Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
ΔΠΜΣ ΕΔΕΜΜ

Evaluating deep instance segmentation methods for mushroom detection on proximate sensing datasets

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΧΑΡΙΣΗΣ ΧΡΗΣΤΟΣ

Επιβλέπων : Καράντζαλος Κωνσταντίνος

Αναπληρωτής Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2023

Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
ΔΠΜΣ ΕΔΕΜΜ

Evaluating deep instance segmentation methods for mushroom detection on proximate sensing datasets

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΧΑΡΙΣΗΣ ΧΡΗΣΤΟΣ

Επιβλέπων : Καράντζαλος Κωνσταντίνος

Αναπληρωτής Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 23η Φεβρουαρίου 2023.

.....	.....	.....
Κωνσταντίνος Καράντζαλος	**Αθανάσιος Βουλόδημος**	**Δημήτριος Αργυρόπουλος**
Αναπλ. Καθηγητής Ε.Μ.Π.	Επίκουρος Καθηγητής Ε.Μ.Π.	Assistant Professor UCD

Αθήνα, Φεβρουάριος 2023

……

Χρήστος Χαρίσης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Τα τελευταία χρόνια, ο τομέας του Deep Learning (DL) έχει γνωρίσει μεγάλη πρόοδο, καθώς παρατηρείται μια τάση προς τη χρήση μοντέλων DL για πολλές εφαρμογές. Ένας μεγάλος αριθμός εφαρμογών εστιάζει στην επεξεργασία εικόνας και πιο συγκεκριμένα, πάνω στο instance segmentation, το οποίο έχει αποκτήσει μεγάλη δημοτικότητα λόγω της προόδου των πιο πρόσφατων μελετών.

Ένας πολύ σημαντικός τομέας που αποτελεί σημαντικό μέρος της ανθρώπινης ζωής και ενσωματώνει καθημερινά νέες τεχνολογίες είναι ο τομέας της γεωργίας. Η υιοθέτηση νέων τεχνολογιών κατέστησε δυνατή τη συλλογή μεγάλου όγκου δεδομένων που απαιτούνται για την εκπαίδευση των πολύπλοκων αρχιτεκτονικών, όπως αυτών του DL. Μεταξύ των δεδομένων διαφόρων μορφών, οι εικόνες των καρπών που καλλιεργούνται είναι ένα από τα πιο κοινά δεδομένα που συλλέγονται στη γεωργία. Η ικανότητα ανίχνευσης και εξαγωγής πληροφοριών για τους καλλιεργούμενους καρπούς μέσω εικόνων είναι υψίστης σημασίας για τους γεωργικούς στόχους καλύτερης ποιότητας και ποσότητας των καλλιεργειών.

Από τις πολλές περιπτώσεις στη γεωργία, η καλλιέργεια των μανιταριών έχει αποδειχθεί μια πολύ τεχνική διαδικασία που απαιτεί τη χρήση όλων των διαθέσιμων μέσων συλλογής πληροφοριών κατά την ανάπτυξή τους, προκειμένου να επιτευχθεί καλύτερη απόδοση και ποιότητα. Επιπλέον, τα μανιτάρια στη φύση παρουσιάζουν μια ενδιαφέρουσα περίπτωση για τον εντοπισμό νέων ειδών, την παρακολούθηση του μικροκλίματος μιας περιοχής και τη δυσκολία εντοπισμού των μανιταριών εικόνες με πολύπλοκα φόντα φυσικού περιβάλλοντος στην εικόνα.

Σε αυτή τη διπλωματική εργασία, συλλέχθηκαν δύο σύνολα δεδομένων μανιταριών και χειροκίνητα δημιουργήθηκαν μάσκες για την υλοποίηση instance segmentation. Το ένα σύνολο δεδομένων περιλαμβάνει συστάδες μανιταριών Pleurotus, που καλλιεργούνται σε τεχνητό περιβάλλον και το άλλο περιέχει διαφορετικά είδη μανιταριών σε φυσικά περιβάλλοντα. Μοντέλα instance segmentation τελευταίας τεχνολογίας εκπαιδεύτηκαν και δοκιμάστηκαν πάνω σε αυτά τα σύνολα δεδομένων προκειμένου να εξαχθούν πολύτιμες πληροφορίες σχετικά με την απόδοσή τους και τον τρόπο με τον οποίο τα διαφορετικά χαρακτηριστικά αυτών των δύο συνόλων δεδομένων επηρεάζουν τα αποτελέσματα. Τα αριθμητικά και απεικονιστικά αποτελέσματα μεταξύ των μοντέλων παραθέτονται και παρουσιάζεται σύγκριση τους.

Τα συμπεράσματα αυτής της εργασίας είναι ότι τα χαρακτηριστικά ενός συνόλου δεδομένων και ο όγκος των διαθέσιμων δεδομένων για εκπαίδευση έχουν μεγάλο ρόλο στην απόδοση ενός μοντέλου. Ωστόσο, πιο σύνθετα μοντέλα που μπορούν να εξάγουν καλύτερα χαρακτηριστικά από τα αρχεία εισόδου και να τα διαδώσουν με πιο αποτελεσματικό τρόπο στα διάφορα δομικά στοιχεία της αρχιτεκτονικής τους μπορούν να επιτύχουν καλά αποτελέσματα ακόμη και με μικρό αριθμό διαθέσιμων δεδομένων.

Abstract

In the recent years, the domain of Deep Learning (DL) has experienced a great progress, as a trend towards the use of DL models for many applications is observed. A great number of applications is implemented on image processing and more recently the task of instance segmentation has gained great popularity due to breakthroughs in the latest studies.

A very important domain, that is a crucial part of the human life and integrates every day new technologies, is the domain of agriculture. The adoption of new technologies has made possible the collection of great amounts of data that are required to train the complex architectures of DL. Among the data of various formats, images of crops, that are cultivated, are one of the most common data collected in agriculture. The ability to detect and extract information for the cultivated crops through images is of paramount importance for the agricultural targets of better crops' quality and quantity.

From the many agricultural cases, the cultivation of mushrooms has proven to be a very technical process, that requires the use of all the available means to collect information during their growth, in order to achieve better yield and quality. Moreover, mushrooms in nature present an interesting case for identifying new species, monitoring the micro-climate and the difficulty of finding mushroom instances in natural complex backgrounds.

In this thesis, two mushroom datasets were collected and manually annotated for the task of instance segmentation. One dataset includes Pleurotus mushroom clusters, cultivated in a controlled environment and the other contains different mushroom species in natural environments. State-of-the-art instance segmentation models were finetuned and tested on these datasets in order to extract valuable information regarding their performance and how the different characteristics of these two datasets affect the results. Arithmetic and imaging results are provided and comparison between the models is presented.

The conclusions of this work are that the dataset's characteristics and the amount of available data for training exert a great role in the performance of a model. However, more complex models that can extract better information and circulate it in a more efficient way through their different components can achieve good results even with small number of available data.

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον κ. Κωνσταντίνο Καράντζαλο ο οποίος ήταν ο επιβλέπων και μου έδωσε την δυνατότητα να εκπονήσω την διπλωματική εργασίας μου, καθώς και για την κατανόηση και τη βοήθεια που μου προσέφερε σε όλα τα στάδια του μεταπτυχιακού.

Επίσης, θα ήθελα να ευχαριστήσω θερμά τον κ. Δημήτριο Αργυρόπουλο για τη βοήθεια και την καθοδήγησή του εντός και εκτός αυτής της διπλωματικής εργασίας. Παράλληλα, θα ήθελα να ευχαριστήσω τους συναδέλφους της εταιρείας SCiO, στην οποία εργάζομαι, για την κατανόηση και την υποστήριξή τους καθ' όλη τη διάρκεια του μεταπτυχιακού προγράμματος.

Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου, αλλά και φίλους μου οι οποίοι υπήρξαν δίπλα μου κατά τη διάρκεια της μεταπτυχιακής μου πορείας, βοηθώντας με ο καθένας με το δικό του τρόπο σε καλές και κακές στιγμές.

Χαρίσης Χρήστος,

Αθήνα, Φεβρουάριος 2023

Content table

1 Introduction

In the recent years, a trend towards the use of Deep Learning (DL), a sub-domain of machine learning (ML), is being observed. DL is constituted from more complex and larger models than the traditional ML. This complexity leads to the requirement of greater amounts of data for the training. However, a well-trained DL model in most cases will outperform a traditional ML model [1]. Many applications of DL models have been observed in various domains, with one of them being the important domain of agriculture, where tasks of yield prediction, crop management, growth analysis, disease detection among others are being explored [1]. At the same time, the technology advancement of the recent years introduced the use of many sensors in agriculture, enabling the extensive data collection [2].

The combination of better achievable results from DL models and the ability to collect large amounts of data, has led to many DL implementations in agriculture in general, but also specifically in controlled environment agriculture (CEA) [3]. As both natural environment and crop production environment are complex environments regarding background, elements present and illumination among others, the need for proper and robust detection and localization of crops/fruits in these environments becomes apparent. This need can be fulfilled utilizing DL models, as they are able to learn complex patterns and features in the images and detect the targeted crops in challenging conditions. The accurate localization can be considered as the first part of a series of applications, for example species classification, growth monitoring and yield estimation. Moreover, in CEA, the detailed detection can be integrated in disease monitoring and automated robotic harvesting applications.

The first steps in the detection and localization utilized Computer Vision (CV) methods with human-designed features extractors based on color, shape and textual characteristics [4,5,6,7,8]. The extracted features were analyzed by traditional ML models for example SVM [5,9,10] and Artificial Neural Networks [5,11]. These implementations produced good results for the dataset they analyzed but cannot be generalized in different illumination settings and backgrounds, due to the manually engineered features. An advantage of DL over ML is the more effective feature extraction from raw data done in an automatic manner [1]. This feature extraction advantage results to DL models outperforming their ML counterparts in agriculture applications of image processing [1]. Researchers started to shift their focus on

implementing DL architectures to image processing to achieve superior overall results in areas such as fruit detection and counting [12,13,14,15,16,17,18]. Many of these studies are using the two very popular object detection architecture families of Faster R-CNN [19] and You Only Look Once (YOLO)[20], which dominated the object detection domain and are the basis for many other developed models to this day.

However, after the breakthrough publication of Mask R-CNN [21], a DL architecture that expanded the object detection of Faster R-CNN into an instance segmentation architecture, many studies integrated Mask R-CNN and custom variations of it to instance segmentation task in agriculture, as well as other implementations of instance segmentation architectures. Fruit detection is the most popular task of the instance segmentation in agriculture, as many studies focus in detecting and localizing fruits for example apple [22], strawberry [23], mango [24] and grapes [25] among others.

Yu et al. [23] used vanilla Mask R-CNN to find strawberries and classify them to two classes (ripe, unripe). The high accuracy of the localization was then utilized for position estimation of the strawberries, allowing their collection from a harvesting robot. Perez-Borrero et al. [26] modified Mask R-CNN in order to create a lighter architecture for strawberry detection with lower inference time (achieved 10 fps vs. vanilla 5 fps) for potential use in robotic harvesting. In his next attempt to develop a faster architecture, Perez-Borrero et al. [27] implemented a custom architecture based on UNet fully convolutional network combined with a grouping and filtering algorithm, outperforming his previous work, meeting the precision and speed requirements needed for real-time harvesting systems.

Gene-Mola et al. [28] aimed on constructing precise 3D apple localization, using Mask R-CNN architecture for the 2D image analysis before applying structure-from-motion (SfM) photogrammetry. Despite the good results for the instance segmentation part, the whole implementation was deemed unfit for real-time robotic harvesting as it presented high processing times for the 3D analysis. In an effort to develop a real-time robotic harvesting application, Kang et al. [29] implemented a custom single-stage instance segmentation architecture, based on the YOLO models family, with a lighter backbone network, achieving robust and efficient results.

Experimenting with the offline task of detecting apple flowers in different growth stages, Tian et al. [22] proposed a custom architecture, combining an extended version of Mask R-CNN, namely Mask Scoring R-CNN, with a U-Net backbone, which

outperformed other state-of-the-art models. Shen et al. [30] proposed a new backbone network, introducing an attention mechanism and dense up-sampling convolution, in order to improve the performance of Mask R-CNN on problems introduced by illumination, occlusion conditions and the variability of grape cluster shape in the dataset.

Regardless of the examined crop/fruit, robotic application of localizing and harvesting was taken into consideration in many of the recent studies. Ganesh et al. [31] proposed the combined use of RGB and HSV images from a Mask R-CNN model for improved localization of oranges on trees, for robotic harvesting. Liu et al. [32] utilizing Mask R-CNN, proposed a custom implementation for cucumber localization in a greenhouse, taking into account the color and the shape of the target crop. The results achieved were compared with other state-of-the-art architectures, outperforming them. Moving away from the Mask R-CNN, Chen et al. [33], created a custom autoencoder architecture, based on VGG16 network, applying it on a synthetic dataset of sweet pepper images. Santos et al. [25] introduced Embrapa dataset, containing images of grape clusters for instance segmentation. A comparison between two architectures of the YOLO family and Mask R-CNN, to examine the performance on the new dataset, with Mask R-CNN presenting superior results compared to the YOLO models. An end-to-end vision system for mango picking robot was designed by Zheng et al. [24] integrating a Mask R-CNN model inside a framework of detecting the fruits and then locating picking points for the harvesting robot. The model was proven robust against various illuminations and complex image backgrounds. Wang et al. [34] developed a custom implementation using Mask R-CNN as the baseline model. A comparison was made on a dataset of multiple fruits, with the proposed model achieving better results compared to other instance segmentation models. Extending an object detection architecture, Jia et al. [35], developed FoveaMask, an instance segmentation architecture in which the dependency on anchor boxes, a characteristic of the Mask R-CNN family, is eliminated and an attention mechanism is utilized. The model was trained on a dataset comprised of immature green apples and persimmon, achieving better results compared with 11 different types of detection and segmentation models.

One of the most significant concerns in agriculture disease control [36], as disease outbreaks can diminish the yield and/or the quality of any cultivar at any environment. Detecting diseases through images has become a task of paramount importance in terms of rapid reaction and disease countermeasures. Instance

segmentation architectures not only can detect the area of interest but also, classify the disease and extract geometrical information through the detailed instance mask being created. Afzaal et al. [37] used Mask R-CNN to detect disease on strawberries and to classify the detected areas into 7 disease classes. Different data augmentation techniques are explored and a new custom strawberry disease dataset is presented. Tassis et al. [38] developed a three-part framework with Mask R-CNN as the first part, responsible for identifying and segmenting green leaves, while the next parts detected disease areas and classified them between five diseases. With the introduction of a new dataset for disease detection on grape clusters and leaves, Rossi et al. [39] applied Mask R-CNN and Recursively Refined R-CNN (architecture in the family of Mask R-CNN) with good results, to prove the validity of their dataset.

Despite the many recent studies incorporating instance segmentation for fruit/crop detection in robotic harvesting, disease detection and growth analysis, to the best of our knowledge little to no such work has been recorded so far on mushrooms. A gap is identified, in applying instance segmentation on images of mushroom both in natural environment and in production farms.

Mushrooms are an important part of our ecosystem due to their nutritional value, medicinal properties, and potential as a sustainable and eco-friendly crop [40]. They can be found in the natural environment [41,42] but also, cultivated in dedicated mushroom farms [43,44,45]. Regarding the natural environment, different mushrooms species can be found in areas where specific conditions are ideal for their growth [46]. Mushroom detection and localization in a natural environment present an important first task that can help scientists and conservationists to track, classify and study different mushroom species, as well as help ordinary people define their edibility [47]. However, detection in natural environments involves challenging conditions such as bad lighting conditions, complex background or wide variety of changes in mushroom shape and size due to different species.

On the other hand, the nutritional value of the mushrooms has increased the demand for big amounts for consumption. World production of cultivated, edible mushrooms has increased more than 30-fold since 1978, which is combined with an increase in per capita consumption at a relatively rapid rate [48]. Some of the most widespread mushrooms are several species of the genus Pleurotus. These are of particular interest because their production accounts to ca. 30% of the total, corresponding to the fastest growing and most profitable section of the mushroom

market [49]. The production involves four main steps: preparation of substrate, inoculation, incubation and fruiting, [50], that are highly dependent on environmental parameters, mainly temperature, humidity and CO2 concentration in the growing rooms [51]. These factors exert a great effect on both growth and yield, thus the correlation between environmental parameters and the visual growth through collected images is of great importance. Unlike the fruiting bodies of other mushrooms, those of oyster mushrooms grow in clusters, which make them more difficult to measure their growth based on cap size and stipe length. This fact demands as a first step to detect and monitor the mushroom clusters and the individual instances that comprise them.

For both aforementioned mushroom cases, the existence of big amounts of data is necessary to achieve the required detection result. An abundance of mushroom images in natural environments that can be found in public datasets [52], including a great variety of mushroom species images. In the case of mushroom cultivation in dedicated farms, the special conditions, under which the cultivation is conducted, render it a very technical process and represents a special case of CEA. This controlled environment is protected and allows a space with precisely regulated environmental and cultural variables to produce yield in a more efficient way, by adopting the use of advanced technology. The use of sensors and actuators, which are located inside the structure to properly monitor the crop/microclimate interaction, control environmental parameters, manage cultivation factors and collect mushroom growth data. Many different types of sensors and IoT implementations were reported in the recent literature [53,54], including cameras for image acquisition. In this way, data can be collected, creating large data banks that contain environmental and most importantly, image data from inside the cultivation installations.

This thesis will explore the implementation of different instance segmentation DL architectures with the aim of detecting mushroom instances on two datasets with different characteristics. The instance detection is a useful first step in extracting fine-grained information, for example the number of individual mushrooms comprising a cluster, the shape and the size among others. This information can be useful for bigger applications, exploring yield prediction and monitoring mushroom growth with correlation on the environmental variables inside the production environment, or localizing mushrooms in natural environments and classify them as edible or not.

The rest of the thesis is structured as follows. In chapter 2, the two datasets are presented and information is given regarding their images and the annotation task. The

DL architectures that are used are described in chapter 3. In chapter 4, the results are presented and discussed and finally, in chapter 5s, conclusions and suggestions for future work are made.

2 Datasets

2.1 Pilze dataset

The first dataset is comprised of oyster mushroom clusters (Pleurotus ostreatus (Jacq. Ex Fr.) P. Kumm). It was created using sets of images acquired from a commercial mushroom farm in June 2022 at Pilze-Nagy Ltd (Budapest, Hungary). Images of unharvested mushroom clusters were carefully acquired at different angles using the dual 12MP camera system of an iPhone 12 mobile phone with a high resolution of 4032x3024 pixels. Field photography mainly occurred during the daytime under artificial lighting conditions, capturing in detail every single mushroom in a cluster. A total of 200 RGB images including different sizes, shapes and distribution densities of raw oyster mushrooms in clusters were captured. Attention was given in capturing images of the clusters from all sides. In Figure 1, the mushroom farm is shown as well as three images from the horizontal, bottom and top views.

Figure 1 Mushroom farm (a) with artificial lighting. Three sample mushroom cluster images from the collected dataset at different angles: front (b), bottom (c), top (d) views obtained from a real production environment.

Moreover, in the dataset exist clusters with different numbers of individual mushrooms, ranging from numbers smaller than 10 up to even more than 30. In Figure 2, two clusters with different number of mushroom instances are presented.

(a) (b)

Figure 2 Mushroom cluster with small (a) and large (b) number of mushroom instances.

As it can be observed from both Figures 1 and 2, many instances are semi-occluded. The visible fraction of some mushrooms is relatively small, posing a challenge for the instance segmentation task.

Figure 3 Mushroom cluster for top view before and after annotation.

The collected raw oyster mushroom dataset was processed into formats that can be used to train and evaluate deep neural network architectures. After image acquisition, the regions of oyster mushroom clusters in the images were manually annotated using an open-source image and video annotation tool (CVAT: Computer Vision Annotation Tool), creating the labels required for model training, validation and testing.

Specifically, the mushrooms in the images were carefully annotated, defining the boundaries of the individual mushrooms present in each image (Figures 3 and 4). This process resulted in a collection of user-defined masks for each mushroom that was considered as an instance of interest for the segmentation.

Figure 4 Mushroom cluster for bottom view before and after annotation. On the right image the points used to define the borders of a mushroom instance are visible.

The annotated dataset was exported and saved in COCO data format. The dataset was divided into three subsets, 75% of the data were randomly selected for training, 10% of the data for validation and 15% of the data for testing.

2.2 Google V6 dataset

Open Images is a dataset of ~9M images annotated with image-level labels, object bounding boxes, object segmentation masks, visual relationships, and localized narratives [52]. Overall, there are 19,958 distinct classes with image-level labels. However, this number is slightly higher than the number of human-verified labels. Trainable classes are those with at least 100 positive human-verifications in the V6

Figure 5 Original mushroom annotation from V6 google dataset for instance segmentation task.

training set. Based on this definition, 9,605 classes are considered trainable and machine-generated labels cover 9,034 of these. For this thesis, the class “mushroom” was extracted from the dataset, for the segmentation task subset. The annotation of the mushroom dataset was not extensive or in the correct format for the training of the models (Figure 5) and the need arose for the annotation of the dataset from scratch.

The class “mushroom” contained many different mushrooms species, in contrast to the other dataset that is used in this thesis. Furthermore, some of the images inside the dataset contained images from artificial or cooked mushrooms. These images were filtered out of the dataset as the purpose was to examine the detailed detection and localization of mushroom instances in natural environment. In Figure 6 are presented images from different mushroom species contained in the dataset.

Figure 6 Example mushroom images from V6 google dataset for instance segmentation task.

From a total number of almost 1500 mushroom images, 1407 were used after the filtering of artificial mushrooms, very low-resolution images and mushrooms in

artificial environments. The V6 dataset has the data split by default in train, validation and test sets. After the elimination of the aforementioned bad samples, 1238 (~88%), 39 (~3%) and 130 (~9%) images were used respectively for train, validation and test set. The eliminated images where proportional of the size of each set, without affecting the final percentage of each set. The resolution of the images never exceeds 1024x1024.

The different mushroom species contained in the dataset, introduce a great variance of number, shape and colors in the images. Some mushroom images are single instances, for example, in Figure 6.f, while other mushroom species grow in compact clusters where the instances cannot be distinguished easily, as it can be seen from Figure 6.c, leading to the characterization of the whole cluster as a mushroom instance. Regarding the shape of the mushrooms, it is visible from Figure 6.b and 6.e, how different can be the instance of a mushroom for different species. All this complexity can be a difficult task, even for a DL architecture. Another important factor that adds complexity in the instance segmentation task is the background in each image. In some cases, for example in Figure 6.a, the background is very distinct from the mushrooms, as the colors are different. This is not the case for Figure 6.f, where the colors of the background are very similar with the ones of the mushroom instance inside the image.

Figure 7 Mushroom images before and after annotation. On the right images the points used to define the borders of a mushroom instance are visible.

The annotation of this dataset was conducted in a similar fashion as the previous dataset, using the online CVAT tool. The mask of each mushroom instance was manually and carefully defined in a time-consuming procedure. In Figure 7, examples of annotated images from the V6 dataset are presented.

3 Instance segmentation architectures

For the instance segmentation task on mushroom instances in this thesis, the open-source object detection toolbox based on PyTorch named MMDetection [55] was utilized. It is a part of the OpenMMLab project. MMDetection contains and supports many popular and state-of-the-art architectures for object detection, instance segmentation and panoptic segmentation. Moreover, the toolbox contains weights for more than 200 pre-trained networks, making the toolbox an instant solution if the available dataset is not big enough to train an architecture from scratch. One of the most important advantages of MMDetection is that many simple modular components of a typical object detection/instance segmentation frameworks are implemented. With selecting and combining them, custom pipelines or a custom model can be built. Also, building a new detector framework on top of an existing framework and comparing its performance is easily possible with this toolbox's benchmarking capabilities.

For this thesis, six instance segmentation architectures where selected, trained and tested on the two aforementioned mushroom datasets. The models were pretrained on COCO dataset, with only the final parts of each architecture being finetuned with the training set of the datasets. The training and testing of each dataset were done independently from each other, with the aim of examining how each architecture could perform on artificial and natural environments for mushroom detection. The six architectures will be discussed in further detail in the following sub-chapters. The first one is Mask R-CNN, on which other four are based on. Moreover, some of these 4 Mask R-CNN based architectures are extensions of each other.

All the examined architectures have as their first part a feature extractor component or else named backboned network. This network is responsible for extracting features from the initial input data provided to the network. Usually, this is comprised of a pre-trained CNN model such as ResNet [56]. For five out of the six architectures, two different pre-trained backbones were examined, one that had fewer layers or was based on a simpler model and one that had more layers or was based on more complex model, giving the name "light" and "heavy" respectively to the whole model. These architectures were selected among many, that were provided by the toolbox, based on their accuracy on the benchmark dataset of COCO, but also with respect to the ability that hardware demands could be met by the available system. The only architecture that was not examined with two different backbones was DetectorRS [57], due to hardware limitations for the provided "heavy". The one DetecoRS

architecture used is categorized as “light”. Another quantitative differentiating factor between the “light” and “heavy” models is the amount of GPU RAM memory that the models required to run, with the “light” models demanding a lot less than the “heavy” ones.

Finally, the computer hardware configuration used for image processing and model training in this experiment is as follows: CPU: AMD Ryzen Threadripper 1950X 16-Core Processor 3.40 GHz; GPU: NVIDIA GeForce RTX 3090 (24 GB memory); RAM: 64 GB; SSD: 512G. The network model is trained under Windows 10 Professional 64-bit operating system. The next sub-chapters will present the architectures used and highlight differences and similarities between them.

3.1 Mask R-CNN

Mask R-CNN is considered as a breakthrough in the domain of instance segmentation, expanding the object detection architecture of Faster R-CNN [33], with the inclusion of a fully convolutional network (FCN) for semantic segmentation. This model is part of the family of the two-stage detection models, where the first step is comprised from a model tasked with the extraction of the regions of interest (ROIs), while the second step is the classification and the detailed localization of the object inside each ROI. Overall, the architecture of Mask R-CNN can be divided into three parts, if the feature extractor part is considered as a distinct part (else it can be considered as part of stage 1), as it can be seen in figure 8.

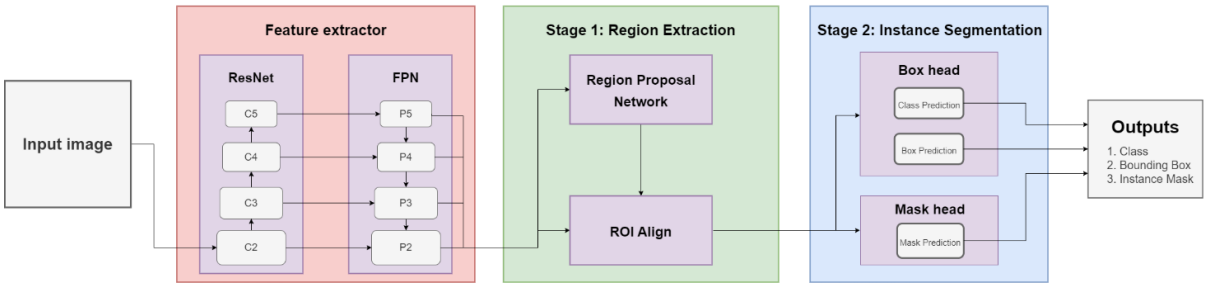


Figure 8 Mask R-CNN architecture, divided into distinct parts.

The first part of the network, namely the backbone, is acting as a feature extractor for the input image, with convolutional neural networks (CNNs) being utilized in an FCN architecture. In Mask R-CNN case, the backbone is comprised of a ResNet [56] + Feature Pyramid Network (FPN) [58] network, which is visualized in Figure 9. ResNet architecture is can effectively solve the gradient vanishing problem of deeper architectures with many layers, making possible the use of multi-layers networks (for

example ResNet50, ResNet101). Moreover, ResNet is a bottom-up architecture, which detects low-level features (in the first layers) and high-level features (in the last layers). With the introduction of the FPN, the backbone achieves the integration of a top-down architecture, combining effectively low- and high-level features in a multi-scale feature fusion.

The output of the backbone is a feature map, that is used from the second part of the network, namely the Region Proposal Network (RPN). RPN is tasked with finding objects inside the image, through the feature map, by using a set of boxes with predefined size and scale, named anchors. The anchors are being slid through the feature map and at each location try to decide whether they contain a possible object or not, after comparing it with the ground truth data. The output of RPN is the collection of all anchors, with two additional values, the anchor class (foreground or background) and the coordinates of the bounding box of the anchor. The anchors with the highest confidence of containing an object are selected in order to alleviate the computation cost of processing all the anchors.

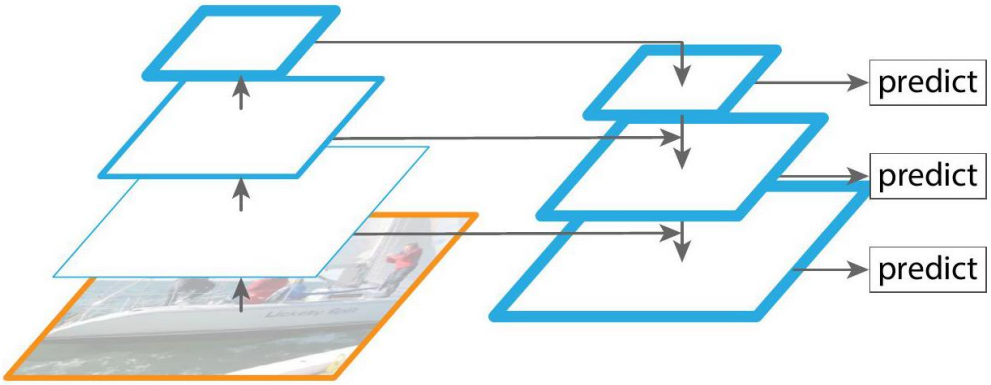


Figure 9 FPN architecture [58], divided into distinct parts. The left part of the architecture is a backbone network (usually ResNet or similar implementation) and the right part is the FPN addition. Depending on the application one or more of the "predict" stages can be utilized as feature from the following components of an architecture.

The third and final part is comprised of the Region of Interest (ROI) Align component and the object detection/mask generation branches. ROI Align is responsible for extracting the features of the feature map (produced from the backbone) that are contained in the proposed regions of the RPN and transforming them into fix-sized arrays for standardized processing. The fix-sized arrays are then pass through two branches producing on one end, the final class prediction and bounding box coordinates

(box head branch) and on the other end the pixel-wise mask of the detected instance (mask branch).

In this thesis, two different backbones have been examined, to give insight how different feature extractors can affect the results. The first backbone, belonging in the aforementioned category of "light" models, was comprised from a ResNet network of 50 layers combined with an FPN, while the second backbone, in the "heavy" category, was comprised of a ResNeXt [59] network of 101 layers with 32x8d (cardinality x bottleneck width) combined with an FPN. The models used were pretrained on COCO dataset and finetuning was done on the stage 2 of the architecture using the two mushroom datasets.

3.2 Mask Scoring R-CNN (MS R-CNN)

Mask Scoring R-CNN [60] is an extension of the Mask R-CNN architecture. The main difference between Mask R-CNN and Mask Scoring R-CNN is the addition of a scoring branch that is used to predict the quality of the instance segmentation mask. This branch can be visually seen in Figure 10, with the name of MaskIoU head.

The MaskIoU head aims to estimate the IoU between the predicted mask and its ground truth mask. The input of this component is concatenation of feature from RoI Align layer and the predicted mask and generates a score for each mask, indicating the quality of the mask. This score is computed by multiplying the predicted MaskIoU and classification score. Thus, mask score is aware of both semantic categories and the instance mask completeness [60]. This allows the model to distinguish between high-

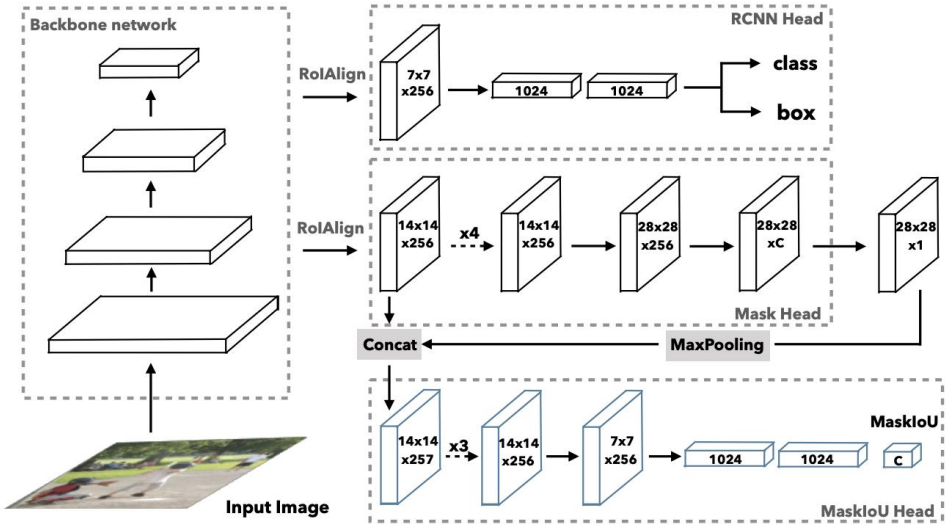


Figure 10 Mask Scoring R-CNN architecture, divided into distinct parts [60].

quality and low-quality masks, and to only keep the high-quality masks for better training.

Two different backbones have been examined for this architecture. The first backbone, considered "light", was comprised from a ResNet network of 50 layers combined with an FPN, while the second backbone, in the "heavy" category, was comprised of a ResNeXt [59] network of 101 layers with 64x8d (cardinality x bottleneck width) combined with an FPN. The models used were pretrained on COCO dataset and finetuning was done on the three heads of the architecture that are visible in Figure 8.

3.3 Cascade Mask R-CNN

Cascade Mask R-CNN [61] is a state-of-the-art object detection and instance segmentation algorithm that improves upon the Mask R-CNN architecture. The main idea behind Cascade Mask R-CNN is to break down the object detection and instance segmentation task into multiple stages, or cascades, each with a different IoU threshold. The stages are trained sequentially, using the output of a stage as training set for the next. Training with small IoU threshold results in noisy samples, and training with large thresholds degrades the performance because of overfitting and low-quality region proposals at inference compared to training. Moreover, models trained using specific IoU thresholds might be suboptimal when evaluated on other IoU thresholds or region proposals of different level of IoU threshold is given. The use of different IoU thresholds for the training of the model can solve these problems and improve the overall performance.

In Figure 11, the difference between the architectures of Mask R-CNN and Cascade Mask R-CNN is shown. *B** are the bounding box regressors, *S** are the mask predictor, *C** are classifiers, *I* is the input image, *H** are network heads.

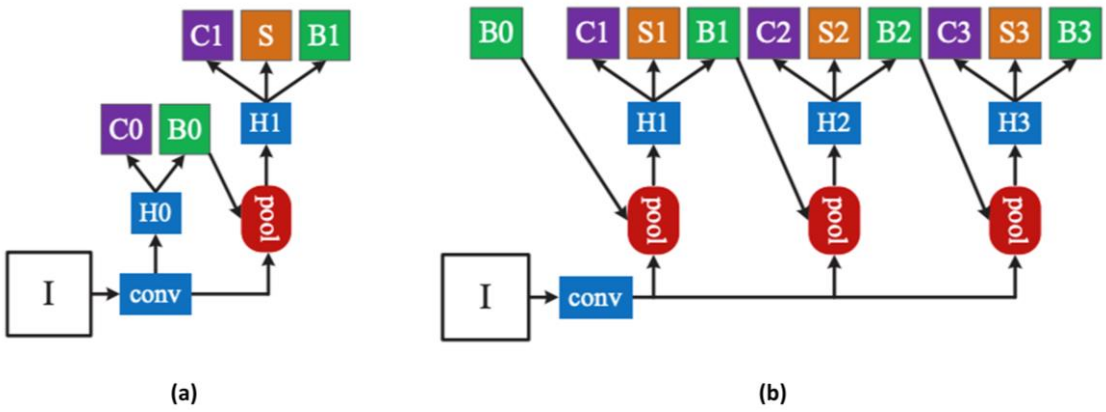


Figure 11 Comparison between Mask R-CNN (a) and Cascade Mask R-CNN (b) [61]

The Cascade Mask R-CNN is comprised of three specialized regressors, each trained with an increasing IoU, the selected values by the authors being [0.5, 0.6, 0.7]

for each regressor. The predicted bounding boxes were forwarded to the next regressor in the pipeline in order to provide a stronger baseline for producing the new bounding boxes with the stricter IoU. It was shown that the use of the previous regressor's bounding box generally improved the quality of the next stage objects proposal. This improvement led to the conclusion that increasing IoU threshold can be used for training to achieve better results. Even though results are produced from all three regressors, the outputs of the last one is regarded as the final result.

Two different backbones have been examined for this architecture. The first backbone, considered "light", was comprised from a ResNet network of 50 layers combined with an FPN, while the second backbone, in the "heavy" category, was comprised of a ResNeXt [59] network of 101 layers with 64x4d (cardinality x bottleneck width) combined with an FPN. The models used were pretrained on COCO dataset and finetuning was done on the heads of each of the three regressors. More specifically the three *H** parts of the architecture that are visible in Figure 9b.

3.4 Hybrid Task Cascade (HTC)

Hybrid Task Cascade (HTC) [62] is a state-of-the-art deep learning architecture for object detection and instance segmentation. It is an extension of the Cascade Mask R-CNN architecture and is considered a multi-stage object detector that uses a cascaded architecture to improve the performance of the object detection and instance segmentation tasks.

In Figure 12, the difference between the architectures of Cascade Mask R-CNN and HTC is shown. *B** are the bounding box regressors, *M** are the mask predictors, *S* is the semantic branch and *F* is the extracted feature map from the backbone.

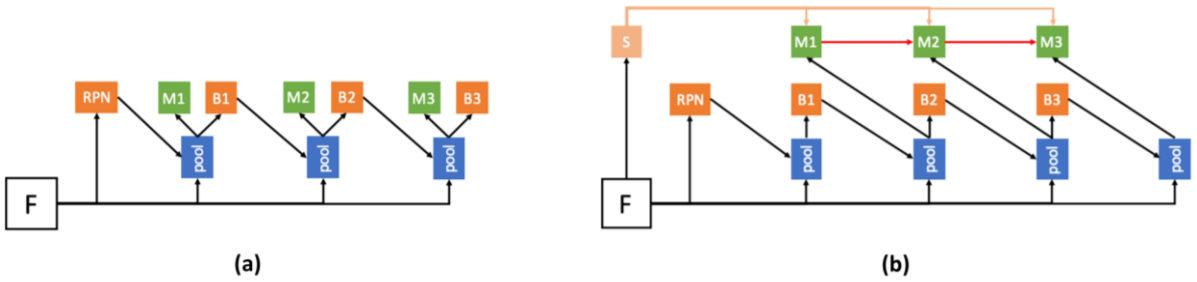


Figure 12 Comparison between Cascade Mask R-CNN (a) and Hybrid Task Cascade (b) [62]

The authors of HTC identified and tried to improve two specific parts of Cascade Mask R-CNN. The first part of the update of the architecture has to do with the mask heads. The new implementation had direct connections between the mask branches of

the cascade part. This strengthens the flow of information between the mask task across all the stages, leading to a progressive refinement of masks, instead of just predicting the masks on progressively refined bounding boxes. Regarding the mask heads, another update was that each mask prediction is done utilizing the updated bounding boxes predicted from the previous stage. This interleaved execution was found to yield better results than executing the bounding box and mask prediction in parallel.

The second update was the introduction of a new branch which predicts pixel-wise semantic segmentation for the whole image. This branch is implemented with FCN and is jointly trained with the other branches. The combination of this semantic segmentation information with the bounding box and mask features produces better predictions, as it is easier to distinguish between the objects and the background area, even if the background is more cluttered and complex than usual.

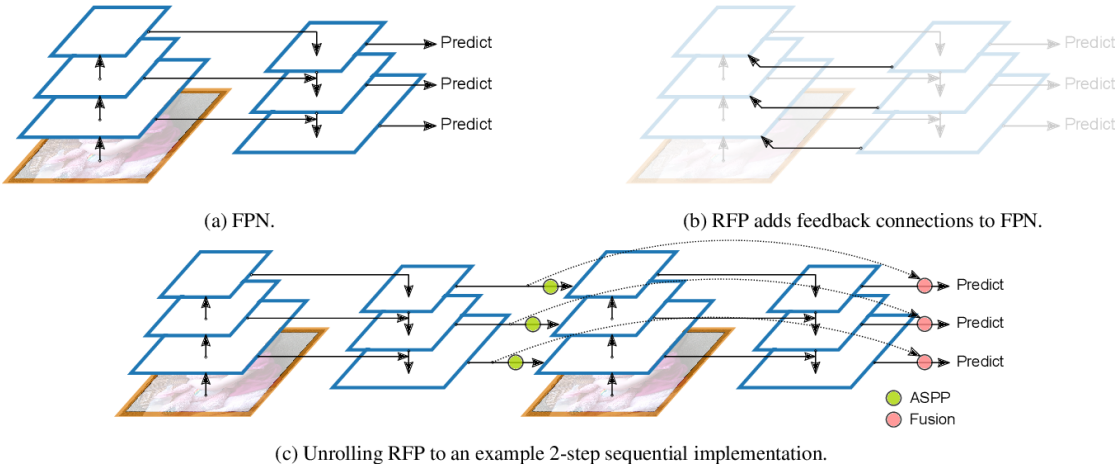
Even though HTC is heavier than Cascade Mask R-CNN, the more complete implementation has led to better results on benchmark datasets, for example COCO Dataset.

Two different backbones have been examined for this architecture. The first backbone, considered “light”, was comprised from a ResNet network of 50 layers combined with an FPN, while the second backbone, in the “heavy” category, was comprised of a ResNeXt [59] network of 101 layers with 64x4d (cardinality x bottleneck width) combined with an FPN. The models used were pretrained on COCO dataset and finetuning was done on the bounding box, mask and semantic components of each of the three regressors. More specifically the three *M**, the three *B** and the *S* components of the architecture that are visible in Figure 10b.

3.5 DetectoRS

DetectoRS [63] is a state-of-the-art object detection algorithm that aims in the improvement of the backbone of instance segmentation architectures. The original implementation was incorporated in an HTC architecture significantly improving the results.

The improvements of the backbone come into two scales, as the authors describe them. The first one is on the macro-level, proposing a Recursive Feature Pyramid (RFP) network. This network builds on the aforementioned FPN, by adding extra feedback connections from the layers of the FPN to the bottom-up backbone layers (usually a ResNet or similar implementation), as it shown in Figure 13.



(a) FPN. (b) RFP adds feedback connections to FPN.

(c) Unrolling RFP to an example 2-step sequential implementation.

Figure 13 Recursive Feature Pyramid (RFP) [63]

In Figure 13a the original FPN network is shown, with connections only from the bottom-up part to the top-down part. By adding connections the opposite way (Figure 13b), the whole architecture becomes recursive and in can be unfolded as in Figure 13c, leading to the architecture looking at the images twice or more. This approach recursively enhances the original FPN leading to the creation of better feature representations.

The second update was done on the micro-level, as the authors describe it. More specifically, each 3x3 convolutional layer in the ResNet backbone was converted to a Switchable Atrous Convolution (SAC) layer. This layer controls the dilation of the kernel used by adding zeros between the kernel values, effectively enlarging the field-of-view. With this technique an object of the same class but with different size can be detected more easily, using the same convolutional weights without adding increasing the existing parameters. In Figure 14, the SAC intuition is presented.

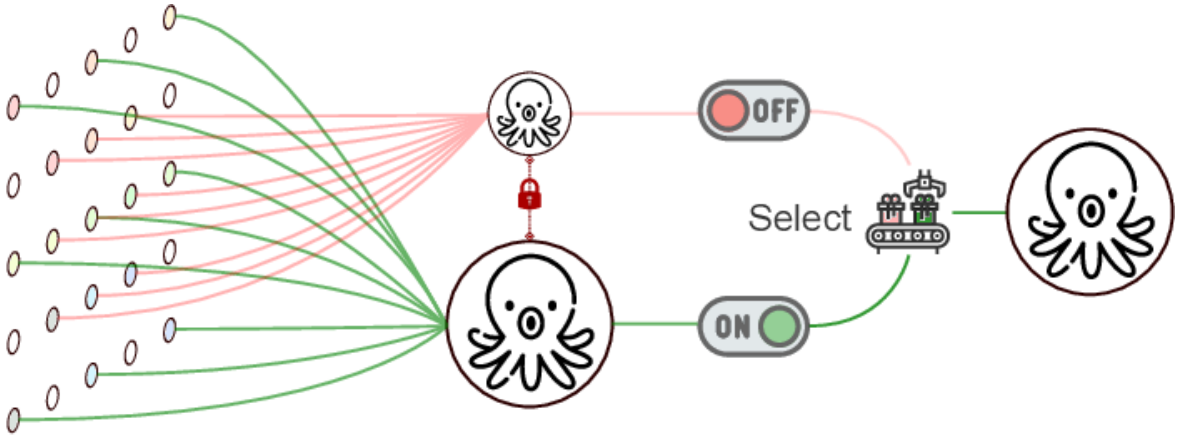


Figure 14 Switchable Atrous Convolution (SAC) layer logic [63].

In Figure 14, the red convolution kernel is the default convolution with 3x3 kernel, while the green convolution kernel has been dilated using 0 between the kernel values. As it can be seen, with the dilated convolution bigger objects can be detected roughly with efficiency using the same kernel. The selection mechanism, controlling which convolution result will be forwarded, is dependent on the input image and the location examined, leading to the model being able to be trained to adapt to different scales as needed.

For this architecture, only one configuration was explored, combining an HTC with ResNet50 as its backbone. As described earlier, the ResNet backbone was updated with the RFP and SAC. A provided implementation of the MMDetection toolbox of a backboned comprised from an HTC with a ResNet101 was exceeding the available hardware capacities and could not be examined. The final parts of the architecture that were finetuned are the same parts described in sub-chapter 3.4 for the HTC architecture and not the updated parts of the backbone.

3.6 Evaluation

Performance metrics such as precision (*P*), recall (*R*) and *F1-score* were used to quantitatively evaluate the detection performance. All detection results were divided into four types: true positive (*TP*), false positive (*FP*), true negative (*TN*) and false negative (*FN*), where TP is the number of oyster mushrooms detected correctly, FP is the number of oyster mushrooms detected incorrectly, and FN is the number of raw oyster mushrooms missed in the cluster. Precision (*P*) and recall (*R*) are defined as follows:

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

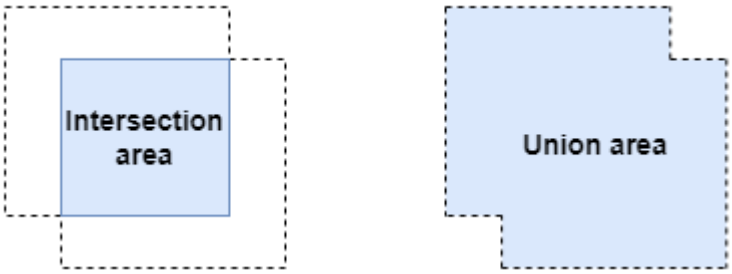
To illustrate how these performance metrics were computed in this work, the Intersection over Union (IoU) is defined in equation 3 and Figure 4. The IoU score ranges from 0.0 to 1.0 and IoU > 0.5 with the ground truth is generally regarded as a

good prediction. In addition, the confidence score of a bounding box reflects how likely the box contains a single mushroom.

With the selection of a specific threshold for both the IoU and the Confidence, each prediction will be considered as a True Positive (TP, correct prediction) if both values are greater than the respective thresholds. Following the same logic, a prediction will be considered as a False Positive (FP, erroneous prediction) if both values are lesser than the thresholds. The greater the values of these two metrics, the more accurate is the prediction made from the model.

The standard COCO metrics were used to evaluate network performance. The mean average precision (mAP) was calculated as the mean of all classes (in this case only one mushroom class) over 10 IoU thresholds, starting from 0.5 to 0.95 with a step size of 0.05. In addition, AP_{50} represents the calculation under the IoU = 0.50, whereas AP_{75} is a stricter metric and represents the calculation under the IoU = 0.75.

$$IoU = \frac{Area\ of\ Inersection}{Area\ of\ Union} \tag{3}$$



F1-score is defined as the harmonic mean of precision and recall instead of the common arithmetic mean. The result is always a number between 0 and 1, with values closer to 1 indicating better overall performance of the model. The formula used for its calculation is the following:

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{4}$$

4 Results and Discussion

In this chapter, the results of the experiments made on both datasets will be presented. The Pilze dataset will be examined first, by presenting the results of all architectures in a table and then comments will be made. Then selected images from the test set will be presented in order to highlight good and bad detection performance and compare the results between the "light" and "heavy" implementations of the same architecture, but also between different architectures. The same approach will be followed for the V6 Google dataset. Finally, overall comments on the performance of the architectures will be made, taking into account the characteristics of each dataset.

4.1 Pilze dataset results

All models were trained for a total of 15 epochs. It should be noted that after the 10^{th} epoch, the models showed little or no improvement, hence this checkpoint was selected for the final testing of the architectures. The learning rate for the finetuning was set at 0.001 and all the models were pre-trained on COCO Dataset.

Table 1 Performance comparison between the architectures on Pilze dataset. Red indicates the best results column-wise.

Architecture	**Backbone**	**Precision**			**Recall**	**F1 score**
		AP_{50}	**AP_{75}**	**mAP**		
Mask R-CNN	light	**0.921**	0.766	0.685	0.731	0.707
	heavy	0.912	0.815	0.710	0.750	0.729
MS R-CNN	light	0.851	0.705	0.638	0.675	0.660
	heavy	0.837	0.742	0.656	0.685	0.670
Cascade Mask R-CNN	light	0.825	0.702	0.625	0.663	0.643
	heavy	0.817	0.732	0.642	0.676	0.659
HTC	light	0.911	0.772	0.689	0.738	0.716
	heavy	0.876	0.794	0.698	0.738	0.717
DetecoRS	light	0.912	**0.818**	**0.724**	**0.773**	**0.748**
	heavy	-	-	-	-	-

The results of Table 1 show that, regarding the AP50 metric, the light models achieve better results, while the AP75 is better on heavy models. The simpler features extracted from a light backbone perform better with a more lenient restriction of IoU than the more complex from a heavy backbone. However, when stricter IoU thresholds are placed, the more sophisticated features of the heavy models outperform the light ones. The logical assumption would be that the more complex backbones would produce more informative features that would benefit the model in all cases, leading to heavy models producing better results in all AP metrics. This phenomenon has to do with the amount of available data, which in this dataset is relatively low, in comparison with the V6 Google Dataset (chapter 4.2). The overall metrics of mAP, Recall and their product, F1-score, show that the heavy models outperform their light counterparts, as the stricter threshold of IoUs are used.

By examining the results of each architecture, it obvious that Mask R-CNN performed very well, despite being the older and simpler architecture of all. It was able to detect the majority of the mushroom instances with decent masks as it will be shown in the next few Figures. The performance achieved by Mask R-CNN will be regarded as the baseline for the comparisons and the comments that will follow. The architectures of MS R-CNN and Cascade Mask R-CNN reported worse performance than the simpler Mask R-CNN. Especially Cascade Mask R-CNN achieved the worst performance among all the architectures. This fall of performance can be attributed to the introduction of more parameters that need to be trained, but at the same time the amount of training data and the feature extraction ability and information flow of the architectures remained the same. In other words, bigger architectures were trained with the same training power of extracted from the dataset.

As described in Chapter 3, MS R-CNN introduced a new branch used for applying a score on the mask produced by a classic Mask R-CNN architecture. This extra branch is comprised of new trainable weights, that were finetuned with the Pilze dataset. However, the feature extractor component of the architecture was the same as with Mask R-CNN, while no extra data was introduced in the training process. In an similar manner, Cascade Mask R-CNN has three instead of one Mask R-CNN head component (comprised from classification + bounding box branch and mask branch), as each one of these three head components is trained with a different IoU threshold. Despite the 3x times more trainable weights introduced, the feature extractor component

of Cascade Mask R-CNN remained the same as the one Mask R-CNN has and no more training data was included during the training.

By examining these three architectures not only as a whole, but examining distinctively light and heavy models in each one of them, it is obvious and stated before, that using a more complex feature extractor component (heavy models) improves the performance. However, the number of the newly introduced weights in both MS R-CNN and Cascade Mask R-CNN cannot be counterbalanced, as both heavy models of these two architectures perform worse than the light model of the simple Mask R-CNN. Meaning that using just deeper feature extraction backbones do not give enough training power to the architecture. This setback can be overcome with two solutions, introduction of more data to counterbalance the extra parameters or modify the architecture to increase information flow and achieve even better feature extraction.

The architecture of HTC shows great improvement when is compared to both MS R-CNN and Cascade Mask R-CNN, but almost similar to Mask R-CNN. This is happening even though is based on the Cascade Mask R-CNN (using three heads) and despite using the same amount of training data with the same feature extraction backbone. This improvement can be attributed to the new connections between the mask heads and the new added branch for the extra segmentation, as described in chapter 4.4. This improvement in architecture has allowed a better flow of information on the final stages and components, even though the extracted feature map from the backbone has not changed.

From all the architectures, DetecoRS has the best performance as not only has the improved information flow of HTC but also with the updates in its backbone (described in chapter 3.5) it is able to produce even better-quality feature map. The combination of better feature maps and better information flow leads to the best results overall, surpassing the both light and heavy models of the simple Mask R-CNN architecture, despite the low amount of available data for training.

Based on the results of Table 1, the behavior of these architectures can be predicted in the scenario where sufficient amount of training data was available. As all the architectures are descendances of the original Mask R-CNN, integrating further updates, it is expected all of them to have better performance, following the official results of each of the authors on benchmark datasets. MS R-CNN introduces an improvement module in a different direction than the three multi-stage, cascade logic architectures of Cascade Mask R-CNN, HTC and DeterctoRS. Based on the collected

results of the MMDetection toolbox, the performance of MS R-CNN is similar to the performance of Cascade Mask R-CNN and both of them better than the simple Mask R-CNN. Regarding the three multi-stage architectures, each of them improves on deficiencies of the previous one, resulting in DetectoRS performing best among them, followed by HTC and finally Cascade Mask R-CNN.

A better understanding, on how each of the architectures performed, can achieved through visualizing and comparing the predictions on the test set. The different shades of green color in the images refer to different detected mushroom instances and it is not a measure of confidence or accuracy of the models regarding their prediction. In order to present in a better and more efficient way the prediction images, the numbering of the images will correspond to a specific model (light or heavy) of an architecture as follows:

(a) = original image
(b) = light Mask R-CNN
(c) = heavy Mask R-CNN
(d) = light MS R-CNN
(e) = heavy MS R-CNN
(f) = light Cascade Mask R-CNN
(g) = heavy Cascade Mask R-CNN
(h) = light HTC
(i) = heavy HTC
(j) = DetectoRS

In the following Figures, different results on the test set will be presented and commented. It is important to state the mushroom instances of this dataset are more homogenous than the V6 dataset. Here, only the species of Pleurotus is detected, in an artificial environment with specific and similar characteristics between the images. The black background of the plastic cover of the substrate makes easier the detection and segmentation of mushrooms which naturally have a much brighter color. The other background which can have slightly similar texture or color with the mushrooms does not appear to be a problem as all the models do not mistake any background for mushrooms. Moreover, the shape of the mushroom instances is quite similar albeit in different sizes and mature levels. The immature mushrooms are not detected as they were not included in the annotation of the dataset. The mature mushrooms that are of bigger shape are more easily identified, especially if they are adequately lighted. However, there is a significant number of mature instances that either are not detected or masked with low quality masks of arbitrary shapes.

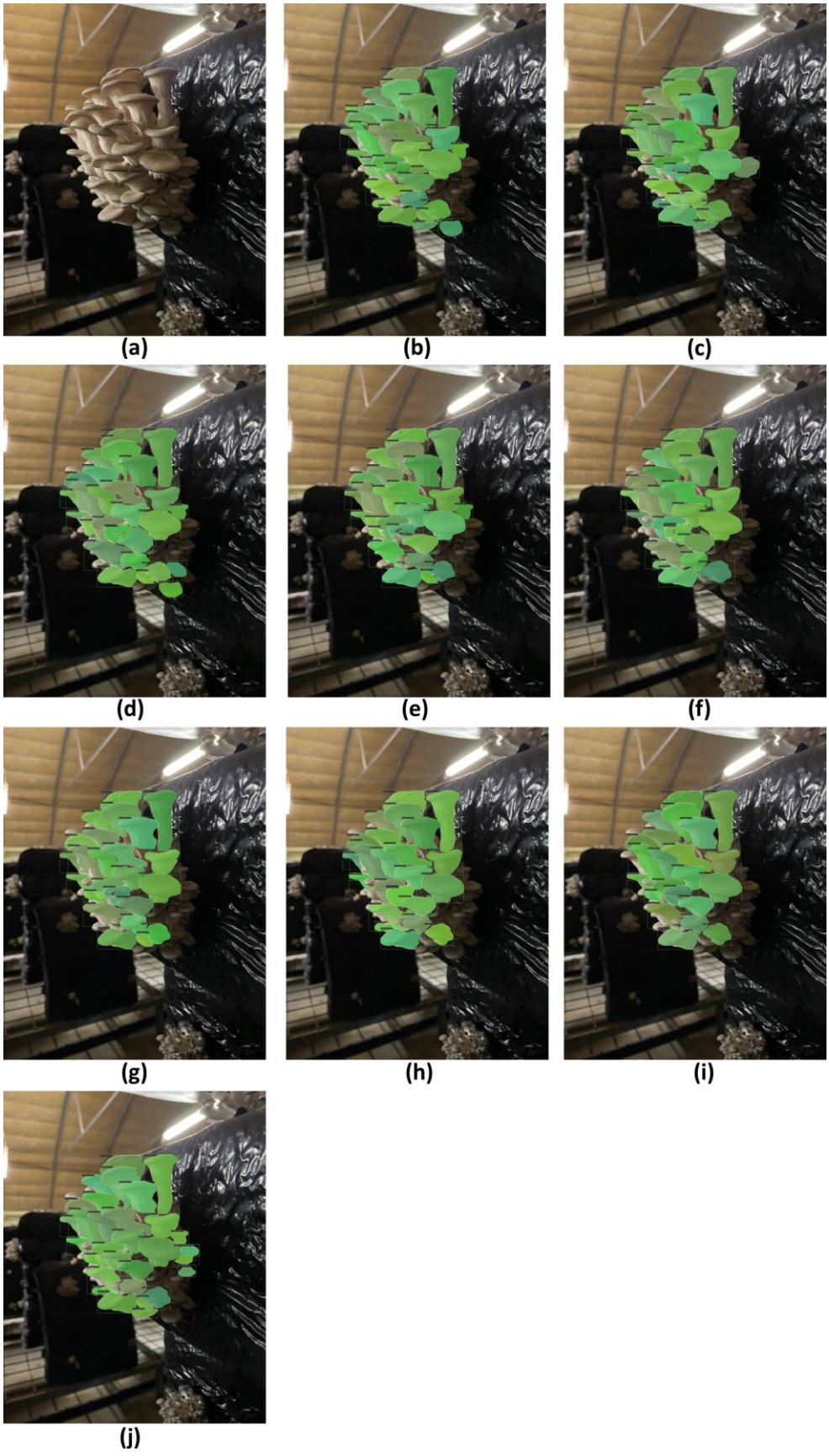


Figure 15 Mushroom cluster, sub-lighted area and arbitrary masks. DetectoRS best overall results.

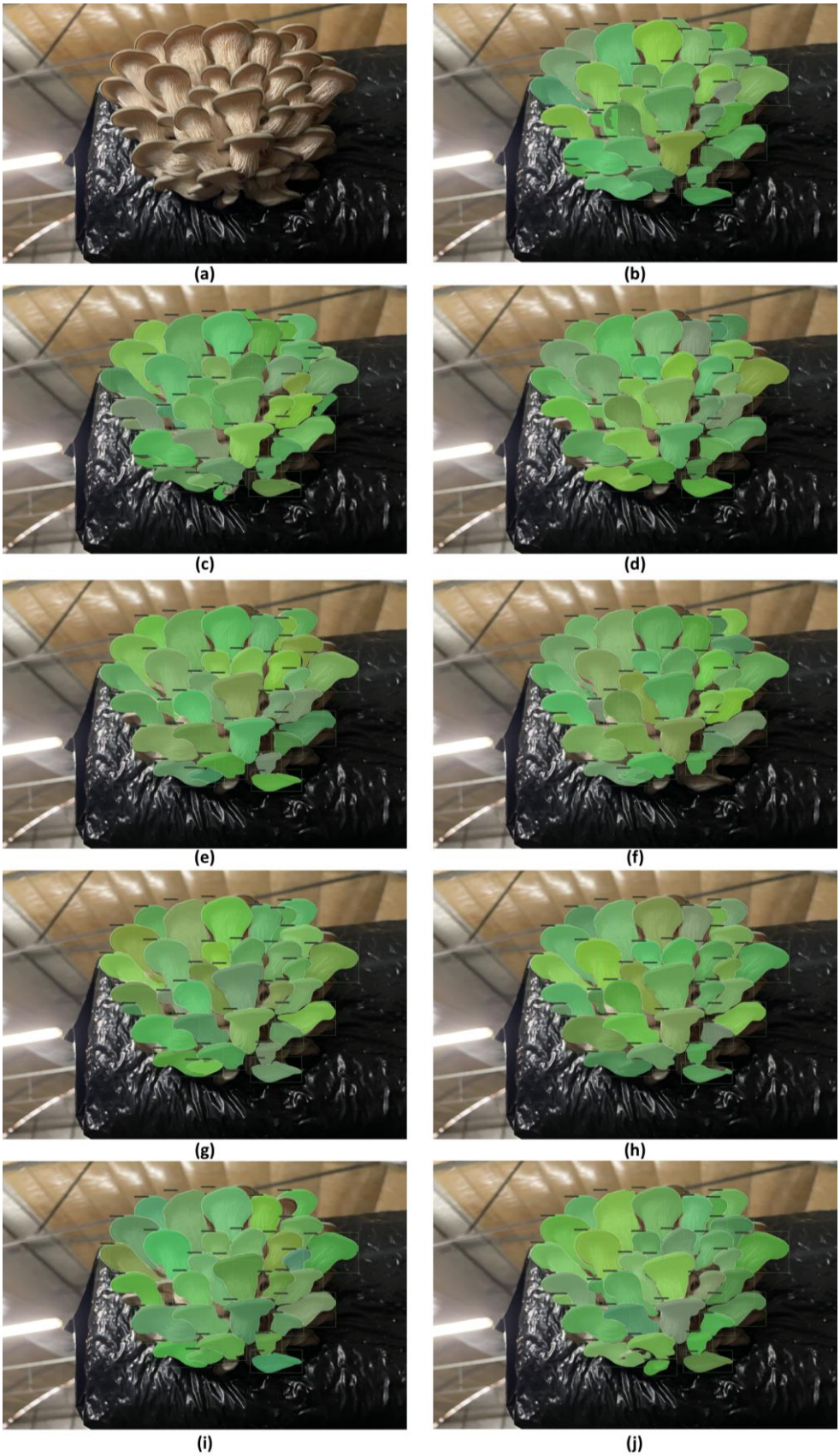


Figure 16 Good performance but problems with occluded, sub-light and neighboring instances observed.

In Figure 15, the overall performance of all the architectures was quite good, given the numerous instances included in the cluster and the extend of occlusion and sub-lighting of some samples. With the exception of DetectoRS, Figure 15.j, the other models produced at some extend masks with arbitrary shapes as it can be easily distinguished in Figures 15.a-g. All of the models failed to detect the mushrooms on the bottom right of the cluster, something that can be attributed to the bad lighting of these specific instances. Moreover, throughout the cluster some highly occluded mushrooms could not be identified by the small parts that were visible, a behavior reported by all the models. Finally, a secondary mushroom cluster on the top right of the image is not detected at all, something that is desirable as the focus should be at one cluster per image. In Figure 17, the dark area of the cluster is shown for some architectures, resulting in the mushroom instances not being able to be detected, due to low light conditions.

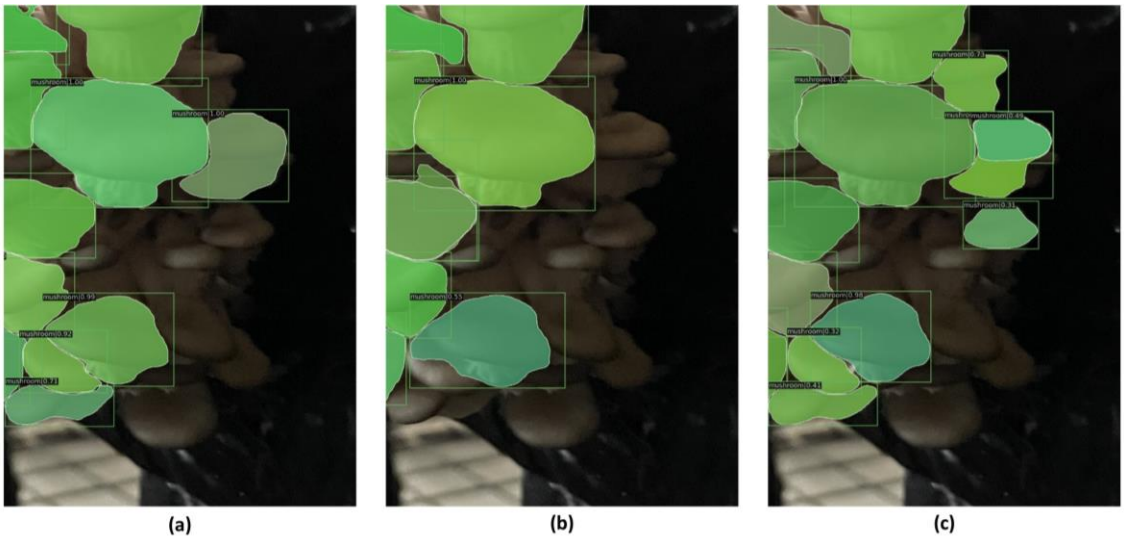


Figure 17 Low light instances failed to be correctly detected. Heavy Mask R-CNN (a), light Cascade Mask R-CNN (b), DetectoRS (c)

Examining Figure 16, the overall performance of the models was good regarding most of the big instances. However, smaller occluded instances throughout the cluster were not detected by the models. More specifically, different models could not identify different instances that other models could. Furthermore, the instances on the right area of the cluster, that are not sufficiently light, unlike the mushrooms on the rest of the cluster, are not detected by any of the models. Finally, it was observed that some models fused two distinct instances under a single mask, mistaking two bordering mushrooms as one, Figures 16.b,d,h,j. In Figure 18, the problematic area of the bordering mushrooms is shown for some of the architectures that fail to distinguish between the two instances.

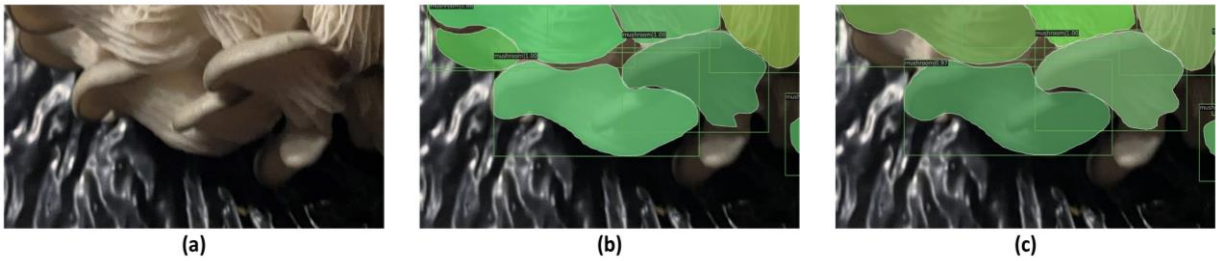


Figure 18 Original image with two mushroom instances (a), light Mask R-CNN produced mask (b), light HTC produced mask.

Another example of good performance of all the architectures can be seen in Figure 20. In this case, all of the models detected all of the depicted mushrooms with few exceptions of masks having arbitrary shapes. The only two models that were not able to detect a small mushroom instance at the bottom of the cluster are light MS R-CNN (Figure 20.d) and light Cascade Mask R-CNN (Figure 20.f).

A similar case with many similar problems observed, as in Figure 16, can be seen in Figure 21. The difference in that image is that there is no light problem but the many of the models split a single mushroom instance into two with their produced masks. This can be seen in Figures 21.b,c,d,f,g,h at the bottom of the cluster, where a mushroom has grown upside down. Figure 19 shows a zoomed version of the mushroom instance for a couple of architectures that fail to correctly detect it. Overall, both models of Mask R-CNN architecture, detect most of the occluded instances but introduce some arbitrary masks in other more well-defined instances.

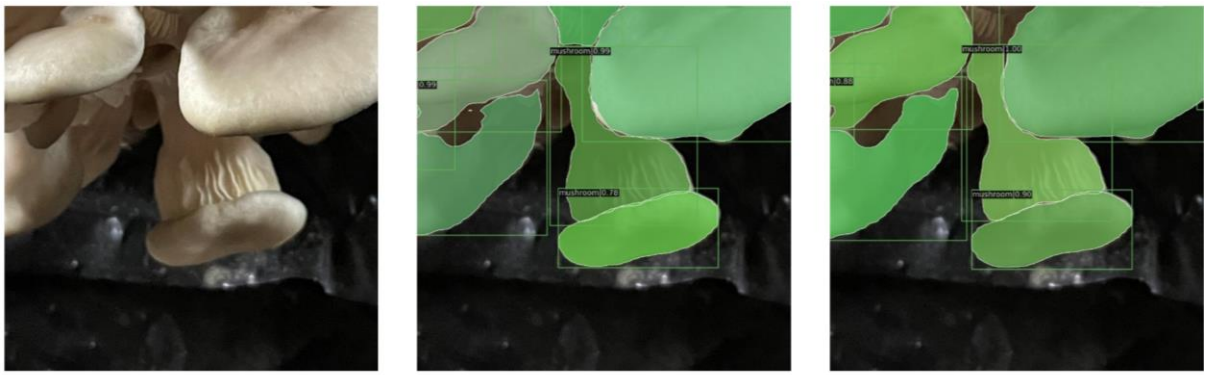


Figure 19 Original image with one mushroom instance (a), heavy Mask R-CNN produced mask (b), light HTC produced mask.

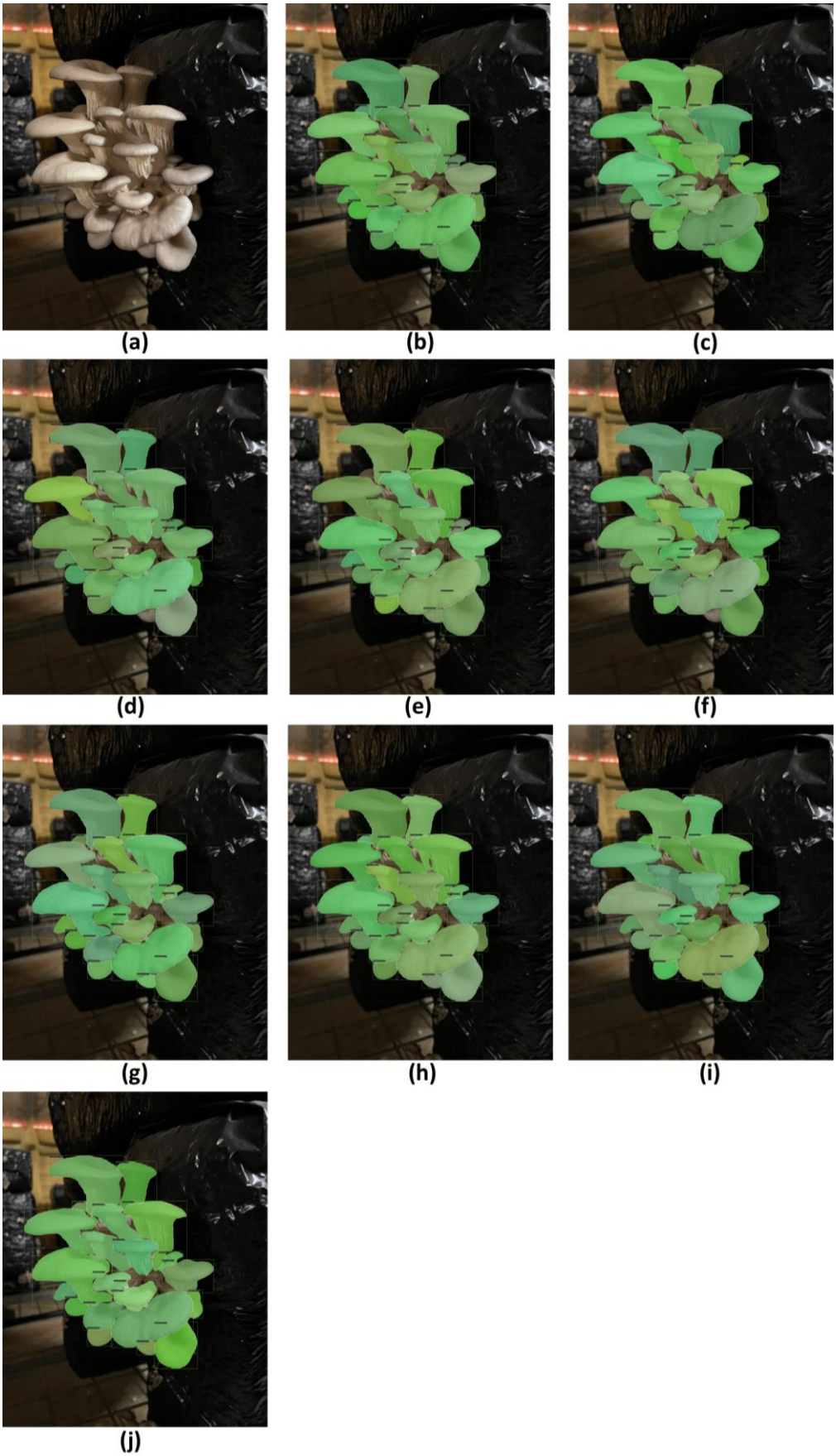


Figure 20 Overall good performance by all architectures.

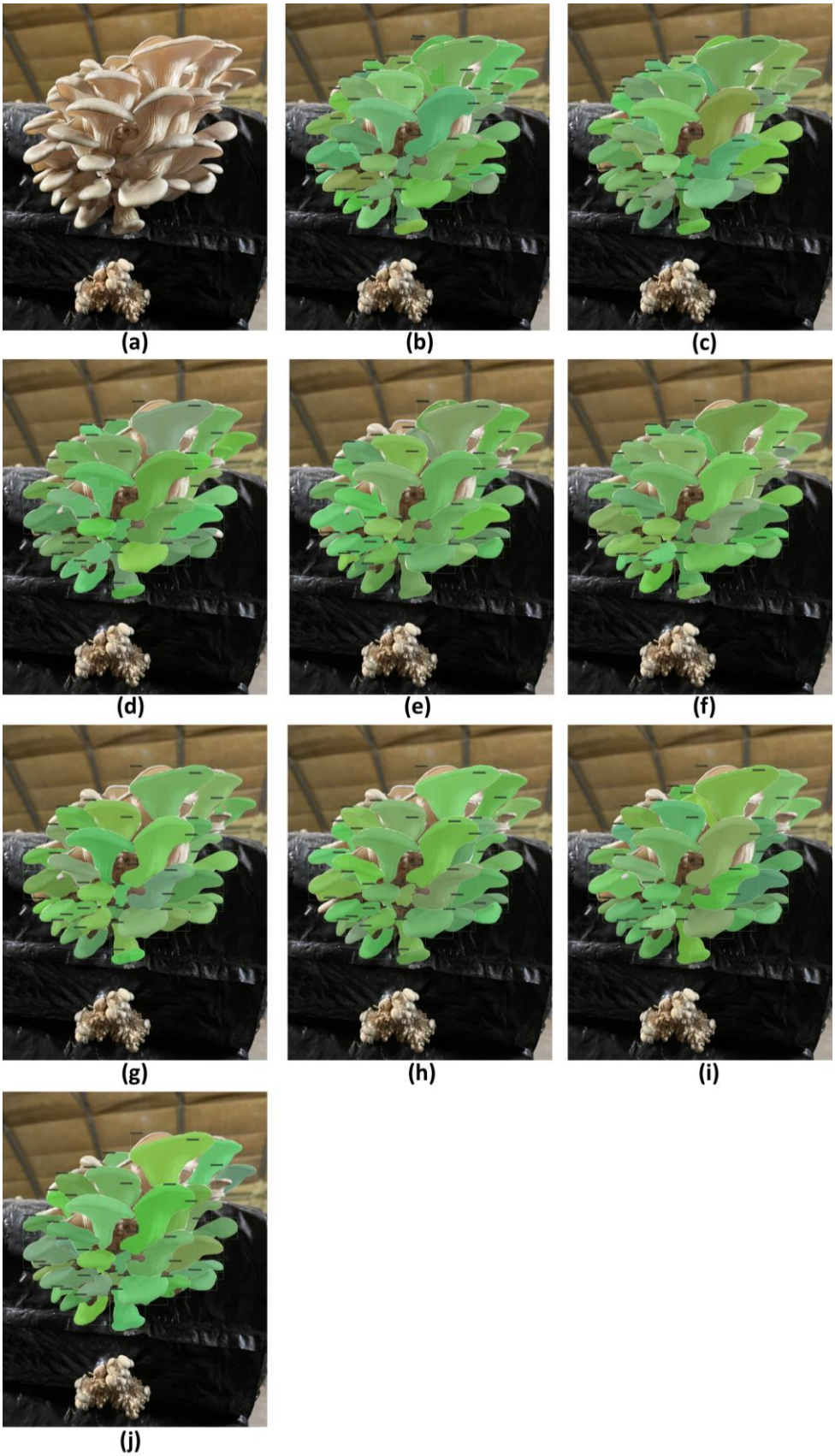


Figure 21 Decent performance but problems with occluded and splitting one mushroom instance to two is observed.

In both Figures 23 and 24 it can be observed that both models of Mask R-CNN produce the best results among all the other models. They are able to detect and produce fairly good masks for almost all of the mushroom instances. An interesting observation is that they are able to produce masks even for the smaller instances in the bottom of the cluster in Figure 23 even though the other models, even DetectoRS cannot. Also, in Figure 24, while both Mask R-CNN and DetectoRS identify accurately the mushroom instances in the big cluster, only Mask R-CNN is able to detect many of the smaller mushrooms in the secondary cluster at the bottom of the image.

Finally, in Figure 25, the performance of DetectoRS is superior to all the other models, producing good quality masks without overlapping or arbitrary shapes. All the other architectures cannot achieve that and even falsely split an instance on the right side of the cluster into two instances. Even though DetectoRS misses some of the smaller instances at the bottom of the cluster compared to both Mask R-CNN models, the mask quality compensated that drawback. In Figure 22 a more detailed comparison between the arbitrary masks produced from some architectures and the well-shaped masks of DetectoRS can be seen.

Figure 22 Arbitrary masks produced by light MS R-CNN (a) and light Cascade Mask R-CNN, while good quality masks by DeterctoRS (c).

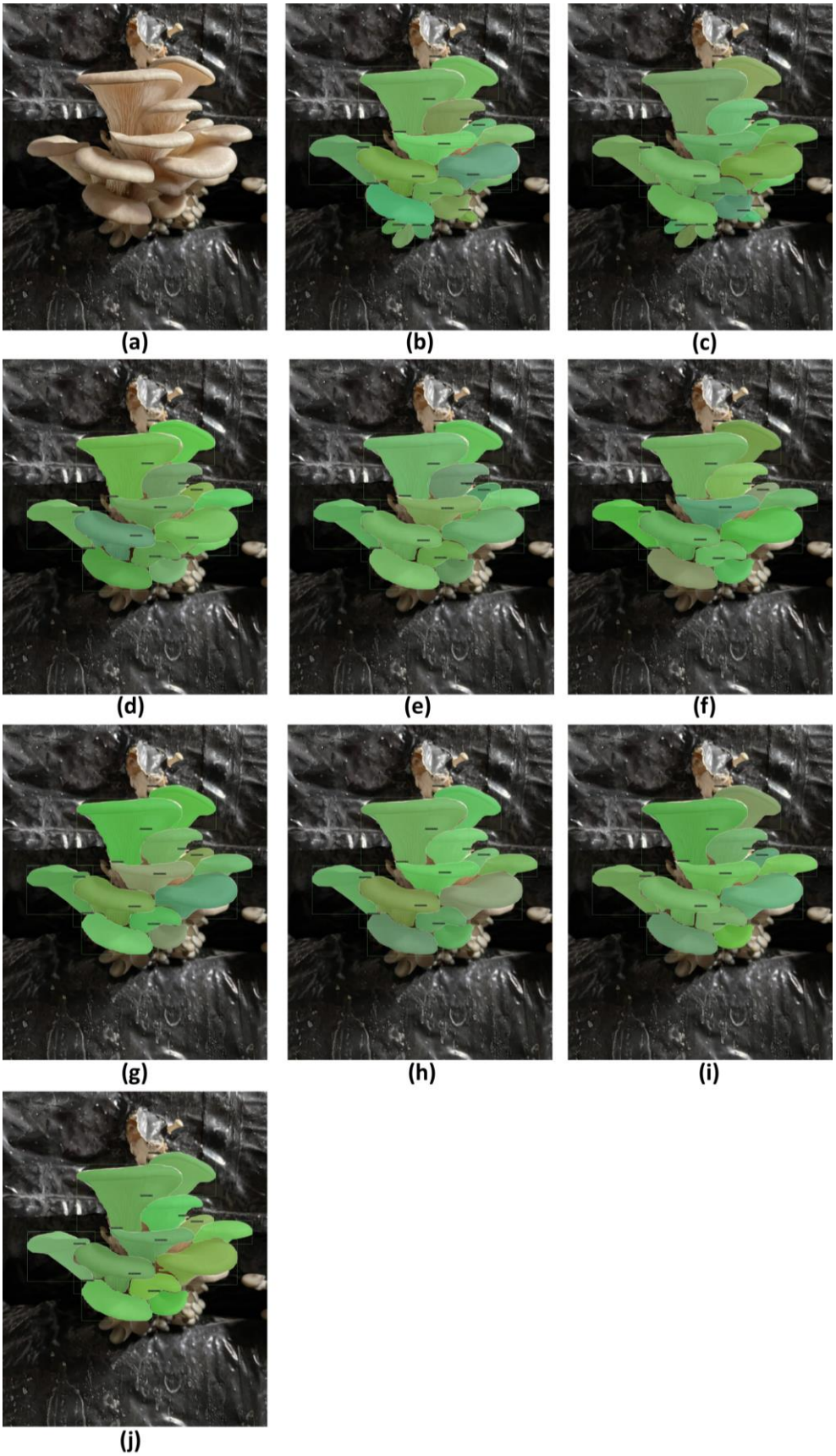


Figure 23 Overall good performance, both Mask R-CNN models detect smaller instances at the bottom of the cluster.

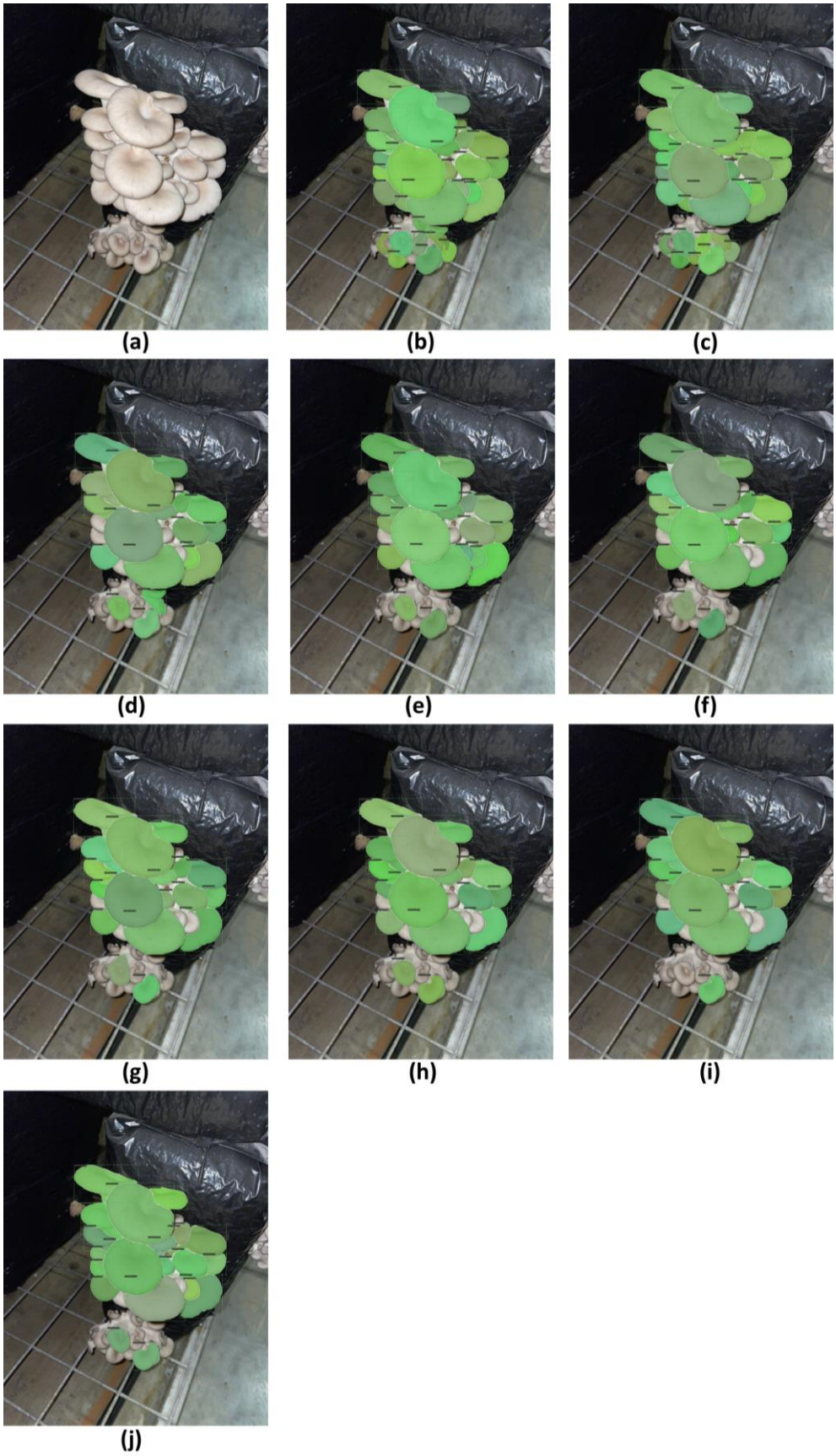


Figure 24 Overall decent performance from both Mask R-CNN and DetectoRS, but only the former detects smaller mushroom instances.

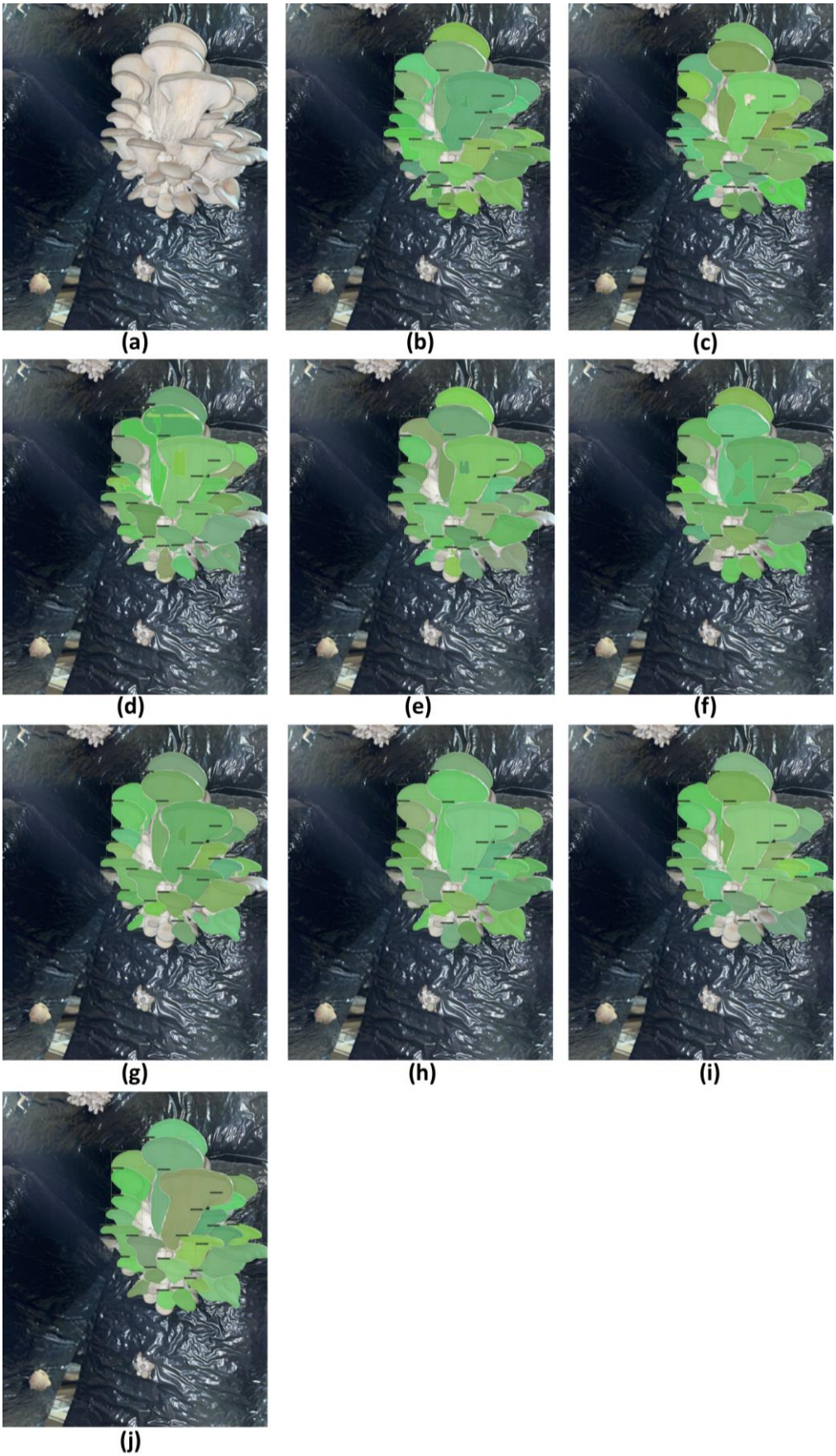


Figure 25 DetectoRS producing better results without arbitrary masks.

4.2 V6 Google dataset results

All models were trained for a total of 15 epochs. It should be noted that after the 10th epoch, the models showed little or no improvement, hence this checkpoint was selected for the final testing of the architectures. The learning rate for the finetuning was set at 0.001 and all the models were pre-trained on COCO Dataset.

Table 2 Performance comparison between the architectures on V6 Google dataset. Red indicates the best results column-wise.

Architecture	**Backbone**	**Precision**			**Recall**	**F1 score**
		AP_{50}	**AP_{75}**	**mAP**		
Mask R-CNN	light	0.870	0.744	0.670	0.739	0.703
	heavy	0.888	0.770	0.693	0.759	0.725
MS R-CNN	light	0.854	0.724	0.658	0.730	0.692
	heavy	0.874	0.771	0.688	0.744	0.715
Cascade Mask R-CNN	light	0.864	0.737	0.665	0.733	0.697
	heavy	0.869	0.766	0.678	0.739	0.707
HTC	light	0.873	0.747	0.671	0.764	0.714
	heavy	0.878	0.764	0.691	0.773	0.730
DetecoRS	light	**0.889**	**0.774**	**0.707**	**0.784**	**0.744**
	heavy	-	-	-	-	-

One of the first comments on the numerical results of the V6 Google dataset is that all the heavy models outperform their light counterparts. This can be attributed to the more complex backbones of the heavy ones, that execute better the task of feature extraction, providing better information for the rest of the architecture. As expected, DetectoRS architecture produced the best results in all metrics due to the extensive updates on the whole architecture.

Even though the total number of data is bigger than the other dataset, the same observation can be done regarding the performance of the architectures based on the available data and the number of parameters that must be finetuned with regard to the extracted feature maps and flow of information. This observation extends on both light and heavy models of each architecture.

The performance of Mask R-CNN can be considered as the base performance. Then, by introducing updates in the architectures of MS R-CNN and Cascade Mask R-CNN, the respective heads have more parameters to be finetuned, but the power of the feature extractor remains the same. However, this is not the case for HTC and DetectoRS. On the one hand, HTC manages to slightly outperform Mask R-CNN due to higher recall, as the mAP is almost similar. On the other hand, DetectoRS light model is able to outperform both light and heavy models of Mask R-CNN architecture.

A better understanding of the performance of each architecture can achieved through visualizing and comparing the predictions on the test set. An important information for the upcoming figures is that the differences in the green color of each mushroom instance indicate a difference mushroom and has nothing to do with the confidence or the accuracy of the models regarding their prediction. In order to present in a better and more efficient way the prediction images, the numbering of the images will correspond to a specific model (light or heavy) of an architecture as follows:

(a) = original image
(b) = light Mask R-CNN
(c) = heavy Mask R-CNN
(d) = light MS R-CNN
(e) = heavy MS R-CNN
(f) = light Cascade Mask R-CNN
(g) = heavy Cascade Mask R-CNN
(h) = light HTC
(i) = heavy HTC
(j) = DetectoRS

From the many results of the test set, the most notable will be presented and discussed, including different kinds of performances from the architectures. In some cases, all of the models managed to successfully segment the image, while on others they all failed. This has to do with the mushroom(s) that is/are inside the image. In other cases, the simpler models produced mediocre results while the more complex were able to visually outperform them. In most of the cases, the exact borderline between different instances or between an instance and the background is not detected. However, as this is an exploratory analysis and the task at hand is not critical (e.g. segmentation of human anatomy in surgical operation), the accuracy of the borderlines with an error of some pixels is considered enough. Moreover, in order to detect this small misalignment of produced masks and the actual mushroom instances, extreme zoom is needed.

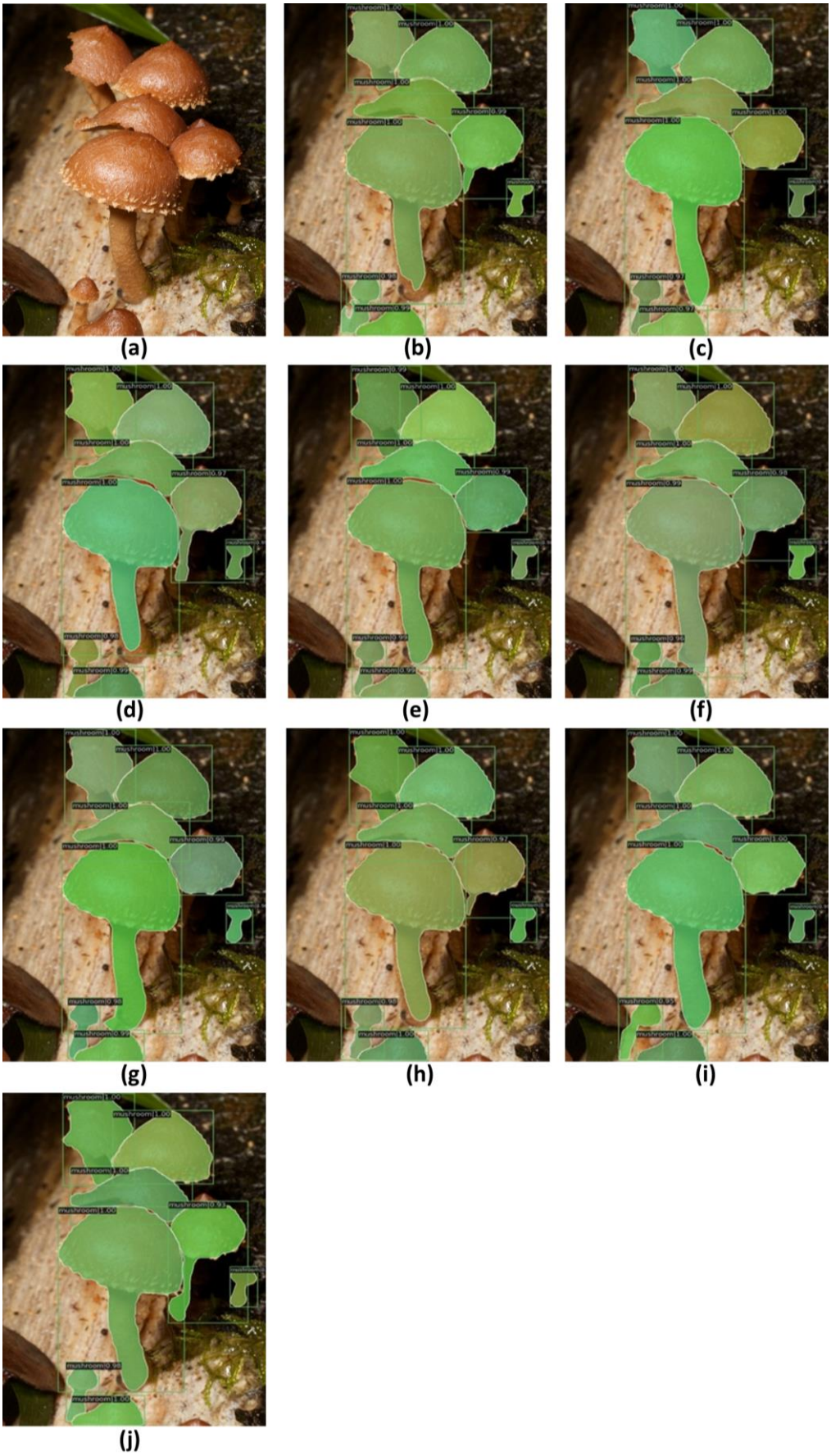


Figure 26 Good detection performance by all models.

In Figure 26, a case where all the models performed very well on the instance segmentation task is presented. All the instances are detected, including those on the foreground and in the background. For the semi-occluded instances were visible only the upper part of the mushrooms (cups) but all of the models were able to detect them an accurate way. Furthermore, some far smaller instances, in the right and in the bottom areas of the image, are also accurately detected. The only small part of the image that most of the models have missed is on the center-right mushroom instance where the stalk of the mushroom is detected in a complete way only by light MS R-CNN in Figure 26.d and by DetectoRS in Figure 26.j. Between those two, DetectoRS has detected the stalk at its full. Overall, in this image the correct number of the instances is detected and with a very good accuracy for each individual mask.

An interesting case can be seen in Figure 27. Here all of the models were able to detect and differentiate between the mushroom instances and the background. However, this was not the case for the border detection between the two mushroom instances that exist in the image. At one side they are in contact with each other, presenting a difficult task for the models to work on. As it can be seen most of the models produced overlapping masks, that from little to big extend intruded into the other instance. The only model that managed to overcome this difficult situation is DetectoRS in Figure 27.j. From all the other models only heavy Mask R-CNN produced decent masks but still a small, overlapped area can be observed. Overall, despite the quality of the masks all the models managed to detect all of the mushroom instances existing in the image.

Another example of the supremacy of DetectoRS can be seen in Figure 28 as it was the only architecture able to detect and produce detailed masks, Figure 28.j. The other architectures struggled, producing masks with arbitrary shapes, as it can be seen in Figures 28.b,d, combined more than one actual mushroom instances in one mask, Figures 28.e,g or missed some mushroom instances, Figures 28.f,h. The only model that produced results close to the ones from DetectoRS were heavy HTC, a result that is logical as this model was second in overall performance as it is stated in Table 2. The performance of DetectoRS is further proved in Figure 29, where it was the only model able to detect the depicted mushroom instances in Figure 29.j. This example was one of the hardest based on the similarity between background and instances with respect to the color and texture. The improved implementation of DetectoRS feature extractor backbone, combined with the information flow provided on the heads of the architecture was able to solve the difficult task of segmenting the two mushroom instances. Surprisingly, the light Mask R-CNN model was able to fairly detect one out of two mushroom instances while other models, with better overall results from Table 2, could not.

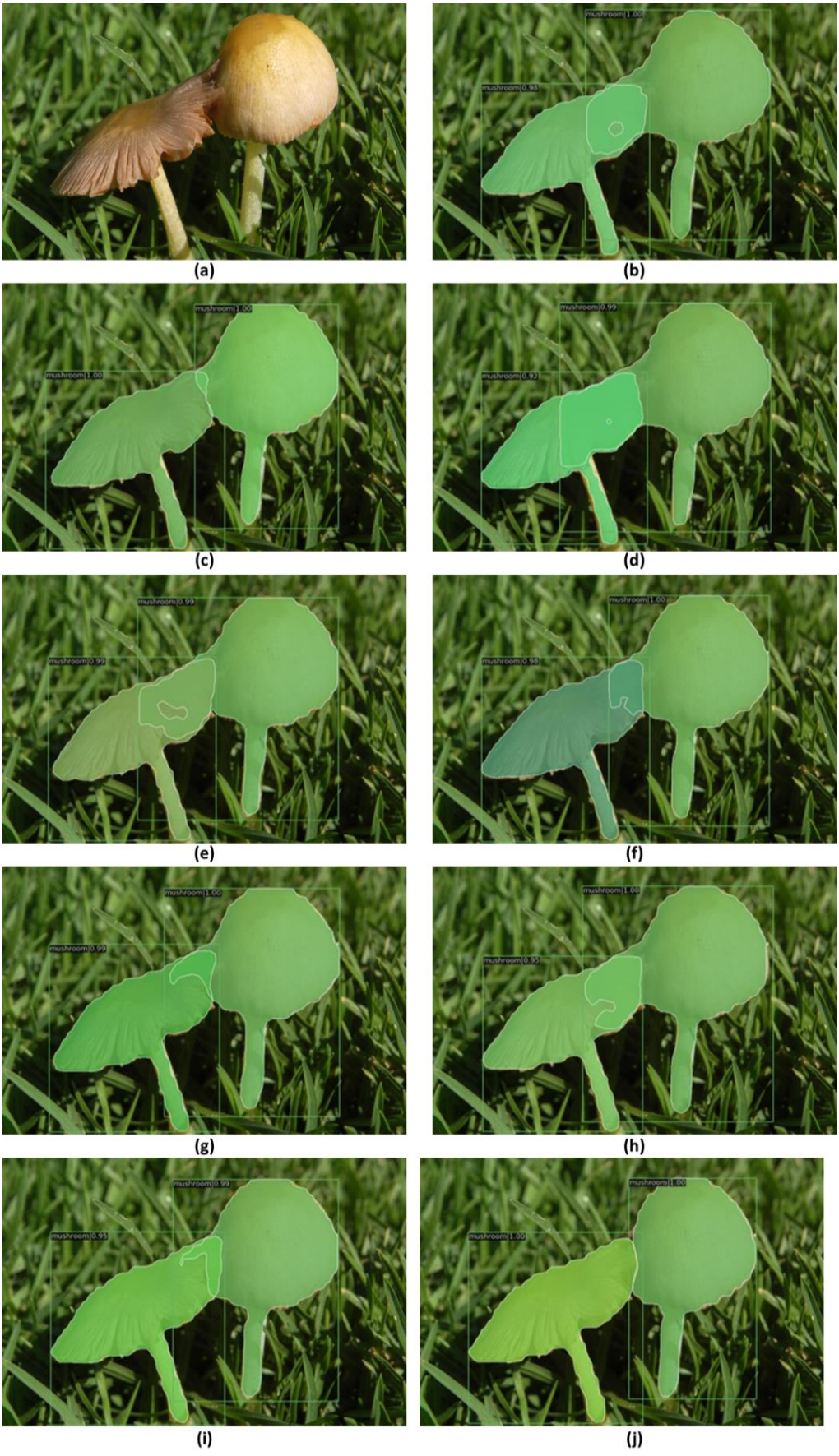


Figure 27 Dual mushrooms with overlapping produced masks from most models.

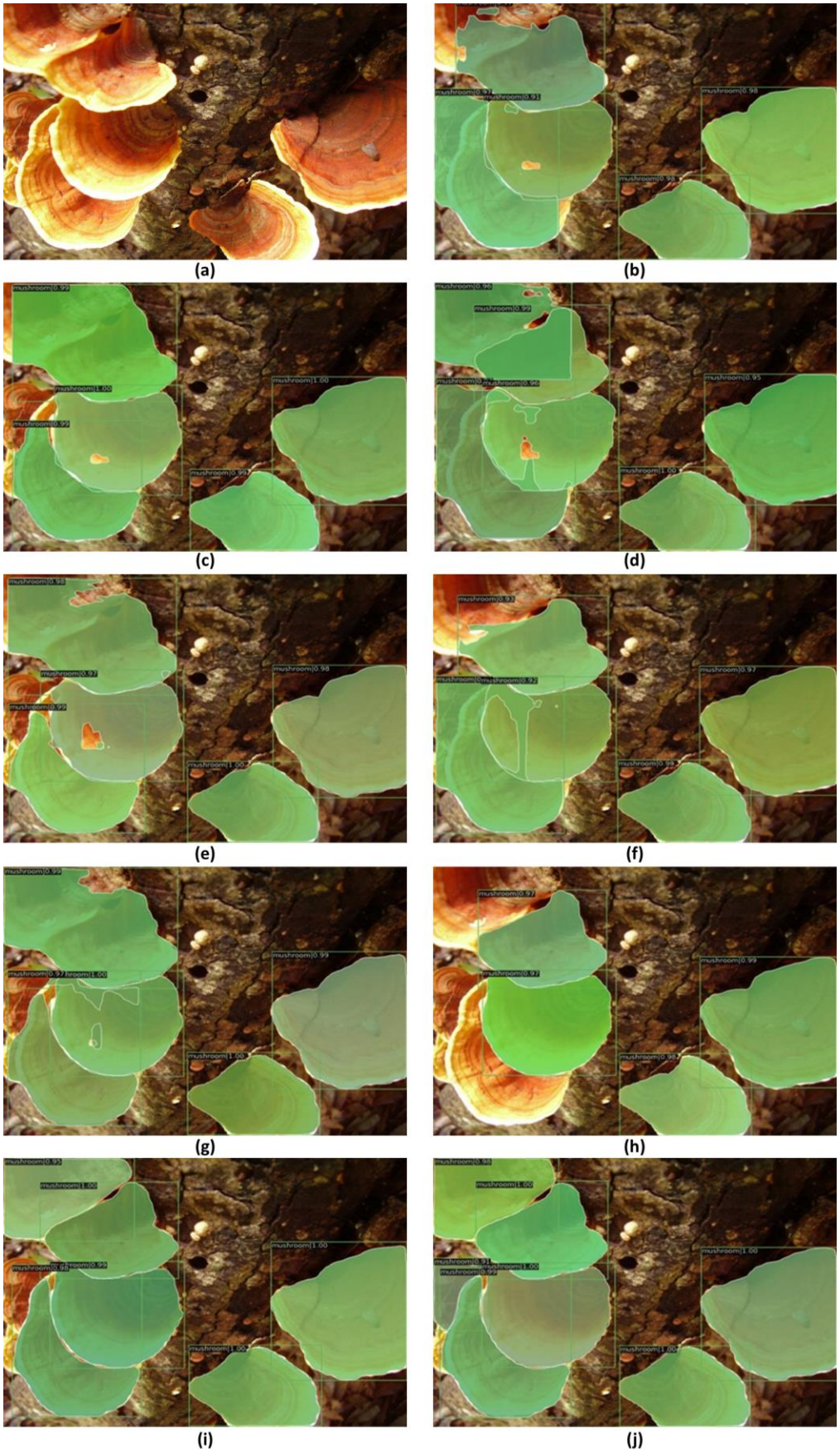


Figure 28 Arbitrary masks and detection oversight by most models except DetectoRS.

Figure 29 Dual mushrooms with complex background, DetectoRS good performance.

Despite the good overall performance of all the models and more specifically the DetectoRS', there were cases where all of them were not able to successfully segment all the instances in an image. This can be seen in Figures 30 and 31, where to different kind of mushrooms are depicted. In Figure 30, distinct mushroom instances, but with arbitrary shapes, are packed together creating a cluster. However, the borders between the instances are not always well defined and occlusion exists. None of the models was able to find all the instances in the image, especially the ones on the top of the cluster. Regarding the mushrooms in the front of the cluster and at the center of the image, all the architectures struggled to produce high quality masks, creating masks with even more arbitrary shapes than the mushroom instances and sometimes missing specific instances. Overall, only DetectoRS was the model with the best result creating higher quality masks for the mushroom instances that was able to identify.

Another example is shown in Figure 31, where the mushroom instances are multiple smaller mushrooms with the classic shape with a cup and a stalk. In this image, even if the mushroom shapes are more standardized, the problems of very small instances and occlusion lead to many instances not being detected. This is a phenomenon noticed in all the examined models. However, the difference in the performance compared to Figure 31, is that the more well-defined shape of this mushroom species facilitates the creation of better masks. So, even if some instances were not detected, the ones that actually detected were provided relatively good quality masks.

A case where the architectures struggle collectively is shown in Figure 32. In this image the depicted mushroom can be considered a cluster-like instance. Nevertheless, there are distinct parts of the mushroom cluster that can be falsely detected as individual instances from a network, even if the whole cluster should be considered as one entity. From all the models, only heavy MS R-CNN, managed to produce a mask containing most of the depicted cluster, but smaller overlapping ones were also produced. If more mushroom of this shape were included during training, the models could have performed better. Finally, in a similar case in Figure 33, only the heavy models of each architecture, plus the DetectoRS, were able to correctly identify and segment the whole mushroom cluster as one instance. On the other hand, the light models could not identify the cluster at all or falsely detected smaller parts of it as individual mushrooms.

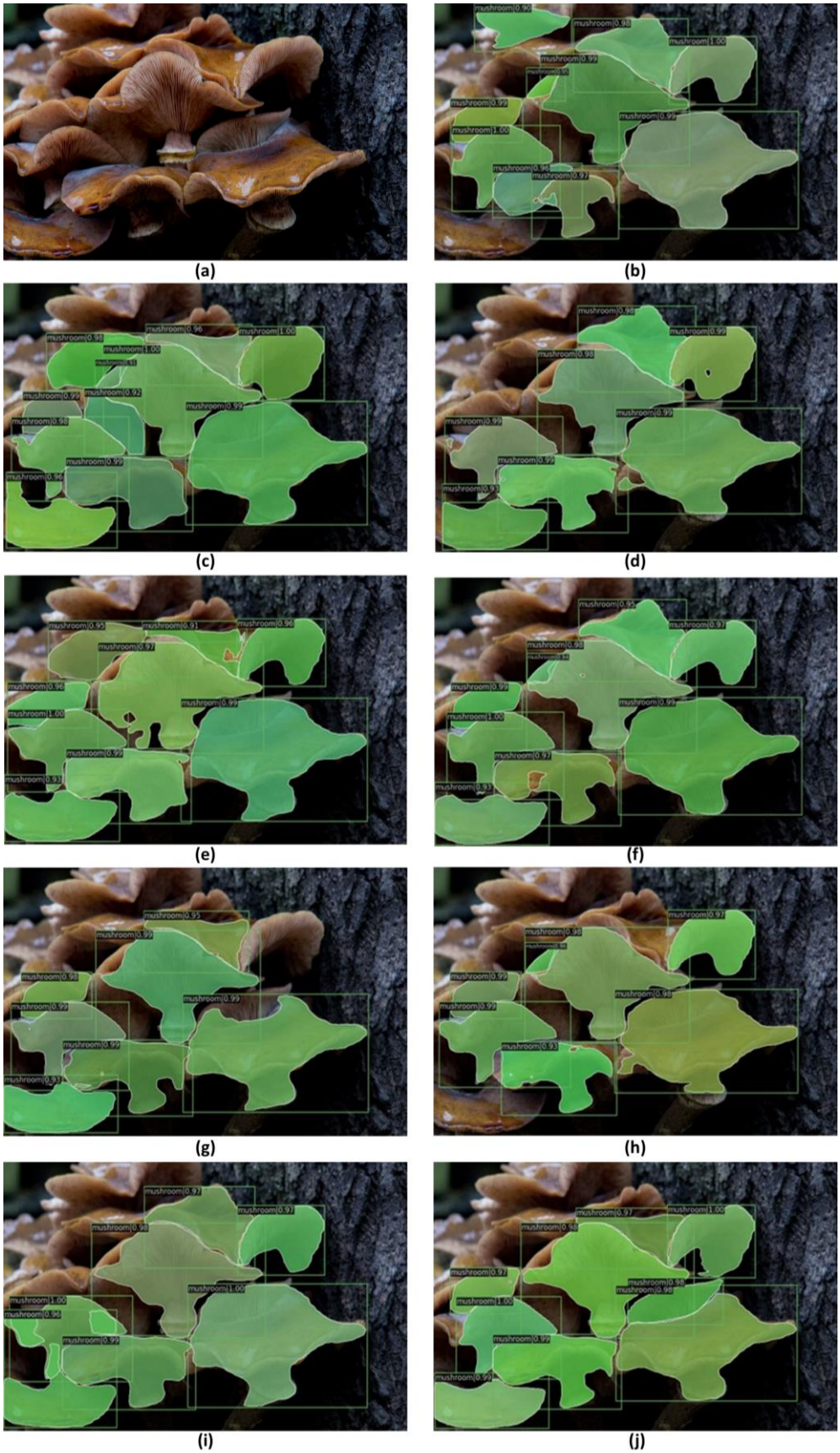


Figure 30 Arbitrary mushrooms instances in cluster leading to lower quality masks and detection performance.

Figure 31 Many smaller mushroom instances with well-defined shape, resulting in better quality masks but missing detection of some instances.

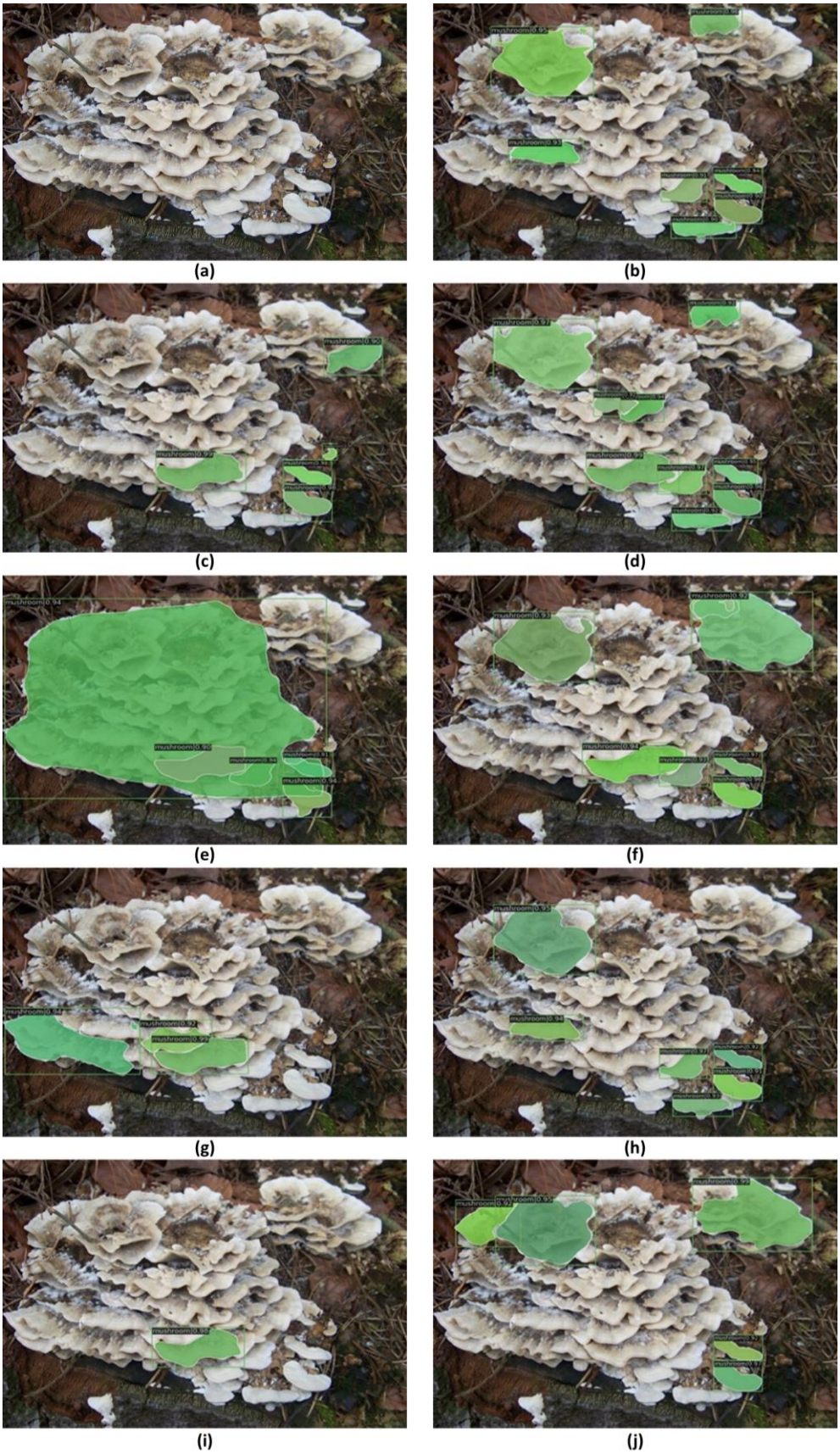


Figure 32 Mushroom cluster considered as one instance proving difficult task for all architectures.

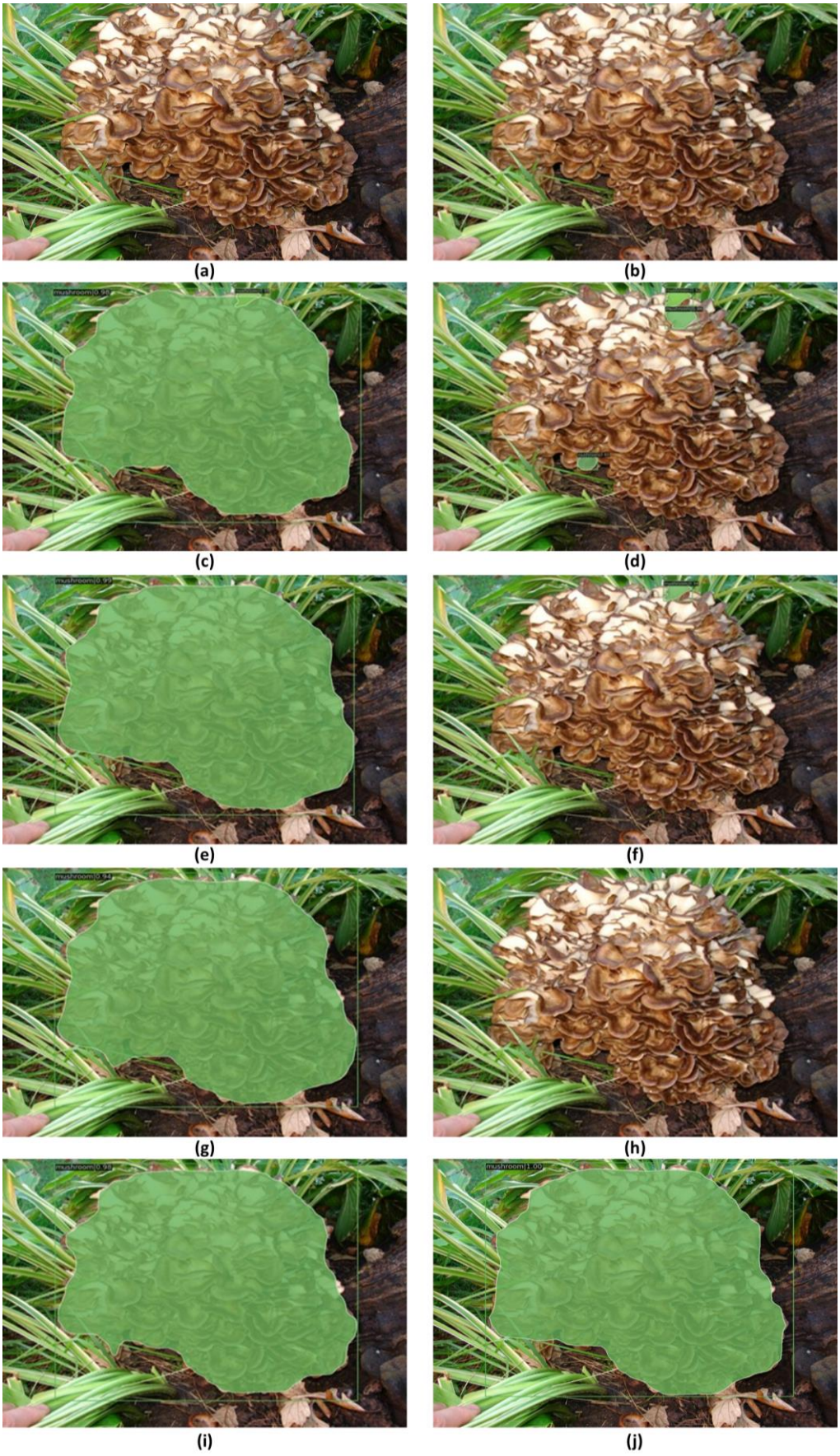


Figure 33 Mushroom cluster correctly detected as one instance by all heavy models.

4.3 Overall notes

Despite the bigger number of images in V6 Google dataset, the performance of the architectures is overall slightly worse than the Pilze dataset and this has to do with the nature of the datasets. On the one hand, Pilze dataset has fewer images, which is a drawback, but the target mushroom class is well defined and uniform in shape/color, the background is less complex and the images are of much higher resolution (4032x3024). On the other hand, V6 Google dataset has more images, but the target mushroom class is not uniform as there are many different mushrooms species with different shape/color/size, the background in nature is more complex and the images are of lower resolution (up to 1024x1024).

In both datasets, the two architectures of MS R-CNN and Cascade Mask R-CNN do not manage to perform better than the simple Mask R-CNN due to the low amount of data available for finetuning the updated head architectures that have more parameters, in the same manner as described in section 4.1 for Pilze dataset. However, the difference in the results is not as obvious in the V6 Google dataset due to the larger number of available images for training, introducing more information to the models.

5 Conclusions – Future work – Acknowledgments

In this thesis, instance segmentation was implemented on two dataset, testing 5 different architectures. One of the architectures was the breakthrough model of Mask R-CNN, while the rest are based on it with specific expansion on different parts of the architecture. For each architecture (except DetectoRS), two different models were tested, with their difference located in the feature extraction backbone, where a lighter and heavier network was used with regard to the depth and the implementation. All of the above models were pre-trained from COCO Dataset, but their final stages were finetuned on the two datasets of this thesis, in an independent manner. The two datasets were comprised of mushroom images, with the first being a collection of Pleurotus clusters inside a mushroom farm (Pilze dataset), while the second being a collection of different mushroom species in natural environment (V6 Google dataset).

The experiments conducted showed that the task of instance segmentation on mushroom instances in both controlled and natural environment is valid. All of the architectures managed a performance ranging from acceptable to good. The characteristics of the datasets played an important role in the quality of the results. The dataset of the controlled environment was more homogenous, with only one mushroom species included and with high resolution images. However, the total number of images contained was relatively small. This has led to architectures that expanded the baseline architecture of Mask R-CNN at the final stages (which were finetuned in this thesis) with extra parameters to produce worse results (MS R-CNN, Cascade Mask R-CNN). On the other hand, architectures that expanded Mask R-CNN's architecture in its all extend (HTC, DetectoRS), showed same or improved results despite the small training set. The dataset of natural environment showed the same behavior with the previous dataset, but the underperformance of the architectures was not so apparent, as more data were available for the training. The slightly worse overall performance of the architectures on this dataset is attributed to the inclusion of many different species of mushrooms and the low resolution images. This counterbalanced the bigger number of training images.

Both numerical and visual results showed that the more complex and newer models can achieve very good results, if they are provided with big amounts of data. More specifically, the DetectoRS has managed to produce the best results in both cases even with small number of training images. If more training data is available then it is expected for that architecture to perform even better with a greater margin from the

others. However, if small number of data is available, simpler architectures with less trainable parameters can perform as well and even better than more complex and extended models. This becomes apparent with Mask R-CNN architecture in the Pilze dataset, outperforming most architectures and slightly underperforming against DetectoRS.

This thesis sets the ground for further possible work in this direction. Following are some suggestions for further work:

1) Expansion of the used datasets with more samples to validate the observed behavior of the models.
2) For a specific species of mushroom, find/create dataset for instance segmentation that also classifies instances as mature or immature.
3) Create model that except from instance segmentation, classifies the species of the detected mushrooms. So far, these two tasks are examined independently.
4) Examine more state-of-the-art architectures and compare them with the Mask R-CNN family of models.
5) Develop customized instance segmentation architecture based on any of the examined architectures of this thesis.
6) Examine the use of the presented or other instance segmentation architectures for robotic applications on mushroom collection.

Acknowledgements

The collection of the aforementioned Pilze dataset was funded by the ICT-AGRI-FOOD ERA-NET project MUSHNOMICS and was financially supported by the Department of Agriculture, Food and the Marine (DAFM) in Ireland, Grant Agreement number 2020EN506.

6 References

[1] A. Kamilaris, F.X. Prenafeta-Boldú, Deep learning in agriculture: a survey Comput. Electron. Agric., 147 (2018), pp. 70-90

[2] Ayaz M., Ammad-Uddin M., Sharif Z., Mansour A., Aggoune E.-H.M., (2019) Internet-of-Things (IoT)-based smart agriculture: Toward making the fields talk, IEEE Access, 7, art. no. 8784034, pp. 129551 - 129583, DOI: 10.1109/ACCESS.2019.2932609

[3] Ojo MO, Zahid A. Deep Learning in Controlled Environment Agriculture: A Review of Recent Advancements, Challenges and Prospects. Sensors. 2022; 22(20):7965. https://doi.org/10.3390/s22207965

[4] Juan Wu, Bo Peng, Zhenxiang Huang, Jietao Xie. Research on Computer Vision-Based Object Detection and Classification.6th Computer and Computing Technologies in Agriculture (CCTA),Oct 2012, Zhangjiajie, China.pp.183-188, 10.1007/978-3-642-36124-1_23

[5] A. Gongal, S. Amatya, M. Karkee, Q. Zhang, K. Lewis, 2015, Sensors and systems for fruit detection and localization: A review, Comput. Electron. Agric., 116 (2015), pp. 8-19, 10.1016/j.compag.2015.05.021

[6] Guo-Quan Jiang and Cui-Jun Zhao, "Apple recognition based on machine vision," 2012 International Conference on Machine Learning and Cybernetics, Xi'an, China, 2012, pp. 1148-1151, doi: 10.1109/ICMLC.2012.6359517.

[7] J. Lu, N. Sang, 2015, Detecting citrus fruits and occlusion recovery under natural illumination conditions, Comput. Electron. Agric., 110 (2015), pp. 121-130

[8] H. Lu, Z. Cao, Y. Xiao, Y. Li, Y. Zhu, 2016, Region-based colour modelling for joint crop and maize tassel segmentation Biosyst. Eng., 147 (2016), pp. 139-150

[9] Kurtulmuş, F., Kavdir I. 2014. Detecting corn tassels using computer vision and support vector machines. Expert Systems with Applications, 41, 7390-7397.

[10] Y. Xu, K. Imou, Y. Kaizu, K. Saga, 2013, Two-stage approach for detecting slightly overlapping strawberries using HOG descriptor, Biosyst. Eng., 115 (2) (2013), pp. 144-153, 10.1016/J.BIOSYSTEMSENG.2013.03.011

[11] Bargoti, Suchet and James Patrick Underwood. “Image classification with orchard metadata.” 2016 IEEE International Conference on Robotics and Automation (ICRA) (2016): 5164-5170.

[12] Mureşan, Horea and Oltean, Mihai. "Fruit recognition from images using deep learning" Acta Universitatis Sapientiae, Informatica, vol.10, no.1, 2018, pp.26-42. https://doi.org/10.2478/ausi-2018-0002

[13] P. Lin and Y. Chen, "Detection of Strawberry Flowers in Outdoor Field by Deep Neural Network," 2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC), Chongqing, China, 2018, pp. 482-486, doi: 10.1109/ICIVC.2018.8492793.

[14] P.A. Dias, A. Tabb, H. Medeiros Apple flower detection using deep convolutional networks Comput. Ind., 99 (2018), pp. 17-28, 10.1016/j.compind.2018.03.010

[15] Liu G, Nouaze JC, Touko Mbouembe PL, Kim JH. YOLO-Tomato: A Robust Algorithm for Tomato Detection Based on YOLOv3. Sensors. 2020; 20(7):2145. https://doi.org/10.3390/s20072145

[16] Kuznetsova A, Maleva T, Soloviev V. Using YOLOv3 Algorithm with Pre- and Post-Processing for Apple Detection in Fruit-Harvesting Robot. Agronomy. 2020; 10(7):1016. https://doi.org/10.3390/agronomy10071016

[17] S. Parvathi, S. Tamil Selvi Detection of maturity stages of coconuts in complex background using Faster R-CNN model Biosyst. Eng., 202 (2021), pp. 119-132, 10.1016/j.biosystemseng.2020.12.002

[18] Sa I, Ge Z, Dayoub F, Upcroft B, Perez T, McCool C. DeepFruits: A Fruit Detection System Using Deep Neural Networks. Sensors. 2016; 16(8):1222. https://doi.org/10.3390/s16081222

[19] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In Advances in Neural Information Processing Systems (NIPS).

[20] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016 pp. 779-788. doi: 10.1109/CVPR.2016.91

[21] He, K., Gkioxari, G., Dollár, P., & Girshick, R.B. (2017). Mask R-CNN. 2017 IEEE International Conference on Computer Vision (ICCV), 2980-2988.

[22] Y. Tian, G. Yang, Z. Wang, E. Li, Z. Liang Instance segmentation of apple flowers using the improved mask R–CNN model Biosystems Eng., 193 (2020), pp. 264-278, 10.1016/j.biosystemseng.2020.03.008

[23] Yang Yu, Kailiang Zhang, Li Yang, Dongxing Zhang, Fruit detection for strawberry harvesting robot in non-structural environment based on mask-RCNN, Comput. Electron. Agric., 163 (April) (2019), Article 104846, 10.1016/j.compag.2019.06.001

[24] C. Zheng, P. Chen, J. Pang, X. Yang, C. Chen, S. Tu, Y. Xue, A mango picking vision algorithm on instance segmentation and key point detection from RGB images in an open orchard, Biosyst. Eng., 206 (2021), pp. 32-54, 10.1016/j.biosystemseng.2021.03.012

[25] Santos T.T., de Souza L.L., dos Santos A.A., Avila S., Grape detection, segmentation, and tracking using deep neural networks and three-dimensional association, Comput. Electron. Agric., 170 (2020), Article 105247

[26] Perez-Borrero, I., Marin-Santos, D., Gegundez-Arias, M.E., & Cortes-Ancos, E., 2020. A fast and accurate deep learning method for strawberry instance segmentation. Comput. Electron. Agric. 178, 105736, 10.1016/j.compag.2020.105736.

[27] Perez-Borrero, I., Marin-Santos, D., Vasallo-Vazquez, M.J. et al. A new deep-learning strawberry instance segmentation methodology based on a fully convolutional neural network. Neural Comput & Applic 33, 15059–15071 (2021). https://doi.org/10.1007/s00521-021-06131-2

[28] Gené-Mola J., Sanz-Cortiella R., Rosell-Polo J.R., Morros J.-R., Ruiz-Hidalgo J., Vilaplana V., Gregorio E., Fruit detection and 3D location using instance segmentation neural networks and structure-from-motion photogrammetry, Comput. Electron. Agric., 169 (2020), Article 105165

[29] H. Kang, C. Chen, Fruit detection, segmentation and 3D visualisation of environments in apple orchardsm, Computers and Electronics in Agriculture, 171 (February) (2020), Article 105302, 10.1016/j.compag.2020.105302

[30] Shen L, Su J, Huang R, Quan W, Song Y, Fang Y and Su B (2022) Fusing attention mechanism with Mask R-CNN for instance segmentation of grape cluster in the field. Front. Plant Sci. 13:934450. doi: 10.3389/fpls.2022.934450

[31] P. Ganesh, K. Volle, T.F. Burks, S.S. Mehta, Deep orange: Mask R-CNN based orange detection and segmentation, IFAC PapersOnLine, 52 (30) (2019), pp. 70-75, 10.1016/j.ifacol.2019.12.499

[32] X. Liu, D. Zhao, W. Jia, W. Ji, C. Ruan and Y. Sun, "Cucumber Fruits Detection in Greenhouses Based on Instance Segmentation," in IEEE Access, vol. 7, pp. 139635-139642, 2019, doi: 10.1109/ACCESS.2019.2942144.

[33] Chen C., Li B., Liu J., Bao T., Ren N., Monocular positioning of sweet peppers: An instance segmentation approach for harvest robots, Biosyst. Eng., 196 (2020), pp. 15-28

[34] Wang S, Sun G, Zheng B, Du Y. A Crop Image Segmentation and Extraction Algorithm Based on Mask RCNN. Entropy. 2021; 23(9):1160. https://doi.org/10.3390/e23091160

[35] W. Jia, Z. Zhang, W. Shao, S. Hou, Z. Ji, G. Liu, X. Yin, FoveaMask: A fast and accurate deep learning model for green fruit instance segmentation, Comput. Electron. Agric., 191 (2021), p. 106488

[36] Liakos KG, Busato P, Moshou D, Pearson S, Bochtis D. Machine Learning in Agriculture: A Review. Sensors. 2018; 18(8):2674. https://doi.org/10.3390/s18082674

[37] Afzaal U, Bhattarai B, Pandeya YR, Lee J. An Instance Segmentation Model for Strawberry Diseases Based on Mask R-CNN. Sensors. 2021; 21(19):6565. https://doi.org/10.3390/s21196565

[38] Tassis L.M., de Souza J.E.T., Krohling R.A., A deep learning approach combining instance and semantic segmentation to identify diseases and pests of coffee leaves from in-field images, Comput. Electron. Agric., 186 (2021), Article 106191

[39] Rossi, L., Valenti, M., Legler, S.E., Prati, A. (2022). LDD: A Grape Diseases Dataset Detection and Instance Segmentation. In: Sclaroff, S., Distante, C., Leo, M., Farinella, G.M., Tombari, F. (eds) Image Analysis and Processing – ICIAP 2022. ICIAP 2022. Lecture Notes in Computer Science, vol 13232. Springer, Cham.

[40] J.A. Bonet, M. Palahí, C. Colinas, T. Pukkala, C.R. Fischer, J. Miina, J. Martínez de Aragón. Modelling the production and species richness of wild mushrooms in pine forests of the Central Pyrenees in northeastern Spain. Canadian Journal of Forest Research. 40(2): 347-356. https://doi.org/10.1139/X09-198

[41] Bonet, J.A., GonzÁlez-Olabarria, J.R. & Martínez De Aragón, J. Mushroom production as an alternative for rural development in a forested mountainous area. J. Mt. Sci. 11, 535–543 (2014). https://doi.org/10.1007/s11629-013-2877-0

[42] Bonet, J.A., Pukkala, T., Fischer, C.R. et al. Empirical models for predicting the production of wild mushrooms in Scots pine (Pinus sylvestris L.) forests in the Central Pyrenees. Ann. For. Sci. 65, 206 (2008). https://doi.org/10.1051/forest:2007089

[43] Modeling and comparison of fuzzy and on/off controller in a mushroom growing hall. Measurement, 90:127–134, 2016.

[44] G. M. Fuady, A.H. Turoobi, M. N. Majdi, M. Syaiin, R.Y. Adhitya, Isa Rachman, F. Rachman, M. A. P. Negara, A. Soeprijanto, and R.T. Soelistijono. Extreme learning machine and back propagation neural network comparison for temperature and humidity control of oyster mushroom based on microcontroller. In 2017 International Symposium on Electronics and Smart Devices (ISESD), pages 46–50, 2017.

[45] Jennifer de la Cruz-del Amen and Jocelyn Flores Villaverde. Fuzzy logic-based controlled environment for the production of oyster mushroom. In 2019 IEEE 11th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM), pages 1–5, 2019.

[46] Karavani et al., 2018 A. Karavani, M. De Cáceres, J.M. de Aragón, J.A. Bonet, S. de-Miguel Effect of climatic and soil moisture conditions on mushroom productivity and related ecosystem services in Mediterranean pine stands facing climate change Agric. For. Meteorol., 248 (2018), pp. 432-440.

[47] Jitdumrong Preechasuk, Orawan Chaowalit, Fuangfar Pensiri, and Porawat Visutsak. 2020. Image Analysis of Mushroom Types Classification by Convolution Neural Networks. In Proceedings of the 2019 2nd Artificial Intelligence and Cloud Computing Conference (AICCC 2019). Association for Computing Machinery, New York, NY, USA, 82–88. https://doi.org/10.1145/3375959.3375982

[48] Royse, D. J., Baars, J., & Tan, Q. (2017). Current overview of mushroom production in the world. Edible and Medicinal Mushrooms, 5-13.

[49] Jegadeesh Raman, Kab-Yeul Jang, Youn-Lee Oh, Minji Oh, Ji-Hoon Im, Hariprasath Lakshmanan, and Vikineswary Sabaratnam. Cultivation and nutritional value of prominent ipleurotus/i spp.: An overview. Mycobiology, 49(1):1–14, November 2020.

[50] Carmen S´anchez. Cultivation of pleurotus ostreatus and other edible mushrooms. Applied Microbiology and Biotechnology, 85(5):1321–1337, December 2009.

[51] Marcelo Barba Bellettini, Fernanda Assump¸c˜ao Fiorda, Helayne Aparecida Maieves, Gerson Lopes Teixeira, Suelen ´Avila, Polyanna Silveira Hornung, Agenor Maccari J´unior, and Rosemary Hoffmann Ribani. Factors affecting mushroom pleurotus spp. Saudi Journal of Biological Sciences, 26(4):633–646, May 2019.

[52] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov, T. Duerig, and V. Ferrari. The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale. IJCV, 2020.

[53] Rakiba Rayhana, Gaozhi Xiao, and Zheng Liu. Internet of things empowered smart greenhouse farming. IEEE Journal of Radio Frequency Identification, 4(3):195–211, September 2020.

[54] Mohamed Rawidean Mohd Kassim, Ibrahim Mat, and Ismail Mat Yusoff. Applications of internet of things in mushroom farm management. In 2019 13th International Conference on Sensing Technology (ICST), pages 1–6, 2019.

[55] Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J. et al. (2019). MM Detection: Open MM Lab Detection Toolbox and Benchmark. arXiv.org. https://arxiv.org/abs/1906.07155. Last accessed 14 December 2022.

[56] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[57] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. arXiv preprint arXiv:2006.02334, 2020.

[58] Lin, T.-Y., Dollar, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature Pyramid Networks for Object Detection. 2017 IEEE Conference on Computer Vision and Pattern Recogni-tion (CVPR).

[59] Xie, S., Girshick, R., Dollar, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[60] Huang, Z., Huang, L., Gong, Y., Huang, C., & Wang, X. (2019). Mask Scoring R-CNN. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 6402-6411.

[61] Cai, Z., & Vasconcelos, N. (2019). Cascade R-CNN: High Quality Object Detection and Instance Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 43, 1483-1498.

[62] Chen, K., Pang, J., Wang, J., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Shi, J., Ouyang, W., Loy, C.C., & Lin, D. (2019). Hybrid Task Cascade for Instance Segmentation. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 4969-4978.

[63] Qiao, S., Chen, L., & Yuille, A.L. (2020). DetectoRS: Detecting Objects with Recursive Feature Pyramid and Switchable Atrous Convolution. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 10208-10219.

Figures catalog

Tables catalog