up-Block	43.22	45.34	164.89	96.10	85.62
6	**Fusion-Block**	**SC-Conv**	**Up-Block**	**42.59**	**43.19**	**162.46**	**96.61**	**86.76**

5.4.2.4 ESeNet-D Evaluation with CamerFood15 Dataset

In this experiment, we compare the performances of our model ESeNet-D with popular RGB and RGB-D models. the obtained results are presented in Tab. 5.4). ESeNet-D achieves superior results compared to competitors. It has better computational load than other RGB-D models and even RGB models except our previous contribution in this thesis, mid-DeepLabv3+ (ResNet50). We observed that, models using less efficient backbones like VGG and ResNet struggle to perform well. Transformer-based models like SegFormer show good results but exhibit high GPU usage during training. Our model, compared to SegFormer, has fewer parameters, smaller size, and lower FLOPs with better results. Additionally, RGB-D models in the literature generally show lower performance on the CamerFood15 dataset, potentially due to the fact that there are designed for scene segmentation with very high intra-class color and texture variation. In this type of data, it is not possible for the models to exploit the colour and texture characteristics correctly.

Moreover, they employ VGG16 and ResNet as encoder backbones which are less efficient than the EfficientNetV2S (see Tab. 5.1) used in ESeNet-D.

Tab. 5.4 ESeNet-D evaluation results and comparison with state-of-the-art models with the CamerFood15 dataset. Best values for each metric are highlighted in green and performances of our proposed models in bold font.

	Model	Backbone	Params(M) ↓	FLOPs(B) ↓	Size(MB) ↓	PA(%) ↑	mIoU(%) ↑
RGB	FCN-8s [60]	VGG16	134.34	110.88	537.40	93.92	76.44
	UNet [61]	VGG16	37.46	223.09	149.90	93.53	75.46
	mid-DeepLabv3+	ResNet101	31.04	62.30	118.43	95.53	82.72
	DeepLabv3+ [45]	ResNet50	40.44	73.30	154.25	95.14	81.60
	DeepLabv3+ [45]	ResNet101	59.51	92.71	227.00	95.44	82.88
	GourmetNet [59]	ResNet101	47.36	104.85	180.68	95.31	82.95
	DANet [195]	ResNet101	67.06	78.19	255.80	95.09	80.71
	EANet [196]	ResNet101	53.88	62.55	205.52	95.22	81.35
	SegFormer [211]	MiT-B4	64.01	95.70	245.30	95.55	84.15
	ESeNet (Ours)	**EffNetV2S**	**22.47**	**28.27**	**85.72**	**95.84**	**84.83**
RGB-D	RedNet [203]	2×ResNet50	52.64	53.15	200.81	94.09	76.33
	FuseNet [3]	2×VGG16	44.20	242.63	168.61	94.85	78.34
	ESANet [5]	2×ResNet50	68.45	53.02	261.12	94.63	79.24
	SSMA [204]	2×ResNet50	56.32	99.65	214.84	93.12	74.68
	ESeNet-D (Ours)	**2×EffNetV2S**	**42.59**	**43.19**	**162.46**	**96.61**	**86.76**

The results shown in Tab. 5.4 are visualized in Fig. 5.19, where mIoU is plotted against the number of parameters, model size, and FLOPs. The top-left corner represents a high mIoU (y-axis) and low values for parameters, size, or FLOPs (x-axis, depending on the subplot). Thus, models closest to the top-left corner offer the best balance between performance and computational load. By comparing the proximity of each model point to the top-left corner across different plots, we found that our proposed models, ESeNet-D and ESeNet, are the most efficient in terms of performance relative to computational load, showing a clear improvement over our previous contribution, mid-DeepLabv3+, discussed in the previous chapter.

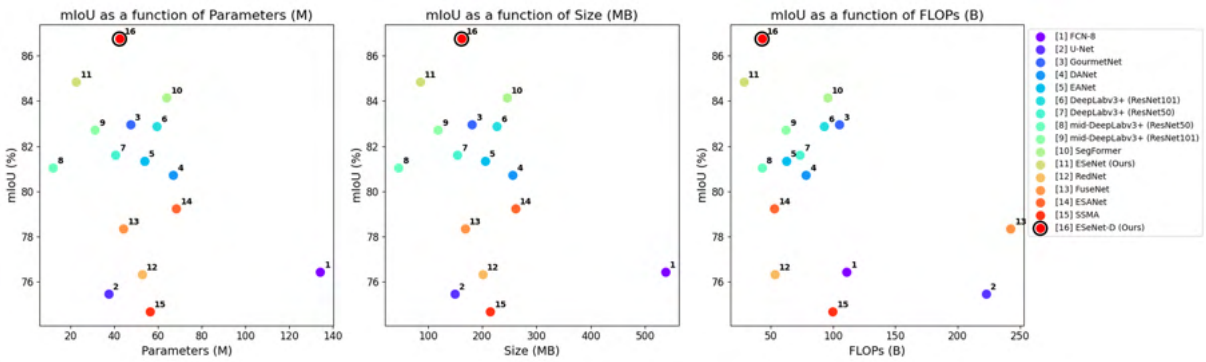


Fig. 5.19 Analysis of ESeNet-D performance and computational load. The best models are those closest to the top left-hand corner. These are the ones with the best ratio of performance to computing load.

5.4.2.5 ESeNet-D Evaluation with MyFood Dataset

Here we evaluate our model on another food image dataset named MyFood [1], and results are shown in Tab. 5.5. These results underscore that model performance can vary depending on the dataset characteristics. Unlike CamerFood15, MyFood exhibits minimal intra-class variation in classes like *apple*, *boiled egg*, *fried egg*, and *rice*, but one class with ambiguous variation adversely affects overall segmentation performance. However, this experiment validates that integrating a depth modality enhances our model's performance.

Tab. 5.5 ESeNet-D evaluation results and comparison with state-of-the-art models with MyFood dataset. Best values for each metric are indicated in bold font. (*) are the obtained results of the paper of Freitas et al. [1]

	Model	Backbone	Params(M) ↓	FLOPs(B) ↓	Size(MB) ↓	PA(%) ↑	mIoU(%) ↑
RGB	mid-DeepLabv3+	ResNet101	29.48	59.88	112.45	93.86	81.69
	DeepLabv3+ [45]	ResNet101	59.51	92.37	226.99	93.82	81.32
	DeepLabv3+ [45]	ResNet50	40.43	72.96	154.25	93.59	81.19
	GourmetNet [59]	ResNet101	47.36	104.51	180.67	93.85	81.99
	DANet [195]	ResNet101	67.05	78.19	255.79	93.57	80.47
	EANet [196]	ResNet101	53.87	62.55	205.51	93.85	82.12
	SegFormer [211]	MiT-B4	64.00	95.70	243.30	93.69	81.39
	ESeNet (Ours)	**EffNetV2S**	**22.47**	**27.85**	**85.71**	**93.54**	**80.73**
RGB-D	RedNet [203]	2×ResNet50	52.64	53.06	200.80	90.57	72.14
	FuseNet [3]	2×VGG16	44.20	241.87	168.60	91.64	76.22
	ESANet [5]	2×ResNet50	68.45	52.91	261.10	92.68	78.46
	SSMA [5]	2×ResNet50	56.32	99.66	214.85	90.70	73.37
	ESeNet-D (Ours)	**2×EffNetV2S**	**42.58**	**42.75**	**162.45**	**94.18**	**83.05**

5.4.2.6 ESeNet-D Evaluation Results with AfricaFoodSeg Dataset

We also trained the ESeNet-D model on the AfricaFoodSeg dataset, with the results summarized in Tab.5.6. This experiment further demonstrated the effectiveness of our RGB-D model.

Tab. 5.6 ESeNet-D results on AfricaFoodSeg dataset.

	Model	Backbone	Params(M) ↓	FLOPs(B) ↓	Size(MB) ↓	PA(%) ↑	mIoU(%) ↑
RGB	mid-DeepLabv3+	ResNet101	31.04	61.36	118.41	96.68	93.52
	DeepLabv3+ [45]	ResNet50	40.43	72.36	154.24	96.53	93.25
	DeepLabv3+ [45]	ResNet101	59.50	91.77	226.98	96.77	93.70
	GourmetNet [59]	ResNet101	47.36	103.91	180.66	96.85	93.80
	DANet [195]	ResNet101	67.05	78.19	255.77	96.66	93.50
	EANet [196]	ResNet101	53.87	62.54	205.50	96.78	93.72
	ESeNet	**EffNetV2S**	**22.46**	**27.46**	**85.69**	**96.92**	**94.01**
	ESeNet-D	**2×EffNetV2S**	**42.58**	**42.37**	**162.43**	**97.55**	**95.19**

5.4.2.7 Qualitative Results on CamerFood15 Dataset

In Fig. 5.20, we visualize the segmentation results for some images from the CamerFood15 validation set. Due to space constraints, we only present the predictions from the some models, RGB (SegFormer) and RGB-D (ESANet, FuseNet), which yielded the best results. A qualitative analysis of these results reveals that FuseNet provides imprecise predictions for complex images (see. **[c]**). SegFormer, on the other hand, struggles with edge accuracy, often showing significant separation between two closely connected objects (see. **[a,d,f]**). Additionally, the masks predicted by SegFormer exhibit strides along the edges, mainly due to the tokenization process in transformer architectures (see. **[e,g]**). Visually, our models, ESeNet-D and ESeNet, deliver better results across most examples.

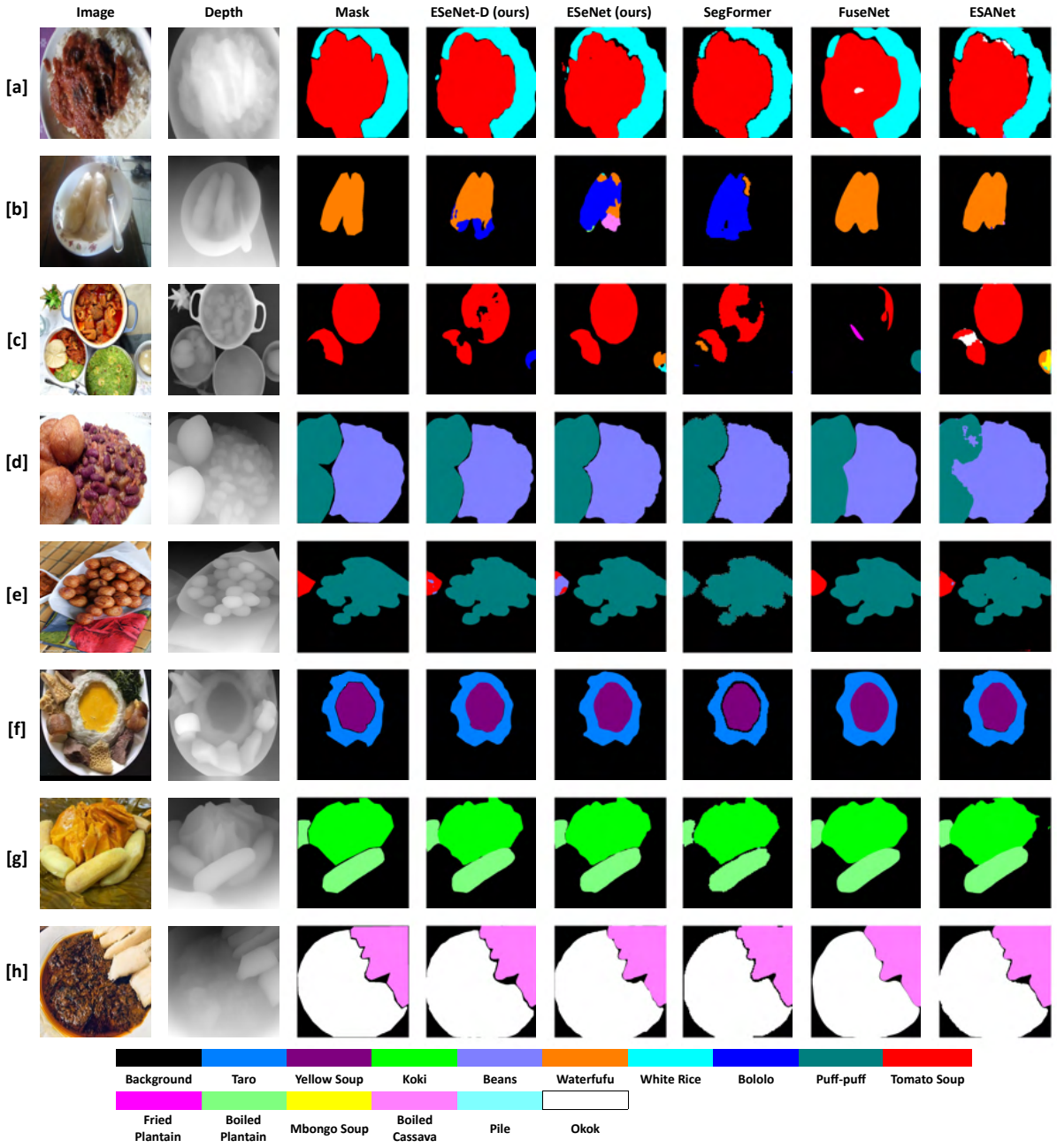


Fig. 5.20 Qualitative comparison of ESeNet-D and ESeNet with related best RGB and RGBD models on CamerFood15 dataset.

5.4.2.8 Class-wise Performance Analysis on CamerFood15 dataset

We analyzed model performance for each class in CamerFood15, presented in Tab. 5.7. To keep the table concise, we focused on the top-performing RGB and RGB-D models. ESeNet-D consistently performs best across most classes. Classes with fewer occurrences

(see Fig. 4.5) show poorer performance with other RGB-D models. There is also noticeable class-level improvement with ESeNet-D compared to ESeNet, highlighting the impact of depth maps on segmentation accuracy improvement. Overall, our results demonstrate that ESeNet-D achieves high segmentation performance compared to other benchmark models, even for less represented classes such as *"12-Mbongo soup"*, *"13-Boiled cassava"*, and *"15-Okok"*.

5.5 Conclusion

In this paper, we propose an efficient model for multimodal image (RGB + Depth) semantic segmentation. Previous research on scene segmentation has shown that incorporating depth information improves semantic image segmentation by providing essential geometric details, which are crucial for accurately distinguishing between objects with similar colors and textures. However, multimodal food image semantic segmentation remains under-explored, with few works utilizing RGB-Depth data. In this study, we propose ESeNet-D, a model that leverages self-calibrated convolutions and an efficient modality fusion technique to combine RGB and depth data effectively. ESeNet-D not only outperforms existing state-of-the-art models but also maintains a lightweight architecture. Validation on diverse datasets containing food images from Brazilian and African contexts demonstrated the robustness and effectiveness of ESeNet-D. Given the challenge of building RGB-Depth image datasets, we demonstrate that MDE models can facilitate the generation of RGB-D datasets for food segmentation. In addition to demonstrating the usefulness of depth maps for semantic segmentation, we investigate how the quality of these depth maps impacts performance.

Tab. 5.7 ESeNet-D classwise IoU scores analysis with CamerFood15 dataset. Best values for each class are highlighted in green.

Id	Class Name	ESeNet-D	ESeNet	SegFormer	mid-DeepLabv3+	DeepLabv3+	GourmetNet	EANet	ESANet	SSMA	FuseNet	RedNet
0	*background*	**95.26**	**94.69**	94.61	94.55	94.36	94.13	94.07	93.64	91.97	93.73	93.24
1	*taro*	**80.94**	**80.81**	80.52	80.61	78.92	78.51	80.47	74.93	76.06	73.97	75.65
2	*yellow soup*	**79.16**	**80.06**	79.78	**81.89**	79.15	80.65	78.02	77.48	67.3	74.85	70.09
3	*koki*	**88.01**	**85.92**	84.30	86.27	89.13	88.66	85.05	83.34	74.31	84.10	78.01
4	*beans*	**81.40**	**75.37**	80.32	80.11	79.87	73.70	80.29	81.19	66.78	**83.33**	74.19
5	*waterfufu*	**82.64**	**79.31**	80.84	80.85	78.78	79.96	77.91	75.91	71.91	63.65	66.51
6	*white rice*	**92.08**	**90.72**	89.52	89.28	89.16	89.29	88.40	88.68	86.06	87.08	87.19
7	*bobolo*	**83.93**	**80.36**	78.16	81.36	80.36	78.96	79.34	**85.86**	73.16	83.10	75.39
8	*puff-puff*	**92.80**	**91.64**	89.11	91.70	90.32	91.23	91.27	83.52	85.67	86.56	85.31
9	*tomato soup*	**85.64**	**86.08**	84.83	83.27	81.58	81.85	82.50	79.87	74.79	82.32	75.31
10	*fried plantain*	**93.15**	**92.65**	91.01	91.59	91.50	91.08	90.03	86.94	80.94	92.05	86.51
11	*boiled plantain*	**89.42**	**85.15**	80.61	83.79	84.39	82.84	83.29	80.36	80.22	83.69	81.88
12	*mbongo soup*	**81.26**	**75.94**	83.07	64.41	67.63	79.07	42.73	43.73	29.19	42.34	36.63
13	*boiled cassava*	**77.66**	**70.63**	77.07	68.86	72.47	69.74	**78.28**	64.43	69.15	62.72	67.48
14	*pile*	**96.07**	**96.04**	93.68	91.49	95.23	91.31	95.68	95.36	93.37	89.63	86.44
15	*okok*	**88.72**	**91.88**	79.01	73.48	73.28	76.22	74.32	72.28	74.02	69.99	81.43
	mIoU	**86.76**	**84.83**	84.15	82.72	82.88	82.95	81.35	79.24	74.68	78.32	76.33

CHAPTER 6

Conclusions and Future Work

6.1 Contributions

With the global rise in diet-related diseases, VBDA has become a very popular research field. VBDA systems aim to take a meal image as input and automatically extract relevant dietary information. Compared to traditional dietary assessment methods, VBDA eliminates subjectivity, saves time, and improves the accuracy and comprehensiveness of dietary intake assessments. VBDA typically involves three stages : **food image analysis, portion estimation,** and **nutrient derivation**. Food image semantic segmentation, which occurs during the image analysis stage, is essential, assigning a label to each pixel to segment the image into different food items or categories. In the literature, food segmentation methods are classified into three main types : **automatic approaches using machine learning with handcrafted features**, **semi-automatic approaches**, and **automatic deep learning-based approaches**. Among these, supervised deep learning methods have

proven its effectiveness and require minimal user input. However, their accuracy heavily depends on the performance of deep learning models and the quality of the used training datasets.

This thesis contributes to development of new datasets and deep learning models for food image segmentation. To this end, we conducted a literature review, providing an analysis of the state of the art of food image segmentation in **Chapter 3**. The review highlighted two main issues : first, food image datasets for segmentation are scarce, and most research has focused on Asian and Western cuisines, with no datasets available for African foods. However, African dishes often consist of mixed food classes, making accurate segmentation particularly challenging. Second, despite its success in other fields, RGB-D segmentation in VBDA remains underexplored due to difficulties in collecting food depth images. As a result, research has predominantly focused on RGB images, which provide color and texture but may lack crucial geometric details.

This research leverages theses issues making several key contributions to the field of food image segmentation, detailed in Chapters 4 and 5. Our work introduces new deep learning models for RGB (mid-DeepLabv3+) and RGB-D (ESeNet-D) segmentation, along with the first food segmentation datasets focused on African cuisines. In **Chapter 4**, we present two datasets we built : **AfricaFoodSeg** for *food/non-food* segmentation with 3,067 images (2,525 for training, 542 for validation), and CamerFood, which focuses on Cameroonian dishes. CamerFood includes **CamerFood10**, with 1,422 images from ten food classes, and **CamerFood15**, an improved version with 15 food classes, 1,684 training images, and 514 validation images. Later in this chapter, the second contribution **mid-DeepLabv3+** is introduce. It is based on DeepLabv3+, with a simplified ResNet backbone, an added skip layer (middle layer) in the decoder, and a SimAM attention mechanism. This model achieves a strong balance between performance and efficiency, matching DeepLabv3+'s accuracy while reducing the computational load by half.

In **Chapter 5**, we address RGB-D food image segmentation with the development of **ESeNet-D**, a new model using two encoder branches with EfficientNetV2 as the backbone, a fusion block for multi-scale integration, and a decoder that employs self-calibrated convolution and learned interpolation for precise segmentation. ESeNet-D outperforms many RGB and RGB-D benchmarks with fewer parameters and FLOPs. Our experiments show that properly integrating depth information significantly improves food segmentation accuracy. When trained on CamerFood15, the ESeNet-D RGB-D model achieved a PA of 96.61% and mIoU of 86.76%, compared to 95.84% PA and 84.83% mIoU with the RGB-only network. Finally, we address the scarcity of depth data in RGB-D segmentation by demonstrating that monocular depth estimation (MDE) models

can help generate effective depth maps. We also evaluate the impact of depth map quality on segmentation performance by testing two top MDE models, MiDASv3.1 and DepthAnything. Depth maps from the more accurate DepthAnything model led to better performance with ESeNet-D.

6.2 Limitations

Despite promising results, our research faces some limitations and unresolved issues.

- Our datasets AfricaFoodSeg and CamerFood focuse on African food images. However, Africans, like people worldwide, consume dishes from various cultures. For instance, one might eat Japanese sushi in France or Cameroonian Ndolé in the United States. Therefore, to be truly effective, a VBDA system should be capable of recognizing a wide variety of foods from different countries and continents. An ideal dataset would not solely focus on African cuisine but include a diverse range of foods from around the world. Creating such a dataset is a significant challenge, not only in gathering a large number of images but also in finding skilled teams for annotation. Food images present complex issues like *"intra-class variation"* and *"inter-class resemblance"*, requiring a good understanding to accurately identify and differentiate dishes in photos.
- Models developed in this research, mid-DeepLabv3+ and ESeNet-D, have primarily been tested on food image datasets. It would be interesting to see how they perform on entirely different types of datasets.
- In this thesis, we explored MDE models as practical alternatives for generating depth maps for datasets. However, an important question remains : *how will a model trained on depth maps generated with MDE models perform in real-world scenarios using RGB-D images captured with portable cameras ?*

6.3 Future Work

For future work, this PhD opens several avenues to expand both the methodological and applications of our research, including the following research avenues.

— We aim to enhance our CamerFood dataset by incorporating more African food classes and images. The initial focus is on Cameroonian cuisine, but with the support of other research teams, we plan to include foods from various African countries. This expansion could pave the way for valuable international collaborations. We

also plan to merge it with other datasets like UECFoodPixComplete [134] to create a more global dataset which will include african, western and asian foods.

— We aim to create a test dataset with real depth images captured by smartphones to evaluate how closely generated depth maps resemble those from mobile devices. This will help determine if a model trained with generated depth maps performs effectively on RGB-D images captured by smartphones.

— VBDA systems aim to help people assess the nutritional composition of their meals. We plan to improve our ESeNet-D model for integration into a mobile app, allowing users to test the system in real-life scenarios. The app should enable users to take food photos, view segmentation results, and optionally upload their images to contribute to expanding the dataset.

— This thesis focuses on one stage of VBDA systems. In the future, we plan to address the next stage : food portion or volume estimation. Recent work [218, 219] explores food volume estimation using depth maps. Graikos et al. [218] approach involves three parts : depth estimation, RGB image segmentation, and a point cloud-to-volume algorithm. By integrating this approach with our research, we aim to develop a system where depth maps enhance food segmentation and create point clouds to assess the volume of each identified food item.

Bibliography

[1] C. N. Freitas, F. R. Cordeiro, and V. Macario, "Myfood : A food segmentation and classification system to aid nutritional monitoring," in *2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. IEEE, 2020, pp. 234–239.

[2] A. Valada, G. L. Oliveira, T. Brox, and W. Burgard, "Deep multispectral semantic scene understanding of forested environments using multimodal fusion," in *2016 international symposium on experimental robotics*. Springer, 2017, pp. 465–477.

[3] C. Hazirbas, L. Ma, C. Domokos, and D. Cremers, "Fusenet : Incorporating depth into semantic segmentation via fusion-based cnn architecture," in *Computer Vision–ACCV 2016 : 13th Asian Conference on Computer Vision, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part I 13*. Springer, 2017, pp. 213–228.

[4] S.-J. Park, K.-S. Hong, and S. Lee, "Rdfnet : Rgb-d multi-level residual feature fusion for indoor semantic segmentation," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 4980–4989.

[5] D. Seichter, M. Köhler, B. Lewandowski, T. Wengefeld, and H.-M. Gross, "Efficient rgb-d semantic segmentation for indoor scene analysis," in *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2021, pp. 13 525–13 531.

[6] L. Yang, B. Kang, Z. Huang, X. Xu, J. Feng, and H. Zhao, "Depth anything : Unleashing the power of large-scale unlabeled data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 10 371–10 381.

[7] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[8] M. Tan and Q. Le, "Efficientnetv2 : Smaller models and faster training," in *International conference on machine learning*. PMLR, 2021, pp. 10 096–10 106.

[9] Keras, "Keras 3 api documentation," https://keras.io/api/applications/, 2022, accessed : (03/06/2024).

[10] World Health Organization, "Fact sheets - malnutrition," https://www.who.int/news-room/fact-sheets/detail/malnutrition, may 2024, [Accessed 12-09-2024].

[11] ——, "Fact sheets - obesity and overweight," https://www.who.int/news-room/fact-sheets/detail/obesity-and-overweight, march 2024, [Accessed 12-09-2024].

[12] D. Chilot, D. G. Belay, M. W. Merid, A. A. Kibret, A. Z. Alem, M. H. Asratie, N. W. Teshager, and F. M. Aragaw, "Triple burden of malnutrition among mother–child pairs in low-income and middle-income countries : a cross-sectional study," *BMJ open*, vol. 13, no. 5, p. e070978, 2023.

[13] W. H. Organization *et al.*, *Noncommunicable diseases : progress monitor 2022*. Genève, Switzerland : World Health Organization, apr 2022.

[14] U. World Health Organization, Food and Agriculture Organization of the United Nations, *Guidance for Monitoring Healthy Diets Globally*. Geneva : World Health Organization, 2024, licence : CC BY-NC-SA 3.0 IGO. [Online]. Available : https://www.who.int/publications/i/item/9789240094383

[15] W. H. Organization, *World health statistics 2020 : monitoring health for the SDGs, sustainable development goals*. Genève, Switzerland : World Health Organization, may 2020.

[16] UNICEF, "The state of the world's children 2019. children, food and nutrition : Growing well in a changing world," New York, 2019.

[17] P. Seferidi, T. Hone, A. C. Duran, A. Bernabe-Ortiz, and C. Millett, "Global inequalities in the double burden of malnutrition and associations with globalisation : a multilevel analysis of demographic and health surveys from 55 low-income and middle-income countries, 1992–2018," *The Lancet Global Health*, vol. 10, no. 4, pp. e482–e490, 2022.

[18] J. de Toro-Martín, B. J. Arsenault, J.-P. Després, and M.-C. Vohl, "Precision nutrition : a review of personalized nutritional approaches for the prevention and management of metabolic syndrome," *Nutrients*, vol. 9, no. 8, p. 913, 2017.

[19] W. Wang, W. Min, T. Li, X. Dong, H. Li, and S. Jiang, "A review on vision-based analysis for automatic dietary assessment," *Trends in Food Science & Technology*, vol. 122, pp. 223–237, 2022.

[20] Z. van der Heijden, F. de Gooijer, G. Camps, D. Lucassen, E. Feskens, M. Lasschuijt, E. Brouwer-Brolsma *et al.*, "User requirements in developing a novel dietary assessment tool for children : Mixed methods study," *JMIR Formative Research*, vol. 8, no. 1, p. e47850, 2024.

[21] L. Bell, A. Manson, D. Zarnowiecki, S. N. Tan, R. Byrne, R. Taylor, M. Zheng, L. M. Wen, and R. Golley, "Development and validation of a short dietary questionnaire for assessing obesity-related dietary behaviours in young children," *Maternal & Child Nutrition*, vol. 20, no. 2, p. e13613, 2024.

[22] M. Chopra and A. Purwar, "Recent studies on segmentation techniques for food recognition : A survey," *Archives of Computational Methods in Engineering*, vol. 29, no. 2, pp. 865–878, 2022.

[23] W. Min, S. Jiang, L. Liu, Y. Rui, and R. Jain, "A survey on food computing," *ACM Computing Surveys (CSUR)*, vol. 52, no. 5, pp. 1–36, 2019.

[24] M. Iriti, E. M. Varoni, and S. Vitalini, "Healthy diets and modifiable risk factors for non-communicable diseases—the european perspective," *Foods*, vol. 9, no. 7, p. 940, 2020.

[25] J.-S. Shim, K. Oh, and H. C. Kim, "Dietary assessment methods in epidemiologic studies," *Epidemiology and health*, vol. 36, 2014.

[26] M. L. Jobarteh, M. A. McCrory, B. Lo, M. Sun, E. Sazonov, A. K. Anderson, W. Jia, K. Maitland, J. Qiu, M. Steiner-Asiedu *et al.*, "Development and validation of an objective, passive dietary

assessment method for estimating food and nutrient intake in households in low-and middle-income countries : A study protocol," *Current developments in nutrition*, vol. 4, no. 2, p. nzaa020, 2020.

[27] M. Gibney, D. Allison, D. Bier, and J. Dwyer, "Uncertainty in human nutrition research," *Nature Food*, vol. 1, no. 5, pp. 247–249, 2020.

[28] A. H. Goris, M. S. Westerterp-Plantenga, and K. R. Westerterp, "Undereating and underrecording of habitual food intake in obese men : selective underreporting of fat intake," *The American journal of clinical nutrition*, vol. 71, no. 1, pp. 130–134, 2000.

[29] A. Illner, H. Freisling, H. Boeing, I. Huybrechts, S. Crispim, and N. Slimani, "Review and evaluation of innovative technologies for measuring diet in nutritional epidemiology," *International journal of epidemiology*, vol. 41, no. 4, pp. 1187–1203, 2012.

[30] G. P. Bathalon, K. L. Tucker, N. P. Hays, A. G. Vinken, A. S. Greenberg, M. A. McCrory, and S. B. Roberts, "Psychological measures of eating behavior and the accuracy of 3 common dietary assessment methods in healthy postmenopausal women," *The American journal of clinical nutrition*, vol. 71, no. 3, pp. 739–745, 2000.

[31] M. A. Subhi, S. H. Ali, and M. A. Mohammed, "Vision-based approaches for automatic food recognition and dietary assessment : A survey," *IEEE Access*, vol. 7, pp. 35 370–35 381, 2019.

[32] W. Tay, B. Kaur, R. Quek, J. Lim, and C. J. Henry, "Current developments in digital quantitative volume estimation for the optimisation of dietary assessment," *Nutrients*, vol. 12, no. 4, p. 1167, 2020.

[33] G. A. Tahir and C. K. Loo, "A comprehensive survey of image-based food recognition and volume estimation methods for dietary assessment," in *Healthcare*, vol. 9, no. 12. Multidisciplinary Digital Publishing Institute, 2021, p. 1676. [Online]. Available : https://www.mdpi.com/2227-9032/9/12/1676

[34] Y. Lu, T. Stathopoulou, M. F. Vasiloglou, L. F. Pinault, C. Kiley, E. K. Spanakis, and S. Mougiakakou, "gofoodtm : an artificial intelligence system for dietary assessment," *Sensors*, vol. 20, no. 15, p. 4283, 2020.

[35] F. P. W. Lo, Y. Sun, J. Qiu, and B. Lo, "Image-based food classification and volume estimation for dietary assessment : A review," *IEEE journal of biomedical and health informatics*, vol. 24, no. 7, pp. 1926–1939, 2020.

[36] K. V. Dalakleidi, M. Papadelli, I. Kapolos, and K. Papadimitriou, "Applying image-based food-recognition systems on dietary assessment : a systematic review," *Advances in Nutrition*, vol. 13, no. 6, pp. 2590–2619, 2022.

[37] U.S. Department of Agriculture, Agricultural Research Service, Beltsville Human Nutrition Research Center, "Fooddata central," https://fdc.nal.usda.gov/, 2024, accessed : August 3, 2024.

[38] A. Vincent, F. Grande, E. Compaoré, G. Amponsah Annor, P. Addy, L. Aburime, D. Ahmed, A. Bih Loh, S. Dahdouh Cabia, N. Deflache, F. Dembélé, B. Dieudonné, O. Edwige, H. Ene-Obong, N. Fanou Fogny, M. Ferreira, J. Omaghomi Jemide, P. Kouebou, C. Muller, S. Nájera Espinosa, F. Ouattara, D. Rittenschober, H. Schönfeldt, B. Stadlmayr, M. van

Deventer, A. Razikou Yiagnigni, and U. Charrondière, *FAO/INFOODS Food Composition Table for Western Africa (2019) User Guide & Condensed Food Composition Table*. Rome : Food and Agriculture Organization of the United Nations, 2020. [Online]. Available : http://www.fao.org/publications/card/ru/c/CA7779B/

[39] C. Wang, C. Wang, W. Li, and H. Wang, "A brief survey on rgb-d semantic segmentation using deep learning," *Displays*, vol. 70, p. 102080, 2021.

[40] H. Zhou, L. Qi, Z. Wan, H. Huang, and X. Yang, "Rgb-d co-attention network for semantic segmentation," in *Proceedings of the Asian conference on computer vision*, 2020.

[41] S. Chen, X. Zhu, W. Liu, X. He, and J. Liu, "Global-local propagation network for rgb-d semantic segmentation," *arXiv preprint arXiv :2101.10801*, 2021.

[42] H. Zhang, V. S. Sheng, X. Xi, Z. Cui, and H. Rong, "Overview of rgbd semantic segmentation based on deep learning," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, no. 10, pp. 13 627–13 645, 2023.

[43] A. Lopes, R. Souza, and H. Pedrini, "A survey on rgb-d datasets," *Computer Vision and Image Understanding*, vol. 222, p. 103489, 2022.

[44] L. Zhou, C. Zhang, F. Liu, Z. Qiu, and Y. He, "Application of deep learning in food : a review," *Comprehensive reviews in food science and food safety*, vol. 18, no. 6, pp. 1793–1811, 2019.

[45] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 801–818.

[46] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[47] L. Yang, R.-Y. Zhang, L. Li, and X. Xie, "Simam : A simple, parameter-free attention module for convolutional neural networks," in *International conference on machine learning*. PMLR, 2021, pp. 11 863–11 874.

[48] E. Sobngwi, *Atlas des aliments de consommation courante au Cameroun*, 1st ed. Yaounde, Cameroon : RSD Institute (Recherche Sante et Développement), 2021. [Online]. Available : https://rsd-institute.org/

[49] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[50] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, "Panoptic segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 9404–9413.

[51] M. Ivanovs, K. Ozols, A. Dobrajs, and R. Kadikis, "Improving semantic segmentation of urban scenes for self-driving cars with synthetic images," *Sensors*, vol. 22, no. 6, p. 2252, 2022.

[52] M. Z. Khan, M. K. Gajendran, Y. Lee, and M. A. Khan, "Deep neural architectures for medical image semantic segmentation," *IEEE Access*, vol. 9, pp. 83 002–83 024, 2021.

[53] N. Renuka, "Semantic segmentation-based skin cancer detection," *Soft Computing*, vol. 27, no. 16, pp. 11 895–11 903, 2023.

[54] J. Zhang, X. Lv, H. Zhang, and B. Liu, "Aresu-net : Attention residual u-net for brain tumor segmentation," *Symmetry*, vol. 12, no. 5, p. 721, 2020.

[55] M. Sudhan, M. Sinthuja, S. P. Raja, J. Amutharaj, G. C. P. Latha, S. S. Rachel, T. Anitha, T. Rajendran, and Y. A. Waji, "Segmentation and classification of glaucoma using u-net with deep learning model," *Journal of Healthcare Engineering*, vol. 2022, 2022.

[56] H. Hu, L. Shen, Q. Guan, X. Li, Q. Zhou, and S. Ruan, "Deep co-supervision and attention fusion strategy for automatic covid-19 lung infection segmentation on ct images," *Pattern Recognition*, vol. 124, p. 108452, 2022.

[57] K. D. M. De Silva and H. J. Lee, "Distorted aerial images semantic segmentation method for software-based analog image receivers using deep combined learning," *Applied Sciences*, vol. 13, no. 11, p. 6816, 2023.

[58] L. M. Amugongo, A. Kriebitz, A. Boch, and C. Lütge, "Mobile computer vision-based applications for food recognition and volume and calorific estimation : A systematic review," in *Healthcare*, vol. 11, no. 1. MDPI, 2022, p. 59.

[59] U. Sharma, B. Artacho, and A. Savakis, "Gourmetnet : Food segmentation using multi-scale waterfall features with spatial and channel attention," *Sensors*, vol. 21, no. 22, p. 7504, 2021.

[60] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[61] O. Ronneberger, P. Fischer, and T. Brox, "U-net : Convolutional networks for biomedical image segmentation," in *Medical image computing and computer-assisted intervention–MICCAI 2015 : 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*. Springer, 2015, pp. 234–241.

[62] H. Yu, Z. Yang, L. Tan, Y. Wang, W. Sun, M. Sun, and Y. Tang, "Methods and datasets on semantic segmentation : A review," *Neurocomputing*, vol. 304, pp. 82–103, 2018.

[63] J. Cheng, H. Li, D. Li, S. Hua, and V. S. Sheng, "A survey on image semantic segmentation using deep learning techniques," *Computers, Materials and Continua*, vol. 74, no. 1, p. 19411957, 2023. [Online]. Available : http://dx.doi.org/10.32604/cmc.2023.032757

[64] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning : concepts, cnn architectures, challenges, applications, future directions," *Journal of big Data*, vol. 8, pp. 1–74, 2021.

[65] X. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci, and M. Parmar, "A review of convolutional neural networks in computer vision," *Artificial Intelligence Review*, vol. 57, no. 4, pp. 1–43, 2024.

[66] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[67] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words : Transformers for image recognition at scale," *arXiv preprint arXiv :2010.11929*, 2020.

[68] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit *et al.*, "Mlp-mixer : An all-mlp architecture for vision," *Advances in neural information processing systems*, vol. 34, pp. 24 261–24 272, 2021.

[69] Y. Zhao, G. Wang, C. Tang, C. Luo, W. Zeng, and Z.-J. Zha, "A battle of network structures : An empirical study of cnn, transformer, and mlp," *arXiv preprint arXiv :2108.13002*, 2021.

[70] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of physiology*, vol. 160, no. 1, p. 106, 1962.

[71] Y. Lee, J. Park, and C.-O. Lee, "Two-level group convolution," *Neural Networks*, vol. 154, pp. 323–332, 2022.

[72] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2 : Practical guidelines for efficient cnn architecture design," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 116–131.

[73] P. Gibson, J. Cano, J. Turner, E. J. Crowley, M. OBoyle, and A. Storkey, "Optimizing grouped convolutions on edge devices," in *2020 IEEE 31st International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. IEEE, 2020, pp. 189–196.

[74] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[75] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.

[76] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet : An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6848–6856.

[77] Y. Ioannou, D. Robertson, R. Cipolla, and A. Criminisi, "Deep roots : Improving cnn efficiency with hierarchical filter groups," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1231–1240.

[78] G. Huang, S. Liu, L. Van der Maaten, and K. Q. Weinberger, "Condensenet : An efficient densenet using learned group convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2752–2761.

[79] L. Yang, H. Jiang, R. Cai, Y. Wang, S. Song, G. Huang, and Q. Tian, "Condensenet v2 : Sparse feature reactivation for deep networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3569–3578.

[80] M. Darvish-Motevali, M. K. Sohrabi, and I. Roshdi, "Self-supervised condensenet for feature learning to increase the accuracy in image classification," *Multimedia Tools and Applications*, pp. 1–12, 2024.

[81] Y. Lee, J. Park, and C.-O. Lee, "Balanced group convolution : An improved group convolution based on approximability estimates," *arXiv preprint arXiv :2310.12461*, 2023.

[82] J. Cheng, S. Tian, L. Yu, C. Gao, X. Kang, X. Ma, W. Wu, S. Liu, and H. Lu, "Resganet : Residual group attention network for medical image classification and segmentation," *Medical Image Analysis*, vol. 76, p. 102313, 2022.

[83] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets : Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv :1704.04861*, 2017.

[84] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2 : Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[85] F. Chollet, "Xception : Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[86] S. Shin, S. Lee, and H. Han, "Ear-net : efficient atrous residual network for semantic segmentation of street scenes based on deep learning," *Applied Sciences*, vol. 11, no. 19, p. 9119, 2021.

[87] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian, "A real-time algorithm for signal analysis with the help of the wavelet transform," in *Wavelets : Time-Frequency Methods and Phase Space Proceedings of the International Conference, Marseille, France, December 14–18, 1987*. Springer, 1990, pp. 286–297.

[88] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab : Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[89] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1520–1528.

[90] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 111–118.

[91] M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *2011 international conference on computer vision*. IEEE, 2011, pp. 2018–2025.

[92] S. Cong and Y. Zhou, "A review of convolutional neural network architectures and their optimizations," *Artificial Intelligence Review*, vol. 56, no. 3, pp. 1905–1969, 2023.

[93] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.

[94] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation functions in deep learning : A comprehensive survey and benchmark," *Neurocomputing*, vol. 503, pp. 92–108, 2022.

[95] A. L. Maas, A. Y. Hannun, A. Y. Ng *et al.*, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, no. 1. Atlanta, GA, 2013, p. 3.

[96] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers : Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[97] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv :1511.07289*, 2015.

[98] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv :1710.05941*, 2017.

[99] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1314–1324.

[100] S. Ioffe and C. Szegedy, "Batch normalization : Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. pmlr, 2015, pp. 448–456.

[101] N. Bjorck, C. P. Gomes, B. Selman, and K. Q. Weinberger, "Understanding batch normalization," *Advances in neural information processing systems*, vol. 31, 2018.

[102] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout : a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[103] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.

[104] L. Shao, F. Zhu, and X. Li, "Transfer learning for visual categorization : A survey," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 5, pp. 1019–1034, 2014.

[105] S. Connor and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.

[106] K. Alomar, H. I. Aysel, and X. Cai, "Data augmentation in classification and segmentation : A survey and new strategies," *Journal of Imaging*, vol. 9, no. 2, p. 46, 2023.

[107] T. S. Nazaré, G. B. P. da Costa, W. A. Contato, and M. Ponti, "Deep convolutional neural networks and noisy images," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications : 22nd Iberoamerican Congress, CIARP 2017, Valparaíso, Chile, November 7–10, 2017, Proceedings 22*. Springer, 2018, pp. 416–424.

[108] R. Takahashi, T. Matsubara, and K. Uehara, "Data augmentation using random image cropping and patching for deep cnns," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 9, pp. 2917–2931, 2019.

[109] R. Azad, M. Heidary, K. Yilmaz, M. Hüttemann, S. Karimijafarbigloo, Y. Wu, A. Schmeink, and D. Merhof, "Loss functions in the era of semantic segmentation : A survey and outlook," *arXiv preprint arXiv :2312.05391*, 2023.

[110] S. Jadon, "A survey of loss functions for semantic segmentation," in *2020 IEEE conference on computational intelligence in bioinformatics and computational biology (CIBCB)*. IEEE, 2020, pp. 1–7.

[111] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[112] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. Jorge Cardoso, "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 : Third International Workshop, DLMIA 2017, and 7th International Workshop, ML-CDS 2017, Held in Conjunction with MICCAI 2017, Québec City, QC, Canada, September 14, Proceedings 3*. Springer, 2017, pp. 240–248.

[113] S. S. M. Salehi, D. Erdogmus, and A. Gholipour, "Tversky loss function for image segmentation using 3d fully convolutional deep networks," in *International workshop on machine learning in medical imaging*. Springer, 2017, pp. 379–387.

[114] S. A. Taghanaki, Y. Zheng, S. K. Zhou, B. Georgescu, P. Sharma, D. Xu, D. Comaniciu, and G. Hamarneh, "Combo loss : Handling input and output imbalance in multi-organ segmentation," *Computerized Medical Imaging and Graphics*, vol. 75, pp. 24–33, 2019.

[115] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *Ussr computational mathematics and mathematical physics*, vol. 4, no. 5, pp. 1–17, 1964.

[116] Y. Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence o (1/k2)," in *Dokl. Akad. Nauk. SSSR*, vol. 269, no. 3, 1983, p. 543.

[117] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv :1609.04747*, 2016.

[118] K. Nakamura, B. Derbel, K.-J. Won, and B.-W. Hong, "Learning-rate annealing methods for deep neural networks," *Electronics*, vol. 10, no. 16, p. 2029, 2021.

[119] T. Tieleman, "Lecture 6.5-rmsprop : Divide the gradient by a running average of its recent magnitude," *COURSERA : Neural networks for machine learning*, vol. 4, no. 2, p. 26, 2012.

[120] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization." *Journal of machine learning research*, vol. 12, no. 7, 2011.

[121] D. P. Kingma and J. Ba, "Adam : A method for stochastic optimization," *arXiv preprint arXiv :1412.6980*, 2014.

[122] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv :1711.05101*, 2017.

[123] Z. Li and S. Arora, "An exponential learning rate schedule for deep learning," *arXiv preprint arXiv :1910.07454*, 2019.

[124] L. N. Smith, "Cyclical learning rates for training neural networks," in *2017 IEEE winter conference on applications of computer vision (WACV)*. IEEE, 2017, pp. 464–472.

[125] I. Loshchilov and F. Hutter, "Sgdr : Stochastic gradient descent with warm restarts," *arXiv preprint arXiv :1608.03983*, 2016.

[126] A. Sohail, N. A. Nawaz, A. A. Shah, S. Rasheed, S. Ilyas, and M. K. Ehsan, "A systematic literature review on machine learning and deep learning methods for semantic segmentation," *IEEE Access*, vol. 10, pp. 134 557–134 570, 2022.

[127] F. S. Konstantakopoulos, E. I. Georga, and D. I. Fotiadis, "A review of image-based food recognition and volume estimation artificial intelligence systems," *IEEE Reviews in Biomedical Engineering*, 2023.

[128] I. Ulku and E. Akagündüz, "A survey on deep learning-based architectures for semantic segmentation on 2d images," *Applied Artificial Intelligence*, vol. 36, no. 1, p. 2032924, 2022.

[129] B. Emek Soylu, M. S. Guzel, G. E. Bostanci, F. Ekinci, T. Asuroglu, and K. Acici, "Deep-learning-based approaches for semantic segmentation of natural scene images : A review," *Electronics*, vol. 12, no. 12, p. 2730, 2023.

[130] A. Seth, S. Sharma *et al.*, "Semantic segmentation : A systematic analysis from state-of-the-art techniques to advance deep networks," *Journal of Information Technology Research (JITR)*, vol. 15, no. 1, pp. 1–28, 2022.

[131] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning : A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 7, pp. 3523–3542, 2021.

[132] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge : A retrospective," *International journal of computer vision*, vol. 111, pp. 98–136, 2015.

[133] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco : Common objects in context," in *Computer Vision–ECCV 2014 : 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 740–755.

[134] K. Okamoto and K. Yanai, "Uec-foodpix complete : A large-scale food image segmentation dataset," in *Pattern Recognition. ICPR International Workshops and Challenges : Virtual Event, January 10–15, 2021, Proceedings, Part V*. Springer, 2021, pp. 647–659.

[135] D. Park, J. Lee, J. Lee, and K. Lee, "Deep learning based food instance segmentation using synthetic data," in *2021 18th International Conference on Ubiquitous Robots (UR)*. IEEE, 2021, pp. 499–505.

[136] Y. Matsuda, H. Hoashi, and K. Yanai, "Recognition of multiple-food images by detecting candidate regions," in *2012 IEEE International Conference on Multimedia and Expo*. IEEE, 2012, pp. 25–30.

[137] Y. Kawano and K. Yanai, "Automatic expansion of a food image dataset leveraging existing categories with domain adaptation," in *Computer Vision-ECCV 2014 Workshops : Zurich, Switzerland, September 6-7 and 12, 2014, Proceedings, Part III 13*. Springer, 2015, pp. 3–17.

[138] F. S. Konstantakopoulos, E. I. Georga, and D. I. Fotiadis, "Multiclass semantic segmentation of mediterranean food images," in *International Conference on Pervasive Computing Technologies for Healthcare*. Springer, 2022, pp. 49–59.

[139] M. Chen, K. Dhingra, W. Wu, L. Yang, R. Sukthankar, and J. Yang, "Pfid : Pittsburgh fast-food image dataset," in *2009 16th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2009, pp. 289–292.

[140] A. Meyers, N. Johnston, V. Rathod, A. Korattikara, A. Gorban, N. Silberman, S. Guadarrama, G. Papandreou, J. Huang, and K. P. Murphy, "Im2calories : towards an automated mobile vision food diary," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1233–1241.

[141] Y. Liang, J. Li, Q. Zhao, W. Rao, C. Zhang, and C. Wang, "Image segmentation and recognition for multi-class chinese food," in *2022 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2022, pp. 3938–3942.

[142] T. Ege, W. Shimoda, and K. Yanai, "A new large-scale food image segmentation dataset and its application to food calorie estimation based on grains of rice," in *Proceedings of the 5th international workshop on multimedia assisted dietary management*, 2019, pp. 82–87.

[143] J. Gao, W. Tan, L. Ma, Y. Wang, and W. Tang, "Musefood : Multi-sensor-based food volume estimation on smartphones," in *2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE, 2019, pp. 899–906.

[144] X. Wu, X. Fu, Y. Liu, E.-P. Lim, S. C. Hoi, and Q. Sun, "A large-scale benchmark for food image segmentation," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 506–515.

[145] C. Rother, V. Kolmogorov, and A. Blake, ""grabcut" interactive foreground extraction using iterated graph cuts," *ACM transactions on graphics (TOG)*, vol. 23, no. 3, pp. 309–314, 2004.

[146] G. Ciocca, P. Napoletano, and R. Schettini, "Food recognition : a new dataset, experiments, and results," *IEEE journal of biomedical and health informatics*, vol. 21, no. 3, pp. 588–598, 2016.

[147] M. Jalal, K. Wang, S. Jefferson, Y. Zheng, E. O. Nsoesie, and M. Betke, "Scraping social media photos posted in kenya and elsewhere to detect and analyze food types," in *Proceedings of the 5th International Workshop on Multimedia Assisted Dietary Management*, 2019, pp. 50–59.

[148] L. Bossard, M. Guillaumin, and L. Van Gool, "Food-101–mining discriminative components with random forests," in *Computer Vision–ECCV 2014 : 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VI 13*. Springer, 2014, pp. 446–461.

[149] S. P. Mohanty, G. Singhal, E. A. Scuccimarra, D. Kebaili, H. Héritier, V. Boulanger, and M. Salathé, "The food recognition benchmark : Using deep learning to recognize food in images," *Frontiers in Nutrition*, vol. 9, 2022.

[150] K. N. Lam, M.-K. T. Nguyen, K. D. Nguyen, N. H. Nguyen, K.-Y. T. Nguyen, and A. Ware, "Swvie-food : A dataset for recognizing foods in southwest vietnam based on deep learning," in *2022 RIVF International Conference on Computing and Communication Technologies (RIVF)*. IEEE, 2022, pp. 488–493.

[151] S. Aslan, G. Ciocca, D. Mazzini, and R. Schettini, "Benchmarking algorithms for food localization and semantic segmentation," *International Journal of Machine Learning and Cybernetics*, vol. 11, no. 12, pp. 2827–2847, 2020.

[152] R. Mao, J. He, Z. Shao, S. K. Yarlagadda, and F. Zhu, "Visual aware hierarchy based food recognition," in *International conference on pattern recognition*. Springer, 2021, pp. 571–598.

[153] E. Battini Sönmez, S. Memiş, B. Arslan, and O. Z. Batur, "The segmented uec food-100 dataset with benchmark experiment on food detection," *Multimedia Systems*, vol. 29, no. 4, pp. 2049–2057, 2023.

[154] T. R. Baban A Erep and L. Chaari, "mid-deeplabv3+ : A novel approach for image semantic segmentation applied to african food dietary assessments," *Sensors*, vol. 24, no. 1, p. 209, 2023.

[155] J. Dehais, M. Anthimopoulos, and S. Mougiakakou, "Food image segmentation for dietary assessment," in *Proceedings of the 2nd international workshop on multimedia assisted dietary management*, 2016, pp. 23–28.

[156] A. Mariappan, M. Bosch, F. Zhu, C. J. Boushey, D. A. Kerr, D. S. Ebert, and E. J. Delp, "Personal dietary assessment using mobile devices," in *Computational imaging VII*, vol. 7246. SPIE, 2009, pp. 294–305.

[157] Y. Eskin and A. Mihailidis, "An intelligent nutritional assessment system," in *2012 AAAI fall symposium series*, 2012.

[158] F. Zhu, A. Mariappan, C. J. Boushey, D. Kerr, K. D. Lutes, D. S. Ebert, and E. J. Delp, "Technology-assisted dietary assessment," in *Computational imaging VI*, vol. 6814. SPIE, 2008, pp. 276–285.

[159] F. Zhu, M. Bosch, C. J. Boushey, and E. J. Delp, "An image analysis system for dietary assessment and evaluation," in *2010 Ieee International Conference on Image Processing*. IEEE, 2010, pp. 1853–1856.

[160] P. Pouladzadeh, S. Shirmohammadi, and A. Yassine, "Using graph cut segmentation for food calorie measurement," in *2014 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*. IEEE, 2014, pp. 1–6.

[161] Y. He, C. Xu, N. Khanna, C. J. Boushey, and E. J. Delp, "Food image analysis : Segmentation, identification and weight estimation," in *2013 IEEE international conference on multimedia and expo (ICME)*. IEEE, 2013, pp. 1–6.

[162] F. Zhu, M. Bosch, N. Khanna, C. J. Boushey, and E. J. Delp, "Multiple hypotheses image segmentation and classification with application to dietary assessment," *IEEE journal of biomedical and health informatics*, vol. 19, no. 1, pp. 377–388, 2014.

[163] M. Anthimopoulos, J. Dehais, P. Diem, and S. Mougiakakou, "Segmentation and recognition of multi-food meal images for carbohydrate counting," in *13th IEEE international conference on bioinformatics and bioengineering*. IEEE, 2013, pp. 1–4.

[164] Y. Kawano and K. Yanai, "Foodcam : A real-time food recognition system on a smartphone," *Multimedia Tools and Applications*, vol. 74, pp. 5263–5287, 2015.

[165] S. Inunganbi, A. Seal, and P. Khanna, "Classification of food images through interactive image segmentation," in *Intelligent Information and Database Systems : 10th Asian Conference, ACIIDS 2018, Dong Hoi City, Vietnam, March 19-21, 2018, Proceedings, Part II 10*. Springer, 2018, pp. 519–528.

[166] H. Hassannejad, G. Matrella, M. Mordonini, S. Cagnoni, and H. Hassannejad, "A mobile app for food detection : New approach to interactive segmentation," in *Proceedings of the FORITAAL Conference, Lecco, Italy*, 2015, pp. 19–22.

[167] H. Hassannejad, G. Matrella, P. Ciampolini, I. D. Munari, M. Mordonini, and S. Cagnoni, "A new approach to image-based estimation of food volume," *Algorithms*, vol. 10, no. 2, p. 66, 2017.

[168] P. Pouladzadeh, P. Kuhad, S. V. B. Peddi, A. Yassine, and S. Shirmohammadi, "Food calorie measurement using deep learning neural network," in *2016 IEEE international instrumentation and measurement technology conference proceedings*. IEEE, 2016, pp. 1–6.

[169] K. Okamoto and K. Yanai, "An automatic calorie estimation system of food images on a smartphone," in *Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management*, 2016, pp. 63–70.

[170] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv :1312.4400*, 2013.

[171] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[172] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," *arXiv preprint arXiv :1412.7062*, 2014.

[173] W. Chen and R. Song, "A new deep learning-based food recognition system for mobile terminal," in *2023 IEEE 12th Data Driven Control and Learning Systems Conference (DDCLS)*. IEEE, 2023, pp. 112–117.

[174] M.-L. Chiang, C.-A. Wu, J.-K. Feng, C.-Y. Fang, and S.-W. Chen, "Food calorie and nutrition analysis system based on mask r-cnn," in *2019 IEEE 5th International Conference on Computer and Communications (ICCC)*. IEEE, 2019, pp. 1721–1728.

[175] P. Poply and J. A. A. Jothi, "Refined image segmentation for calorie estimation of multiple-dish food items," in *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*. IEEE, 2021, pp. 682–687.

[176] B. Artacho and A. Savakis, "Omnipose : A multi-scale framework for multi-person pose estimation," *arXiv preprint arXiv :2103.10180*, 2021.

[177] X.-Y. Kong, X.-H. Sun, Y.-Z. Wang, R.-Y. Peng, X.-Y. Li, Y.-H. Yang, Y.-R. Lv, and S.-P. Tseng, "Food calorie estimation system based on semantic segmentation network." *Sensors & Materials*, vol. 35, 2023.

[178] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "Yolact : Real-time instance segmentation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 9157–9166.

[179] X. Lan, J. Lyu, H. Jiang, K. Dong, Z. Niu, Y. Zhang, and J. Xue, "Foodsam : Any food segmentation," *IEEE Transactions on Multimedia*, 2023.

[180] H.-T. Nguyen, Y. Cao, C.-W. Ngo, and W.-K. Chan, "Foodmask : Real-time food instance counting, segmentation and recognition," *Pattern Recognition*, vol. 146, p. 110017, 2024.

[181] A. Dutta and A. Zisserman, "The via annotation software for images, audio and video," in *Proceedings of the 27th ACM international conference on multimedia*, 2019, pp. 2276–2279.

[182] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv :1706.05587*, vol. 5, 2017.

[183] M.-H. Guo, T.-X. Xu, J.-J. Liu, Z.-N. Liu, P.-T. Jiang, T.-J. Mu, S.-H. Zhang, R. R. Martin, M.-M. Cheng, and S.-M. Hu, "Attention mechanisms in computer vision : A survey," *Computational Visual Media*, vol. 8, no. 3, pp. 331–368, 2022.

[184] L. Chen, H. Zhang, J. Xiao, L. Nie, J. Shao, W. Liu, and T.-S. Chua, "Sca-cnn : Spatial and channel-wise attention in convolutional networks for image captioning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5659–5667.

[185] J. Park, S. Woo, J.-Y. Lee, and I. S. Kweon, "Bam : Bottleneck attention module," *arXiv preprint arXiv :1807.06514*, 2018.

[186] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam : Convolutional block attention module," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.

[187] D. Misra, T. Nalamada, A. U. Arasanipalai, and Q. Hou, "Rotate to attend : Convolutional triplet attention module," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2021, pp. 3139–3148.

[188] S. Das, A. A. Fime, N. Siddique, and M. Hashem, "Estimation of road boundary for intelligent vehicles based on deeplabv3+ architecture," *IEEE Access*, vol. 9, pp. 121 060–121 075, 2021.

[189] J. Jia, J. Song, Q. Kong, H. Yang, Y. Teng, and X. Song, "Multi-attention-based semantic segmentation network for land cover remote sensing images," *Electronics*, vol. 12, no. 6, p. 1347, 2023.

[190] R. Azad, M. Asadi-Aghbolaghi, M. Fathy, and S. Escalera, "Attention deeplabv3+ : Multi-level context attention mechanism for skin lesion segmentation," in *Computer Vision–ECCV 2020 Workshops : Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*. Springer, 2020, pp. 251–266.

[191] H. Zeng, S. Peng, and D. Li, "Deeplabv3+ semantic segmentation model based on feature cross attention mechanism," in *Journal of Physics : Conference Series*, vol. 1678, no. 1. IOP Publishing, 2020, p. 012106.

[192] B. S. Webb, N. T. Dhruv, S. G. Solomon, C. Tailby, and P. Lennie, "Early and late mechanisms of surround suppression in striate cortex of macaque," *Journal of Neuroscience*, vol. 25, no. 50, pp. 11 666–11 675, 2005.

[193] Q. Wang, B. Wu, P. Zhu, P. Li, W. Zuo, and Q. Hu, "Eca-net : Efficient channel attention for deep convolutional neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 534–11 542.

[194] Q. Hou, D. Zhou, and J. Feng, "Coordinate attention for efficient mobile network design," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 13 713–13 722.

[195] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu, "Dual attention network for scene segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 3146–3154.

[196] M.-H. Guo, Z.-N. Liu, T.-J. Mu, and S.-M. Hu, "Beyond self-attention : External attention using two linear layers for visual tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 5436–5447, 2022.

[197] S. Piramanayagam, E. Saber, W. Schwartzkopf, and F. W. Koehler, "Supervised classification of multisensor remotely sensed images using a deep learning framework," *Remote sensing*, vol. 10, no. 9, p. 1429, 2018.

[198] D. Balouek, A. Carpen Amarie, G. Charrier, F. Desprez, E. Jeannot, E. Jeanvoine, A. Lèbre, D. Margery, N. Niclausse, L. Nussbaum, O. Richard, C. Pérez, F. Quesnel, C. Rohr, and L. Sarzyniec, "Adding virtualization capabilities to the Grid'5000 testbed," in *Cloud Computing and Services Science*, ser. Communications in Computer and Information Science, I. I. Ivanov, M. van Sinderen, F. Leymann, and T. Shan, Eds. Springer International Publishing, 2013, vol. 367, pp. 3–20.

[199] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv :1409.1556*, 2014.

[200] N. U. Gilal, K. Al-Thelaya, J. K. Al-Saeed, M. Abdallah, J. Schneider, J. She, J. H. Awan, and M. Agus, "Evaluating machine learning technologies for food computing from a data set perspective," *Multimedia Tools and Applications*, pp. 1–28, 2023.

[201] G. Sinha, K. Parmar, H. Azimi, A. Tai, Y. Chen, A. Wong, and P. Xi, "Transferring knowledge for food image segmentation using transformers and convolutions," *arXiv preprint arXiv :2306.09203*, 2023.

[202] C. Couprie, C. Farabet, L. Najman, and Y. LeCun, "Indoor semantic segmentation using depth information," *arXiv preprint arXiv :1301.3572*, 2013.

[203] J. Jiang, L. Zheng, F. Luo, and Z. Zhang, "Rednet : Residual encoder-decoder network for indoor rgb-d semantic segmentation," *arXiv preprint arXiv :1806.01054*, 2018.

[204] A. Valada, R. Mohan, and W. Burgard, "Self-supervised model adaptation for multimodal semantic segmentation," *International Journal of Computer Vision*, vol. 128, no. 5, pp. 1239–1285, 2020.

[205] G. Lin, A. Milan, C. Shen, and I. Reid, "Refinenet : Multi-path refinement networks for high-resolution semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1925–1934.

[206] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, "Towards robust monocular depth estimation : Mixing datasets for zero-shot cross-dataset transfer," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 3, pp. 1623–1637, 2020.

[207] A. Masoumian, H. A. Rashwan, J. Cristiano, M. S. Asif, and D. Puig, "Monocular depth estimation using deep learning : A review," *Sensors*, vol. 22, no. 14, p. 5353, 2022.

[208] V. Arampatzakis, G. Pavlidis, N. Mitianoudis, and N. Papamarkos, "Monocular depth estimation : A thorough review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[209] R. Ranftl, A. Bochkovskiy, and V. Koltun, "Vision transformers for dense prediction," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 12 179–12 188.

[210] R. Birkl, D. Wofk, and M. Müller, "Midas v3.1–a model zoo for robust monocular relative depth estimation," *ArXiv*, vol. abs/2307.14460, 2023. [Online]. Available : https://api.semanticscholar.org/CorpusID:260202837

[211] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "Segformer : Simple and efficient design for semantic segmentation with transformers," *Advances in neural information processing systems*, vol. 34, pp. 12 077–12 090, 2021.

[212] A. Valada, J. Vertens, A. Dhall, and W. Burgard, "Adapnet : Adaptive semantic segmentation in adverse environmental conditions," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4644–4651.

[213] J.-J. Liu, Q. Hou, M.-M. Cheng, C. Wang, and J. Feng, "Improving convolutional networks with self-calibrated convolutions," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 096–10 105.

[214] Z. Xue, X. Yu, X. Tan, B. Liu, A. Yu, and X. Wei, "Multiscale deep learning network with self-calibrated convolution for hyperspectral and lidar data collaborative classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–16, 2021.

[215] C. Zheng, J. Zhang, J.-N. Hwang, and B. Huang, "Double-branch dehazing network based on self-calibrated attentional convolution," *Knowledge-Based Systems*, vol. 240, p. 108148, 2022.

[216] R. Zheng, H. Zhu, X. Wu, and W. Meng, "T-psd : T-shape parking slot detection with self-calibrated convolution network," *Journal of Real-Time Image Processing*, vol. 21, no. 3, p. 82, 2024.

[217] L. Wang, L. Wang, Q. Wang, and L. Bruzzone, "Rscnet : A residual self-calibrated network for hyperspectral image change detection," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–17, 2022.

[218] A. Graikos, V. Charisis, D. Iakovakis, S. Hadjidimitriou, and L. Hadjileontiadis, "Single image-based food volume estimation using monocular depth-prediction networks," in *Universal Access in Human-Computer Interaction. Applications and Practice : 14th International Conference, UAHCI 2020, Held as Part of the 22nd HCI International Conference, HCII 2020, Copenhagen, Denmark, July 19–24, 2020, Proceedings, Part II 22*. Springer, 2020, pp. 532–543.

[219] Y. A. Sari and A. Gofuku, "Measuring food volume from rgb-depth image with point cloud conversion method using geometrical approach and robust ellipsoid fitting algorithm," *Journal of Food Engineering*, vol. 358, p. 111656, 2023.

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

Université de Toulouse

Titre : Contribution au développement d'un système intelligent de quantification des nutriments dans les repas d'Afrique subsaharienne.

Mots clés : Segmentation Semantique, Image Alimentaire, Segmentation Multimodale, RGB-D Image, Apprentissage Profond, Réseaux Convolutifs

Résumé : La malnutrition, qu'elle soit liée à un apport insuffisant ou excessif en nutriments, représente un défi mondial de santé publique touchant des milliards de personnes. Elle affecte tous les systèmes organiques en étant un facteur majeur de risque pour les maladies non transmissibles telles que les maladies cardiovasculaires, le diabète et certains cancers. Évaluer l'apport alimentaire est crucial pour prévenir la malnutrition, mais cela reste un défi. Les méthodes traditionnelles d'évaluation alimentaire sont laborieuses et sujettes aux biais. Les avancées en IA ont permis la conception de VBDA, solution prometteuse pour analyser automatiquement les images alimentaires afin d'estimer les portions et la composition nutritionnelle. Cependant, la segmentation des images alimentaires dans un VBDA rencontre des difficultés en raison de la structure non rigide des aliments, de la variation intra-classe élevée (où le même type d'aliment peut apparaître très différent), de la ressemblance inter-classe (où différents types d'aliments semblent visuellement très similaires) et de la rareté des ensembles de données disponibles publiquement.

Presque toutes les recherches sur la segmentation alimentaire se sont concentrées sur les aliments asiatiques et occidentaux, en l'absence de bases de données pour les cuisines africaines. Cependant, les plats africains impliquent souvent des classes alimentaires mélangées, rendant la segmentation précise difficile. De plus, la recherche s'est largement concentrée sur les images RGB, qui fournissent des informations sur la couleur et la texture mais pourraient manquer de suffisamment de détails géométriques. Pour y remédier, la segmentation RGB-D combine des données de profondeur avec des images RGB. Les images de profondeur fournissent des détails géométriques cruciaux qui enrichissent les données RGB, améliorent la discrimination des objets et sont robustes face à des facteurs tels que l'illumination et le brouillard. Malgré son succès dans d'autres domaines, la segmentation RGB-D pour les aliments est peu explorée en raison des difficultés à collecter des images de profondeur des aliments.

Cette thèse apporte des contributions clés en développant de nouveaux modèles d'apprentissage profond pour la segmentation d'images RGB (mid-DeepLabv3+) et RGB-D (ESeNet-D) et en introduisant les premiers ensembles de données axés sur les images alimentaires africaines. Mid-DeepLabv3+ est basé sur DeepLabv3+, avec un backbone ResNet simplifié et une couche de saut (middle layer) ajoutée dans le décodeur, ainsi que des couches mécanisme d'attention SimAM. Ce model offre un excellent compromis entre performance et efficacité computationnelle. ESeNet-D est composé de deux branches d'encodeurs utilisant EfficientNetV2 comme backbone, avec un bloc de fusion pour l'intégration multi-échelle et un décodeur employant des convolutions auto-calibrée et interpolations entrainées pour une segmentation précise. ESeNet-D surpasse de nombreux modèles de référence RGB et RGB-D tout en ayant une charge computationnelle plus faible. Nos expériences ont montré que, lorsqu'elles sont correctement intégrées, les informations relatives à la profondeur peuvent améliorer de manière significative la précision de la segmentation des images alimentaires.

Nous présentons également deux nouvelles bases de données : AfricaFoodSeg pour la segmentation « aliment/non-aliment » avec 3067 images (2525 pour l'entraînement, 542 pour la validation), et CamerFood, axée sur la cuisine camerounaise. Les ensembles de données CamerFood comprennent CamerFood10 avec 1422 images et dix classes alimentaires, et CamerFood15, une version améliorée avec 15 classes alimentaires, 1684 images d'entraînement et 514 images de validation. Enfin, nous abordons le défi des données de profondeur rares dans la segmentation RGB-D des aliments en démontrant que les modèles MDE peuvent aider à générer des cartes de profondeur efficaces pour les ensembles de données RGB-D.

Title: Contribution to the development of an intelligent system of quantification of nutrients in meals from subsaharan Africa.

Key words: Semantic Segmentation, Food Image, Multimodal Segmentation, RGB-Depth Image, Deep Learning, CNNs

Abstract: Malnutrition, including under- and overnutrition, is a global health challenge affecting billions of people. It impacts all organ systems and is a significant risk factor for noncommunicable diseases such as cardiovascular diseases, diabetes, and some cancers. Assessing food intake is crucial for preventing malnutrition but remains challenging. Traditional methods for dietary assessment are labor-intensive and prone to bias. Advancements in AI have made Vision-Based Dietary Assessment (VBDA) a promising solution for automatically analyzing food images to estimate portions and nutrition. However, food image segmentation in VBDA faces challenges due to food's non-rigid structure, high intra-class variation (where the same dish can look very different), inter-class resemblance (where different foods appear similar) and scarcity of publicly available datasets.

Almost all food segmentation research has focused on Asian and Western foods, with no datasets for African cuisines. However, African dishes often involve mixed food classes, making accurate segmentation challenging. Additionally, research has largely focus on RGB images, which provides color and texture but may lack geometric detail. To address this, RGB-D segmentation combines depth data with RGB images. Depth images provide crucial geometric details that enhance RGB data, improve object discrimination, and are robust to factors like illumination and fog. Despite its success in other fields, RGB-D segmentation for food is underexplored due to difficulties in collecting food depth images.

This thesis makes key contributions by developing new deep learning models for RGB (mid-DeepLabv3+) and RGB-D (ESeNet-D) image segmentation and introducing the first food segmentation datasets focused on African food images. Mid-DeepLabv3+ is based on DeepLabv3+, featuring a simplified ResNet backbone with and added skip layer (middle layer) in the decoder and SimAM attention mechanism. This model offers an optimal balance between performance and efficiency, matching DeepLabv3+'s performance while cutting computational load by half. ESeNet-D consists on two encoder branches using EfficientNetV2 as backbone, with a fusion block for multi-scale integration and a decoder employing self-calibrated convolution and learned interpolation for precise segmentation. ESeNet-D outperforms many RGB and RGB-D benchmark models while having fewer parameters and FLOPs. Our experiments show that, when properly integrated, depth information can significantly improve food segmentation accuracy. We also present two new datasets: AfricaFoodSeg for "food/non-food" segmentation with 3,067 images (2,525 for training, 542 for validation), and CamerFood focusing on Cameroonian cuisine. CamerFood datasets include CamerFood10 with 1,422 images from ten food classes, and CamerFood15, an enhanced version with 15 food classes, 1,684 training images, and 514 validation images. Finally, we address the challenge of scarce depth data in RGB-D food segmentation by demonstrating that Monocular Depth Estimation (MDE) models can aid in generating effective depth maps for RGB-D datasets.