Master Thesis
at the University of Osnabrück

(for the degree of Master of Cognitive Science (M. Sc.))

Topic

Vegetation and Agriculture zone classification: Instance Segmentation

Submitted By
Saurabh Kumar Mishra
Jahnplatz 6, W 371, 49080 Osnabrück

Matriculation Number
982024

Submitted on
13.09.2021

First Supervisor
Prof. Dr. Joachim Hertzberg

Second Supervisor
Dr. Thomas Jarmer

Universität Osnabrück
Fachbereich Humanwissenschaften
Institute of Cognitive Science

Universität Osnabrück

Declaration of Authorship

I hereby certify that the work presented here is, to the best of my knowledge and belief, original and the result of my own investigations, except as acknowledged, and has not been submitted, either in part or whole, for a degree at this or any other university.

Osnabrueck, 13.09.2021

City, Date

Signature

Acknowledgement

I would like to express my gratitude to Professor Dr. Joachim Hertzberg for agreeing to supervise this project. I thank my mentor, M.Sc. Julius Autz for introducing me to the research area of instance segmentation on Satellite images. His advice has been priceless in helping me gain a deeper understanding of the subject. I cannot thank him enough for his valuable insights related to the research problem, his extreme levels of patience while discussing the general topic and his suggestions on other super-specific problems that I encountered; and lastly his extremely helpful inputs on writing and structuring the draft.

I also extend my heart-felt gratitude to my second supervisor Dr. Thomas Jarmer for introducing me to this fascinating research domain on multi-spectral Satellite images and later agreeing to support me in carrying out this particular study. I also thank Lucas Wittstruck for some fruitful discussions on the extraction of Satellite images from Google Earth Engine.

Many thanks to M.Sc. Taher Habib, M.Sc. Christian Johnson and Hunaid Hameed for proofreading parts of this thesis and giving constructive advice on editing the draft.

Lastly, I am grateful to my parents and my love for just being there for me, whenever and in whichever way I needed them.

Contents

List of Figures

List of Abbreviations

A
Adam Adaptive moment estimation
AP Average Precision

C
CNN Convolutional Neural Network
COCO Common Objects in Context
CRS Coordinate reference system

E
EM Electromagnetic

F
FC Fully-connected
FCIS Fully Convolutional Instance-aware Semantic Segmentation
FPN Feature Pyramid Network

I
IoU Intersection over union

M
mAP Mean average precision
MNC Multi-task Network Cascades

N
NAG Nesterov Accelerated Gradient
NMS Non-maximum Suppression

R
R-CNN Regions with CNN features
RMSprop Root Mean Square Propagation
RoI Region of Interest
RPN Region proposal network

S
SVM Support Vector Machine

Abstract

Deep learning models are state-of-the-art in many fields, and their application on satellite images is an active field of research. This thesis seeks to analyse the instance segmentation tasks on satellite images for vegetation/agricultural zones. The thesis focuses on classifying crops using deep CNNs models in the context of crop-cycles of a crop year.

1The two instance segmentation frameworks – FCIS and Mask R-CNN are employed and compared on vegetation/agricultural datasets. Furthermore, the study explores the above problem by designing an automated dataset creation pipeline for extracting polygons of crop fields from the shapefiles provided by the food ministry of Denmark and mapping them to satellite images. The document determines if FCIS can outperform Mask R-CNN and analyzes the reason for this.

1 Introduction

This chapter begins with motivation and inspiration for the thesis. Subsequently, the objectives of the thesis are introduced. Finally, a brief overview of the thesis structure is mentioned.

1.1 Motivation

Many of the decisions we make and actions we take, both individually and in groups, have an impact on our environment. To make informed decisions about the environment, we are equipped with senses that allow us to observe it. Observed data enables us to make effective changes around us when it is appropriate. Thus arises the need for reliable, detailed, timely and affordable geospatial data to understand urban growth, climate change and socioeconomic trends—the process of gathering geospatial data known as Earth Observation.

There is a wide range of different methods and techniques for Earth Observation; in-situ sensors observe by taking samples and measurements on-site, while remote sensing methods obtain measurements from distant sensors such as those on the space shuttle or Earth resource satellites [5].

The technological rise in satellite industries has increased the number of remote sensing satellites with diminishing launch and manufacturing costs. Consequently, the availability of satellite images has increased; thus, the research on satellite imaging has experienced an expansion in interest and demand over the last few years.[1]. In the past, analyses of satellite and aerial images were feasible primarily because the volume of images available was relatively low - but that is not the case now. Information extraction from images thus becomes a problem with the high volume of data. A significant part of these problems is annotation, where one identifies the structures and patterns visible in a satellite image. This problem is addressed in subsubsection 2.1.2, annotating large scale images using multispectral bands of satellites.

Multispectral sensors have several bands (channels); each spectral band detects EM energy reflected from objects at a specific wavelength range. For example, our eyes and the digital camera only see visible RGB bands. Near-infrared and shortwave infrared bands from the EM spectrum are used in satellite imagery to detect green pigment (chlorophyll) in plants [8].

Machine learning is the study of computer algorithms that automatically allow computer programs to improve through experience[1]. Deep learning is a subset of machine learning and refers to applying a set of algorithms called neural networks and their variants. In such methods, one provides the network (or model) with a set of labelled (annotated) examples that it learns or trains on[2]. A more detailed discussion of deep learning and neural networks is provided in subsection 2.2.

1.2 Objective and Scope

With this background in mind, the main objective of this thesis is to develop, implement, and experimentally examine a deep neural network for the instance segmentation of vegetation/agriculture zone in satellite images. This study aims to achieve two goals. First, evaluate the performance of convolutional neural network based instance segmentation models on Sentinel-2 imagery. This experiment determines if CNN based models can make use of Sentinel-2 images to provide a vegetation/agriculture zone segmentation model, which can be fine-tuned with images taken at a different point in time (same month but different year). Based on crop production cycle knowledge, it can be assumed that images of a specific crop will be identical during the same months each year. The second goal is to, compare the results to Rieke's [50] approach, which applied FCIS to the task of segmenting large agricultural fields in Denmark. In summary, this thesis addresses three research questions:

1. Does better model Mask R-CNN (on Imagenet dataset) outperforms the FCIS model on vegetation/agriculture zone segmentation dataset?

2. Which crops perform better on CNN networks?

3. Do individual crop images taken at different points of time lead to similar results?

[1] Google Earth provides satellite imagery from Landsat, Sentinel, etc publicly via its code interface. `https://earthengine.google.com/`

[2] This particular process is called supervised learning. Unsupervised learning, on the other hand, is characterised by training on unlabeled examples. In such cases, one deploys an autoencoder, beyond the scope of this work

1.3 How to read this Thesis

There are three sequential stages (parts) of the thesis:

- **Background** Chapter 2 provides an overview of how satellites are used in remote sensing and a short review of deep learning techniques. In Chapter 3, the reader learns how to formulate a computer vision problem using convolutional neural networks by diving deeper into the theory of convolutional neural networks. A detailed description of object detection and segmentation models are provided in chapter 4. Lastly, Chapter 5 presents the data used in this thesis and details how annotated polygons for vegetation/agriculture zone were obtained.

- **Development and Approach** Chapter 6 discusses the methods used in the thesis experiments and how instance segmentation approaches are built. Chapter 7 builds on the understanding from Chapter 6, explains the implementation details of FCIS and Mask R-CNN. Chapter 8 discusses the experiments used in the thesis to carry out the research. In chapter 9, the results are presented, both qualitatively and quantitatively, by using the evaluation metrics introduced in chapter 5.

- **Conclusion** In chapter 10, the results from chapter 9 are discussed and analysed. Chapter 11 concludes the thesis and reviews the objectives set out above in subsection 1.2 and, finally, makes recommendations on possible extensions to the existing work.

Part I
Background

2 Overview

In this chapter, we take a look at satellite imagery and introduce deep learning. The first section summarizes how one obtains satellite imagery today and its different use cases. After that, there is a brief introduction to deep learning, which is currently showing the most promising results in computer vision.

2.1 Satellites and Earth Observation

Earth observation gathers information about Earth's physical, chemical and biological systems from remote sensing platforms such as satellites, aircraft and subsurface measurements and mapping. The first spacecraft to have taken pictures of Earth was Explorer 6, launched on August 7, 1959 [60]. Since then, the number of Earth observation and remote sensing satellites has increased. Earth observation satellites have found applications in many fields, ranging from cartography, urban planning, disaster relief, real estate management, military intelligence and climate studies. Earth science is a broad discipline encompassing land cover issues, but not all satellite missions can be used for global land cover mapping. To be helpful for this, a satellite must be able to produce images of the land surface. This thesis study focuses on satellite images in the understanding of vegetation and agricultural land structures.

2.1.1 Remote sensing

Remote sensing is the art, science, and technology of observing an object, scene, or phenomenon by instrument-based techniques from a far distance without physical contact with an object of interest[5]. In remote sensing, sensors used to sense the interaction between earth surface materials and electromagnetic energy. The interaction between the radiation and the object of interest conveys information required on the nature of the object, such as reflection, coefficient, emittance and roughness[49]. These sensors broadly categorised as active and passive sensors. Passive sensors use existing energy sources, while active sensors produce their energy [5].

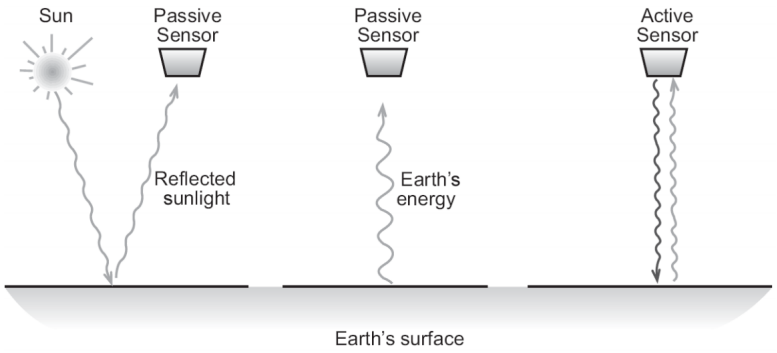


Figure 1: *A remote sensor measures reflected or emitted energy. An active sensor has its source of energy. [5]*

A satellite consists of passive sensors which detect the emitted electromagnetic radiation of different bands (i.e. radio, microwaves, infra-Red and visible light) to capture an image with multiple bands. The subsubsection 2.1.2 addresses satellite imagery in more detail. The human eye and digital cameras capture objects which reflect a visible spectrum of electromagnetic radiation with a wavelength in the range of 380nm to 760nm. The reason for using the EM spectrum lies in the fact that each object reflects, transmits and absorbs light energy differently. This property of an object is referred to as its spectral signature and makes remote sensing possible. It is also essential to note interference by the Earth's atmosphere, which absorbs specific wavelengths in the EM spectrum. Sensors are therefore designed to measure only a specific range of wavelengths, known as atmospheric windows[44]. Figure 2 shows the atmospheric windows for the earth atmosphere.

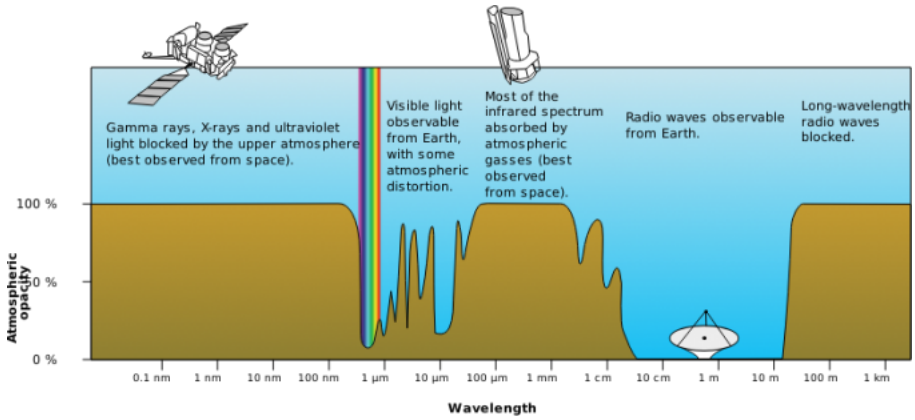


Figure 2: *Atmospheric Electromagnetic Opacity* [44]

2.1.2 Satellite Imagery

As discussed in the previous subsubsection 2.1.1, different surface's reflect and absorb the sun's electromagnetic radiation in different ways. The spectral signature of a material varies with the wavelength of EM energy, also known as spectral reflectance. The spectral reflectance measures how much energy a surface reflects at a specific wavelength. Knowing the spectral reflectance at a given wavelength in the EM spectrum can determine the type of object. Figure 3, Figure 4, Figure 5 and Figure 6 are spectral reflectance of water, soil, vegetation and ice, respectively.

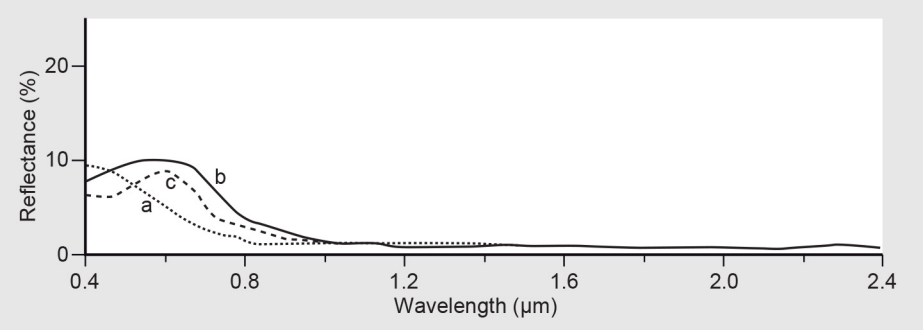


Figure 3: *Spectral Reflectance of Water (a) ocean water, (b) turbid water, (c) water with chlorophyll [55]*

The satellite captures data within specific wavelengths (bands) across the electromagnetic spectrum, combining any three bands in a multispectral image creates a different depiction of earth surfaces. For example, Landsat 8 (Figure 7) has eight bands in total, from visible light to shortwave infrared. A combination of any three of eight bands will produce several depictions of the Earth surface. Figure 5 shows the near-infrared region where plants can be easily perceived; as a result, the near-infrared band could be mapped to RGB channels to annotate vegetation and agriculture zones. In this work, a different approach for annotation was employed. Instead of using near-infrared bands, vegetation/agriculture polygons extracted from food ministry of Denmark[42] were directly applied to Sentinel-2 raster image. More detailed information about dataset preparation process is mentioned in subsection 5.4. Figure 8 shows the bands' comparison between Landsat and Sentinel-2.

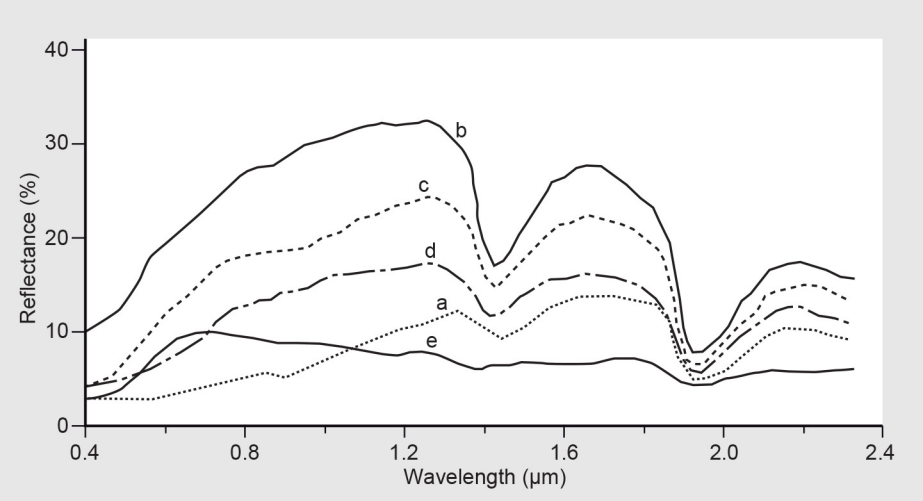


Figure 4: *Spectral Reflectance of Soil (a) organic dominated, (b) minimally altered, (c) iron altered, (d) organic affected and (e) iron dominated [55]*

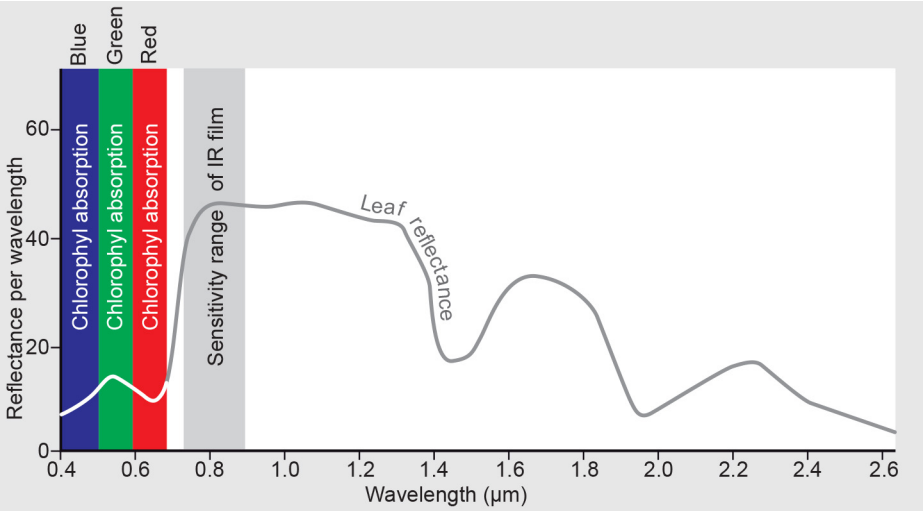


Figure 5: *Spectral Reflectance of Vegetation [55]*

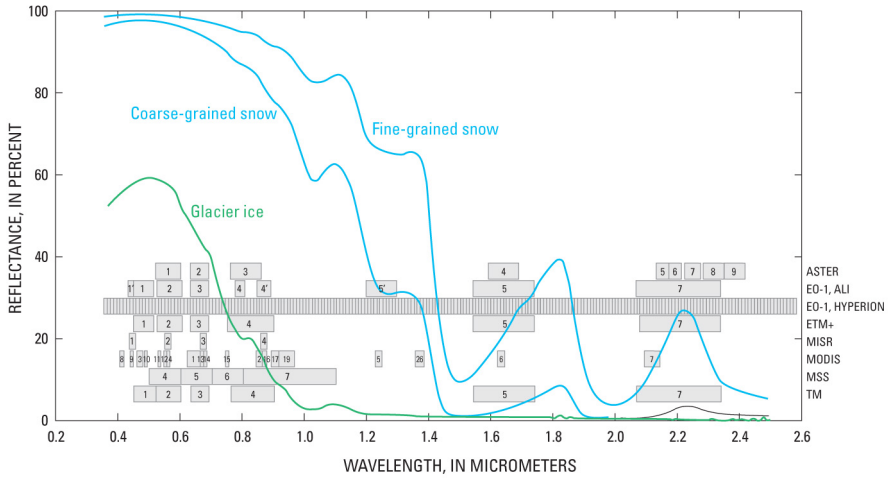
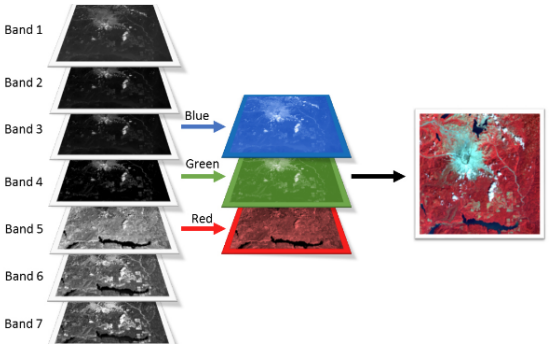


Figure 6: *Spectral Reflectance of Ice and Snow [57] [58]*



(a) *Landsat bands*

Composite Name	Bands
Natural Color	4 3 2
False Color (urban)	7 6 4
Color Infrared (vegetation)	5 4 3
Agriculture	6 5 2
Healthy Vegetation	5 6 2
Land/Water	5 6 4
Natural With Atmospheric Removal	7 5 3
Shortwave Infrared	7 5 4
Vegetation Analysis	6 5 4

(b) *Band Combinations for Landsat 8*

Figure 7: *Landsat 8 bands combination [56]*

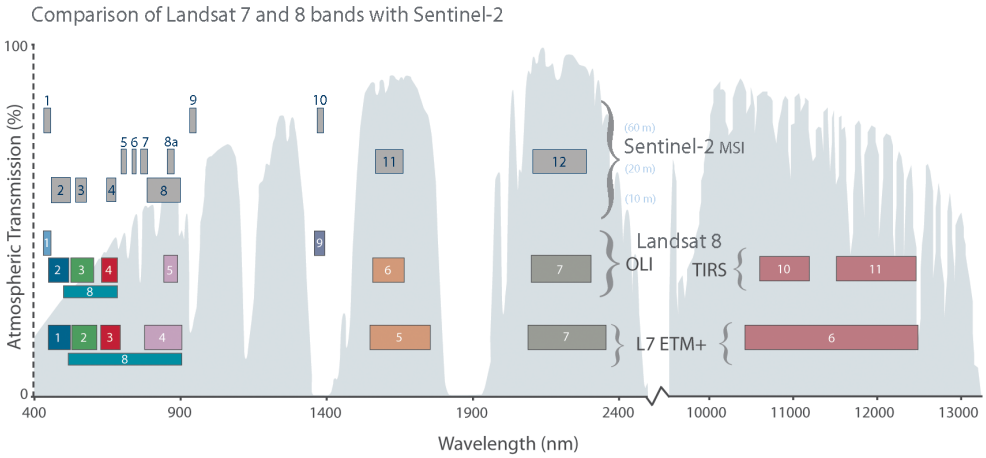


Figure 8: *Comparison of Landsat 7 and 8 bands with Sentinel-2. In figure, sentinel-2 has 13 bands and different combinations is used to analyses behaviour of earth environment. Sentinel-2 bands are shown in grey colored box with blue boundaries. [58]*

2.2 Deep Learning

Neural networks have garnered much attention only in the last few years in the computer vision and machine learning communities, despite having been around for decades. A network defines a mapping $\boldsymbol{y = f\,(x;\theta)}$ and learns the value of the parameters θ that result in the best function approximation[24]. Artificial neural networks are inspired by human brains and are generally visualised as layers of neurons on top of one another. Figure 9 shows one hidden layer neural network.

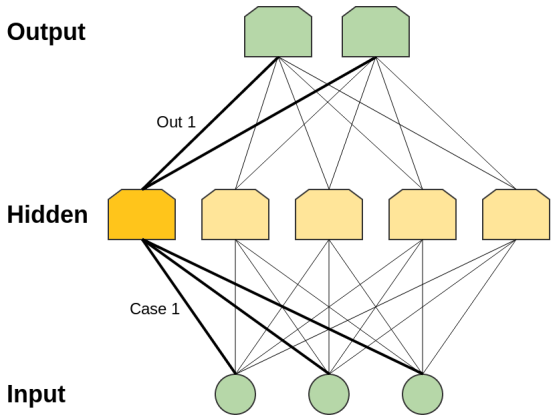


Figure 9: *One Hidden Layer Neural Network [47]*

The first layer is the input layer, which has a finite size; in our case, inputs are the pixel values of an image. Hidden layer (2^{nd} layer) neurons get the input by weighing up the results from the input layer. Similarly, the last output layer takes input from the hidden layer, passing through an activation function. In this way, a many-layer network of neurons can engage in complex decision-making. The parameters of a neural network are learned and trained by several optimisation techniques on various types of loss functions. Technically, the objective of the neural network is to minimise the loss by propagating the error back through the neurons from output to input, which is achieved by adapting the weights using a method known as backpropagation. In feed-forward networks, there is an input neuron for each pixel in the input image (for 28x28 image, the network has 784 input neurons). Here, implicit assumptions have been made that an image is a series of pixels that contrast spatial structure concepts. It leads to feed-forward networks treating input pixels close together and far away on the same standing[40]. Thus, a special neural architecture called the Convolutional Neural Network (CNN) was introduced for classifying images. A more detailed discussion of CNN is presented in next chapter.

3 Convolutional Neural Network

This chapter gives a brief overview of Convolutional Neural Network (CNN). It presents a description of the various layers of CNNs. Afterwards, we examine the backpropagation training algorithm on CNNs. Lastly, several modern CNN architectures are discussed.

3.1 Components of CNN

The regular neural network consists of linear functions where the input is a single vector of defined size, which later transforms into series of hidden layers. In contrast to a fully connected neural network (a.k.a linear layers), convolutional neural networks take input as an image. For example, if an image is 28x28 , then CNN input would look like Figure 10.

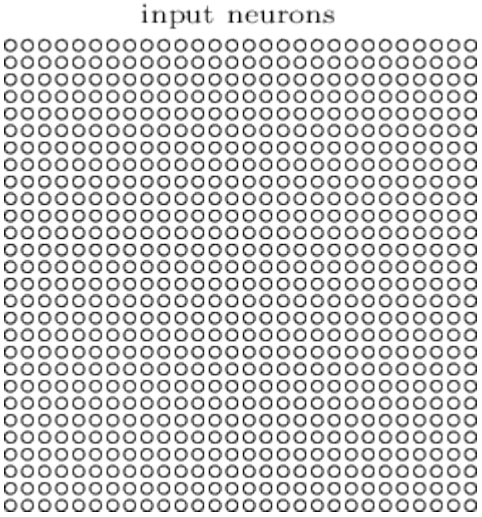


Figure 10: *28 x 28 input neurons [45]*

CNNs consist of several layers responsible for different essential operations, which are explained in the following subsections: convolutional layers in subsubsection 3.1.1, pooling layers in subsubsection 3.1.2, activation layers in subsubsection 3.1.3, fully connected layers in subsubsection 3.1.4 and classifier layers in subsubsection 3.1.5.

3.1.1 Convolutional Layer

Convolution is a mathematical operation on two functions that produces a third function that expresses how the shape is modified by the other. It is a specified as the integral of the product of the two functions after one is reversed and shifted[59][40]. Mathematically :

$$(f * g)(x) = \int f(z)g(x - z)dz \tag{1}$$

For discrete objects, the integral turns into a summation :

$$(f * g)(i) = \sum_{a} f(a)g(i - a) \tag{2}$$

Two-dimensional convolution

$$(f * g)(i, j) = \sum_{a} \sum_{b} f(a, b)g(i - a, j - b). \tag{3}$$

In image processing, instead of 1-D convolution, 2-D convolution is used. Images can be thought of as two-dimensional functions. Hence image transformations can be carried out by convolving the image with a very small, local function called a “Kernel". The kernel slides over the image and computes new pixel values by doing element-wise multiplication, this method known as

cross-correlation. In practice, cross-correlation operations are carried out in convolutional layers. Figure 11 shows the convolution operation on an image by a kernel.

While representing an image into vector form, it has the shape of $\boldsymbol{h}$ x $\boldsymbol{w}$ x $\boldsymbol{d}$, whereas $\boldsymbol{d}$ signifies the number of channels (depth) of an image. E.g. a standard RBG image has three channels. As the kernel slides over the $\boldsymbol{d}$ channels of an image, it produces a two-dimensional activation map that gives the responses of that kernel at every spatial position of $\boldsymbol{d}$ depth. Within one activation map (feature map), all neurons share the same parameters (weights and biases) and the number of output activation ($\boldsymbol{N}$) maps is stacked along the third dimension. E.g., if an image of size 7x7x3 has padding 1 convolved by two kernels of size 3x3x3 would give an output activation map of size 3x3x2 (see Figure 11). For a convolution layer to have the same height and width as the previous layer, it is common to add zeros around the inputs, known as zero padding (or padding by 1 pixel). Convolution takes another parameter stride that defines how much kernel moves in each spatial dimension and can be considered a means of subsampling the input. Hyperparameters $\boldsymbol{N}$ (number of filters), $\boldsymbol{f}$ (size of the kernel), $\boldsymbol{s}$ (stride) and $\boldsymbol{p}$ (padding) decide the output activation maps (feature maps) of the convolutional layer.

3.1.2 Pooling Layer

CNNs extract local features at a high resolution and successively combine these into more complex features at lower resolutions. The purpose of the pooling layer in CNNs is to achieve spatial invariance by down-sampling the input features spatially. A pooling layer is added after the convolutional layer, especially after a nonlinear activation function (see subsubsection 3.1.3).

In a pooling operation, a pooling window (kernel window) slides over each feature map and extracts a single value. This pooling window can be of arbitrary size, and windows may be overlapping. There are two different pooling operations: max pooling and average pooling. In the case of max-pooling, the maximum value from each slide is returned—an alternative is the average pooling, where the average value from each slide is returned instead (see Figure 12). Like the Convolutional layer, the pooling layer also has a stride parameter to control the output dimensions. As mentioned above, pooling provides spatial invariance in the image and also decreases the computational cost of the network by eradicating unnecessary information, which makes the network more efficient [24].

3.1.3 Activation Layer

Initially, neural networks were created to mimic the human brain and were inspired by how neurons in a brain are connected. By analogy with the human brain, nonlinear functions are used in artificial neural networks to model the firing of specific neurons in a network layer, and thus, they are called activation functions.

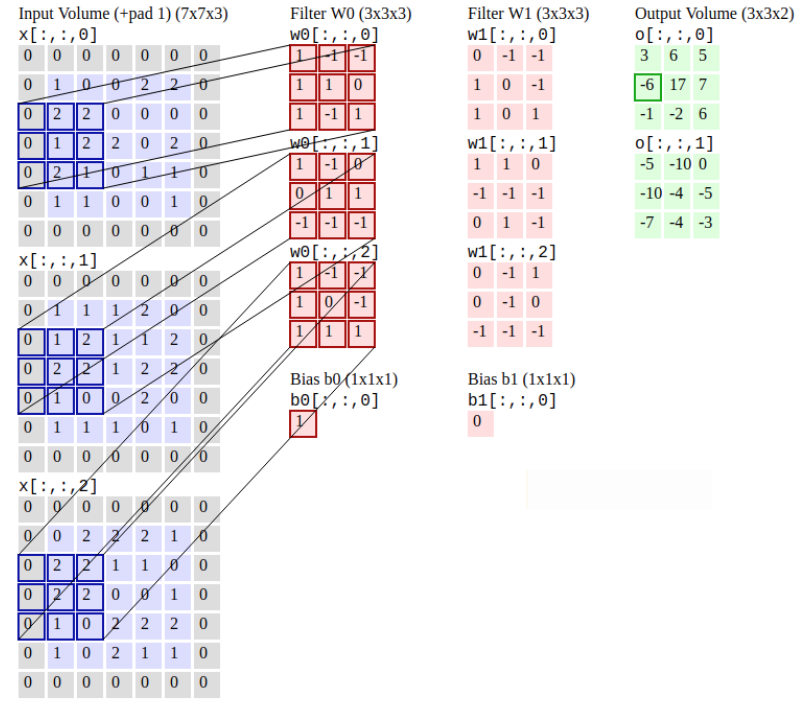
Traditionally, the outputs of a convolution layer are fed into activation functions at each layer to know whether the neuron in the kernel (filter) is activated or not. Several activation functions exist; notably, the Sigmoid, Tanh, Rectified Linear Unit (ReLU) functions are among them. In modern neural networks, the default recommendation is to use the rectified linear unit (ReLU depicted in Figure 13) due to vanishing and exploding gradient problems [53]. A ReLU unit activates by thresholding the negative inputs at zero and passing the positive inputs unchanged as in Equation 4.

$$g(z) = max(0, z). \tag{4}$$

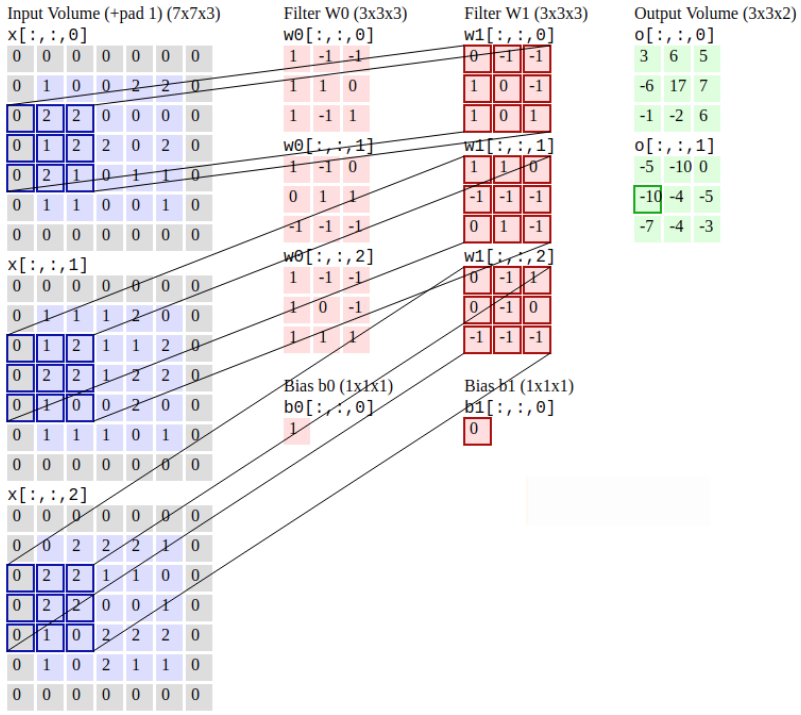
3.1.4 Fully Connected Layer

After the extraction of features from Convolutional and Pooling layers, a Fully Connected (FC) layer is usually attached to determine which features correspond with which class the most[53]. Since a neuron in a FC layer receives activations from all previous layer neurons, spatial information is lost, which is crucial in semantic and instance segmentation problems.

One way to overcome the lost information in FC layers is to use 1x1 convolution, an equivalent convolutional layer representation for FC layers known as a Fully Convolutional Network. The basic intuition behind Fully Convolutional Networks, introduced in [19]



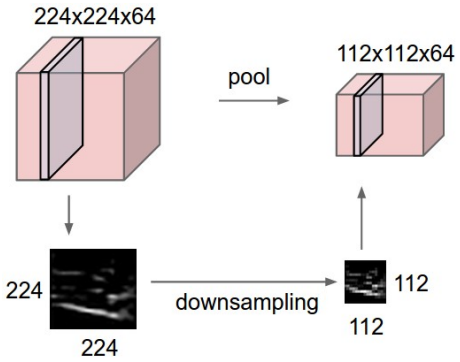
(a)



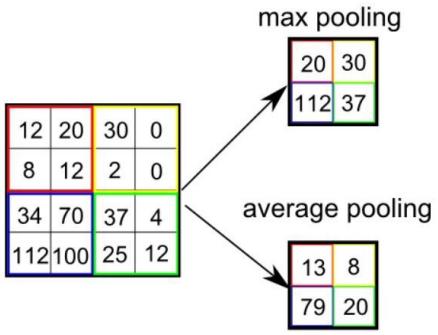
(b)

Figure 11: *Illustration of a convolution operation by two 3x3x3 kernels - w0 and w1 on 7x7x3 image. Output size is 3x3x2. [53]*

1x1 convolution was first introduced in the paper Network in Network (NIN)[14]. The goal was to generate a deeper network without simply stacking more layers and reducing the inception module dimension[14].



(a) *Illustration of a 2 × 2 pooling operation. The reduction in spatial resolution in the output layer, thereby making it invariant to local transformations in the input. [53]*



(b) *Illustration of a 2 × 2 max and average pooling operation. [51]*

Figure 12: *Pooling operations*

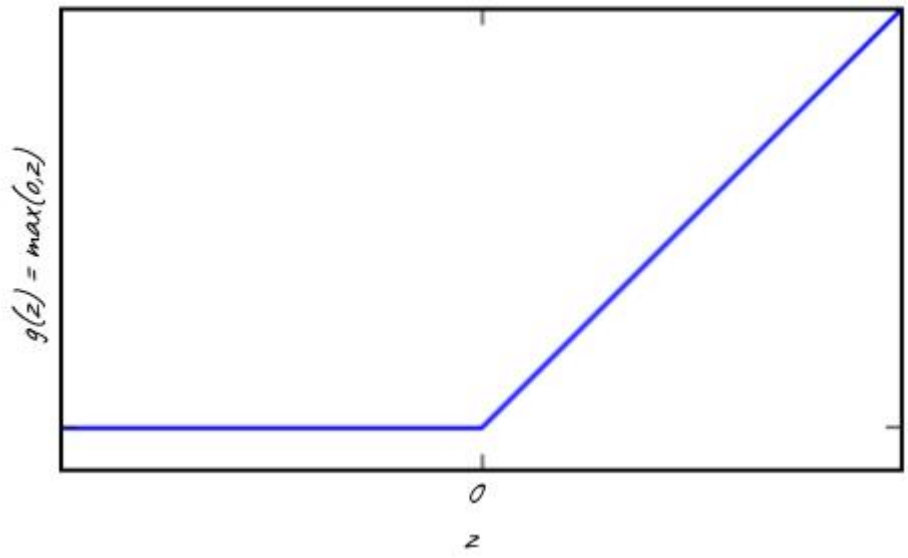


Figure 13: *Illustration of ReLU activation function. [24]*

3.1.5 Classifier

A classifier at the end depends on the problem statement. The softmax function is used for the multiclass problem; for binary classification, one may use logistic regression. Softmax converts the scores to a normalised probability distribution. In other words softmax function gives a probability value for a certain input, x_i belonging to a certain class $\boldsymbol{k}$ as in Equation 5, where $\boldsymbol{s}$ is the particular class from the previous layers.

$$p_i(s) = Pr(y = k|s = s_i) = \frac{exp(s_i)}{\sum_j^J exp(s_j)} \tag{5}$$

3.1.6 CNN Architecture

CNNs are commonly made up of three layers, i.e. Convolutional layers (subsubsection 3.1.1), Pooling layers (subsubsection 3.1.2) and Fully connected layers (subsubsection 3.1.4). LeNet was the first CNN architecture introduced to recognise the handwritten digits in images[40]. It consists of two convolution layers followed by three FC layers. The final output FC layer consists of 10 neurons, each for digits 1-9 holds the class scores. The subsection 3.3 discusses some modern CNN architectures.

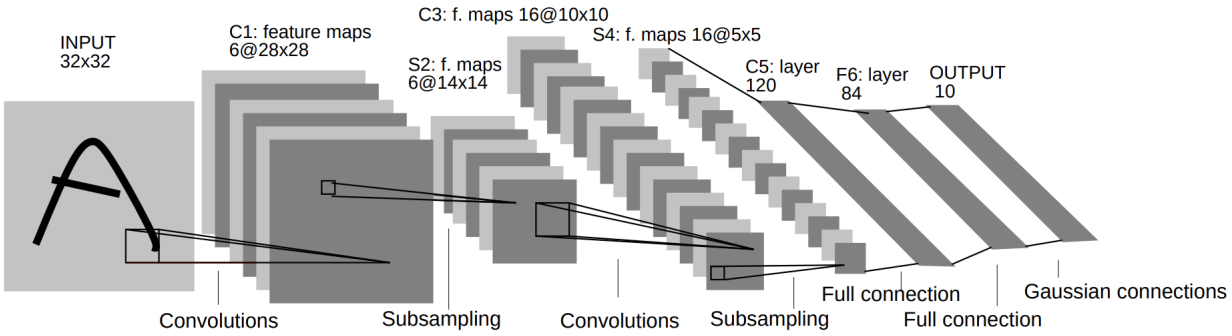


Figure 14: *Illustration of LeNet. [2]*

3.2 Training of CNN

In convolution layers, each layer has its own set of convolution kernels that have to be learned. Based on traditional computer vision techniques, we know which kernel is responsible for detecting edges, sharpening images, and blurring them[40]. In CNNs, it is challenging to assign kernels for large networks to specify what each kernel should be doing manually. Thus, CNNs try to learn kernel parameters and decide in the learning phase how kernels should be adapted throughout the network. In [13], authors showed that at different layers, the network learns to detect different features such as edges and curves at initial layers, and the last layers are responsible for learning a high-level representation of objects in an image.

There are three major steps for the learning process in convolutional neural networks.

1. Forward propagation
2. Error/loss optimisation
3. Backpropagation

3.2.1 Forward Propagation

The input image is fed through CNN architecture, which generally consists of the layers described in subsection 3.1 in mixed combinations. Generally, CNN layers stack alternative convolution and pooling layers, followed by fully connected layers at the end. The network returns the class score depending upon the classifier. For semantic and instance segmentation, a class score is provided for each pixel in the image, which are discussed in subsubsection 4.3.1 and subsubsection 4.3.2.

3.2.2 Loss Optimisation

In order to determine the network performance, some sort of error function needs to be defined. For the softmax classifier, cross-entropy loss (Equation 6) is used.

$$CE(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_{i=1}^{\mathrm{N}_c} y_i \log(\hat{y}_i) \tag{6}$$

where y is actual class and $\hat{y}$ is softmax output of the network.

In the forward step, the network gives output in terms of scores which need to be optimised by adjusting the parameter values (weights and biases) being learned in the network. The goal of the optimisation step is to minimise the total loss of the network, which is carried by the famous algorithm backpropagation.

3.2.3 Backpropagation

Backpropagation is a fundamental algorithm in fitting the neural network by computing the gradients of the loss with respect to the weights of the network. An example of backpropagation is illustrated in the Figure 15.

After the calculation of gradients, weights are adapted using the update rule given in Equation 7.

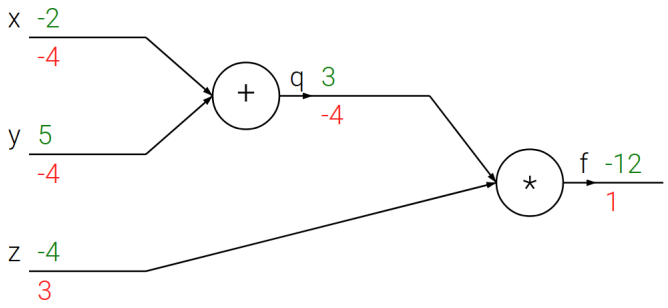


Figure 15: *Illustration of a simple computational graph for the equation* $(x+y)z$ *where* $x = 0-2$, $y = 5$, $z = 0-4$ *depicting backpropagation. Forward pass,values in green, computes from left to right, while backward pass, values in red, iteratively applies chain rule to calculate gradients first for the output, and flows back to the input. [54]*

$$\Theta_t = \Theta_{t-1} - \epsilon * \nabla J(\Theta_{t-1}) \tag{7}$$

where Θ_t is weights of network at time step t, ϵ is learning rate, J is loss function and ∇ is partial derivative.

This rule is known as the gradient descent method. Simple gradient descent and its variants might not be enough to train big deep networks. Therefore, optimisers are used to make gradient descent more efficient and faster while training. There are many gradient descent optimisers[3] such as RMSprop, Adam, NAG, among others. All the mentioned optimisers use another parameter, momentum, which helps in accelerating gradient descent in a relevant direction; it does this by adding a fraction γ of the weight change of the previous step to the weight vector at the current time (Equation 8 and Equation 9)[3]. That is, the modification of the weight vector at the current time step depends on both the current gradient and the weight change of the previous step.

The modification of the weight vector at the current time step depends on both the current gradient and the weight change of the previous step

$$v_t = \gamma v_{t-1} + \epsilon * \nabla_\theta J(\Theta) \tag{8}$$

$$\Theta = \Theta - v_t \tag{9}$$

3.3 Various Architectures

After the introduction of LeNet in 1998, it took around 14 years for CNNs to achieve a breakthrough in 2012. Alex et al. proposed the AlexNet in 2012, which won the championship in the ImageNet 2012 competition[9]. Since then, many researchers have invented a variety of CNN models which are deeper, wider and lighter. This section focuses on architectures that propose promising ideas in object detection and segmentation problems. Figure 16 shows the timeline of such architectures.

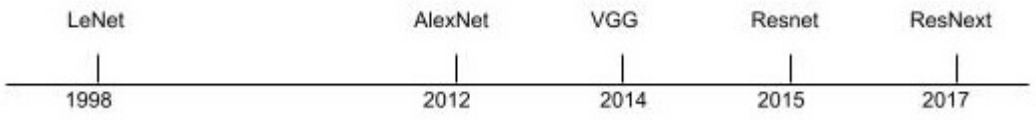


Figure 16: *Illustration of CNN Timeline in context of object detection and instance Segmentation*

[3]This link gives more detail information about gradient based optimisers - https://ruder.io/optimizing-gradient-descent/index.html

3.3.1 AlexNet

AlexNet[9] takes LeNet's ideas and applies the basic principles of CNN to a deep and wide network. AlexNet has eight layers, containing five convolutional layers and three fully-connected layers. It successfully leverages ReLU activation function, dropout, and local response normalisation (LRN) for the first time on CNN. At the same time, AlexNet also makes use of GPUs for computing acceleration. AlexNet shows that the use of data augmentation can substantially mitigate overfitting problem and improve generalisation ability. Alexnet was the backbone CNN architecture for the RCNN.

3.3.2 VGG

VGG are the series of convolutional neural networks proposed by the Visual Geometry Group of Oxford University. VGG secured first place in the localisation track of Imagenet Challenge 2014[22]. VGG introduced the concept of blocks [40], one VGG block consists of a sequence of convolutional layers followed by a maximum pooling layer. VGG uses 3x3 convolution kernels with padding of 1 (keeping height and width same) rather than 5x5, since small kernels have the same receptive field and add more nonlinearity to the network. VGG-16 and VGG-19 were the backbone CNN architecture for the Fast R-CNN.

3.3.3 ResNet

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 17: *Illustration of ResNet-18/34/50/101/152 with 5 convolution stages for ImageNet dataset, building blocks (conv layers) are shown in brackets with number of blocks stacked. Downsampling is performed by conv3 , conv4 , and conv5 with a stride of 2.[16]*

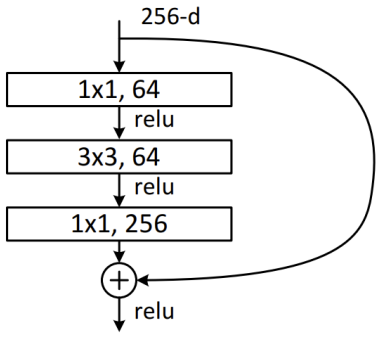


Figure 18: *Illustration of a "bottleneck" building block for ResNet-50/101/152 [16]*

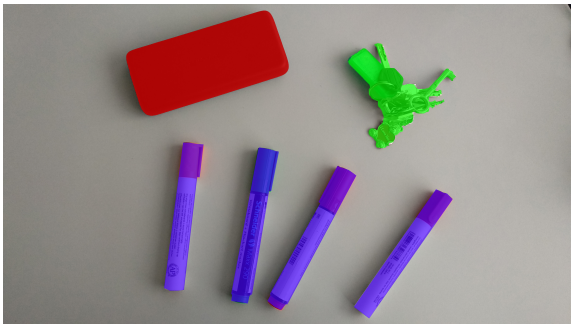
Historically, researchers have deepened networks to increase their complexity and expressiveness. He et al. [16] proposed a 34-layer Residual Network in 2016, which was the winner of the ILSVRC 2015 image classification and object detection algorithm. The idea of the ResNet was to use skip connections to preserve the gradient, which solves the vanishing gradient problem in deeper networks. In segmentation tasks, skip connections proved to be beneficial as lower semantic information from initial layers carried to later layers. Figure 18 shows three-layer residual blocks, called the "bottleneck" module, because the two ends of the block are narrower than the middle. Using 1×1 convolution kernel can reduce the number of parameters in the network and greatly improve the network's nonlinearity. 50-layer ResNet, 101-layer ResNet, and 152-layer ResNet employ three-layer residual blocks[16]. ResNet-50, ResNet-101 and variants of ResNet are still the backbones of state-of-the-art object detection and instance segmentation frameworks. The next chapter introduces object detection and instance segmentation in detail.

4 Object Detection and Segmentation

The purpose of this chapter is to provide a short conceptual introduction to object detection and segmentation. It presents an overview of the different frameworks and approaches used in object detection and segmentation pipelines. It begins with an introduction to common techniques and performance metrics. This is followed by a review of object detection. Finally, segmentation methods are discussed.

(a) *Classification of Power Bank, Keys and Pens*

(b) *Object Detection*



(c) *Semantic Segmentation*

(d) *Instance Segmentation*

Figure 19: *Illustration of Object Detection, Semantic Segmentation and Instance Segmentation*

Object detection is the process of detecting objects and locating them in digital images, which is an incremental step on the path from coarse to fine inference. Additionally, it provides the

locations of the image objects with which the classes have been classified. Bounding boxes and centroids provide the location information (Figure 19b). Using semantic segmentation, a fine inference can be made based on the predicted labels for every pixel in the input image. Pixels are labelled by the object class within which they are contained (Figure 19c). Instance segmentation gives different labels for separate instances of objects belonging to the same class (Figure 19d). Therefore, instance segmentation can be understood as the technique of solving both the object detection problem and the semantic segmentation problem simultaneously. Prior to explaining object detection and instance segmentation, it's imperative that we examine performance metrics and common techniques.

4.1 Performance Metrics and Common Techniques

4.1.1 Intersection over Union (IoU)

A metric for evaluating tasks such as segmentation, object detection and tracking is intersection over union (IoU), also known as the Jaccard index. It is defined as the ratio of intersection between the predicted bounding box and the ground truth bounding box and their union (Equation 10)[46].

$$IoU = \frac{(Areaofoverlap)}{(Areaofunion)} = \frac{|A \cap B|}{|A \cup B|} \tag{10}$$

where A and B are ground truth and predicted bounding box respectively.

After the calculation of IoU, a prediction matched with ground truth is considered to be True Positive (TP) if IoU is greater than a threshold; otherwise, it is a False Positive (FP). If a ground truth object is not matched by any predicted instance, it can be considered a False Negative (FN). The IoU threshold works as a hyperparameter in object detection and segmentation algorithms.

4.1.2 Non-Maximum Suppression

In the process of object detection, an algorithm may find multiple detections of the same object. Non-max suppression (NMS) ensures that each object is detected only once by the algorithm. NMS cleans up multiple detections by considering the highest probability box as a central bounding box and suppressing the remaining rectangles with high overlap or with a high IoU threshold[27]. In the end, NMS outputs one bounding box for every object in an image.

4.1.3 Mean average precision (mAP)

In the COCO dataset, mAP is used as a metric to evaluate the performance of object detection and segmentation algorithms. The mAP is calculated by averaging the Average Precision (AP) over all the categories (class) and hence named as mean average precision[18]. So, to calculate mAP, we first need precision. Precision (P) is the percentage of all predicted instances that match a ground truth object given in Equation 11. Recall (R) is the percentage of ground truth objects that were correctly predicted given in Equation 12.

$$P = \frac{TP}{TP + FP} \tag{11}$$

$$R = \frac{TP}{TP + FN} \tag{12}$$

AP is calculated by taking the area under the precision-recall curve to summarise the precision-recall curve into a single value representing the average of all precisions. In other words, the AP is the weighted sum of precisions at each threshold, with the increase in recall from the previous threshold used as the weight (Equation 13)[52]. AP and AR are calculated for different IoU thresholds, e.g. 0.5 and 0.75. AP@[0.5:0.95], which is the standard COCO segmentation challenge metric, averages the AP at IoU thresholds from 0.5 to 0.95 in 0.05 steps. For the assessment of multi-class problems, the AP over all classes or mean average precision (mAP) is used[18]. COCO segmentation challenge divides the assessment into three categories based on size of the area covered by object, i.e. S (small), M (medium) and L (Large)[4].

[4]More details about the COCO dataset can be found here `https://cocodataset.org/`

$$AP = \sum_{n} (R_n - R_{n-1}) P_n \tag{13}$$

where P_n and R_n are the precision and recall at the nth threshold.

4.2 Object Detection

There are chiefly two types of state-of-the-art object detectors[31]. On the one hand, we have detectors that employ two stages, such as Faster R-CNN (Region-based Convolutional Neural Networks) or Mask R-CNN, that (i) generate regions of interest in the first stage and (ii) pass the region proposals on for object classification and bounding-box regression. Despite their high accuracy rates, such models tend to take longer to calculate. In contrast, single-stage detectors, such as YOLO (You Only Look Once) and SSD (Single Shot MultiBox Detector), use an input image and learn the class probabilities and bounding box coordinates. These models achieve much lower accuracy rates but are much faster than multi-stage object detectors [31].

4.2.1 Single Stage Detector

Single stage detectors use only one feed-forward CNN to predict class probabilities and bounding box offsets directly from full images, encapsulating all computations into a single network without requiring post-classification or feature resampling. Single stage detector does not require prior region proposal generation, making them efficient in terms of speed. The first such architecture was DetectorNet by Szegedy[11]. DetectorNet formulated object detection as a regression problem to object bounding box masks. They use AlexNet[9] and replace the final softmax classifier layer with a regression layer. The development of several advanced and fast networks has occurred since DetectorNet, including Overfeat, Single shot multibox (SSD), You only look once (YOLO), and others[31].

4.2.2 Two Stage Detector

Two stage detectors are also known as region based networks, where category-independent region proposals are generated from an image. R-CNN architecture was the first architecture to explore region proposal algorithms, followed by CNNs for locating objects in regions and predicting objects' class through a classifier[31].

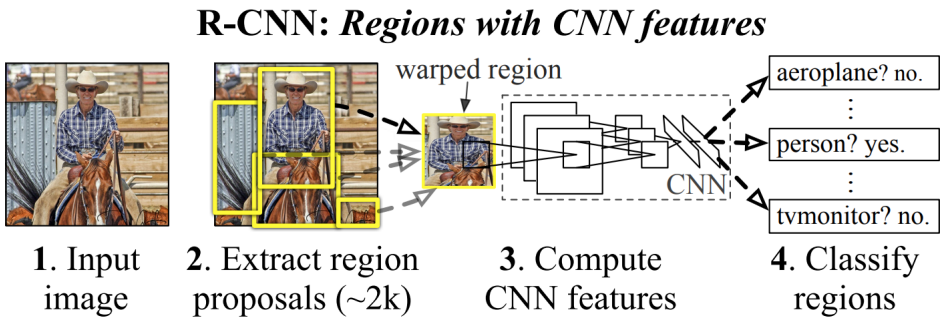


Figure 20: *Illustration of R-CNN workflow [10]*

4.2.2.1 Region Based Convolutional Neural Networks (R-CNN)

As the first paper of the R-CNN family[10], Region-based Convolutional Neural Network (R-CNN) used CNNs to improve detection performance to a large degree. To convert detection into classification and localization problems, R-CNN uses a class agnostic region proposal module. The main idea of R-CNN is composed of two steps. First, using selective search algorithm (region proposal module)[12], which proposes a defined number of regions. Second, for classification, it extracts CNN features using CNN architecture (AlexNet) for each region independently. The Workflow of R-CNN is as follows :

1. A mean-subtracted input image is first passed through the region proposal module–selective search, which produces 2000 object candidates. R-CNN training does not include a region proposal module– Stage 2 in Figure 20.

2. Region proposals are then warped to have a fixed size as per the input requirement of the pre-trained CNN network. The warped proposals are then propagated through a CNN network, which extracts a 4096-dimension feature vector for each proposal. The Original R-CNN [10] paper used pre-trained AlexNet on Imagenet dataset. The last fully connected layer is removed to give an output size of 4096. So, given every image region, one forward propagation through CNN generates a 4096 feature vector (in the case of AlexNet) – Stage 3 in Figure 20.

3. The resulting feature vectors from CNN are then passed to the trained, class-specific Support Vector Machines (SVMs) to obtain confidence scores. That is one SVM classifier for each object class. A non-maximum suppression (NMS) is then applied to the scored regions based on their IoU and class. Once the class has been identified, the algorithm predicts its bounding box using a trained bounding-box regressor, which predicts four parameters, i.e., centre coordinates of the box along with its width and height. – Stage 4 in Figure 20.

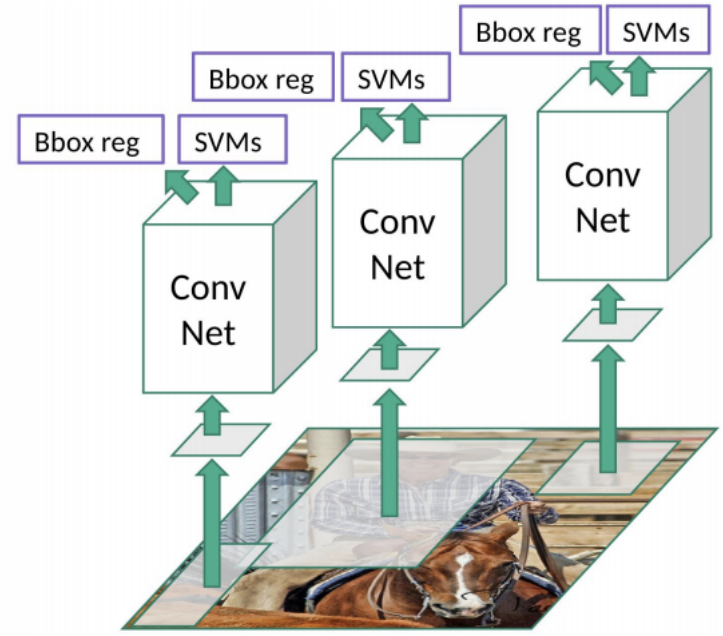


Figure 21: *R-CNN Architecture [43]*

All three of the above steps are independent and are coupled sequentially to make one network during inference. The training process for each stage performed independently. First, CNN architecture is trained as a regular classification problem on a large classification dataset (Imagenet[9]). Afterwards, a SVM gets trained for object classification by replacing the classification layer of CNN with a randomly initialized N+1-way classifier, N being the number of classes. The region proposals passed through the NMS technique to filter out negative samples. The positive samples are proposed regions with IoU (intersection) overlap threshold, in the paper threshold used as 0.3-0.5. At last, a bounding box regressor as shown in Figure 22 is used to correct the positive region (bounding box) for object localisation by using ridge regression given by Equation 14. Even though RCNN achieves high object detection results, it suffers from several drawbacks:

1. Each stage of training is performed separately, making training a slow and inefficient process.

2. Both, disk space and time are a concern when training SVMs and bounding box regressions as it is necessary to extract CNN features from every proposed object in every image, which poses great challenges for large-scale detection, particularly with very deep networks such as VGG16.

3. Due to the fact that CNN features are extracted per object proposal in each image without sharing computation, the process requires a long inference time.

$$
\begin{aligned}
\hat{g}_x &= p_w d_x(\mathbf{p}) + p_x \\
\hat{g}_y &= p_h d_y(\mathbf{p}) + p_y \\
\hat{g}_w &= p_w \exp(d_w(\mathbf{p})) \\
\hat{g}_h &= p_h \exp(d_h(\mathbf{p}))
\end{aligned}
$$

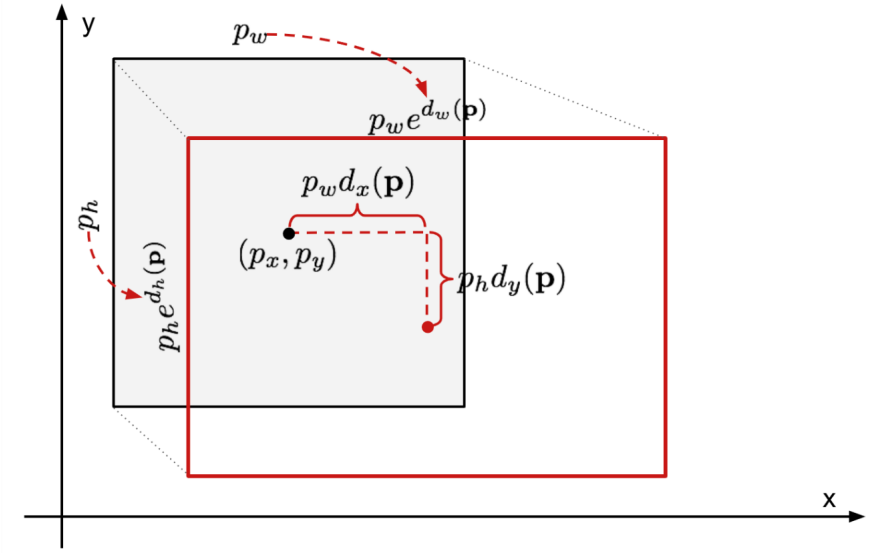


Figure 22: *Illustration of transformation between predicted and ground truth bounding boxes.* $p = (p_x, p_y, p_w, p_h)$ *are center coordinates, width and height of predicted box respectively.* $g = (g_x, g_y, g_w, g_h)$ *are center coordinates, width and height of ground truth box respectively. The goal is to learn a transformation that maps a proposed box p to a ground-truth box g. Authors parameterise the transformation in terms of four functions* $d_x(p), d_y(p), d_w(p), d_h(p)$*. The first two specify a scale-invariant translation of the center of p's bounding box, while the second two specify log-space translations of the width and height of p's bounding box. After learning these functions, we can transform an input proposal p into a predicted ground-truth box* $\hat{g}$ *by applying the transformation. Bounding box regression is achieved by using ridge regression given equation 10. [27][10].*

$$
L_{reg} = \sum_{i \in x,y,w,h} (t_i - d_i(p))^2 + \lambda ||w||^2 \tag{14}
$$

where w is learnable model parameters and λ is regularisation term. The regression targets t_i for training pair (p,g) are defined as

$$
\begin{aligned}
t_x &= (g_x - p_x)/p_w \\
t_y &= (g_y - p_y)/p_h \\
t_w &= log(g_w/p_w) \\
t_h &= log(g_h/p_h)
\end{aligned} \tag{15}
$$

4.2.2.2 Fast R-CNN

Girshick proposed the Fast R-CNN (Figure 23)[15], which overcomes some of the R-CNN's weaknesses while improving its detection speed and accuracy. Fast R-CNN enables end-to-end detector training by unifying three independent models into one jointly trained framework and increasing shared computation results rather than separately training a softmax classifier, SVMs, and Bounding Box Regressors (BBRs) as in RCNN. The workflow of Fast the R-CNN is as follows:

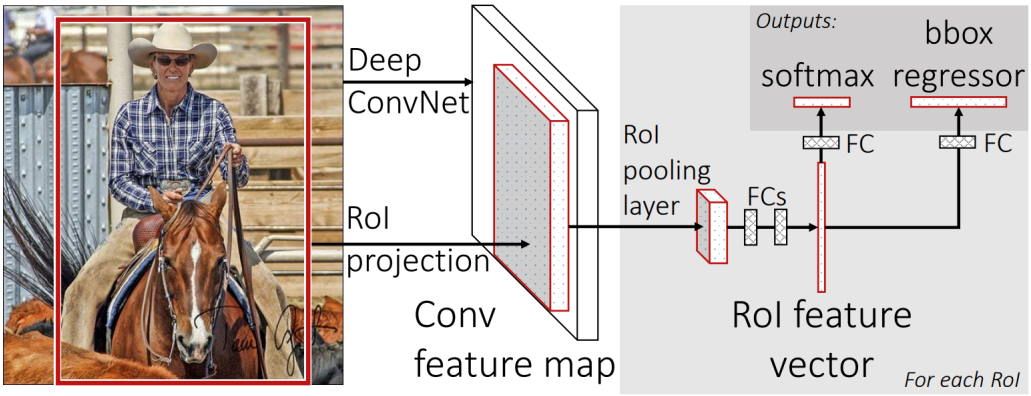


Figure 23: *Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are fed into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. [15]*

1. Image is passed through region proposal algorithm (selective search) to propose approximately 2000 regions per image.
2. CNN processes the entire image in order to produce feature maps (activation maps). . Then, for each object proposal, a region of interest (RoI) pooling layer extracts a fixed-length feature vector from the feature maps.
3. Each feature vector is fed into a sequence of FC layers. Finally, the model branches into two output layers, first a softmax layer which gives out probabilities for K+1 classes, one extra for background, and second a bounding box regressor that predicts four real-valued numbers relative to the original RoI for each of K classes.

After region proposal and feature extraction steps, there are two parallel outputs, i.e. a high-resolution image with region proposals and CNN feature maps. Now, Region of interest (RoI) pooling layer first projects region proposal onto feature maps (see Figure 24a). Thereafter, it applies a max pool layer to extract fixed-size features for each region proposal that is fed through the FC layer (see Figure 24b).

Fast R-CNN has two outputs: softmax probabilities for object category prediction and class-specific bounding box regression offsets for proposal refinement. In order to train the entire network end to end, a multi-task loss is introduced, which is a combination of classification loss and bounding box regressor given by Equation 16. Fast R-CNN in comparison with R-CNN improves the efficiency considerably, typically three times faster in training and ten times faster in testing. Thereby improving detection quality and updating all network layers in one process without requiring feature caching. Despite enhancing the speed of detection, Fast R-CNN depends on an external region proposal module to help identify regions, which makes it inefficient for real-time detection[31]. Faster R-CNN solves this problem by introducing a new CNN network for the region proposal called Region Proposal Network (RPN)[20].

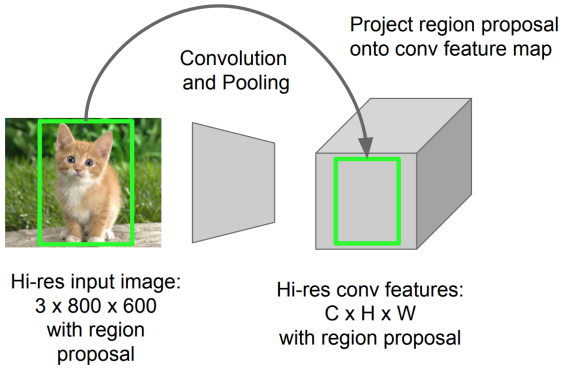
$$L_{p,u,t^u,v} = L_{cls}(p,u) + \lambda[u \geq 1]L_{loc}(t^u, v) \tag{16}$$

where L_{cls} is log loss for true class and L_{loc} is for bounding box regressor.

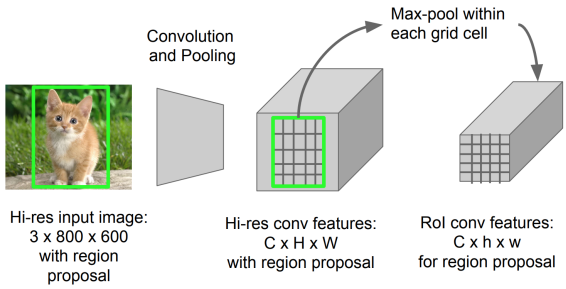
4.2.2.3 Faster R-CNN

Even though Fast R-CNN inched closer to real time object detection, its region proposal generation was still an order of slow magnitude (2 sec per image compared to 0.2 sec per image)

The Faster R-CNN[20] (Figure 25) framework proposed by Ren et al'. offered an efficient and accurate Region Proposal Network (RPN) for generating region proposals. They utilise the same architecture of Fast R-CNN for object detection and classification. The addition of RPN greatly speeds up the generation of region proposals ten times faster than selective search, and also makes Faster R-CNN the first fully differentiable object detection model. [31]. The workflow of Faster R-CNN is as follows:



(a) Region proposals projected on feature maps



(b) *Feature extraction using max pool layer for each region proposal from feature maps*

Figure 24: *Illustration of Region of interest pooling operation [41]*

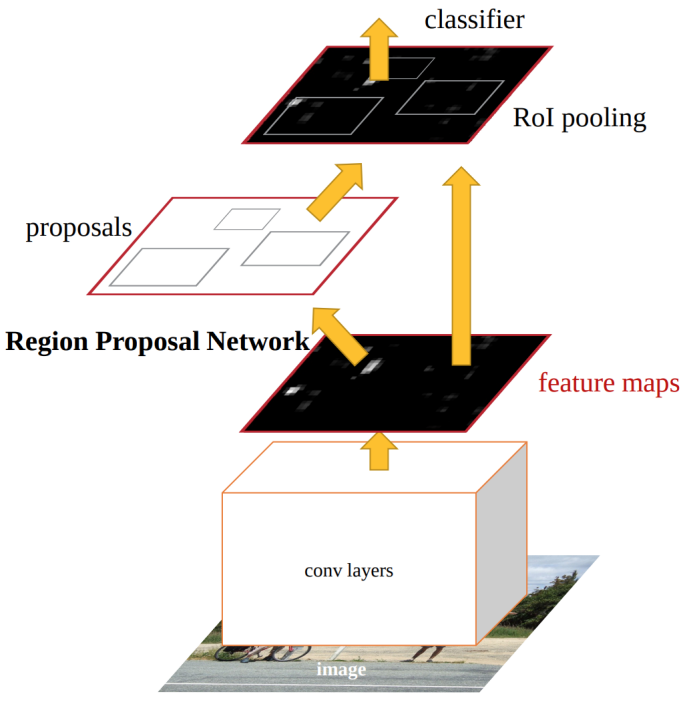


Figure 25: *Faster R-CNN architecture RPN + Fast R-CNN.[20]*

1. The image is passed through a CNN network (VGG16, ResNet), producing output feature maps.
2. A Region proposal network works on feature maps to generate regions with boxes and objectness scores.

RPN is a Fully Convolutional Network, which shares the convolution features with Fast R-CNN. It takes an arbitrary input image and outputs a set of candidate windows. Each such window has an associated objectness score which determines the likelihood of an object. RPN introduces Anchor boxes to solve the size variance of objects. It first initialises k reference boxes known as anchors of different scales and aspect ratios at each convolution feature map location[31]. At each sliding window, it predicts multiple region proposals (anchors). A sliding window is nothing but a kernel of size 3x3, as shown in the Figure 26, which connects with two parallel 1x1 convolutions.

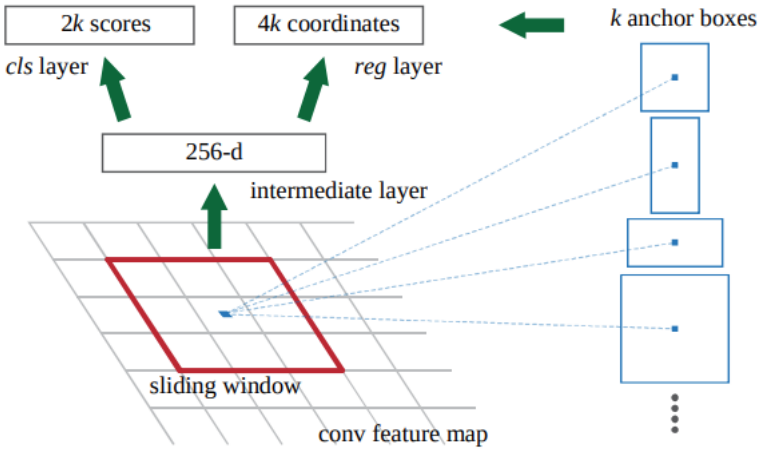


Figure 26: *Illustration of RPN [20].*

One has 2k output channels for background/foreground classification, and the other has 4k output channels for coordinates of regions. Since anchors usually overlap, proposals end up also overlapping over the same object. The NMS technique eliminates duplicate proposals by using an IoU threshold of 0.7[20]. The top N proposals are finally sorted by score following the application of NMS. The paper uses N=2000[20].

Before training, the RPN is first pre-trained on the ImageNet dataset[4] and fine-tuned on the PASCAL VOC dataset[6]. Then, Fast R-CNN is fine-tuned from the updated RPN. From RPN to Fast R-CNN connection, binary cross-entropy loss is used for the classification of foreground and only foreground samples are used for bounding box regression loss. For the calculation of targets, foreground anchor and closest ground truth object are used. It is possible to end up with zero foreground anchor depending on the ground truth objects in the image, the size and ratios of anchors. In above cases, the paper suggests to use the anchors with the biggest IoU along with the ground truth boxes. In total, there are four losses – two from RPN (foreground/background and region losses) and two from Fast R-CNN (classification and bounding box losses) combined to train Faster R-CNN end to end.

4.3 Semantic and Instance Segmentation

Semantic segmentation is defined as “The task of semantic segmentation is to label all pixels within an image as belonging to one of N classes" [34]. On the other hand, instance segmentation is a correct separation of all objects in an image while also semantically segmenting each instance at the pixel level [38].

4.3.1 Semantic Segmentation

As mentioned before, the semantic segmentation process involves tagging each pixel of an image with a class (Figure 19c). For pixel-wise prediction, Long et al. introduced a Fully Convolutional Network (FCN) trained with end-to-end supervised training in 2015 [37]. In the FCN, images are downsampled to generate feature maps, which are then upsampled for semantic segmentation. Later, U-net [21] improved FCN’s upsampling by increasing the number of feature channels to the same amount as in the downsampling; which allows the network to propagate context information to higher resolution layers. In this way, the network has a symmetrical shape that can be drawn as a U; hence it is named as U-net. Similar to FCN, the upsampling and downsampling are linked using

a skip connection to preserve local information in the U-net. DeepLabV3+, proposed by Cheng et al. [29], which improved DeepLabV3 into an encoder-decoder model, is currently at the cutting edge of semantic segmentation. Among numerous other things, DeepLab utilises atrous/dilated convolutions. Atrous convolutions are beneficial for gaining a large view while preserving spatial resolution, which normal pooling does not do[5].

4.3.2 Instance Segmentation

As discussed above, instance segmentation determines segmenting and classifying individual object instances in an image. Microsoft Common Objects in Context (COCO) dataset[18] provide a platform to evaluate the instance segmentation frameworks. In this thesis, two architectures are used and compared "FCIS and Mask-RCNN". The current state-of-the-art model for instance segmentation at the time of writing the thesis, Swin Transformer [35] achieves 50.04 mAP on the COCO dataset. Mask-RCNN with Swim Transformer as the backbone achieves 43.3 mAP [36], whereas novel Mask-RCNN with FPN as backbone achieves 37.1 mAP on the COCO dataset. Fully Convolutional Instance-aware Semantic Segmentation (FCIS) was the first end-to-end trainable fully convolutional network architecture for Instance segmentation. It extends upon the ideas of position-sensitive class score maps for fully convolutional segment proposal prediction from Multi-task Network Cascade (MNC). Compared to Mask R-CNN, FCIS performs object segmentation and classification simultaneously, whereas Mask R-CNN extends a separate fully convolutional branch from Faster R-CNN to predict RoI segmentation masks.

4.3.2.1 Fully Convolutional Instance-aware Semantic Segmentation (FCIS)

FCIS was the first fully convolutional end-to-end solution for the instance-aware semantic segmentation task [25]. The Position-Sensitive Inside/Outside Score Maps allow convolutional representation to be fully shared for detection and segmentation. The workflow of FCIS is as follows:

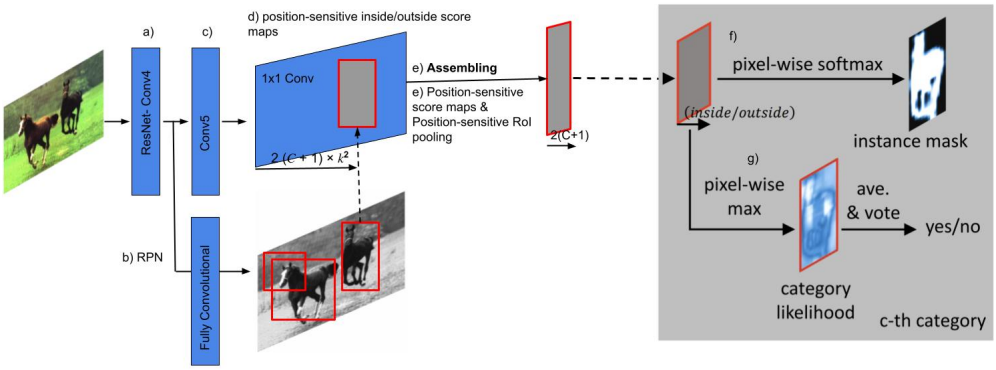


Figure 27: *Illustration of FCIS [25].*

1. The ResNet-101 model is used as convolutional network architecture, where the last fully connected layer for 1000 class classification is discarded. Only the previous convolutional layers are retained. Thus, resulting feature maps have 2048 channels. On top of resnet feature maps, a 1×1 convolutional layer is added to reduce the dimension to 1024 as shown in Figure 27a and Figure 27c [25]. For instance-aware semantic segmentation, 32 feature size on the last convolution (conv5) is reduced to 16 pixels because the downsampling factor of 32 in the last convolutional layer of ResNet produces too coarse feature maps[25].

2. A parallel branch of a fully convolutional network from Conv4 of ResNet (see Figure 17), is used as a region proposal network, as shown in Figure 27b.

3. From the conv5 feature maps (Figure 27c), $2k^2 \times (C + 1)$ score maps are produced (C object categories + one background category and two sets of k^2 position-sensitive inside/outside

[5]This thesis is limited to instance segmentation, a detailed information about Semantic segmentation can be found at [37] `https://arxiv.org/abs/1912.10230`

score maps per category (class), k=7 used by the paper), using a 1×1 convolutional layer (Figure 27d). Each RoI is projected into a 16× smaller region over the score maps as conv5 is reduced to 16 pixels.

4. Given a region-of-interest (RoI), its pixel-wise score maps are produced by the assembling operation within the RoI (Figure 27e). For each RoI, one inside and one outside score map is assembled from the respective RoI area on the sets of inside/outside score maps of the full image. The k^2 (i.e. 7x7) position-sensitive scores give vote by averaging the scores on the RoI, producing a (C + 1)-dimensional vector for each RoI [23] (Figure 27e). This means that each RoI would have C + 1 inside/outside scores.

5. For each pixel in a RoI, there are two tasks: First, detection: whether pixel belongs to an object bounding box at a relative position (detection+) or not (detection-); Second, segmentation: whether pixel is inside an object instance's boundary (segmentation+) or not (segmentation-) [25]. The FCIS paper used two classifiers to achieve object detection and segmentation separately. The class-specific foreground probability mask is determined by pixel-wise softmax operations (the pixel has a high probability to belong to an object mask if it has a high RoI-inside and low RoI-outside score). A pixel-wise maximum operation yields a class-specific category likelihood score map (high pixel-wise probability belong to the bounding box of an actual image object in case of combined low RoI-inside with high RoI-outside scores or low RoI-inside with low RoI-outside scores). The detection score of the whole RoI (category likelihood of object, Figure 27g) is then obtained via average pooling over all pixels' likelihoods followed by a softmax operator across all the categories (C+1). To filter out the overlap in RoIs, non-maximum suppression with an IoU threshold of 0.3 is applied. For the remaining RoIs, the foreground probability mask is selected by the class label and the final segment mask is determined by mask voting. The masks are averaged pixel-wise (weighted by the RoIs' detection scores) and then binarized (Figure 27f). At last, bounding box (bbox) regression is used to refine the RoIs. A sibling 1×1 convolutional layer with $4k^2$ channels is added on the conv5 feature maps to estimate the bounding box shift in location and size.

While training, "each RoI has three loss terms in equal weights: a softmax detection loss over C + 1 categories, a softmax segmentation loss over the foreground mask of the ground-truth category only for postive RoIs, and a bbox regression loss for positive RoIs". The model is first initialised from the pre-trained ImageNet classification, and missing layers are initialised randomly. FCIS is then trained via stochastic gradient descent with Online Hard Example Mining (OHEM) bootstrapping. The addition of OHEM increased the mAP by 1.2%. After the RoI and score map generation, all layers implement simple, fixed functions, enabling very efficient per-RoI computations.

4.3.2.2 Mask R-CNN

Mask R-CNN[30] utilises the same concept of a two-stage detector, where the first stage is the same as RPN. The second stage adds a fully convolutional network branch that outputs a binary mask for each RoI. In contrast to FCIS, where classification depends on mask prediction, Mask R-CNN follows the approach of Faster R-CNN that applies bounding-box classification and regression in parallel. The workflow of Mask R-CNN (mask branch) is as follows:

1. The mask branch consists of a fully convolutional network for each RoI, which outputs a binary mask having a dimension of km^2 for each k class of resolution m x m.

2. Binary pixel-wise classification is done by using sigmoid activation and binary loss.

The Mask R-CNN paper's ablation experiment is performed on two main architectures, i.e., i) Feature pyramid Network, and ii) C4, where features are extracted from conv4 of ResNet.

The Feature pyramid network (FPN) [26] solved the problem of recognising objects at vastly different scales by scanning the model over both positions and pyramid levels. It takes a single-scale image of an arbitrary size as input, and outputs proportionally sized feature maps at multiple levels, in a fully convolutional fashion. The architecture of FPN combines low-resolution, semantically strong features with high resolution, semantically weak features via top-down pathway and lateral connections, as shown in the Figure 28.

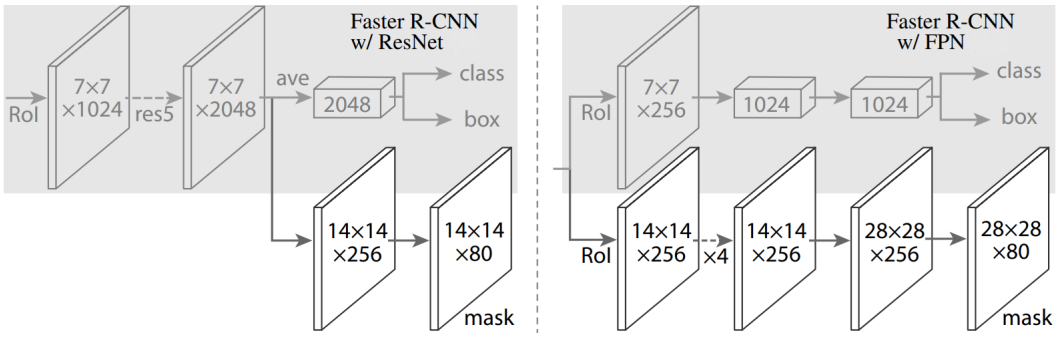


Figure 28: *Illustration of Mask R-CNN head architectures. [30].*

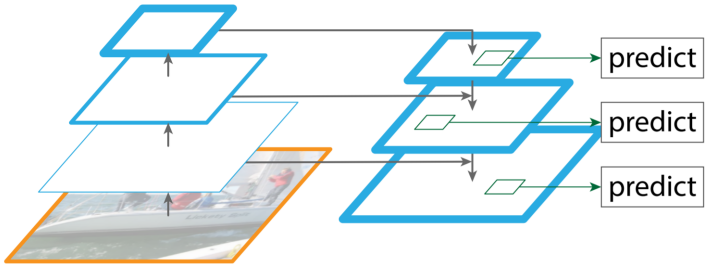


Figure 29: *Illustration of Feature Pyramid Network [26].*

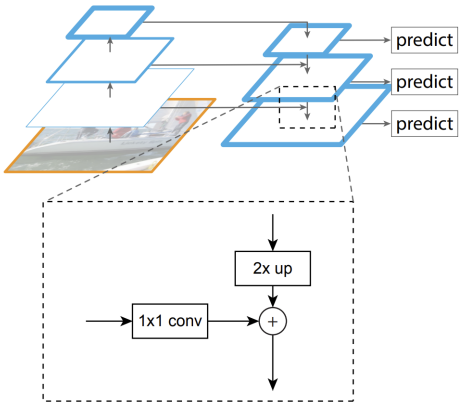


Figure 30: *Illustration of lateral connections in Feature Pyramid Network [26].*

The goal of this (FPN) is to build a pyramidal structure that has strong semantics at all scales, which is achieved by combining two pathways using lateral connections. The bottom-up pathway is the feed-forward computation of the backbone CNN, which computes a feature hierarchy consisting of feature maps at several scales with a scaling step of 2. Figure 31 shows the bottom-up pathway, where each stage output is downscaled by 2. Here, ResNet's stages are used for the pyramid level; hence, the output of each stage is defined as the output of one pyramid level. The FPN[26] paper denotes the ResNet's residual blocks as C2, C3, C4, C5 for conv2, conv3, conv4, and conv5 outputs. The top-down pathway constructs higher resolution layers by upsampling spatially coarser, but semantically stronger, feature maps from higher pyramid levels. The up-sampled feature maps are then merged with the corresponding bottom-up feature maps using a lateral connection (Figure 30). The bottom-up feature passes through a 1x1 convolutional layer before getting merged with up-sampled features, and the lateral connection is element-wise addition. Figure 31 shows the top-down pathway, where pink colour addition is lateral connection adds upsampled features to the corresponding bottom-up feature, which passes through a 3x3 convolution to produce final feature maps. Hence, the top-down process starts with a 1x1 convolutional layer attached on C5 to produce the coarsest resolution map, the output dimension set as 256 for all top-down levels. Finally, a 3x3 convolution is attached to each merged map to generate the final feature maps P2, P3, P4, P5, corresponding to C2, C3, C4, C5 that are respectively of the same spatial sizes.

Each of the generated feature maps is passed through a 3x3 convolution layer for region proposal (RPN). For RPN network, anchors of a single scale to each level are defined, so that, in total, 15 (3 for each P2, P3, P4, P5, P6) anchors are generated over the pyramid[26]. P6 only introduced for RPN for covering larger anchor scale; P6 is not used by further Fast R-CNN heads (Predictor and Bbox heads)[26]. RoI poolers extract 7x7 features for the box head and attach two hidden 1,024-d FC layers (each followed by ReLU) before the final classification and bounding box regression layers. For mask branch from [30] adopt RoI pooling to extract 14x14 features as shown in the Figure 28 and Figure 31.

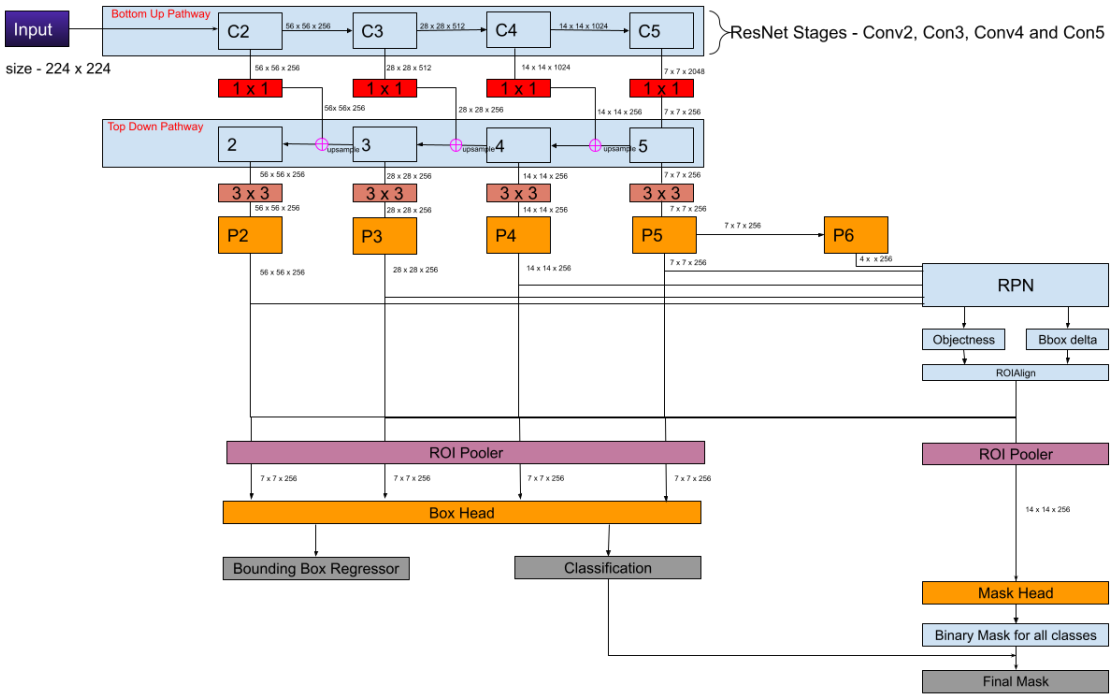


Figure 31: *Illustration of Mask R-CNN head architectures using FPN.*

Another variant of Mask R-CNN is C4 backbone, where ResNet[16] and ResNeXt[28] networks of depth 50 or 101 layers are used (Figure 28 left)[30]. For C4, the mask branch adopted the same idea of the original implementation of Faster R-CNN with ResNets extracting features from the final convolutional layer of the fourth stage. As shown in Figure 28 RoI poolers extract 14x14 features for mask branch for C4 too.

The most significant change from Faster R-CNN to Mask R-CNN is the replacement of RoI pooling with RoIAlign pooling. In RoIPool, quantisation is performed when dividing RoIs into bins (e.g., 7×7), which causes misalignments between the RoI and the extracted features. The RoIPool works well for object detection, but it fails horribly for instance segmentation due to too many quantization steps, which affect the generation of masks, where pixel-to-pixel correspondence is important[30]. Whereas, it's substitution, RoIAlign layer removes the harsh quantization steps and properly align the extracted features with the input. In RoIAlign, the authors used bilinear interpolation to compute the exact values of the input features at four regularly sampled locations in each RoI bin, before aggregating the result (using max or average)[30].

While the training of Mask R-CNN, a multi-task loss on each sampled RoI as $L = L_{cls} + L_{box} + L_{mask}$ is defined. The classification and bounding box training is adapted from Faster R-CNN. The mask branch has a Km^2 - dimensional output for each RoI, which encodes K binary masks of resolution m×m, one for each of the K classes as shown in Fig. 29. Therefore, L_{mask} is defined as the average binary cross-entropy loss for a per-pixel sigmoid to distinguish background and foreground (object), which is only limited for positives RoI (above a certain NMS threshold), i.e. "for an RoI associated with ground-truth class k, L_{mask} is only defined on the k-th mask (other mask outputs do not contribute to the loss)"[30]. During inference, the mask branch is only applied to highest scoring 100 detection boxes to speed up the inference and improves accuracy (due to the use of fewer, more accurate RoIs). At last, m×m floating-number mask output is then resized to the RoI size, and binarised at a threshold of 0.5 (due to sigmoid activation function for binary

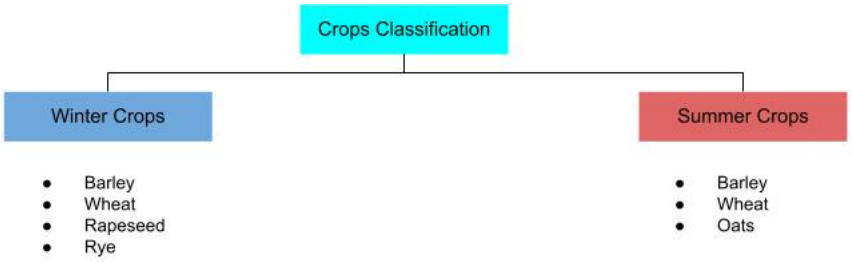
classification)[30] to generate masks.

The next chapter discusses details of the dataset properties and preparation pipeline used in this work.

5 Data Review

This chapter introduces the datasets used in this thesis and briefly describes the possible methods of obtaining ground truth data from publicly available sources.

5.1 Dataset categorisation



Jan-Mar	Apr-Jun	Jul-Sep	Oct-Dec
Barley - Mid	Barley - Mid	Barley - Mid/Harvest	Barley - Planting
Wheat - Mid	Wheat - Mid	Wheat - Mid/Harvest	Wheat - Planting
Rapeseed - Mid	Rapeseed - Mid	Rapeseed - Harvest/Planting	Rapeseed - Planting
Rye - Mid	Rye - Mid	Rye - Harvest	Rye - Planting
	Barley- Planting/Mid	Barley- Mid/Harvest	Barley- Harvest(Nov)
	Wheat- Planting/Mid	Wheat- Mid/Harvest	Wheat- Harvest(Nov)
	Oats - Planting/Mid	Oats - Mid/Harvest	

Figure 32: *Denmark crop classification based on respective crop cycle. Blue colored fonts depicts the winter crops along with stage of respective crops. Red colored fonts depicts summer crops along with stage of respective crops. [39]*

In order to conduct the research and experiments, the following five categories of the dataset were prepared.

1. **Single class classification**: All agricultural fields are set to the same class. Here the task is to predict for each polygon, whether the patch is vegetation land or not. Henceforth, this dataset is referred to as **'wh_si'**.
2. Based on the Denmark crop cycle provided by [39], four other datasets are created to study the crop classification in their respective crop cycle.
   (a) **Entire year multi class classification**: This dataset was collected throughout the year and images were divided into four exclusive classes, 'winter', 'summer', 'grassland' and 'others'. Here, the task was to predict winter and summer crops. In this entire document, this dataset is referred to as **'wh_win_sum'**.
   (b) **January-March multi class classification**: Figure 32 describes the types of crops available during this period of the year along with the status of the crop (planting, mid and harvest). There are four types of winter crops during January-March. In the entire thesis, this dataset is referred to as **'jm_m'**.
   (c) **April-June multi class classification**: Similarly, there are seven crops in April-June, four winter and three summer crops. In the entire thesis, this dataset referred to as **'aj_m'**.
   (d) **July-September multi class classification**: The July-September interval also has seven crops, four winter and three summer crops, but at different stages. In the entire thesis, this dataset is referred to as **'aj_m'**.

Another category, October-December, was not part of the experiment due to the high percentage of clouds in the satellite images.

5.2 Satellite Data

In subsubsection 2.1.2, we discussed satellite imagery and how images are formed using multispectral satellite bands. For this work, the Sentinel-2 multispectral Level-2A dataset was utilised. Sentinel-2 contains 13 spectral bands, four bands at a spatial resolution of 10m, six bands at 20m and three bands at 60m. Instead of considering all the channels in the multiband imagery from the Sentinel-2, we only focus on the RGB channels, i.e. B2 (Blue), B3 (Green) and B4 (Red) at 10m spatial resolution. The spatial resolution of a satellite image refers to the size of features detected by a satellite image, a spatial resolution of 10m means that one pixel represents an area of 10mx10m on the ground.

(a) *Sentinel-2 raster image taken at 2 percent cloud*

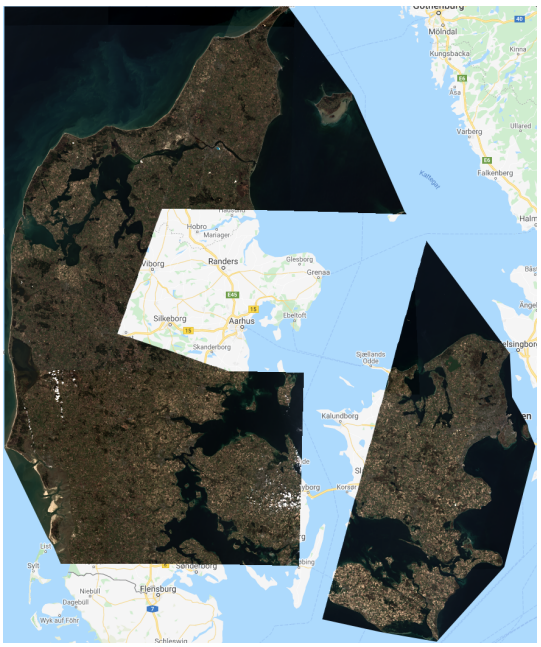
(b) *Sentinel-2 raster image taken at 4 percent cloud*

Figure 33: *Illustration of comparison in RGB intensities for 2 percent cloud coverage and 4 percent cloud coverage of the same area.*

Satellite raster data is also associated with a coordinate reference system (CRS), along with resolution and spatial context. The Coordinate Reference System or CRS of a spatial object tells where the raster is located in geographical space. For this study, 'EPSG:32632' is used as Denmark rest in UTM 32. Another crucial parameter is the cloud percentage, which determines how much cloud should be present when extracting raster images. In the Figure 33, two cloud percentage levels are shown for comparison, and it can be seen that the cloud percentage does not only change the volume of the clouds but also the RGB colour intensities in the image. According to [33], setting the cloud percentage to 2 is ideal for RGB intensities. As mentioned in the previous subsection 5.1, datasets are prepared in five categories to conduct the research. Therefore, four raster images are extracted via the Google Earth engine for Sentinel-2 (Figure 34)[6].

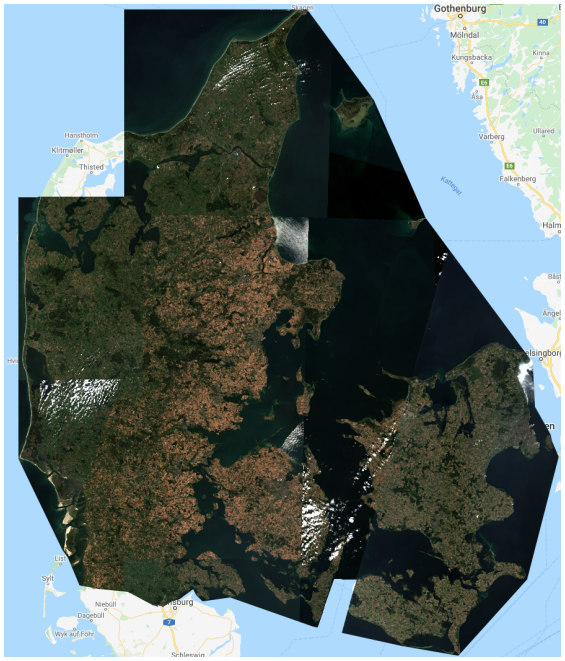
[6]Google Earth Engine code can be found here - `https://code.earthengine.google.com/92cce6f90c632703ea671ec2363738cf`

(a) *Sentinel-2 data for whole year 2020*



(b) *Sentinel-2 data for Jan-Mar 2020*

(c) *Sentinel-2 data for Apr-Jun 2020*



(d) *Sentinel-2 data for July-Sep 2020*

Figure 34: *Illustration of four Sentinel-2 raster image extracted at different interval of the year 2020 over the Denmark. The vacant patches where the black mask is not present are filtered out due to higher cloud percentage than 2 percent at that particular region.*

5.3 Shape Data

The food ministry of Denmark[42] provides vector annotations for raster images in the form of 'shapefiles'. The 'shapefile' format is a geospatial vector format that spatially describes the following vector features: points, lines and polygons. There features represent rivers, roads, buildings and land. These vector attributes are represented in the geospatial coordinate system. Figure 35 is extracted using the Python library geopandas to illustrate Denmark vegetation shape data, which consist of 583674 rows and 10 columns. The "geometry" column carries the polygon in the form of geospatial coordinates, and the corresponding "Afgkode" column represents the crop code for the polygon. Crop codes uniquely define crops in the dataset. The next section describes the mapping of vector files to raster images to generate annotations for crops in COCO format.

	Marknr	IMK_areal	Journalnr	CVR	Afgkode	Afgroede	GB	Markblok	GBanmeldt	geometry
0	VSF 5	15.28	20-0004414	28422733	318.0	MVJ ej udtagning, ej landbrugsareal	0.0	447232-86	0.00	POLYGON ((447095.808 6232154.360, 447097.965 6...
1	VSF 6	18.87	20-0004414	28422733	318.0	MVJ ej udtagning, ej landbrugsareal	0.0	447232-86	0.00	POLYGON ((447414.977 6232743.568, 447416.902 6...
2	VSF 7	21.02	20-0004414	28422733	254.0	Miljøgræs MVJ-tilsagn (0 N), permanent	1.0	447232-86	21.02	POLYGON ((447968.632 6232132.837, 447968.701 6...
3	filsø6	2.66	20-0004414	28422733	250.0	Permanent græs, meget lavt udbytte	1.0	463234-72	2.66	POLYGON ((463635.528 6233882.395, 463592.872 6...
4	fjan10	3.92	20-0004414	28422733	254.0	Miljøgræs MVJ-tilsagn (0 N), permanent	1.0	445243-99	3.92	POLYGON ((445791.067 6243928.038, 445791.721 6...
...	...	...	...	...	...	...	...	...	...	...
583669	65-0	9.18	20-0030324	31106605	NaN	None	NaN	511161-68	NaN	POLYGON ((511539.465 6161423.150, 511531.294 6...
583670	64-0	16.08	20-0030324	31106605	NaN	None	NaN	511161-97	NaN	POLYGON ((511622.497 6161362.253, 511618.345 6...
583671	1-0	5.46	20-0018854	21405051	NaN	None	NaN	479172-61	NaN	POLYGON ((479566.600 6172007.170, 479566.690 6...
583672	80-0	8.09	20-0030324	31106605	NaN	None	NaN	512160-07	NaN	POLYGON ((511350.042 6160285.812, 511350.561 6...
583673	3-1	1.52	20-0018854	21405051	NaN	None	NaN	479172-61	NaN	POLYGON ((479378.050 6171989.210, 479378.740 6...

583674 rows × 10 columns

Figure 35: *Denmark shape file described by Python library geopandas. It contains 583674 rows and 10 columns.*

5.4 Dataset preparation

A pipeline has been deployed to create a large enough dataset suitable for vegetation land classification, detection, and segmentation goals. Five categories (as mentioned in subsection 5.1) of refined raster images of the entire Denmark region were downloaded from the Google Earth Engine. Subsequently, vector shapefiles for vegetation and agricultural land were gathered from the food ministry of Denmark[42]. The research work in this thesis is only restricted to seven crops in total (four winter and three summer crops) (see subsection 5.1).

The workflow of dataset preparation was as follows (see Figure 36):

1. A vector shapefile was created for every dataset category and reclassified into a single class 'agriland'. With a single class classification, all the crops were grouped into one class and saved separately (Table 1). Likewise, a shapefile for other dataset categories (Table 2, Table 3, Table 4) was also created. Figure 37 shows the distribution of classes across all the dataset categories.

2. Preprocessing of raster image and polygon data was carried out using python libraries, rasterio, geopandas and shapely.

3. For the creation of annotations, polygons were projected to the coordinate reference system of the image (epsg:32632) and intersected with the image's boundaries (raster bounds). The rest of the polygons were then simplified within a small distance tolerance of each point of the original geometry. Following this, the polygons were transformed into pixel coordinates based on the size of the entire raster image. Polygons now have pixel coordinates rather than CRS coordinates.

4. Next, raster and vector polygons were cut into chips of three sizes, i.e. 128x128, 224x224 and 400x400 pixels, and raster image were saved as JPEG format.

5. All image chips were then randomly split into 80 % training data and 20 % validation data. Both datasets contain "empty" image chips that do not contain any ground truth polygons.

6. To normalize the image chips to zero mean during training, the mean value of each image channel was calculated based on the training data.

7. The image chips were then saved in RGB JPEG format without any compression. The polygon data was saved in the COCO JSON annotation format.

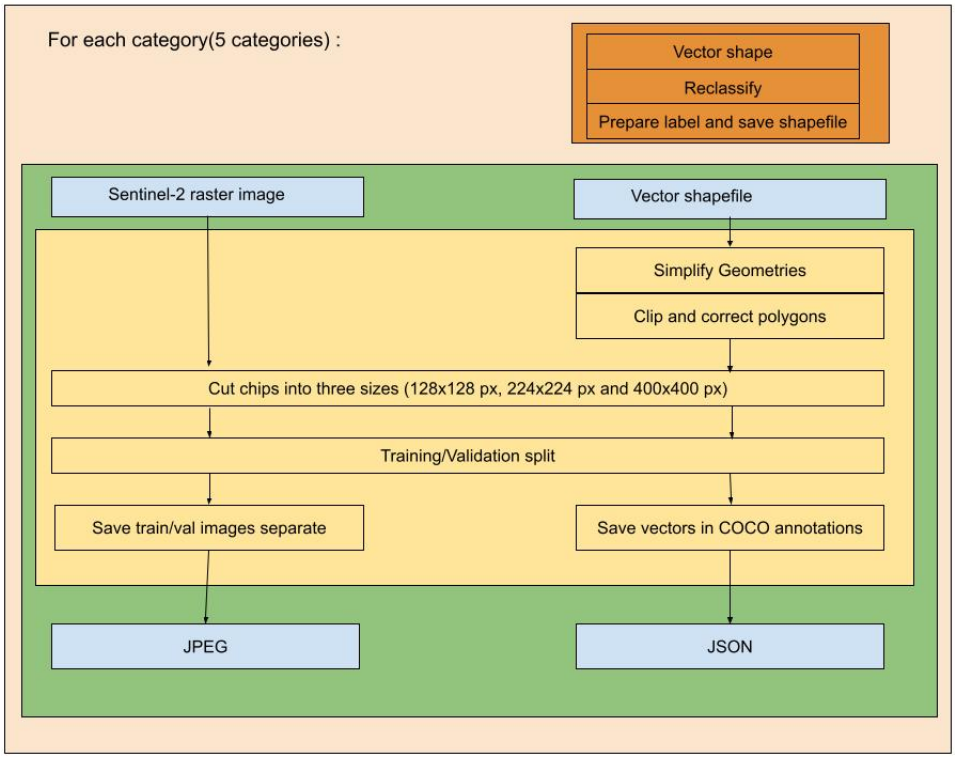


Figure 36: *Vegetation/Agriculture land classification and segmentation dataset preparation pipeline*

Index	Label
01	agriland

Table 1: *Category **'wh_si'**. Whole year single class classification where all crops are in single class for year.*

Index	Label
01	winter
02	summer
03	other
04	grassland

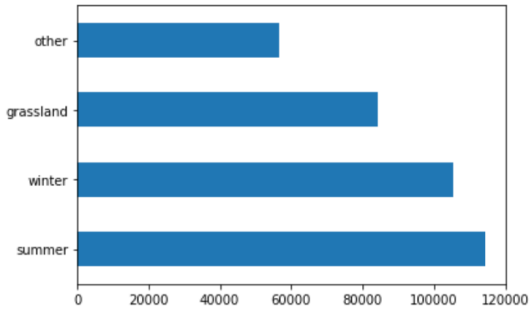
Table 2: *Category **'wh_win_sum'**. Whole year multi class classification where crops are divided into four classes for year.*

Index	Label
01	barley_w
02	wheat_w
03	rye_w
04	rapeseed_w

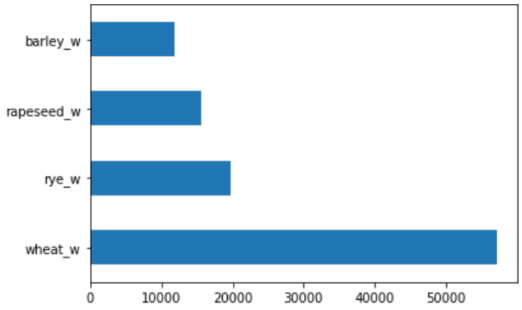
Table 3: *Category **'jm_m'**. January - March crops classification for four crops.*

Index	Label
01	barley_w
02	wheat_w
03	rye_w
04	rapeseed_w
05	barley_s
06	wheat_s
07	oats_s

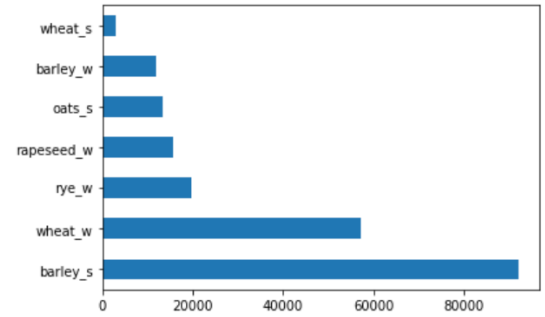
Table 4: *Category **'aj_ m'** and **'js_ m'**. April - June and July-September crops classification for seven crops.*



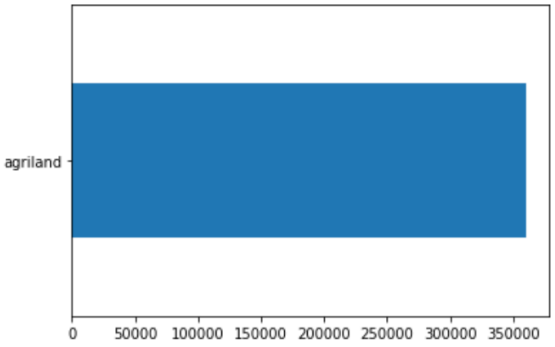
(a) *Distribution of class labels in multi-class classification for year 2020 (see Table 2.)*



(b) *Distribution of class labels in multi-class classification for January - March cycle in year 2020 (see Table 3.)*



(c) *Distribution of class labels in multi-class classification for April - June and July - September cycles in year 2020 (see Table 4.)*



(d) *Distribution of class label agriland in single-class classification for year 2020 (see Table 1.)*

Figure 37: *Illustration of five dataset categories and distribution of class labels for each of them in year 2020.*

In Denmark shapefile, there were multiple cases for single crops with different tags. For example, a crop 'spring barley' had three unique classes 'Spring barley', 'Green grain of spring barley' and 'Spring barley, wholecrop', and these classes had codes, 1, 701 and 210, respectively. A full list of crop classes is given in the appendix 2 at the end of this document. Due to redundancy in crop classes, an assumption was made before the reclassification of crops for dataset categories, 'jm_m', 'aj_m', and 'js_m'. We assumed that all other crop variants are different from each other and don't reflect the same field. Hence, for each crop, only one class was considered. In the above example, class code 1 represented the 'Spring barley' at each stage in its crop cycle (planting, mid and harvest). Figure 54 in appendix 1 outlines the crop class codes used for the creation of dataset categories 'wh_win_sum', 'aj_m', and 'js_m'.

For each dataset category, a raster image is extracted, and a shapefile is created after reclassification (step 1). The above workflow was then applied to all the dataset categories ('wh_si', 'wh_win_sum', 'aj_m', and 'js_m') for the following image sizes, 128x128 px, 224x224 px and 400x400 px. The original raster image's resolution was kept unchanged i.e., three image sizes are created by cropping a large raster image into the required image size of the dataset. In subsec-

tion 8.1, we discussed the reason for choosing the smaller image size for conducting experiments. Table 5, Table 6 and Table 7 show the comparison of dataset sizes for five categories in the three image size variants. The numbers in the tables are solely positive samples, which means only images that are annotated are included.

Image Size	Training set	Validation set
128x128 px	23228	3752
224x224 px	8234	1234
400x400 px	2812	460

Table 5: *Number of positive images (images that have annotations) in training and validation sets for category 'wh_si' and category 'wh_win_sum' (Table 1 and Table 2) dataset in three image sizes 128, 224 and 400 along with total number of instances. The table stats are for year 2020.*

Image Size	Training set	Validation set
128x128 px	20179	2462
224x224 px	7596	1092
400x400 px	2670	442

Table 6: *Number of positive images (images that have annotations) in training and validation sets for category 'jm_m' (Table 3) dataset in three image sizes 128, 224 and 400 along with total number of instances. The table stats are for year 2020.*

Image Size	Training set	Validation set
128x128 px	22235	3668
224x224 px	7968	1142
400x400 px	2735	522

Table 7: *Number of positive images (images that have annotations) in training and validation sets for category 'aj_m' and 'js_m" (Table 4) dataset in three image sizes 128, 224 and 400 along with total number of instances. The table stats are for year 2020.*

As mentioned in the subsection 4.1, the COCO dataset is evaluated on three different area sizes, such as short (S), medium (M) and large (L) polygons. The selection of polygons in S, M and L are defined by the COCO api which are: S ($< 32^2$ pixels), M ($32^2 - 96^2$ pixels) and L ($> 96^2$S pixels). According to the above distribution, approximately 85% of training field polygons belong to the S, 13% to the M, and the rest 2% to the L category in 200x200 px and 400x400 px datasets. However, for image size 128x128 px, the L category doesn't exist. Therefore, all the field areas are distributed between small and medium with 87% and 13%, respectively.

To answer the third research question of whether individual crop images from different years have similar results, the datasets are also prepared for the year 2019 in the respective five categories using the dataset preparation workflow as described above in Figure 36. The 2019 datasets were only prepared in one image size of 128x128 px.

Part II
Development and Approach

6 Methodology

As discussed in subsubsection 4.3.2, instance segmentation of an object specifies which pixels of an image belong to the object. The localisation of an object is typically represented by a bounding box and segmentation by a mask (subsubsection 4.3.2).

This thesis deploys FCIS and Mask-RCNN for object detection and instance segmentation, as mentioned in subsubsection 4.3.2. For the implementation of FCIS in this work, the official code released by its authors has been used[7]. However, in case of Mask R-CNN, Detectron2's implementation has been used[8].

The two training paradigms followed in the work are: training from scratch and transfer learning.

6.1 Training from scratch

Detectron2 provides a configuration that can be used to train a model from scratch to test out the different combinations in model selection process. There are two choices in which the Mask R-CNN framework could be trained from scratch, i.e., i) utilising pre-trained ResNet (on Imagenet) and ii) utilising no pre-trained network. In the first option, pre-trained ResNet on Imagenet dataset is employed as conv layers in Mask R-CNN's first stage, and other parts such as RPN and classifiers are randomly initialised using He[17] or Glorot initialisers[7]. In the second option, the whole Mask R-CNN architecture is trained from scratch. The learning rate to 0.00005 or lower to prevent network divergence after initial epochs.

In case of FCIS, the official implementation does not provide the training from scratch settings due to divergence in deep networks.

6.2 Transfer Learning

Machine learning and computer vision have been advanced largely by using standardized benchmark datasets that have high variance but generally lend themselves to powerful features. The transfer learning approach is popular in computer vision because it leads to more accurate, time-saving models. The use of transfer learning allows one to use an existing model that has learned fairly generalizable weights trained on a standardized benchmark datasets such as ImageNet, and fine-tune the network to suit the particular use-case.

When compared to learning from scratch, transfer learning leads to fast convergence. In addition, it's appropriate for small datasets such as ours. Mask R-CNN and FCIS were trained on the COCO dataset and used as pre-trained models for thesis' experiments.

Table 8 shows the comparison between training from scratch and transfer learning approaches on 'wh_win_sum' dataset. The transfer learning approach performed better and took less training time. The size of the dataset could be one reason for network to perform badly in training-from-scratch settings compared to transfer learning settings. Table 5, Table 6 and Table 7 show the dataset sizes for each dataset category for each image size, which are fairly very low compared to the original COCO dataset that has 120k images in the training set and 46k images in the validation set. Transfer learning is therefore an appropriate method.

[7] FCIS - `https://github.com/msracver/FCIS`
[8] Detectron2 - `https://github.com/facebookresearch/detectron2`

No	Method	Backbone model	mAP	No of epochs	Lr
1.	Transfer learning	ResNet101 FPN	1.08	2k	0.001
2.		ResNet50 FPN	0.92	2k	0.001
3.		ResNet101 C4	0.83	2k	0.001
4.		ResNet50 C4	0.80	2k	0.001
5.	Scratch Training	ResNet101 FPN	0.04	10k	0.00005
6.		ResNet50 FPN	0.03	10k	0.00005
7.		ResNet101 C4	0.01	10k	0.00005
8.		ResNet50 C4	0.004	10k	0.00005

Table 8: *Comparison between transfer learning and scratch training on image size 128x128 px. The mAP on validation set of 'wh_win_sum' (Table 2. ResNet-50/101 are used with backbone FPN and C4 for Mask R-CNN.*

7 Implementation

In this chapter, the implementation of instance segmentation pipelines explained in subsubsection 4.3.2 is described in detail.

7.1 Pytorch and Detectron2

PyTorch is an open-source machine learning python package based on the Torch library. It provides two high-level features: Tensor computation with GPU acceleration and Deep neural networks built on a tape-based autograd system. Detectron2 is a PyTorch-based modular object detection library that has Mask R-CNN implementation from Facebook AI research. The Detectron2 development library offers high customisation of training parameters, integration with custom datasets, several detection methods, and backbone combinations and is currently used by the state-of-the-art object detection algorithm, hence the library is used to conduct Mask R-CNN experiments in this work.

7.1.1 Detectron2 Structure

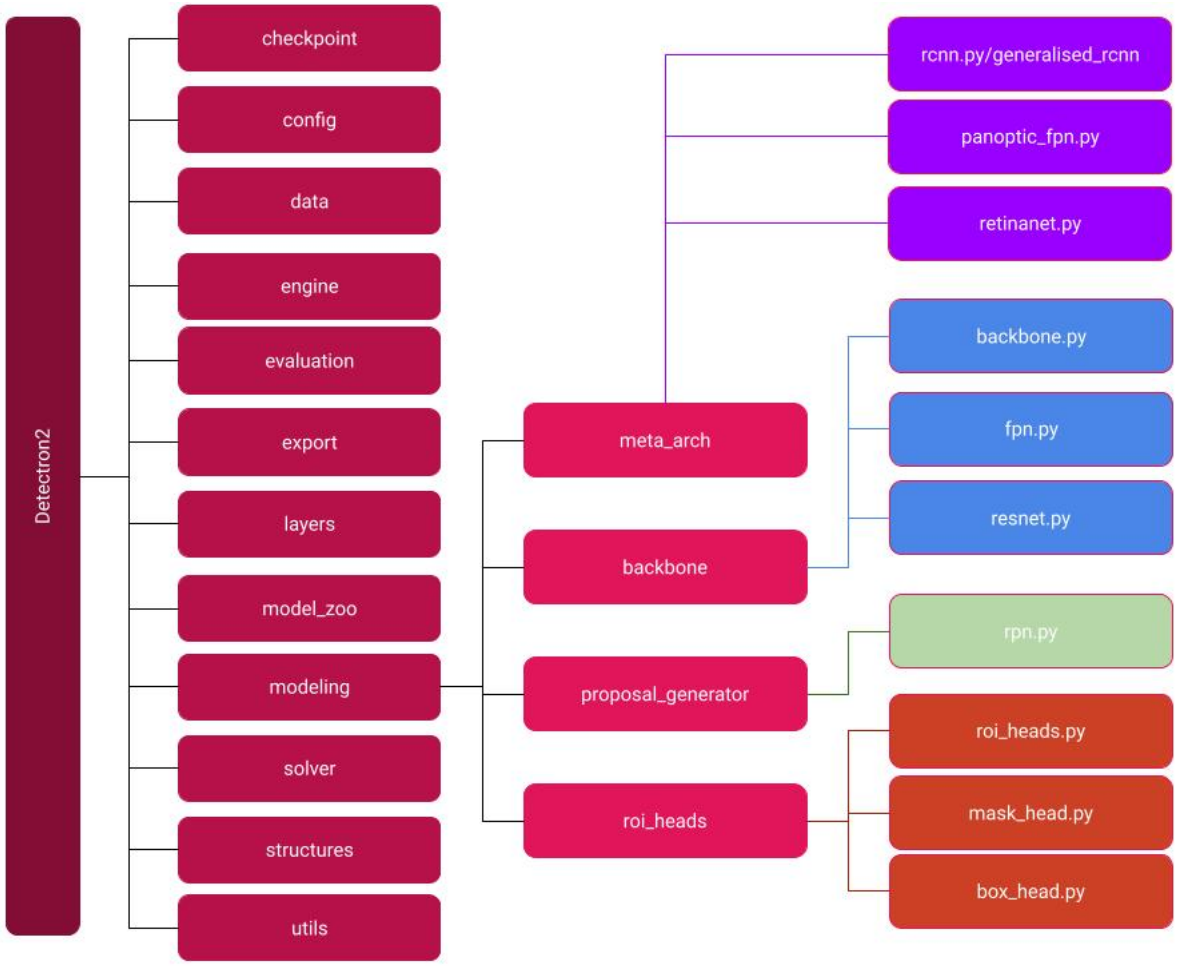


Figure 38: *Illustration of Detectron2 repository. The construction of Mask R-CNN is in modelling package of Detectron2. The starting point is* `generalised_rcnn` *which has modular implementation of FPN and combined each stages of Mask R-CNN in meta architecture. The backbone contains ResNet, RPN is implemented in rpn script and* `roi_heads` *contains heads for classifier, box and mask generation*

Figure 38 shows the overall structure of the Detectron2's official repository. The 'modeling' package is responsible for the construction of Mask R-CNN. All the Python classes and subclasses are registered using the registry package of Detectron2, which makes it very modular and extendable. The 'modeling' package has four major packages - `meta_arch`, `backbone`, `proposal_generator` and `roi_heads`. The `'meta_arch'` is the starting point of Mask R-CNN

architecture, which register the 'GeneralisedRCNN' class responsible for the initiation of Mask R-CNN. The `'backbone'` contains a backbone network, the RPN network is implemented in `'proposal_generator'`, and lastly `'roi_heads'` contains classifier, bounding box head and mask head. For the experiments in the thesis, on top of Detectron2, Python custom methods and classes are extended. A detailed description of the experiments can be found in chapter 8. During model training, in default settings, Detectron2 only considers images with existing ground truth instances, that is, only positive samples. The data augmentation and normalisation were performed according to configured settings; different settings of Detectron2 are discussed in chapter 8. Detectron2 provides a configuration for trying out different sets of hyperparameters. However, most of the chosen hyperparameters follow the default settings of Mask R-CNN implementation. Training checkpoints and configuration files are saved simultaneously. Training and evaluation were performed on DFKI's cluster. For visualisation purpose, Detectron2 supports Tensorboard. TensorBoard is a tool that provides the visualisation and tooling needed for deep learning experiments.

7.1.2 FCIS and MXNet

The official code of FCIS was implemented on MXNet. MXNet is also a deep learning framework that provides efficient training. An open-source Docker image of MXNet was configured and used on DFKI's clusters to conduct the experiment on FCIS[9]. Similar to Mask R-CNN, FCIS also considers positive image samples only during the training. The images are mean-centred by subtracting per-channel mean values, which were extracted during the preprocessing. Additionally, horizontal and vertical flip data augmentation is performed before the training process. The complete FCIS training process is discussed in paragraph 4.3.2.1.

[9]The docker image can be found in docker hub - `https://hub.docker.com/r/smishra03/mxnet`

8 Experiments

8.1 Experiment 1: Image Size

The first experiment was done to evaluate which image size (in pixels) would give the best results. Both instance segmentation frameworks, FCIS and Mask R-CNN were evaluated on three image sizes - 128x128 px, 224x224 px and 400x400 px. Table 5 shows the size of the training set and validation set for a single-class category dataset.

Table 9 shows the image size and evaluation dataset performance (mAPs) comparisons for multi-class classification of 'wh_win_sum' dataset (Table 2). Using higher image sizes than 128x128 px does not lend itself well for effective training – 400x400 px and 224x224 px have a lower mAP than 128x128 px. The 400x400 px image size dataset shows significantly poor result on both the frameworks, FCIS and Mask-RCNN, compared to other image sizes among themselves. A possible reason this could be an increase in the number of objects per image as image size increases (see Table 10). Since both frameworks have a limitation of 100 object detections per image as their default setting, only 128x128 px image sizes were evaluated in further experiments.

Model	128x128 px	224x224 px	400x400 px
Mask R-CNN, ResNet101 with FPN	1.08	0.88	0.45
Mask R-CNN, ResNet50 with FPN	0.92	0.94	0.27
Mask R-CNN, ResNet101 with C4	0.83	0.59	0.12
Mask R-CNN, ResNet50 with C4	0.80	0.45	0.10
FCIS, ResNet101	32.10	20.30	8.90

Table 9: *The mAP on validation set of 'wh_win_sum' (Table 2) . ResNet-50/101 are used with backbone FPN and C4 for Mask R-CNN. ResNet101 is used for FCIS.*

Category	Dataset	128x128 px	224x224 px	400x400 px
'wh_si'	Whole year 2020 single-class	22	52	132
'wh_win_sum'	Whole year 2020 multi-class	22	52	132
'jm_m'	January-March 2020 multi-class	8	17	58
'aj_m'	April-June 2020 multi-class	14	33	92
'js_m'	July-September 2020 multi-class	14	33	92

Table 10: *Table shows average instances (objects) per image in three image sizes for 5 dataset categories.*

8.2 Experiment 2: Single-Class Classification

In this experiment, all the instances of agricultural fields from 2020 were set to the same class (see subsection 5.1) and both frameworks were trained and evaluated on classifying this single class. Figure 37d shows the total number of agricultural field instances in Denmark in year 2020. Altogether, around 26,980 positive sample images of 128x128 px sizes were used for this experiment (Table 5), in which the training set contains 23,228 images and the validation set contains 3,752 images. The experiment results were compared to those of Rieke, 2017[50].

8.3 Experiment 3: Crop Cycle and Multi-Class Classification

In this experiment, multiple categories of the dataset were prepared to examine the variation in image features learnt in a crop cycle (see Table 2, Table 3 and Table 4). The experiment attempts to answer the question of whether a crop-cycle-specific model can be applied to another year's crop cycle. For example, does an image of a winter wheat crop taken in January 2020 possess a similar feature to one taken in January 2019? Accordingly, datasets are prepared by crop cycles and divided into yearly quarters, i.e. Jan-Mar, Apr-Jun and Jul-Sep, to answer such questions. In subsection 5.4 the dataset preparation process is discussed in more detail. Figure 37 shows the distribution of crop fields in each dataset. The quarterly datasets are imbalanced and to counter this problem, Detectron2 provides an additional hyperparameter called 'RepeatFactorTrainingSampler' which samples more from fewer-instances classes based on refactoring value. For instance, `'wheat_s'` class in dataset categories 'aj_m' and 'js_m' is sampled more than `'barley_s'` class during training. The refactoring value is set to 0.001, which is inspired from the LVIS dataset[32]. The implementation of FCIS, on the other hand, lacks any mechanism to deal with the imbalanced dataset. Hence, to gain a deeper understanding of results, AP values per class is summarised in the Results section for each dataset category.

9 Results

This section presents quantitative and qualitative results of the experiments discussed above. Subsection 9.1 and 9.2 present quantitative and qualitative results respectively, including the obtained accuracy, training output graphs, and segmentation results.

9.1 Quantitative

9.1.1 FCIS

FCIS was trained for ten epochs which took roughly 20 hours to train for each dataset category – given the implementation's limitation of having a batch size of one image and only being able to use one GPU at a time. For the FCIS model, no hyperparameters were altered from those used by Rieke, 2017[50]. Table 11 and Figure 39 show the performance metrics after the completion of the training process. As can be seen, single-class yields the best training results with classification accuracy of 82% and foreground (object) of 96%. The dataset category 'js_m' (July-Sep), which contains seven crops, shows the significantly poor training results. The foreground accuracy is 52% for 'js_m' that confers high false positives. Nevertheless, the dataset category 'aj_m' (April-June), which also contains seven crops, yields 82% foreground accuracy. Comparing Figure 43 and Figure 44, we find the same outcome – AP values (for all crops) of 'aj_m' is higher than those of 'js_m'. Appendix 4 shows all FCIS training curves in great detail.

Dataset Category	FCISAcc	FCISAccFG	FCISMaskLoss
'wh_si'	82.04%	95.69%	0.29
'wh_win_sum'	60.01%	61.43%	0.28
'jm_m'	78.89%	59.58%	0.26
'aj_m'	77.83%	82.44%	0.22
'js_m'	61.15%	52.05%	0.24

Table 11: *FCIS's training metrics for each dataset categories (image size is 128x128 px) for year 2020. No of epochs was ten. FCISAcc is classification accuracy (including background). FCISAccFG is only foreground object classification (without background). FCISMaskLoss is mask loss from pixel-wise softmax over RoI inside/outside maps.*

FCIS is evaluated for each dataset category on the validation set to examine the crop and crop-cycle specific performances. In order to check whether it is possible to apply a giver year's crop-specific model to another year's crop cycle, both Mask R-CNN and FCIS frameworks trained on dataset categories from 2020 are tested on respective dataset categories from 2019. Moreover, as highlighted in subsection 8.3, since the datasets are imbalanced, class-wise analysis has been performed in the following.

Single-class classification (Dataset category 'wh_si' [Whole Year])
In Figure 40, the final AP for FCIS with ResNet-101 on the 'wh_si' dataset (see Table 1) can be read. Category 'wh_si' dataset focuses on single-class classification, where all the vegetation fields are labelled onto one class for the whole year. Upon comparison between 2019 and 2020, a decrease of 17.13% in mAP is observed from 2020 to 2019.

Multi-class classification (Dataset category 'wh_win_sum' [Whole Year])
For multi-class classification on the whole year, dataset category 'wh_win_sum' (Table 2) was prepared—the goal of this experiment was to distinguish summer and winter crops in a year. Figure 41 shows the average precision values for each class in the 'wh_win_sum' dataset. Overall mAP @[IoU = 0.5-0.95] is 19.1 which is 41% less than the single-class classification. The bar graph in Figure 41 shows that winter crops have a better AP than summer crops, having an approximately equal amount of instances in both classes (see Figure 37c). It can also be seen from Figure 43 and Figure 44 that winter crops ('barley_w', 'wheat_w', 'rye_w' and 'rapeseed_w') have 51.25% and 15% higher mAP values than summer crops ('barley_s', 'wheat_s' and 'oats_s') in category 'aj_m' (Table 4) and category 'aj_m' (Table 4) datasets, respectively. When compared to the

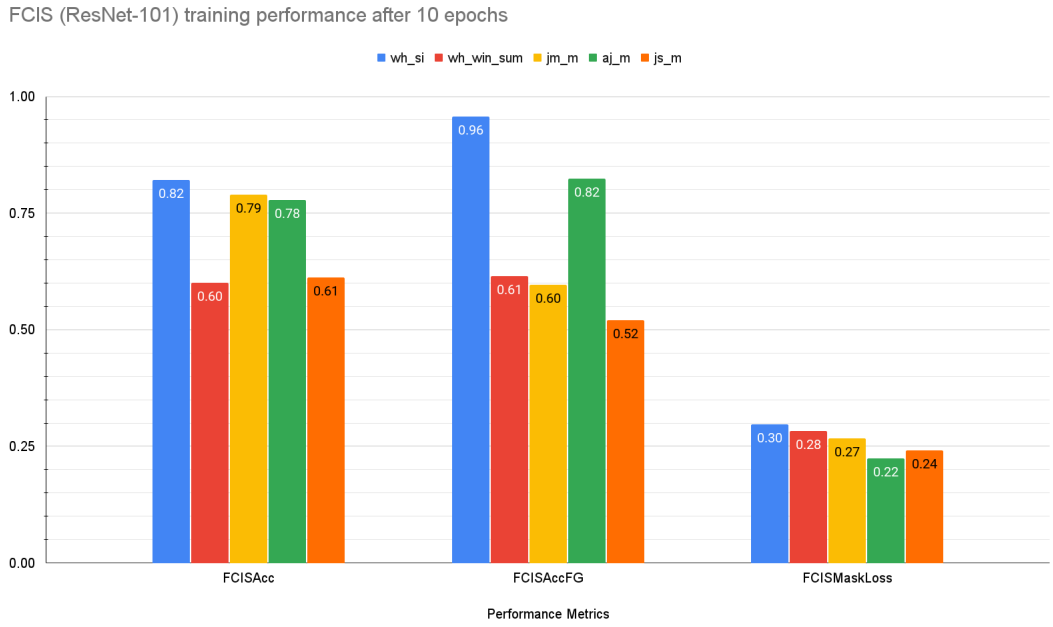


Figure 39: *FCIS training metrics on five datasets (image size is 128x128 px) for year 2020. Number of training epoch was 10. FCISAcc is classification accuracy (including background). FCISAccFG is only foreground object classification (without background). FCISMaskLoss is mask loss from pixel-wise softmax over RoI inside/outside maps.*

Figure 40: *Figure shows the AP comparison between 2019 and 2020 for dataset category 'wh_si' (single class for whole year) . FCIS is trained on training set of 2020 and evaluated on validation set of 2020 and 2019. AP@[IoU = 0.5-0.95] is calculated for each class in dataset. 'All' is mean average precision (mAP) for all the classes. In single-class, all the fields are labelled as 'agriland'.*

year 2020, the results for the year 2019 are not as satisfactory. The mAP drops by 60% when the model is evaluated on 2019 validation sets.

Multi-class classification (Dataset category 'jm_m' [January-March])

The main objective of this experiment was to investigate the performance of crop classification in a supervised fashion. From subsection 5.1, we know that over the course of its life cycle, various visual features of the crop changes. Having this information in mind, the FCIS model was trained

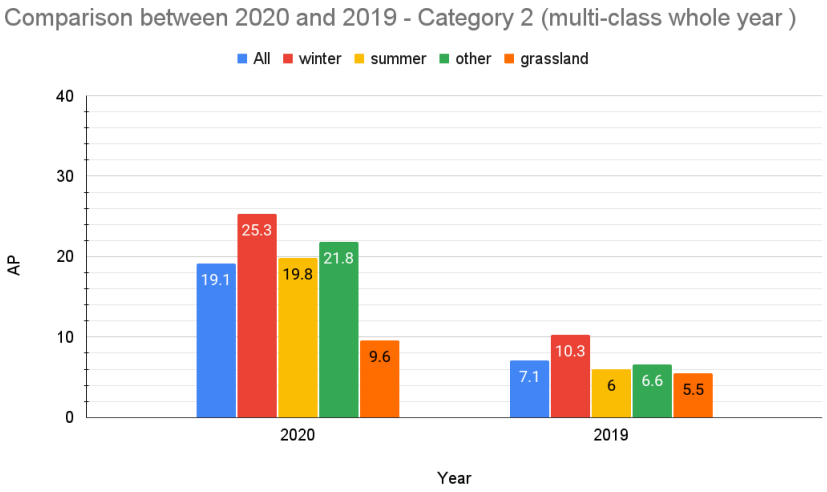


Figure 41: *Figure shows the AP comparison between 2019 and 2020 for dataset category 'wh_win_sum'. FCIS is trained on training set of 2020 and evaluated on validation set of 2020 and 2019. AP@[IoU = 0.5-0.95] is calculated for each class in dataset. 'All' is mean average precision (mAP) for all the classes.*

model for each stage of the crop cycle. In Figure 42, a comparison is made between 2020 and 2019 performances, where the average precision values for each class for the years 2020 and 2019 are shown. The rapeseed winter and barley winter crops have nearly similar results in both years. However, there are large differences in performance for wheat winter and rye winter crops. For dataset category 'jm_m', the FCIS yields 22.1 and 14.9 mAP@[IoU = 0.5-0.95] in 2020 and 2019, respectively.

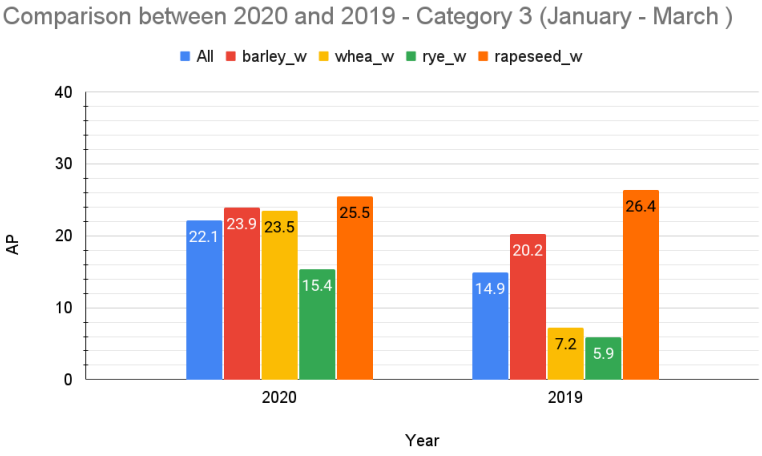


Figure 42: *Figure shows the AP comparison between 2019 and 2020 for dataset category 'jm_m' (multi class January - March). FCIS is trained on training set of 2020 and evaluated on validation set of 2020 and 2019. AP@[IoU = 0.5-0.95] is calculated for each class in dataset. 'All' is mean average precision (mAP) for all the classes.*

Multi-class classification (Dataset category 'aj_m' [April-June])

Dataset 'aj_m' comprises the crop cycle for April-June in which the experiment was performed on the seven crops involved – four winter and three summer (see Table 4). Dataset 'aj_m' yields the best results in multi-class classification with 32.1 mAP@[IoU = 0.5-0.95]. As stated earlier in Figure 9.1.1, winter crops outperform summer crops by a noteworthy margin (see Figure 43). Rapeseed winter crop has the highest 47.3 mAP, whereas wheat summer has the lowest 2.8 mAP. According to Figure 37, dataset 'aj_m' is highly imbalanced, with almost 80k of barley summer

crop fields and fewer than 5k of wheat summer crop fields. Hence, class-based mAP values are very helpful in understanding the metrics. When comparing 2019 datasets with 2020 datasets, it is evident that the results for 2019 are significantly worse.

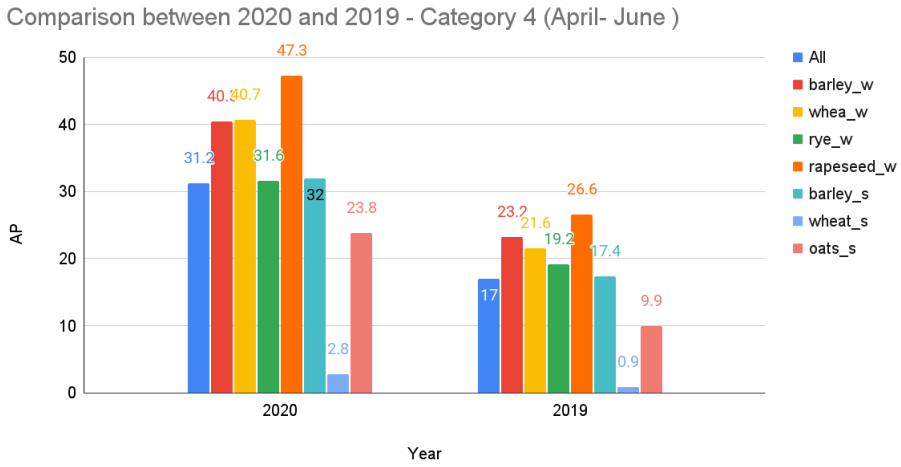


Figure 43: *Figure shows the AP comparison between 2019 and 2020 for dataset category 'aj_m' (multi class for April - June). FCIS is trained on training set of 2020 and evaluated on validation set of 2020 and 2019. AP@[IoU = 0.5-0.95] is calculated for each class in dataset. 'All' is mean average precision (mAP) for all the classes.*

Multi-class classification (Dataset category 'js_m' [July-September])

This dataset comprises the crop cycle for the July-September interval (see Table 4). Dataset 'js_m' yields the worst results among all categories with 15.6 mAP@[IoU = 0.5-0.95]. As with other datasets, dataset 'js_m' follows the same trend – performances on 2019 being lower than that of 2020. It is notable that when the performance of dataset 'js_m' is compared to dataset 'aj_m', crops in the April-June quarter perform well than any other time of year.

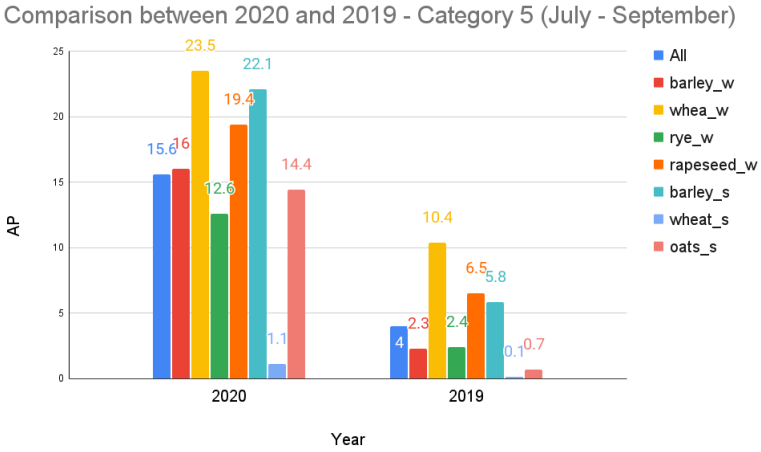


Figure 44: *Figure shows the AP comparison between 2019 and 2020 for dataset 'js_m' (multi class for July - September). FCIS is trained on training set of 2020 and evaluated on validation set of 2020 and 2019. AP@[IoU = 0.5-0.95] is calculated for each class in dataset. 'All' is mean average precision (mAP) for all the classes.*

9.1.2 Mask R-CNN

As mentioned in section 8, the Mask R-CNN instance segmentation framework was experimented on five dataset categories. Most of the hyperparameter settings for the training of Mask R-CNN were similar to those in Detectron2 while those that were changed are shown in Table 12. Instead of the default sampler while loading data, a repeat factor training sampler was used to reduce the effect of imbalance in the data (see subsection 8.3). The Model Freeze setting determines which network parameters are allowed to be learnt at different stages of the pre-trained model. By changing the setting from 0 to 2, parameters that represent higher-level features were learnt. While this did not impact model accuracy compared to using the default, it did speed up the model's training though. For normalising the images while training, pixel mean values were set to values extracted during data preparation.

Data Sampler	Repeat Factor Training Sampler factor	Pretrained model frozen at iteration number
Repeat Factor Training Sampler	0.001	2

Table 12: *Hyperparameters that were changed from the defaults for Mask R-CNN model. The full model configuration file is available in Appendix.*

Figure 45 and Figure 46 show major performance metrics after 2k epochs of training for the five dataset categories where validation was done twice during training. From both the graphs, we found that only single-clas classification for dataset 'wh_si' and multi-class classification for dataset 'wh_win_sum' (see Table 1 and Table 2) had some significant learning. For datasets 'jm_m', 'aj_m' and 'js_m', both variants of Mask R-CNN (50 and 101) showed significantly low foreground object accuracy with high classification accuracy, indicating high false positives for classification. Full training graphs are shown in Appendix 3.

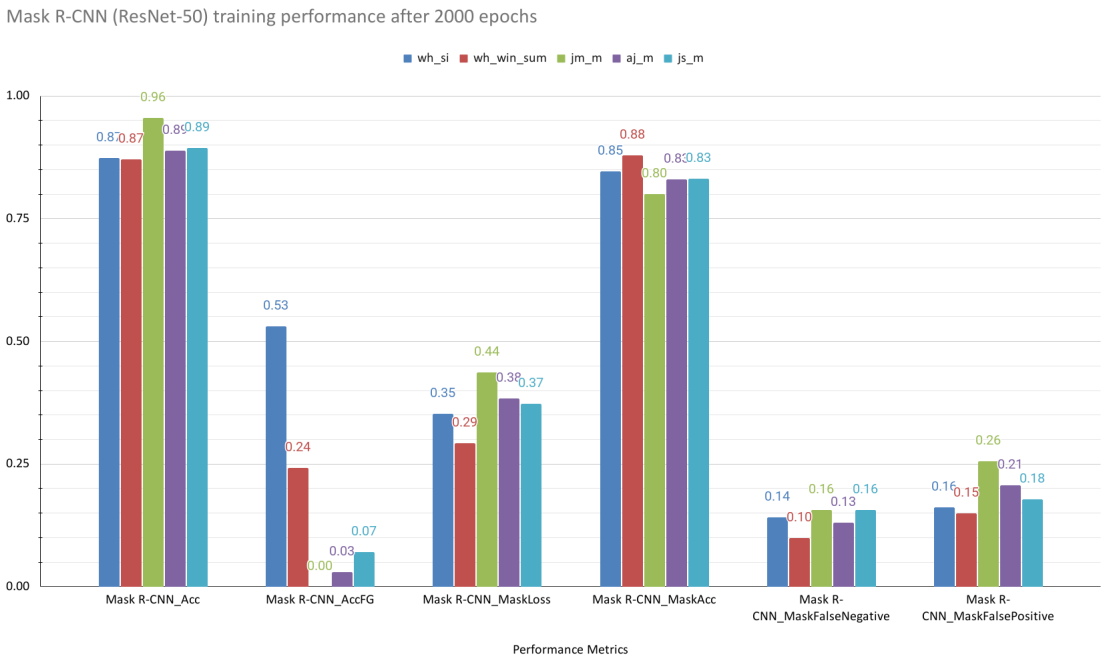


Figure 45: *Mask R-CNN's (backbone ResNet-50) training metrics on five dataset (image size is 128x128 px) for year 2020. Number of training epoch was 2000. Mask R-CNN_Acc is classification accuracy (including background). Mask R-CNN_AccFG is only foreground object classification (without background). Mask R-CNN_MaskLoss is mask head's loss. Mask R-CNN_MaskAcc is mask head's accuracy. Mask R-CNN_MaskFalseNegative & Mask R-CNN_MaskFalsePositive are mask head's false negative and false positive, respectively.*

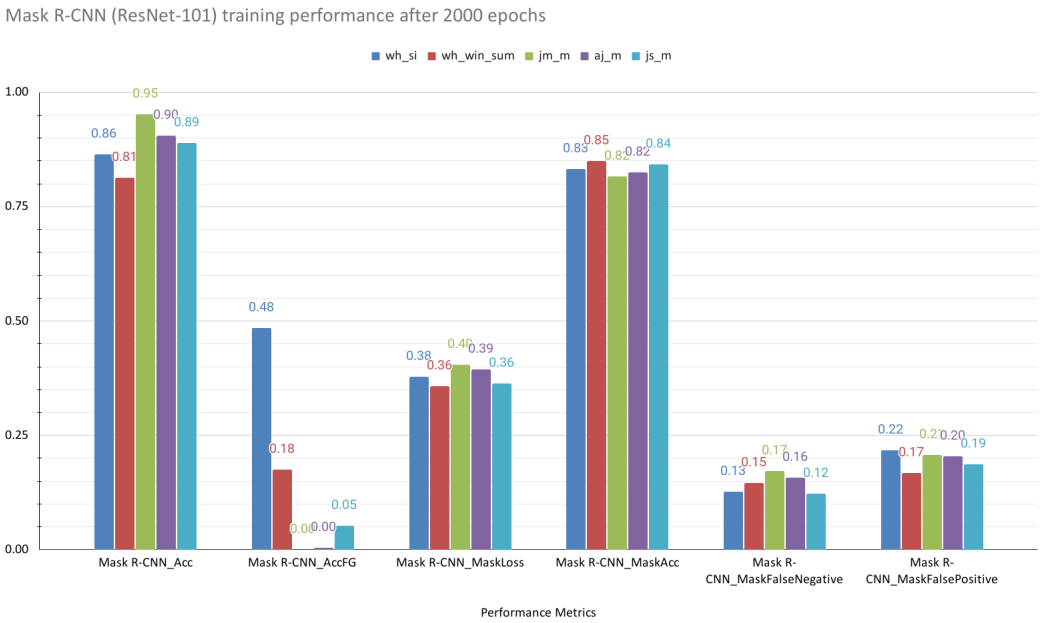


Figure 46: *Mask R-CNN's (backbone ResNet-101) training metrics on five dataset (image size is 128x128 px) for year 2020. Number of training epoch was 2000. Mask R-CNN_ Acc is classification accuracy (including background). Mask R-CNN_ AccFG is only foreground object classification (without background). Mask R-CNN_ MaskLoss is mask head's loss. Mask R-CNN_ MaskAcc is mask head's accuracy. Mask R-CNN_ MaskFalseNegative & Mask R-CNN_ MaskFalsePositive are mask head's false negative and false positive, respectively.*

Mask R-CNN is evaluated for each dataset category on the validation set to compare overall model performance across different crops and respective crop cycles. Table 13 shows the mAP for all the classes in the corresponding dataset categories. Due to a lack of performance on Mask R-CNN, presentation of class-specific results (AP on each class over a range of IoU thresholds) has been omitted.

Backbone	mAP on 'wh_si'	mAP on 'wh_win_sum'	mAP on 'jm_m'	mAP on 'aj_m'	mAP on 'js_m'
ResNet-50	2.52	0.92	0.37	0.63	0.31
ResNet-101	2.98	1.08	0.21	0.30	0.4

Table 13: *The mAP values on validation set for the five dataset categories for Mask R-CNN-50/101.*

9.2 Qualitative

Qualitative evaluation involves inspecting the images and identifying patterns based on human assessment. Figure 47, 48, 49, 50 and 51 show inference on FCIS and Mask R-CNN for datasets 'wh_si', 'wh_win_sum', 'jm_m', 'aj_m' and 'js_m' respectively. The images in these datasets were not included in training the frameworks. For both the frameworks, the inference is performed on RoI IoU at 0.6. When comparing FCIS with Mask R-CNN, there are random shapes generated for agricultural fields. For example, as shown in Figure 47 (fourth row), Mask R-CNN failed to generate field shapes correctly. On the other hand, for FCIS, the ground truth and the prediction results are in good accordance with each other in terms of the number of field instances, their boundaries and the resulting overlap. However, some complex shapes in FCIS predictions are slightly different – large shapes are divided into multiple small shapes of a field. Hence, large field polygons are detected into multiple small field polygons in FCIS. For example, as shown in Figure 47 (first row), the second field on the right is predicted into two fields for FCIS predictions. There are also some instances where FCIS prediction merged multiple small fields into one large field, and overlap is such that the separation between fields is not as strict as in the ground truth polygons. As shown in Figure 48 (second row), multiple field polygons are merged in FCIS predictions contrary to the ground truth. A possible reason for these aforementioned poor segregation results could be that FCIS fails in discriminating field instances in clusters of small, directly adjacent fields, even if they show distinct differences in texture and colour. It is possible that the under-representation of small, partially overlapping segments in the mask of the FCIS operation, coupled with the low spatial resolution of the imagery, has caused the segment masks to be merged mistakenly.

A similar problem of overlapping in FCIS was also observed by authors of Mask R-CNN, when comparing prediction results from the COCO dataset across frameworks, "suggesting that it is challenged by the fundamental difficulty of instance segmentation" [30]. As two overlapping objects with distinct size differences usually have a relatively low IoU score, none of the techniques – NMS, FCIS mask voting operation and the additional NMS after the mask merging steps – can typically remove the redundant prediction.

In 'wh_win_sum' dataset, Mask R-CNN predicted a significantly low number of field polygons, which also supports the quantitative results where mAP is lower in multi-class classification than single-class classification. For datasets 'jm_m', 'aj_m' and 'js_m', Mask R-CNN failed to predict field polygons, showing considerably poor results compared to FCIS. Other noticeable characteristics shown by FCIS are: false negatives as well as false positives in multi-class classification. Rieke[50] suggested non-existing annotations of field instances in ground truth data as a possible reason for false positives.

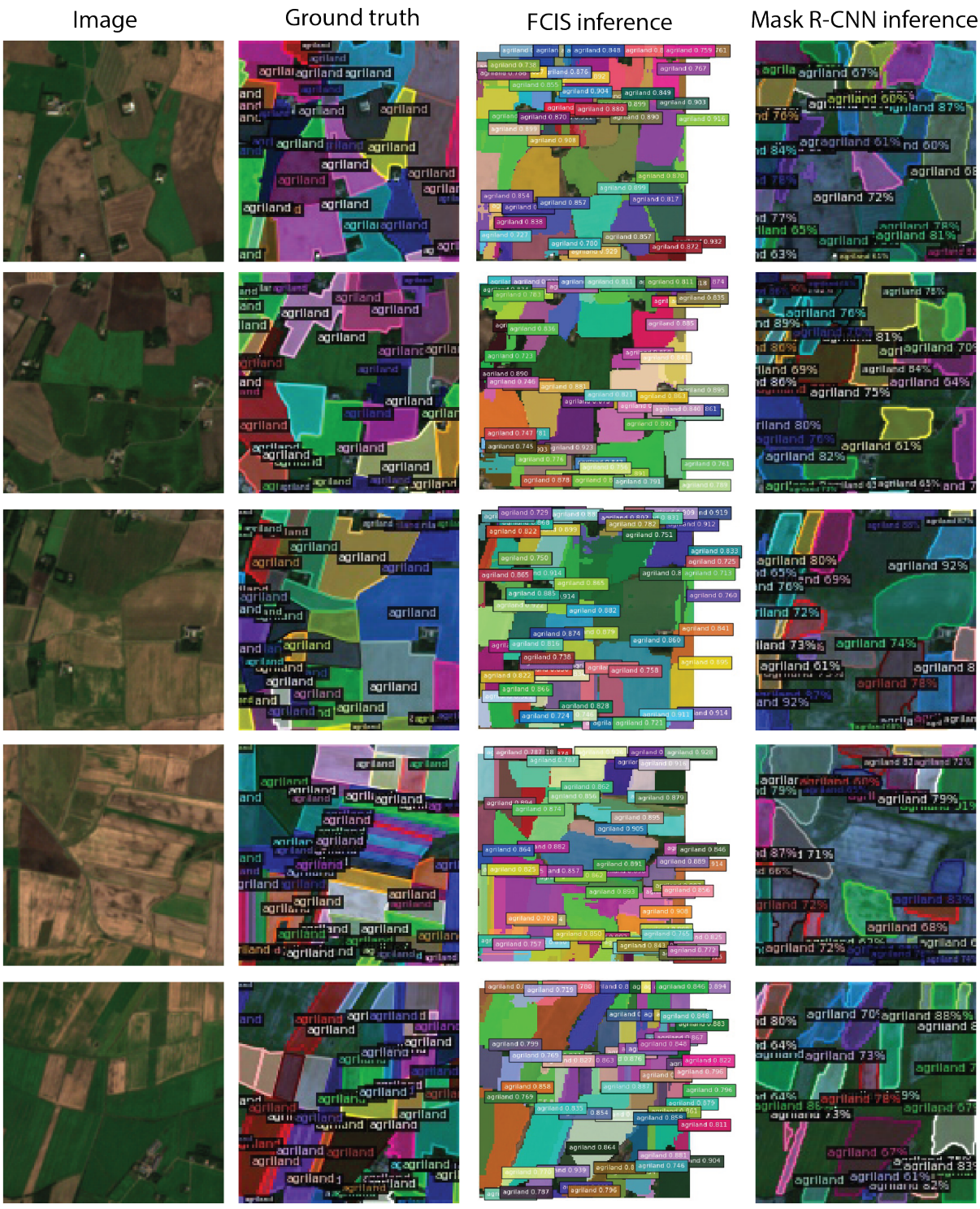


Figure 47: *Example image chips from the validation set of 'wh_si' in 2020. IoU threshold set to 0.6 for both FCIS and Mask R-CNN.*

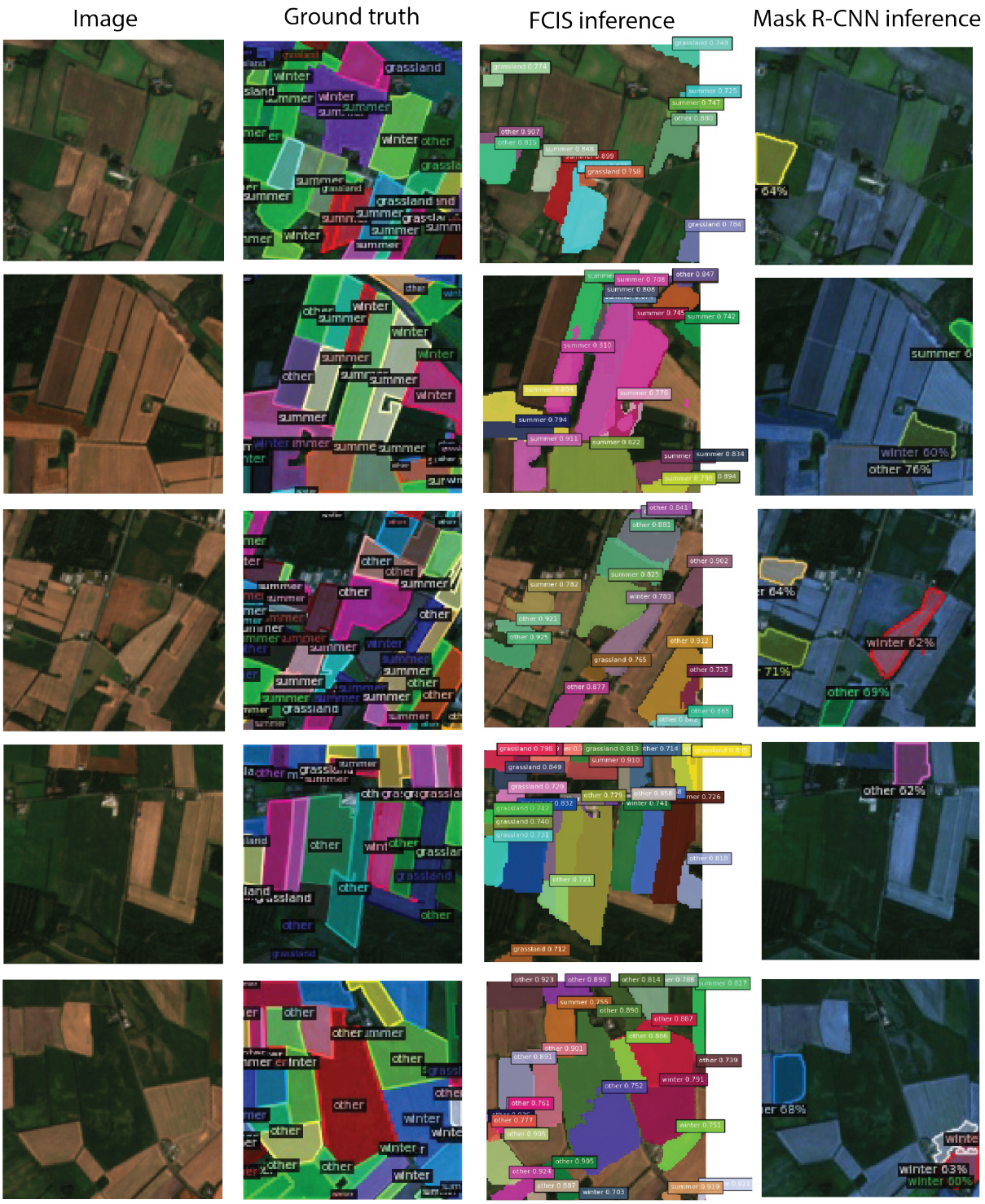


Figure 48: *Example image chips from the validation set of 'wh_win_sum' in 2020. IoU threshold set to 0.6 for both FCIS and Mask R-CNN.*

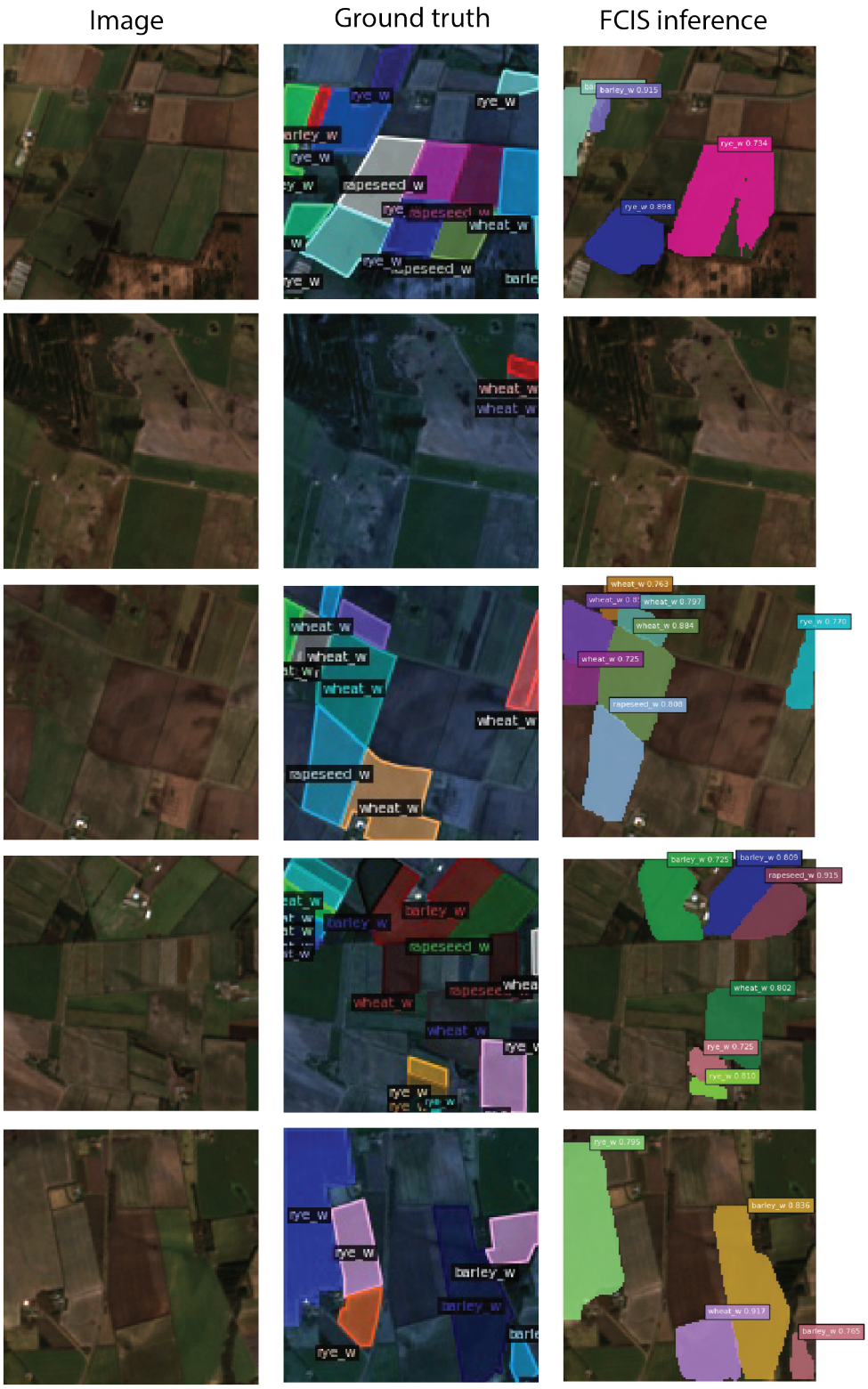


Figure 49: *Example image chips from the validation set of 'jm_m' in 2020. IoU threshold set to 0.6 for both FCIS and Mask R-CNN. There were no detections of fields for Mask R-CNN IoU at 0.6.*

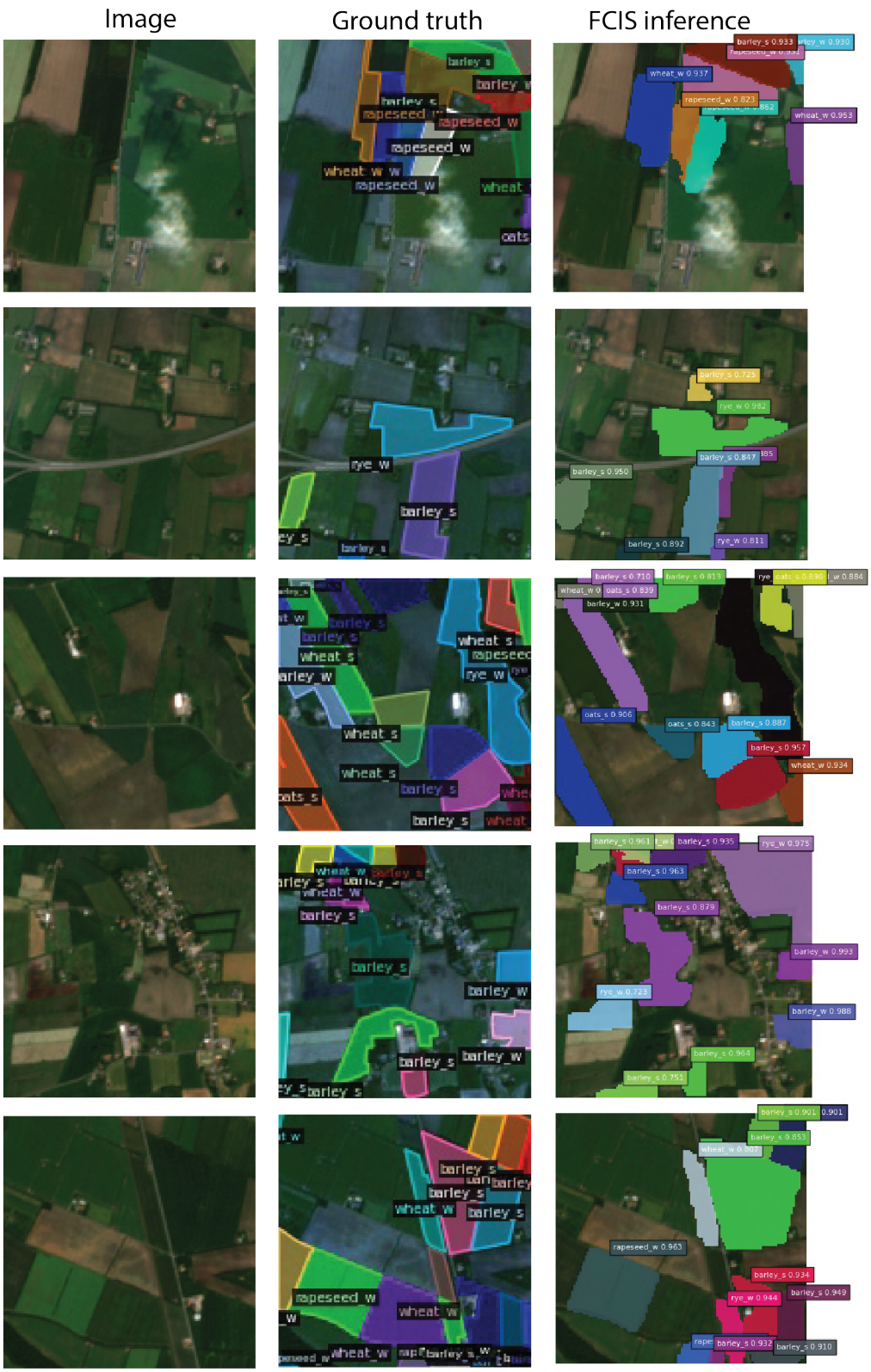


Figure 50: *Example image chips from the validation set of 'aj_m' in 2020. IoU threshold set to 0.6 for both FCIS and Mask R-CNN. There were no detections of fields for Mask R-CNN IoU at 0.6.*

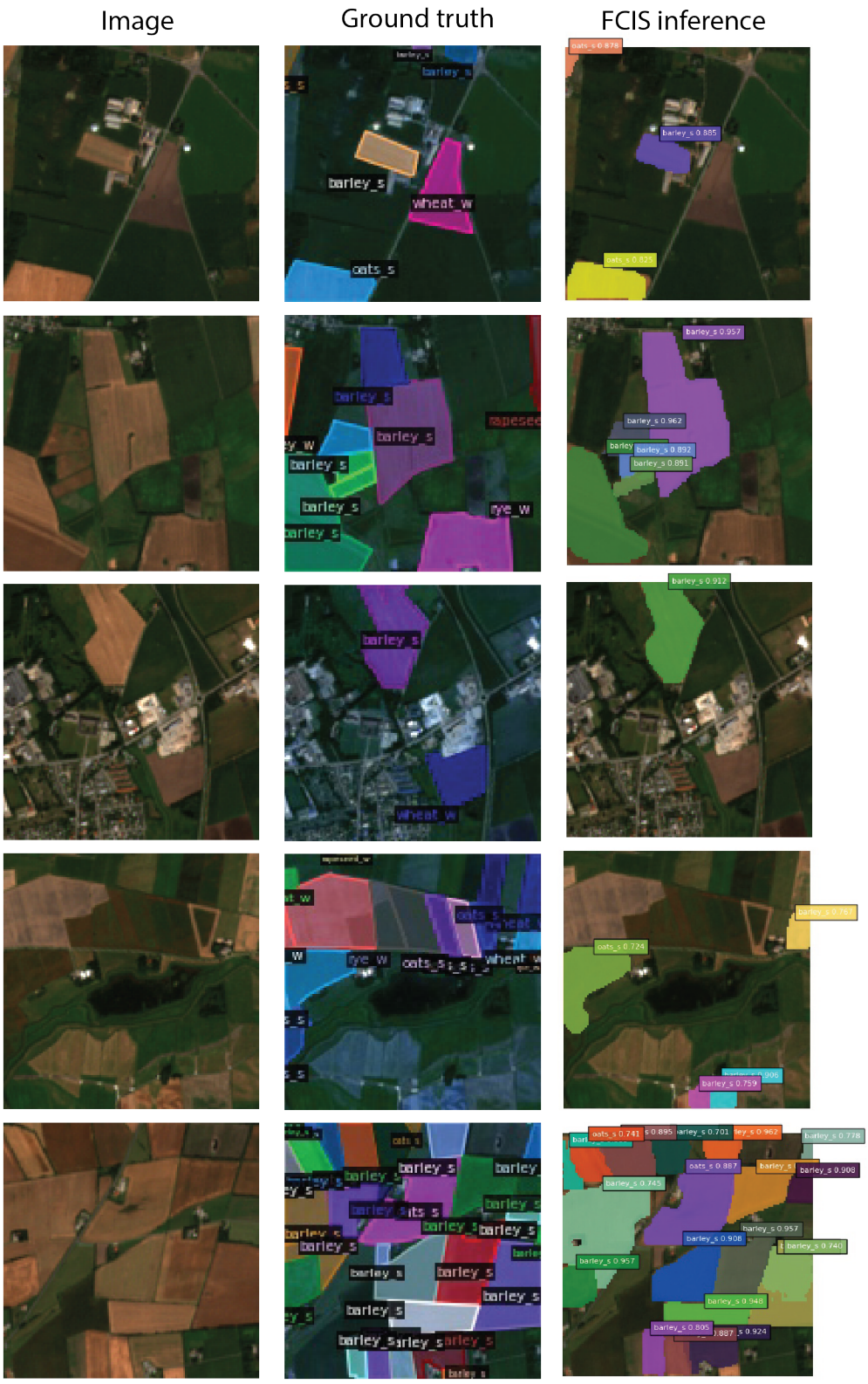


Figure 51: *Example image chips from the validation set of 'js_m' in 2020. IoU threshold set to 0.6 for both FCIS and Mask R-CNN. There were no detections of fields for Mask R-CNN IoU at 0.6.*

Part III
Conclusion

10 Discussion

This study performed an instance segmentation based on the vector polygons of crop fields annotated on Senitel-2 raster image. Mask R-CNN and FCIS were chosen and compared as the two instance segmentation methods. In order to answer the research questions mentioned in the subsection 1.2, three major experiments were conducted (see section 8). The results obtained varied significantly from crop to crop, with some crops performing better in one stage and poorly in another stage of a crop cycle. In the following sections, we will analyse and discuss the results we obtained in section 9.

10.1 Mask R-CNN vs FCIS analysis

Figure 52 shows performance comparison between Mask R-CNN and FCIS. As can be seen, Mask R-CNN fails to learn the features on vegetation land zone substantially. At the same time, FCIS performs significantly better on all the tasks – classification, detection and segmentation. In order to comprehensively illuminate on this distinction between the two frameworks' performances, the nature of the employed dataset must be elucidated. The dataset contains low-resolution satellite images of vegetation fields having arbitrary shapes and sizes, unlike the COCO dataset [18]. Even objects in the same class tend to have different shapes and sizes, unlike the COCO dataset where all classes have their own unique distributions.

The reason for the failure of Mask-RCNN is inherent in its working. As discussed previously in paragraph 4.3.2.2, Mask R-CNN have three branches – classification, bounding box and mask heads. Each of these branches originates from the same RoIs and performs the designated tasks individually. While training on COCO dataset, where each class distribution differs from others, Mask R-CNN a priori adds an inductive bias for the shape and size of the object at the RoI level. Hence, when Mask R-CNN performs training on current dataset, it fails to add such an a prior bias as the class distribution of the vegetation field is particularly random.

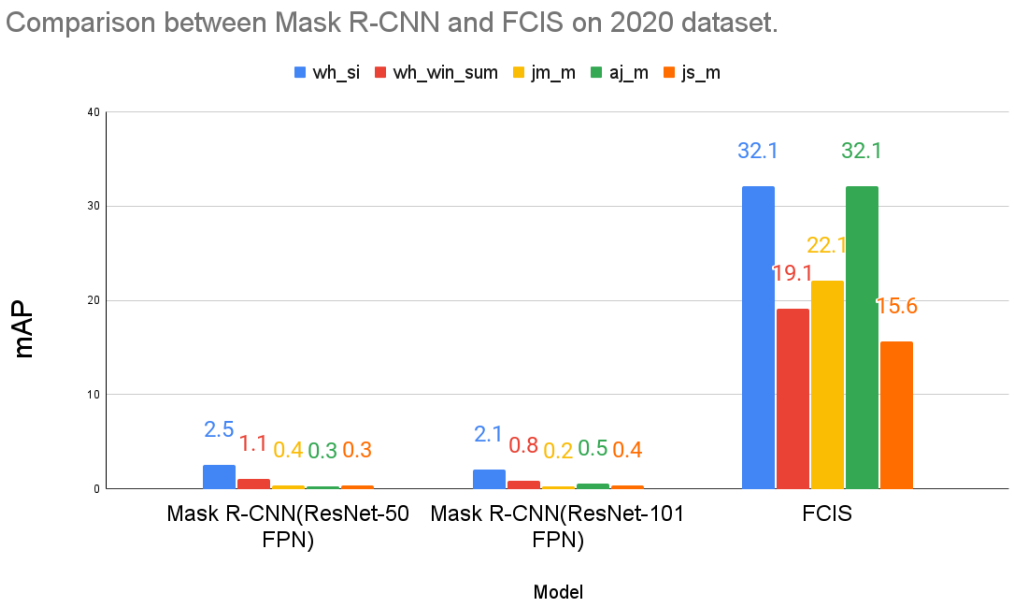


Figure 52: *The bar graph shows performance (mAPs) comparison between Mask R-CNN and FCIS on five dataset categories for year 2020.*

On the other hand, FCIS performs all three tasks – classification, bounding box regressor and mask generation – simultaneously. As explained in paragraph 4.3.2.1, for each object category, two sets of kxk score maps are generated from the preceding convolutional layer where k is the feature map size (resolution). Now, each pixel has two scores representing the likelihood that the pixel belongs to the inside or the outside of the boundary of the object instance. For example, consider Figure 53 with the three numbered regions as shown. If a pixel is in region 3, it will be considered

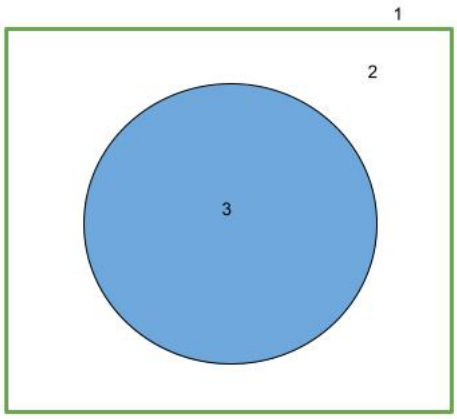


Figure 53: *Illustration of 'how inside/outside score maps are considered for detection and segmentation'. In the figure, there are three regions: region 1, region 2 and region 3. Region 3 denotes Segmentation+, Detection+ : Inside scores (Inside bbx and inside mask). Region 2 denotes Segmentation-, Detection + : Outside scores (Inside bbx and inside mask). Region 3 denotes Segmentation -, Detection - : None (Outside bbx and outside mask). For detection, FCIS performs pixel-wise max of region 2 and region 3. For segmentation, FCIS performs pixel-wise softmax of region 2 and region 3.*

as detection+ and segmentation+; hence, that pixel will have a high inside score and low outside score [25]. Similarly, if a pixel is in region 2, it will be considered as detection+ and segmentation- and will have a high outside score and low inside score. For region 1, there will be no object and hence no classification voting would be performed. For mask prediction, a softmax operation is performed on regions 2 and 3 producing pixel-wise foreground probabilities that would estimate the shape of mask. For mask classification (i.e. object classification), a pixel-wise max operation is carried out, which produces pixel-wise likelihood of the object category. As a result, there is no added bias in FCIS regarding the object's shape and size since it first estimates the mask and subsequently classifies the object.

10.2 Crop-specific analysis

Due to considerably poor results with Mask R-CNN, only FCIS performances are compared for crop-specific analysis on 'jm_m', 'aj_m' and 'js_m' datasets. Based on the results of subsubsection 9.1.1, it can be concluded that the FCIS is more efficient on winter crops than summer crops in general. In the quarter January-March, all crops had mAP of 24.3, except rye winter crop with AP of 15.4. The rapeseed winter crop had the highest AP (mean AP of 30.73) among the seven crops, and emerged as a leading crop in all mentioned datasets. Alternatively, wheat summer (mean AP of 19.1) and oats summer (mean AP of 1.95) crops were among the worst performers in 'aj_m' and 'js_m' datasets. According to imbalance in class distributions (Figure 37), wheat summer and oats summer crops have significantly fewer instances than barley summer and wheat winter crops, which could be a probable cause for the poor performance. Further future research on crop-specific classification should elucidate on the matter.

10.3 Crop-cycle-specific analysis

The term crop cycle refers to the different stages of a particular crop in its development from planting to harvest. The crop cycle is divided into three stages, which have been grouped into four quarters for one year in this thesis work (see subsection 5.1). This categorisation of crop stages for a repetitive interval is also known as crop year [39]. In theory, crops should exhibit the same features every year during each stage of their life cycle. However, results in the subsubsection 9.1.1 show a decreasing trend in performance of FCIS on crops from years 2020 to 2019. Changes in the environment due to climate change could be one of the reasons for the difference in performance for the years 2020 and 2019. The decreasing trend in 2019 crops doesn't mean that 2019 was more affected by climate change than 2020. The performance metrics here only show the difference in image features of crops in 2020 and 2019, as the model was trained on 2020 crop images and tested on 2019 crop images.

In [48], the authors show the effects of the shift in growing season due to climate change on different types of rice yield in South Korea. The paper discussed how the growing period and length of the rice crop year shrank or expanded over the years. As a result, the climatic change in environment may affect the performance of 2019 and 2020. In the January-March quarter, the rapeseed winter crop yields approximately a similar AP for 2020 and 2019, which raises the question of which crops will be most and least affected by climate change at different stages.

10.4 Comparison to Rieke 2017

The work in this thesis is related to Rieke [50]. Rieke [50] used the raster images from Sentinel-2 level 1C, which needed post-processing for the correction of RGB values while converting them into JPEG. In contrast, Sentinel-2 level 2A images, which didn't require any correction measure in the generation of JPEG images, were utilised in this work. Moreover, raster images for the entire Denmark area (80767 km^2), used in this study, were extracted from Google Earth Engine, which was nearly four times that of Rieke [50] (20900 km^2). Additionally, Rieke [50] focused more on regular-shaped fields, and irregular shapes were only extracted for agriculture fields that were near-forest areas. Table 14 shows the mAP comparison between this work and Rieke [50] on single-class classification. The multi-class classification performed by Rieke was only limited to one dataset category for the whole year. The mAP result in Rieke [50] is higher than this thesis' work for single-class classification. This shows that employing more data adds more randomness (irregularities in shapes) to the datasets, which increases false positives. This is a noteworthy point for future research designed to deploy CNN techniques to map fields or other land cover objects. The feasibility of mapping the detection target can be more critical than using an order-of-magnitude more training data to improve a model's generalization capabilities.

Work	mAP
Rieke	39.1
This thesis work	32.1

Table 14: *Table shows the mAP comparison between this thesis and Rieke [50]*

11 Conclusion and Future work

With the purpose of introducing vision-based instance segmentation as a decision-support system to study crop production from earth observation satellite, this thesis work focused on implementation and evaluation of instance segmentation methods on vegetation and agriculture zones from the Sentinel-2 dataset. This study concentrated on providing insights about crop classification in their respective crop-cycles and attempted to answer whether crop classification is possible on low-resolution satellite images.

To achieve these goals, the experiments were based on crop-cycles specific to a given year, and for the preparation of multiple datasets, an iterative dataset creation pipeline was introduced. Furthermore, COCO evaluation metrics were defined to evaluate crop-specific and crop-cycle-specific datasets on two frameworks – FCIS and Mask-RCNN. After incorporating segmentation results of all dataset categories and open-source implementation of the presented frameworks, it can concluded from the above analysis that FCIS with ResNet-101 backbone was better suited for extraction of random shape datasets such as vegetation/agricultural land datasets.

From crop-specific analysis, it was found that winter crops displayed significantly better results than their counter summer crops at each stage of the crop cycle. Also, a significant difference was found between the performances of datasets from years 2019 and 2020, considering AP statistics for multi-class classification. Moreover, it was found that increasing the size of the dataset compared to Rieke [50] did not improve the performance, as previously mentioned in subsection 10.4. Rather, the mAP for single-class classification was lower than that of Rieke [50].

Considering these differences, further experiments on instance segmentation of agricultural fields with regular and irregular shapes with deep CNNs will be helpful. Assessing the positive results in this thesis with FCIS on randomly-shaped datasets, the application of deep learning instance segmentation methods on satellite data has a strong potential for the future, provided FCIS is reimplemented in a modular way. In addition, further research on climatic factors would likely be necessary to determine the reasons for the difference in the performance of FCIS between years 2020 and 2019. Besides the overall aim to provide answers to the aforementioned research questions, this thesis also shed light on a possible reason for the failure of state-of-the-art Mask R-CNN framework on the image instance segmentation task on vegetation/agriculture land datasets.

The methodology presented in this thesis offers potential for many improvements. First, as mentioned above, a modular reimplementation of FCIS is required to conduct smooth, fast and efficient experiments. Second, the Sentinel-2 satellite offers 10m resolution, which is considerably low for pixel-wise segmentation. Instead of using satellite images, areal images from drones would certainly improve the resolution of agricultural zones. For future research, it would be worthwhile to study whether a vegetation/agriculture zone segmentation model could also be deployed as a region-independent model. Just as it was realistic to assume that crops at a specific stage of a crop year within different crop years would share similar image features, it stands to reason that crops in different regions would share the same distribution in terms of image features.

Code Availability

Implementation of the instance segmentation frameworks along with other conducted experiments of this study can be found at:
`https://github.com/MISSEY/Vegetation-Land-Segmentation-and-Classification`

The programming language used is Python 3.8.5 for Detectron2 and Python 2.7 for FCIS. More details of packages and their version can be found in respective requirements.txt files.

References

[1] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997, ISBN: 978-0-07-042807-2.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, 1998, pp. 2278–2324. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.7665.

[3] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Networks*, vol. 12, no. 1, pp. 145–151, 1999, ISSN: 0893-6080. DOI: https://doi.org/10.1016/S0893-6080(98)00116-6. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608098001166.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li, "Imagenet: A large-scale hierarchical image database," Jun. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[5] K. Tempfli, G. Huurneman, W. Bakker, L. Janssen, W. Feringa, A. Gieske, K. Grabmaier, C. Hecker, J. Horn, N. Kerle, F. Meer, G. Parodi, C. Pohl, C. Reeves, F. Ruitenbeek, E. Schetselaar, M. Weir, E. Westinga, and T. Woldai, "Principles of remote sensing : An introductory textbook.," in. Jan. 2009, ISBN: 978-90-6164-270-1.

[6] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, pp. 303–338, Jun. 2010. DOI: 10.1007/s11263-009-0275-4.

[7] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Y. W. Teh and M. Titterington, Eds., ser. Proceedings of Machine Learning Research, vol. 9, Chia Laguna Resort, Sardinia, Italy: PMLR, 2010, pp. 249–256. [Online]. Available: https://proceedings.mlr.press/v9/glorot10a.html.

[8] N. Science. (2010). "National aeronautics and space administration, science mission directorate. (2010). reflected near-infrared waves.," [Online]. Available: https://science.nasa.gov/ems/08_nearinfraredwaves. (accessed: 2021-07-05).

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., vol. 25, Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68Paper.pdf.

[10] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013. arXiv: 1311.2524. [Online]. Available: http://arxiv.org/abs/1311.2524.

[11] C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection," in *Advances in Neural Information Processing Systems*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds., vol. 26, Curran Associates, Inc., 2013. [Online]. Available: https://proceedings.neurips.cc/paper/2013/file/f7cade80b7cc92b991cf4d2806d6bd78-Paper.pdf.

[12] J. Uijlings, K. V. D. Sande, T. Gevers, and A. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, pp. 154–171, 2013.

[13] M. D. Zeiler and R. Fergus, *Visualizing and understanding convolutional networks*, 2013. arXiv: 1311.2901 [cs.CV].

[14] M. Lin, Q. Chen, and S. Yan, *Network in network*, 2014. arXiv: 1312.4400 [cs.NE].

[15] R. B. Girshick, "Fast R-CNN," *CoRR*, vol. abs/1504.08083, 2015. arXiv: 1504.08083. [Online]. Available: http://arxiv.org/abs/1504.08083.

[16] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. arXiv: 1512.03385 [cs.CV].

[17] ——, *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*, 2015. arXiv: 1502.01852 [cs.CV].

[18] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, *Microsoft coco: Common objects in context*, 2015. arXiv: 1405.0312 [cs.CV].

[19] J. Long, E. Shelhamer, and T. Darrell, *Fully convolutional networks for semantic segmentation*, 2015. arXiv: 1411.4038 [cs.CV].

[20] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015. arXiv: 1506.01497. [Online]. Available: http://arxiv.org/abs/1506.01497.

[21] O. Ronneberger, P. Fischer, and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, 2015. arXiv: 1505.04597 [cs.CV].

[22] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, 2015. arXiv: 1409.1556 [cs.CV].

[23] J. Dai, Y. Li, K. He, and J. Sun, *R-fcn: Object detection via region-based fully convolutional networks*, 2016. arXiv: 1605.06409 [cs.CV].

[24] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[25] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, *Fully convolutional instance-aware semantic segmentation*, 2017. arXiv: 1611.07709 [cs.CV].

[26] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, *Feature pyramid networks for object detection*, 2017. arXiv: 1612.03144 [cs.CV].

[27] L. Weng, "Object detection for dummies part 3: R-cnn family," *lilianweng.github.io/lil-log*, 2017. [Online]. Available: http://lilianweng.github.io/lil-log/2017/12/31/object-recognition-for-dummies-part-3.html.

[28] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, *Aggregated residual transformations for deep neural networks*, 2017. arXiv: 1611.05431 [cs.CV].

[29] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, *Encoder-decoder with atrous separable convolution for semantic image segmentation*, 2018. arXiv: 1802.02611 [cs.CV].

[30] K. He, G. Gkioxari, P. Dollár, and R. Girshick, *Mask r-cnn*, 2018. arXiv: 1703.06870 [cs.CV].

[31] L. Liu, W. Ouyang, X. Wang, P. W. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," *CoRR*, vol. abs/1809.02165, 2018. arXiv: 1809.02165. [Online]. Available: http://arxiv.org/abs/1809.02165.

[32] A. Gupta, P. Dollár, and R. Girshick, *Lvis: A dataset for large vocabulary instance segmentation*, 2019. arXiv: 1908.03195 [cs.CV].

[33] C. Corbane, P. Politis, P. Kempeneers, D. Simonetti, P. Soille, A. Burger, M. Pesaresi, F. Sabo, V. Syrris, and T. Kemper, "A global cloud free pixel- based image composite from sentinel-2 data," *Data in Brief*, vol. 31, p. 105 737, 2020, ISSN: 2352-3409. DOI: https://doi.org/10.1016/j.dib.2020.105737. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352340920306314.

[34] A. Tao, K. Sapra, and B. Catanzaro, "Hierarchical multi-scale attention for semantic segmentation," *CoRR*, vol. abs/2005.10821, 2020. arXiv: 2005.10821. [Online]. Available: https://arxiv.org/abs/2005.10821.

[35] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, *Swin transformer: Hierarchical vision transformer using shifted windows*, 2021. arXiv: 2103.14030 [cs.CV].

[36] ——, (2021). "Swin transformer: Hierarchical vision transformer using shifted windows," [Online]. Available: https://github.com/SwinTransformer/Swin-Transformer-Object-Detection. (accessed: 2021-09-05).

[37] I. Ulku and E. Akagunduz, *A survey on deep learning-based architectures for semantic segmentation on 2d images*, 2021. arXiv: 1912.10230 [cs.CV].

[38] X. Wang, R. Zhang, C. Shen, T. Kong, and L. Li, *Solo: A simple framework for instance segmentation*, 2021. arXiv: 2106.15947 [cs.CV].

[39] U. S. D. for Agriculture. (). "Crop calenders for europe," [Online]. Available: https://ipad.fas.usda.gov/rssiws/al/crop_calendar/europe.aspx. (accessed: 2021-08-09).

[40] Z. C. L. Aston Zhang. (). "Dive into deep learning," [Online]. Available: https://d2l.ai/chapter_convolutional-neural-networks/index.html. (accessed: 2021-07-05).

[41] J. J. Fei-Fei Li Andrej Karpathy. (). "Spatial localization and detection," [Online]. Available: http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf. (accessed: 2021-07-26).

[42] fvm. (). "Ministry of environment and food of denmark," [Online]. Available: https://en.fvm.dk/the-ministry/. (accessed: 2021-08-23).

[43] R. gandhi. (). "R-cnn, fast r-cnn, faster r-cnn, yolo — object detection algorithms," [Online]. Available: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e. (accessed: 2021-07-26).

[44] GisGeography. (). "What is remote sensing," [Online]. Available: https://gisgeography.com/remote-sensing-earth-observation-guide/#chapter3. (accessed: 2021-07-05).

[45] M. Nielsen. (). "Neural networks and deep learning," [Online]. Available: hhttp://neuralnetworksanddeeplear com/chap6.html. (accessed: 2021-07-06).

[46] O'reilly. (). "Intersection over union," [Online]. Available: https://www.oreilly.com/library/view/deep-learning-for/9781788295628/a5ce2fa2-8c67-4ead-a9bd-a2d07b5f3fa8.xhtml. (accessed: 2021-08-23).

[47] C. Olah. (). "Deep learning, nlp, and representations," [Online]. Available: http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/. (accessed: 2021-07-05).

[48] J.-Y. C. Pu Reun Yoon1. (). "Effects of shift in growing season due to climate change on rice yield and crop water requirements," [Online]. Available: https://rdcu.be/cwRQt. (accessed: 2021-09-02).

[49] A. Rai. (). "Remote sensing and gis applications in agriculture," [Online]. Available: http://www.gisresources.com/wp-content/uploads/2013/09/Remote-Sensing-and-GIS-Application-in-Agriculture_2.pdf. (accessed: 2021-07-05).

[50] C. Rieke. (). "Deep learning for instance segmentation of agricultural fields - master thesis," [Online]. Available: https://github.com/chrieke/InstanceSegmentation_Sentinel2. (accessed: 2021-08-09).

[51] S. Shah. (). "A comprehensive guide to convolutional neural networks," [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. (accessed: 2021-07-16).

[52] sklearn. (). "Average precision," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.average_precision_score.html. (accessed: 2021-08-23).

[53] Stanford. (). "Convolutional neural networks," [Online]. Available: https://cs231n.github.io/convolutional-networks/. (accessed: 2021-07-16).

[54] ——, (). "Introduction to neural networks," [Online]. Available: https://cs231n.github.io. (accessed: 2021-07-19).

[55] U. of Twente. (). "Spectral reflectance curves," [Online]. Available: https://ltb.itc.utwente.nl/498/concept/81713. (accessed: 2021-07-05).

[56] H. state university. (). "Natural and false color composites," [Online]. Available: https://gsp.humboldt.edu/OLM/Courses/GSP_216_Online/lesson3-1/composites.html. (accessed: 2021-07-05).

[57] ——, (). "Spectral reflectance of earth surface features," [Online]. Available: https://gsp.humboldt.edu/OLM/Courses/GSP_216_Online/lesson2-1/reflectance.html. (accessed: 2021-07-05).

[58] USGS. (). "Usgs spectral library," [Online]. Available: https://pubs.er.usgs.gov/publication/ds1035. (accessed: 2021-07-05).

[59] wikipedia. (). "Convolution," [Online]. Available: https://en.wikipedia.org/wiki/Convolution. (accessed: 2021-07-06).

[60] ——, (). "Timeline of first images of earth from space," [Online]. Available: https://en.wikipedia.org/wiki/Timeline_of_first_images_of_Earth_from_space. (accessed: 2021-08-09).

APPENDICES

Appendix 1

```
jan_mar = {
    'barley_w': [10],
    'wheat_w': [11, 13],
    'rye_w': [14, 15],
    'rapeseed_w': [22]
}

apr_jun = {
    'barley_w': [10],
    'wheat_w': [11,13],
    'rye_w': [14,15],
    'rapeseed_w':[22],
    'barley_s': [1],
    'wheat_s' :[2,6],
    'oats_s':  [3]
}

whole_year = {
    'winter': [9, 10, 11, 13, 14, 15, 16, 17, 22, 57, 220, 221, 222, 223, 224, 235],
    'summer': [1, 2, 3, 4, 6, 8, 21, 55, 56, 210, 211, 212, 213, 214, 230, 234],
    'other': [5, 7, 17, 23, 24, 25, 30, 31, 32, 35, 36, 40, 41, 42, 51, 52, 53, 54, 58, 105, 106, 107, 108, 109, 110,
              111, 120, 121, 122, 123, 124, 125, 126, 149, 150, 152, 153, 160, 161, 162, 180, 182, 215, 216, 281, 283,
              400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 415, 416, 417, 418, 420, 421, 422,
              423, 424, 430, 429, 431, 432, 434, 440, 448, 449, 450, 487, 488, 489, 491, 493, 496, 497, 498, 499, 501,
              502, 503, 504, 505, 507, 509, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526,
              527, 528, 529, 530, 531, 532, 533, 534, 536, 539, 540, 541, 542, 543, 544, 545, 547, 548, 549, 550, 551,
              552, 553, 560, 561, 563, 570, 579],
    'grassland': [101, 102, 103, 104, 112, 113, 114, 115, 116, 117, 118, 170, 171, 172, 173, 174, 260, 261, 262, 263,
                  264, 266, 267, 268, 269, 270, 284]
}

agriculture_land = {
    "agriland": [9, 10, 11, 13, 14, 15, 16, 17, 22, 57, 220, 221, 222, 223, 224, 235, 1, 2, 3, 4, 6, 8, 21, 55, 56, 210,
                 211, 212, 213, 214, 230, 234, 5, 7, 17, 23, 24, 25, 30, 31, 32, 35, 36, 40, 41, 42, 51, 52, 53, 54, 58,
                 105, 106, 107, 108, 109, 110, 111, 120, 121, 122, 123, 124, 125, 126, 149, 150, 152, 153, 160, 161, 162
                 180, 182, 215, 216, 281, 283, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 415
                 416, 417, 418, 420, 421, 422, 423, 424, 430, 429, 431, 432, 434, 440, 448, 449, 450, 487, 488, 489, 491
                 493, 496, 497, 498, 499, 501, 502, 503, 504, 505, 507, 509, 512, 513, 514, 515, 516, 517, 518, 519, 520
                 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 536, 539, 540, 541, 542, 543, 544
                 545, 547, 548, 549, 550, 551, 552, 553, 560, 561, 563, 570, 579, 101, 102, 103, 104, 112, 113, 114, 115
                 116, 117, 118, 170, 171, 172, 173, 174, 260, 261, 262, 263, 264, 266, 267, 268, 269, 270, 284
                 ]
}
```

Figure 54: *Crop codes used for creating labels for five datasets.*

Appendix 2 - Crop classes

Below table lists Denmark crop codes and crop names provided by food ministry of Denmark.

Crop_nr	Name
1	Spring barley
2	Spring wheat
3	Spring oats
4	Mix of spring sown cereals
5	Maize for maturity
6	Spring wheat, bread wheat
7	Cereal + pulse under 50% pulse
8	spring sown Triticum spelta
9	winter sown Triticum spelta
10	Winter barley
11	Winter wheat
13	Winter wheat, bread wheat
14	Winter rye
15	Winter hybrid rye
16	Winter triticale
17	Mix of autum sown cereals
21	Spring rape
22	Winter rape
23	Turnip rape
24	Sunflower
25	Soybeans
30	Peas
31	Broad bean, fava bean
32	Sweet lupin
35	Pulse, perennial mix
36	Pulse, other types for maturity mix
40	Oil flax
41	Fibre flax
42	Hemp
51	Mixed broad-leaved crop, seed/grain
52	Quinoa
53	Buckwheat
54	Pulse mix
55	Spring rye
56	Spring triticale
57	Winter oats
58	Sorghum
101	Rye-grass seeds, regular
102	Rye-grass seeds, regular 1 year, autum undersown
103	Rye-grass seeds, Italian
104	Rye-grass seeds, Italian 1 year, autum undersown
105	Timothy seed
106	Cocksfoot seed
107	Meadow fescue seed
108	Red fescue seed
109	Rye fescue seed
110	Hard fescue seed
111	Tall fescue seed
112	Kentucky bluegrass seed (ley type)
113	Kentucky bluegrass seed (turfgrass type)
114	Rough-stalked meadow grass
115	Browntop and creeping bent
116	Rye-grass, hybrid
117	Rye-grass, autum undersown, hybrid

Crop_nr	Name
118	Rye fescue seed, autum undersown
120	Clover seeds
121	Ley legumes
122	Caraway seed
123	Poppy seeds
124	Spinach seeds
125	Beetroot seed
126	Mix of seeds for seed corn
149	Seed Potatoes (certified)
150	Seed potatoes (own propagation)
151	Potatoes, starch
152	Potatoes, consumption
153	Potatoes, other
154	Potatoes, eating- (processed, peeled boiled)
155	Potatoes, powdered/flaked-
156	Potatoes, fried/chips/french fries
160	Sugar beet for factory
161	Chicory roots
162	Mix, other industrial crops
170	Grass for factory (rotation)
171	Alfalfa, mowing
172	Alfalfagrass, over 25% grass for mowing incl. own feed
173	Clover for mowing
174	Clovergrass for factory
180	White mustard
182	Mix of oil species
210	Spring barley, wholecrop
211	Spring wheat, wholecrop
212	Spring oats, wholecrop
213	Mixed cereal, spring sown, wholecrop
214	Cereal and pulse, wholecrop, under 50% pulse
215	Peas wholecrop
216	Silo maize
220	Winter barley, wholecrop
221	Winter wheat, wholecrop
222	Winter rye, wholecrop
223	Winter triticale, wholecrop
224	Mix of cereals, autum sown wholecrop
230	Mix of spring cereals, green grain
234	Cereal and pulse, green grain, under 50% pulse
235	Mix of winter cereals, green grain
247	Environmental grass AECM scheme 2 (0 N) (rotation)
248	Permanent grass next to water borehole
249	Utilized grass next to water boreholes
250	Permanent grass, very low yield
251	Permanent grass, low yield
252	Permanent grass, standard yield
253	Environmental grass AECM scheme 1 (80 N) (rotation)
254	Environmental grass AECM scheme 2 (0 N) (permanent)
255	Permanent grass, under 50% clover/alfalfa
256	Permanent clovergrass, over 50% clover/alfalfa
257	Permanent grass, without clover
259	Permanent grass, factory, over 6 tons
260	Grass with clover/alfalfa, under 50% legumes (rotation)
261	Clovergrass, over 50% clover (rotation)
262	Alfalfa, alfalfagrass, over 50% alfalfa (rotation)
263	Grass without clovergrass (rotation)

Crop_nr	Name
264	Grass and clovergrass without N norm, under 50% clover (rotation)
266	Grass under 50% clover/alfalfa, extremely low yield (rotation)
267	Grass under 50% clover/alfalfa, very low yield (rotation)
268	Grass under 50% clover/alfalfa, low yield (rotation)
269	Grass, rolled sod
270	Grass for freerange pigs, rotation
271	Recreative purpose
272	Permanent grass for factory
273	Alfalfa for factory
274	Permanent alfalfagrass over 25% grass, for factory
276	Permanent grass and clovergrass without N norm, under 50% clover
277	Clover for factory
278	Permanent alfalfa and alfalfagrass over 50% alfalfa
279	Permanent grass for factory
280	Sugar beet for fodder
281	Swedes
282	Marrow stem kale
283	Fodder carrot
284	Grass with vetch and other legumes, under 50% legumes
285	Grass and clovergrass without N norm, over 50% clover (rotation)
286	Permanent grass and clovergrass without N norm, over 50% clover
287	Grass for freerange pigs, permanent
305	Permanent grass, without disbursement off subsidy for organic farming
306	Grass in rotation, without disbursement off subsidy for organic farming
308	EFA-Fallow with summer mowing
309	Uncultivated area next to water boreholes
310	Fallow with summer mowing
311	Forest establisment on formerly farmland 1
312	20-year set aside
313	20-year side aside of agricultural land with voluntary forest establisment
314	20-year set aside with scheme of forest establiment from NST
317	Wetlands with set aside
318	AECM no set aside, not farmland
319	EFA Wetlands or shallow soils with set aside
321	Environmental measures, not farmland
322	Mini-wetlands with project scheme
323	EFA-Uncultivated area next to water boreholes
324	Fallow with flowers
325	EFA-fallow with flowers
326	Environmental sensitive permanent grassland converted into non-agricultural area
327	EFA-field margins, mowing/grazing
328	EFA-field margins with flower mix
329	EFA-field margins with environmental scheme
361	EFA-fallow for melliferous plants
406	Farmland not warranted for subsidies
407	Gherkin
408	Asparagus
409	Celery
410	Cauliflower
411	Broccoli
412	Courgette, squash
413	Kale, Leaf cabbage
415	Carrot
416	White cabbage
417	Chinese cabbage
418	Celeriac
420	Onion

Crop_nr	Name
421	Parsnip
422	Root parsley
423	Leek
424	Brussels sprouts
429	Beetroot
430	Red cabbage
431	Lettuce, open air
432	Savoy cabbage, oxheart cabbage
434	Spinach
440	Sweet corn
448	Peas, consumption
449	Jerusalem artichokes, comsumption
450	Parsley
486	Chives
487	Herbs (except parsley and chives)
488	Vegetable, others (open air)
489	Coneflower
491	Medicinal plants, annual and biennial
492	Medicinal plants, perennials
493	Vegetable, mixture
494	Henyard, with bare soil
495	Agroforestry
496	Henyard, permanent grass
497	Sea buckthorn
499	Cranberry
501	Lingonberry
502	Black chokeberry
503	Japanese quince
504	Mulberries
505	Medicinal plants, arboreal plants
506	Nursery cultivation, arboreal plants, for sale
507	Closed system
508	Perennials
509	Flower bulb
510	Annual and biennial plants
512	Blackcurrant, multiplication by cutting
513	Red currant, multiplication by cutting
514	Gooseberry, multiplication by cutting
515	Raspberry, multiplication by cutting
516	Other of the genus Vaccinium
517	Quince
518	Melon
519	Rhubarb
520	Strawberries
521	Blackcurrant
522	Red currant
523	Gooseberry
524	Blackberry
525	Raspberry
526	Blueberry
527	Sour cherry without undergrowth of grass
528	Sour cherry with undergrowth of grass
529	Plum without undergrowth of grass
530	Plum with undergrowth of grass
531	Sweet cherry without undergrowth of grass
532	Sweet cherry with undergrowth of grass
533	Elder

Crop_nr	Name
534	Hazel
535	Apples
536	Pears
537	Grapes
538	Other tree fruit
539	Other shrub fruit
540	Rowanberry
541	Rosehip
542	Juneberry
543	Food grape
544	Common Walnut
545	Sweet chestnut
547	Mixed fruit
548	Tomatoes
551	Cucumber
552	Lettuce (green house)
553	Vegetable, others (green house)
563	Cut flowers and foliage
564	Houseplants
570	Nursery cultivation, perennials
576	Seedlings, annual
577	Pumpkin
578	Vegetable marrow
579	Giant pumpkin
580	Mushrooms, champignon
581	Container area
582	Hops
583	Afforestation - the improvement of aquatic environment and groundwater protection
586	Afforestation - the improvement of aquatic environment and groundwater protection
587	African marigold
588	Forestry, common
589	Afforestation with treehight below 3 m
590	Greenery cuttings, organic farming
591	Christmas trees and greenery cuttings
592	Forest establisment in project areas, not included by a scheme
593	Public forest establisment
594	Forest establisment on former farmland 3
596	State funded forest establisment
597	Sustainable forestry
598	Sustainable forestry in Natura 2000 area
599	Coppice forest
602	Willow
603	Poplar
604	Alder
605	Elephant grass
606	Reed canary grass
650	Sorrel
651	Poplar
652	EFA - Willow
653	EFA - Poplar
654	EFA - Alder
655	EFA - Coppice forest
656	EFA - Poplar
657	Chrysanthemum Garland, seeds
658	Dill seeds
659	Chinese cabbage seeds
660	Cress seeds

Crop_nr	Name
661	Rocket, Argula seeds
662	Radish seed (incl. fodder radish seeds)
663	Spinach beet seed, beetroot seeds
664	Green cabbage seeds
665	Carrot seeds
666	Cabbage seeds (white and red cabbage)
667	Parsley seeds
668	Chervil seeds
701	Turnip seeds
702	Parsnip seeds
703	Scorzonera/Scorzonera seeds
704	Salsify seeds
705	Chive seeds
706	Thyme seeds
707	Flower seeds
708	Green grain of spring barley
709	Green grain of spring wheat
710	Green grain of spring oats
711	Green grain of spring rye
900	Green grain of spring triticale
903	Green grain of winter barley
905	Green grain of winter wheat
907	Green grain of winter oats

Appendix 3 - Mask R-CNN Results

Backbone	Acc	AccFG	Mask head's loss	Mask Acc	Mask FN	Mask FP
ResNet-50	87.36%	53.02%	0.35	84.57%	14.13%	16.21%
ResNet-101	88.80%	48.21%	0.37	83.59%	13.75%	17.84%

Table 16: *Mask R-CNN's training metrics for **category 1** dataset (image size is 128x128 px) for year 2020. No of epochs was 2k. Acc is classification accuracy (including background). AccFG is only foreground object classification (without background). Mask Acc is mask head's accuracy. Mask FN & Mask FP are mask head's false negative and false positive, respectively*

Backbone	Acc	AccFG	Mask head's loss	Mask Acc	Mask FN	Mask FP
ResNet-50	87.15%	24.24%	0.29	87.91%	9.9%	14.9%
ResNet-101	80.27%	17.53%	0.37	83.37%	14.38%	17.11%

Table 17: *Mask R-CNN's training metrics for **category 2** dataset (image size is 128x128 px) for year 2020. No of epochs was 2k. Acc is classification accuracy (including background). AccFG is only foreground object classification (without background). Mask Acc is mask head's accuracy. Mask FN & Mask FP are mask head's false negative and false positive, respectively*

Backbone	Acc	AccFG	Mask head's loss	Mask Acc	Mask FN	Mask FP
ResNet-50	94.89%	0.3%	0.39	81.32%	14.60%	22.60%
ResNet-101	96.38%	0.2%	0.41	81.63%	12.01%	25.04%

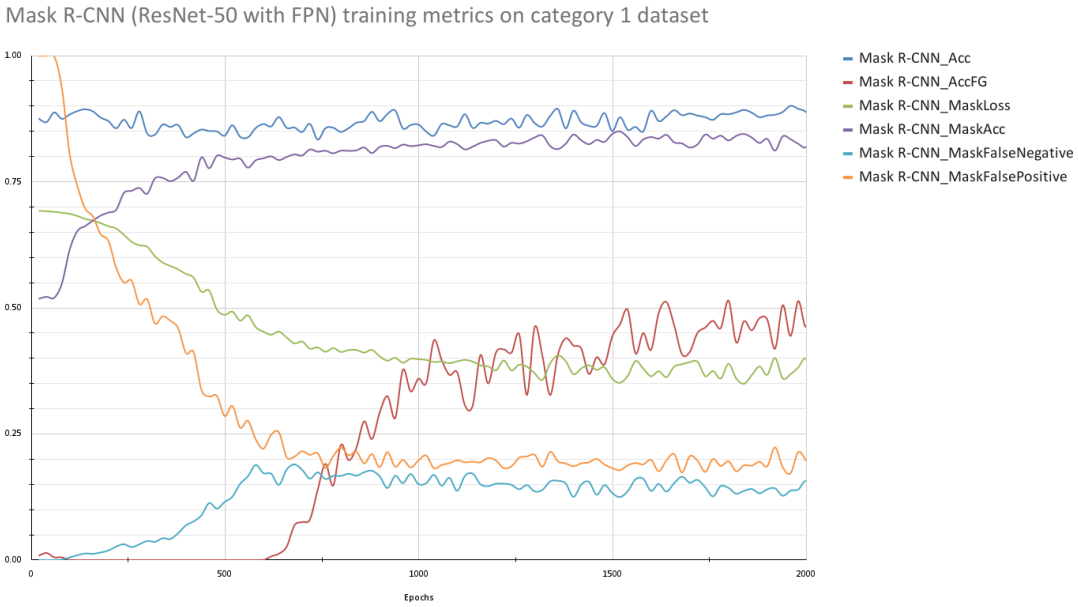
Table 18: *Mask R-CNN's training metrics for **category 3** dataset (image size is 128x128 px) for year 2020. No of epochs was 2k. Acc is classification accuracy (including background). AccFG is only foreground object classification (without background). Mask Acc is mask head's accuracy. Mask FN & Mask FP are mask head's false negative and false positive, respectively*

Backbone	Acc	AccFG	Mask head's loss	Mask Acc	Mask FN	Mask FP
ResNet-50	88.83%	0.02%	0.38	82.99%	13.05%	20.65%
ResNet-101	88.35%	0.02%	0.37	83.61%	14.27%	18.46%

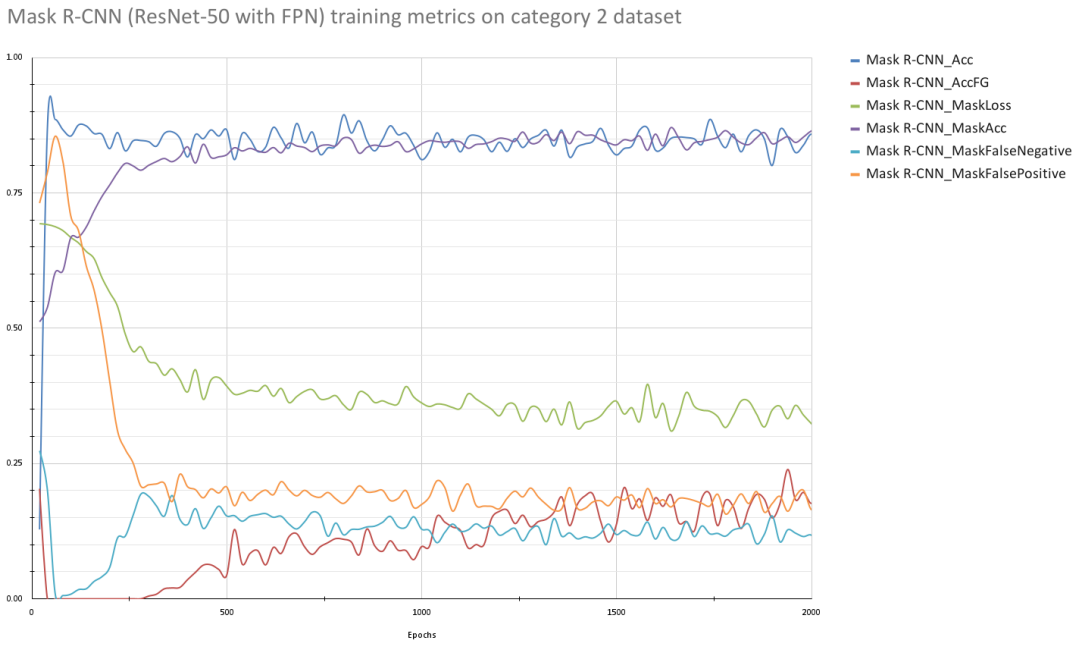
Table 19: *Mask R-CNN's training metrics for **category 4 dataset** (image size is 128x128 px) for year 2020. No of epochs was 2k. Acc is classification accuracy (including background). AccFG is only foreground object classification (without background). Mask Acc is mask head's accuracy. Mask FN & Mask FP are mask head's false negative and false positive, respectively*

Backbone	Acc	AccFG	Mask head's loss	Mask Acc	Mask FN	Mask FP
ResNet-50	91.86%	0.05%	0.37	83.45%	12.04%	20.05%
ResNet-101	90.24%	0.06%	0.35	84.39%	14.20%	16.57%

Table 20: *Mask R-CNN's training metrics for* ***category 5 dataset*** *(image size is 128x128 px) for year 2020. No of epochs was 2k. Acc is classification accuracy (including background). AccFG is only foreground object classification (without background). Mask Acc is mask head's accuracy. Mask FN & Mask FP are mask head's false negative and false positive, respectively*

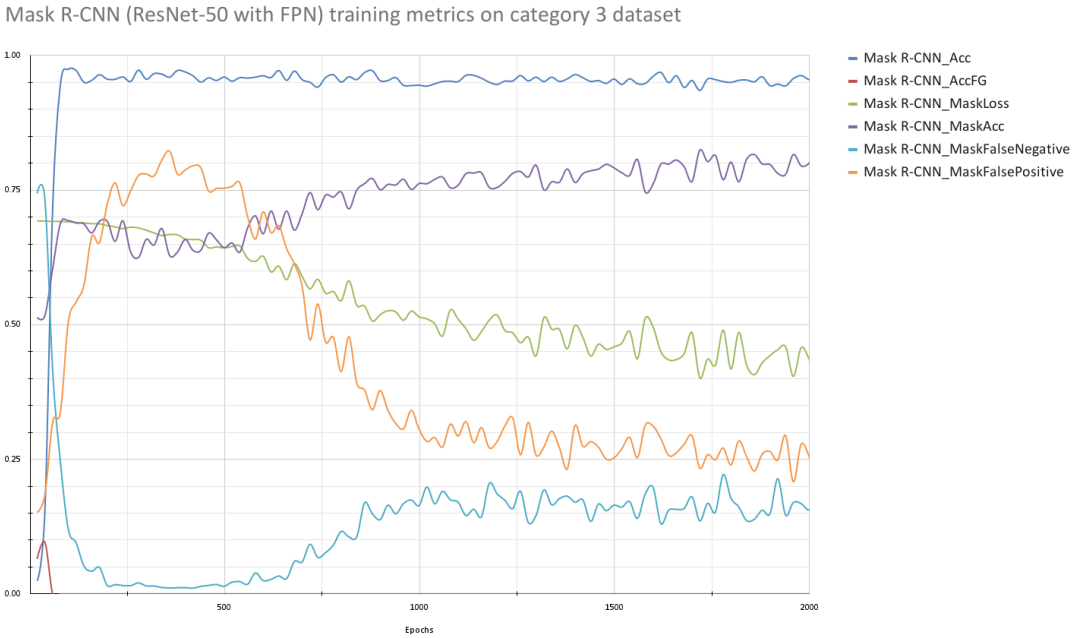


(a) *Mask R-CNN (ResNet-50 with FPN) on category 1 dataset*

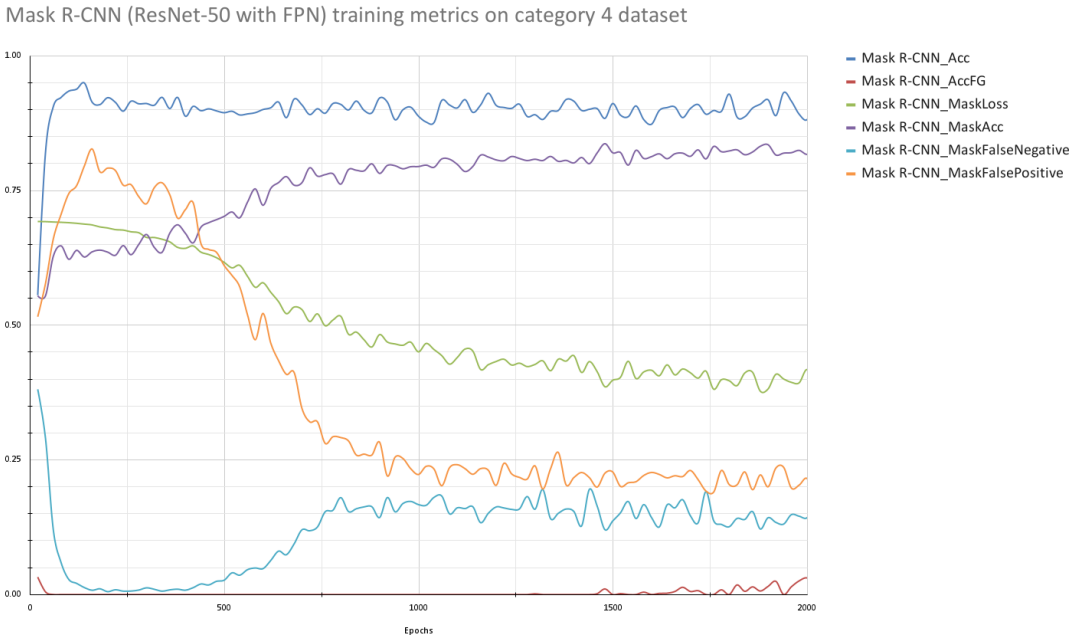


(b) *Mask R-CNN (ResNet-50 with FPN) on category 2 dataset*

Figure 55: *R-CNN's training metrics for image size (128x128) for year 2020. No of epochs was 2k. Mask R-CNN_ Acc is classification accuracy (including background). Mask R-CNN_ AccFG is only foreground object classification (without background). Mask R-CNN_ Maskloss is mask head's loss (L_{mask}). Mask R-CNN_ MaskAcc is mask head's accuracy. Mask R-CNN_ MaskFalseNegative is false negative for mask head. Mask R-CNN_ MaskFalsePositive is false positive for mask head.*

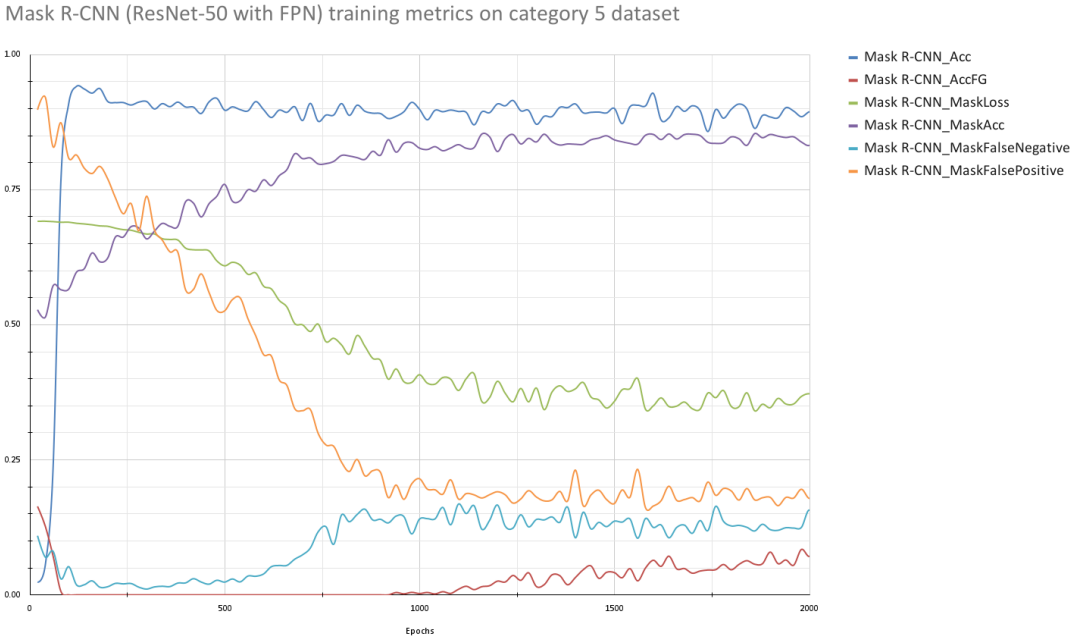


(a) *Mask R-CNN (ResNet-50 with FPN) on category 3 dataset*

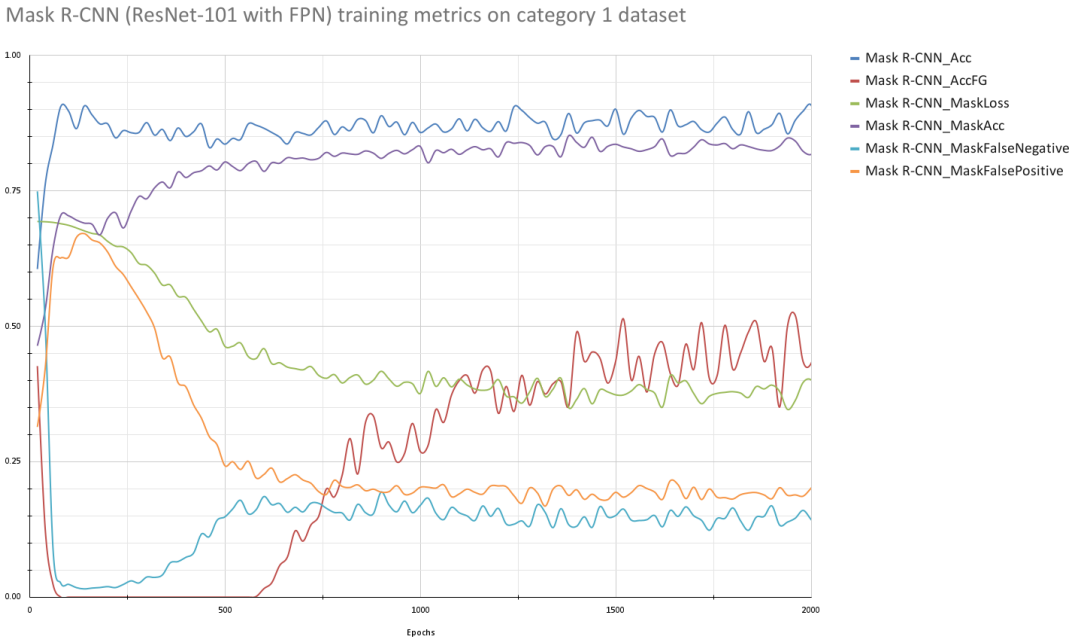


(b) *Mask R-CNN (ResNet-50 with FPN) on category 4 dataset*

Figure 56: *Mask R-CNN's training metrics for image size (128x128) for year 2020. No of epochs was 2k. Mask R-CNN_ Acc is classification accuracy (including background). Mask R-CNN_ AccFG is only foreground object classification (without background). Mask R-CNN_ Maskloss is mask head's loss (L_{mask}). Mask R-CNN_ MaskAcc is mask head's accuracy. Mask R-CNN_ MaskFalseNegative is false negative for mask head. Mask R-CNN_ MaskFalsePositive is false positive for mask head.*

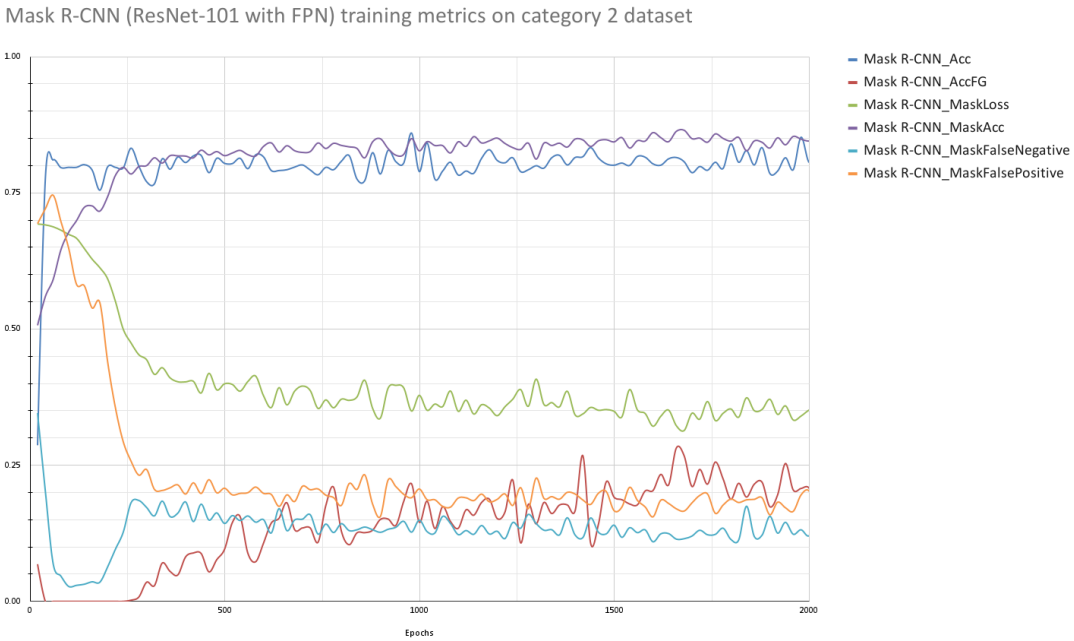


(a) *Mask R-CNN (ResNet-50 with FPN) on category 5 dataset*

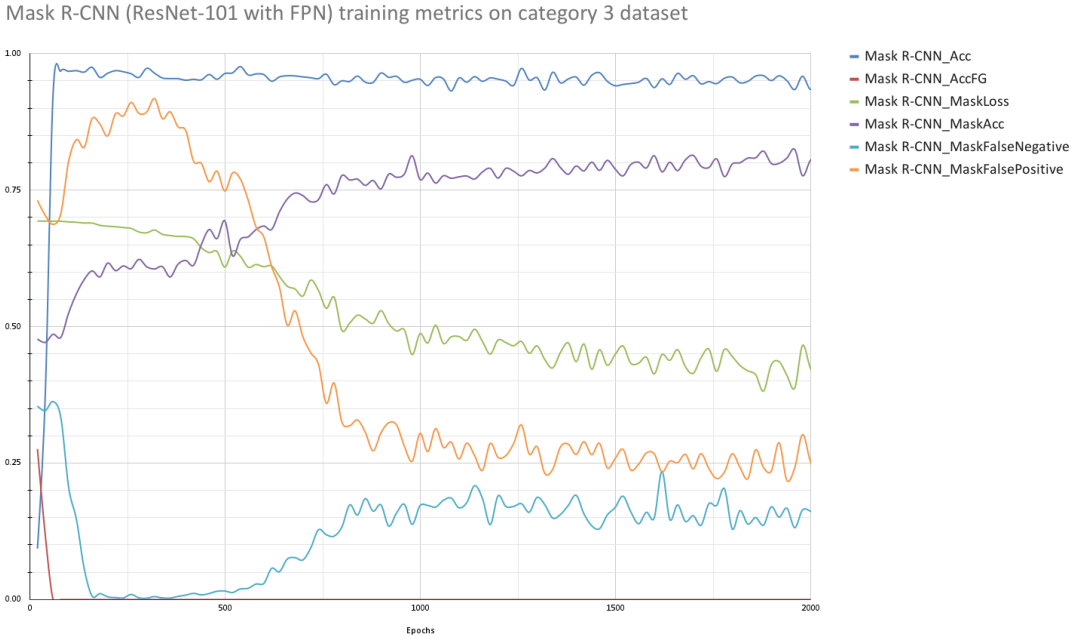


(b) *Mask R-CNN (ResNet-101 with FPN) on category 1 dataset*

Figure 57: *Mask R-CNN's training metrics for image size (128x128) for year 2020. No of epochs was 2k. Mask R-CNN_ Acc is classification accuracy (including background). Mask R-CNN_ AccFG is only foreground object classification (without background). Mask R-CNN_ Maskloss is mask head's loss (L_{mask}). Mask R-CNN_ MaskAcc is mask head's accuracy. Mask R-CNN_ MaskFalseNegative is false negative for mask head. Mask R-CNN_ MaskFalsePositive is false positive for mask head.*

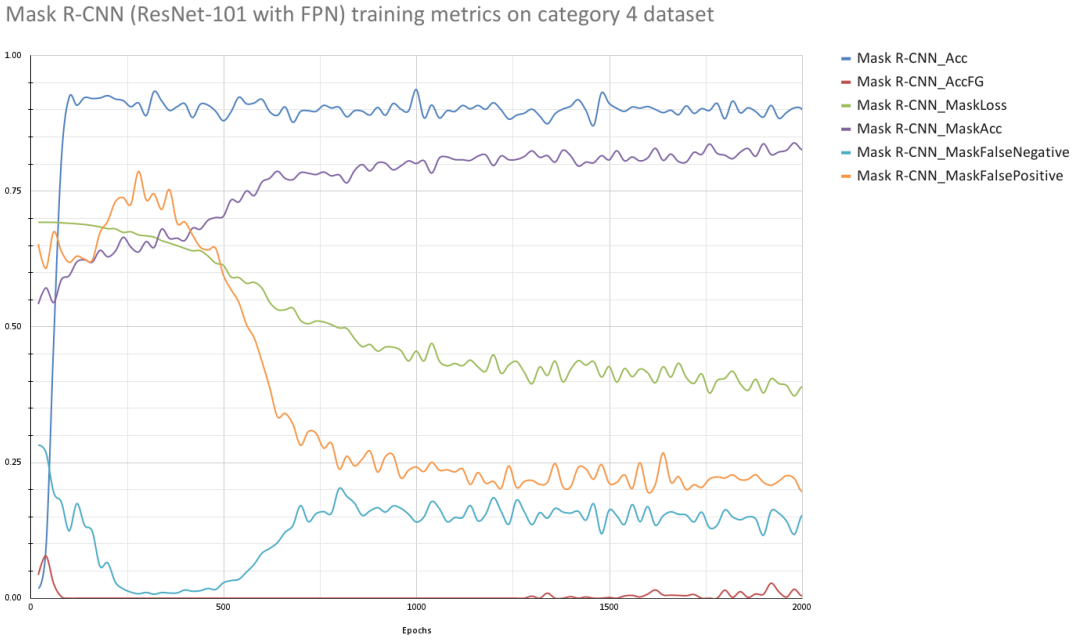


(a) *Mask R-CNN (ResNet-101 with FPN) on category 2 dataset*

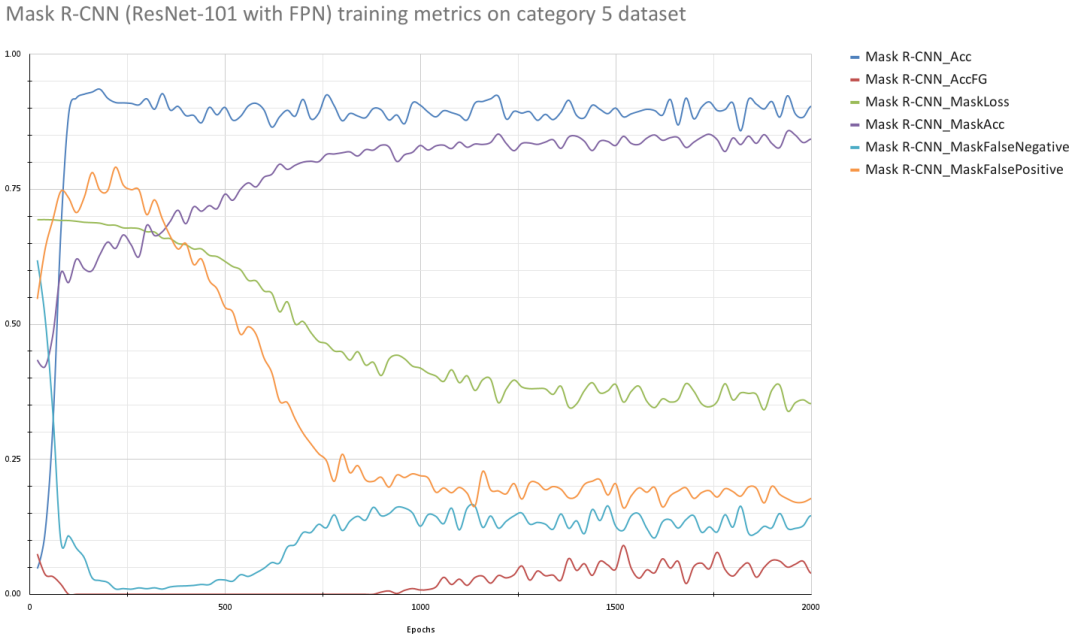


(b) Mask R-CNN (ResNet-101 with FPN) on category 3 dataset

Figure 58: *Mask R-CNN's training metrics for image size (128x128) for year 2020. No of epochs was 2k. Mask R-CNN_ Acc is classification accuracy (including background). Mask R-CNN_ AccFG is only foreground object classification (without background). Mask R-CNN_ Maskloss is mask head's loss (L_{mask}). Mask R-CNN_ MaskAcc is mask head's accuracy. Mask R-CNN_ MaskFalseNegative is false negative for mask head. Mask R-CNN_ MaskFalsePositive is false positive for mask head.*



(a) *Mask R-CNN (ResNet-101 with FPN) on category 4 dataset*



(b) *Mask R-CNN (ResNet-101 with FPN) on category 5 dataset*

Figure 59: *Mask R-CNN's training metrics for image size (128x128) for year 2020. No of epochs was 2k. Mask R-CNN_ Acc is classification accuracy (including background). Mask R-CNN_ AccFG is only foreground object classification (without background). Mask R-CNN_ Maskloss is mask head's loss (L_{mask}). Mask R-CNN_ MaskAcc is mask head's accuracy. Mask R-CNN_ MaskFalseNegative is false negative for mask head. Mask R-CNN_ MaskFalsePositive is false positive for mask head.*

Appendix 4 - FCIS Results

(a)

(b)

Figure 60: *FCIS's training metrics for image size (128x128) for year 2020. No of epochs was 10. FCISAcc is classification accuracy (including background). FCISAccFG is only foreground object classification (without background). Mask FCISMaskloss is mask loss from pixel-wise softmax over ROI inside/outside maps.*

(a)

(b)

Figure 61: *FCIS's training metrics for image size (128x128) for year 2020. No of epochs was 10. FCISAcc is classification accuracy (including background). FCISAccFG is only foreground object classification (without background). Mask FCISMaskloss is mask loss from pixel-wise softmax over ROI inside/outside maps.*

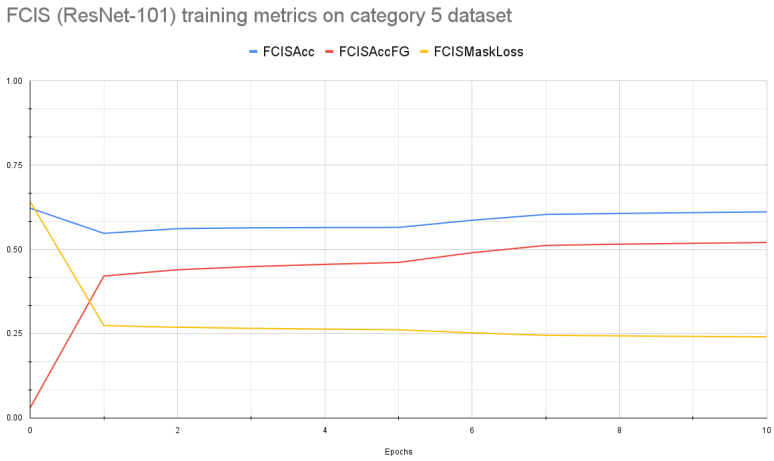


Figure 62: *FCIS's training metrics for image size (128x128) for year 2020. No of epochs was 10. FCISAcc is classification accuracy (including background). FCISAccFG is only foreground object classification (without background). Mask FCISMaskloss is mask loss from pixel-wise softmax over ROI inside/outside maps.*