Technische
Hochschule
Rosenheim

Fakultät für Ingenieurwissenschaften

Masterstudiengang Ingenieurwissenschaften (Mechatronik)

Development of a Neural Network for the automated wood class identification using an RGB-LIDAR camera

Master Project

by

Abdulla Alshaer (987749)

Jerardh Josekutty (1003839)

Submitted on: 13.02.2023
Guided by: Prof. Dr. Ing. Martin Versen

Acknowledgement

We take this opportunity to wish to express our deepest gratitude to, Prof. Dr.-Ing. Martin Versen, who gave us all the significant inputs and assisted us time and again to complete this project successfully. We would also like to express our gratitude towards Ms. Nina Leiter, Mr. Maximilian Wohlschlaeger, and Mr. Maximilian Dietlmeier for all the help and support to complete this project successfully. We also extend our gratitude to our university, Technische Hochschule Rosenheim for providing us with some of the best resources available. Last, but not the least, a word of thanks to all our friends and family for their never-ending encouragement and support.

Rosenheim, 13.02.2023

Abdulla Alshaer,
Jerardh Josekutty

Declaration of Originality

We declare that, We have authored this master project independently, that We have not used other than the declared sources / resources, and that We have explicitly marked all material which has been quoted either literally or by content from the used sources.

Rosenheim, 13.02.2023

Abdulla Alshaer,
Jerardh Josekutty

Abstract

Wood is becoming more widely recognized as a valuable renewable resource. With the increasing demand for wood and the depletion of natural forests, it is essential to find ways to use wood sustainably and efficiently. Recycling wood conserves natural resources, reduces the amount of waste in landfills, and helps reduce greenhouse gas emissions. Furthermore, wood can be recycled and repurposed into a variety of products, from furniture to construction materials. In order to recycle the wood, It must be classified into individual classes A1, A2, and A3. This project uses Mask-RCNN to develop a neural network to classify wood using an RGB-LIDAR camera. An Instance Segmentation model is developed for the classification of the wood. Using Mask-RCNN, waste wood classes A1-A3 were identified with mean Average Precision of 65%.

Keywords: Mask-RCNN, RGB-LIDAR camera, Neural network, Instance Segmentation, mean Average Precision

Abbreviations

ANN	Artificial Neural Networks
Mask-RCNN	Mask Region-Based Convolutional Neural Network
CNN	Convolutional Neural Network
RGB	Red Green Blue
LIDAR	Light Detection and Ranging
FD-FLIM	Frequency Domain Fluorescence Lifetime Imaging Microscopy
RNN	Recurrent Neural Network
NLP	Natural Language Processing
FC	Fully connected layer
ReLU	Rectified Linear Unit
ROI	Region of interest
COCO	Common Objects in Context
HED	Holistically-Nested Edge Detection
SGD	Stochastic Gradient Descent
TP	True positive
FP	False Positives
TN	True Negative
FN	False Negative
AP	Average Precision
mAP	mean Average Precision
mAR	mean Average Recall
JSON	JavaScript Object Notation
fps	frames per second

Contents

List of Figures

List of Tables

1 Introduction

Wood is considered humankind's very first source of energy. Today it is still the most important single source of renewable energy providing about 6% of the global total primary energy supply. Wood is considered a renewable source of energy because it can be regrown through sustainable forestry practices [1]. Using wood as a source of energy reduces dependence on non-renewable energy sources and contributes to a cleaner and more sustainable future. Over one year, a mature tree will take up about 22 kilograms of carbon dioxide from the atmosphere, and in exchange release oxygen. Each year, 1.3 million trees are estimated to remove more than 2500 tonnes of pollutants from the air [2]. Germany is the largest producer of timber in Europe. The forest in Germany is now managed in a more natural way than ever before, more wood grows continuously than is used. Germany currently has a wood reserve of 3.7 billion cubic meters [3]. Due to the high usage of wood products, the amount of waste wood produced annually is increasing. The volume of waste wood produced in Germany is between 7.7 to 10 million tonnes [4]. Hence recycling wood waste will have both economic and environmental benefits. The collective term "waste wood" is used to denote all woods and wood-based products whose end of life as a product has been reached and which therefore fall under the definition of waste [5]. German law stipulates that in order to be recycled, wood must be separated into classes A1-A4. Separation of wood into individual classes requires expert knowledge and is a time-consuming process.

The identification of wood species and waste wood category is currently done by utilizing the Frequency-Domain Fluorescence Lifetime Imaging Microscopy (FD-FLIM), an approach based on their distinctive fluorescence decay periods. This method uses metadata from nd2 files and Phasorplots to separate wood class A1 from the rest of the samples. However, FD-FLIM is not yet suitable when it comes to real-time applications, as it requires more time to take measurements and pre-process the data [6].

The main objective of this project is to use a faster real-time Instance Segmentation model to classify waste wood into classes A1-A3. The model is trained using labeled wood sample data collected using an Intel Realsense LIDAR camera. Instance segmentation is done using Mask-RCNN architecture.

2 Convolutional neural networks

Neural networks are a subset of machine learning and the core component of a deep learning algorithm. The structure of the neural network is similar to the neurons of the brain, thus the name neural network. As shown in Figure 2.1, the neural network consists of node layers, i.e., an input layer, one or more hidden layers, and an output layer. Each individual part is called a node. A node can be considered a linear regression with its own input, weights, bias, and output [8]. The output value of the node should be above a specific threshold (bias) value to activate the node and transfer the data to the next layer. There are different types of neural networks. For example, Artificial neural networks (ANN) are used for data science applications, Recurrent neural networks (RNN) are used for natural language processing (NLP) and speech recognition, and Convolutional neural networks (CNN) are used for computer vision and image processing tasks.

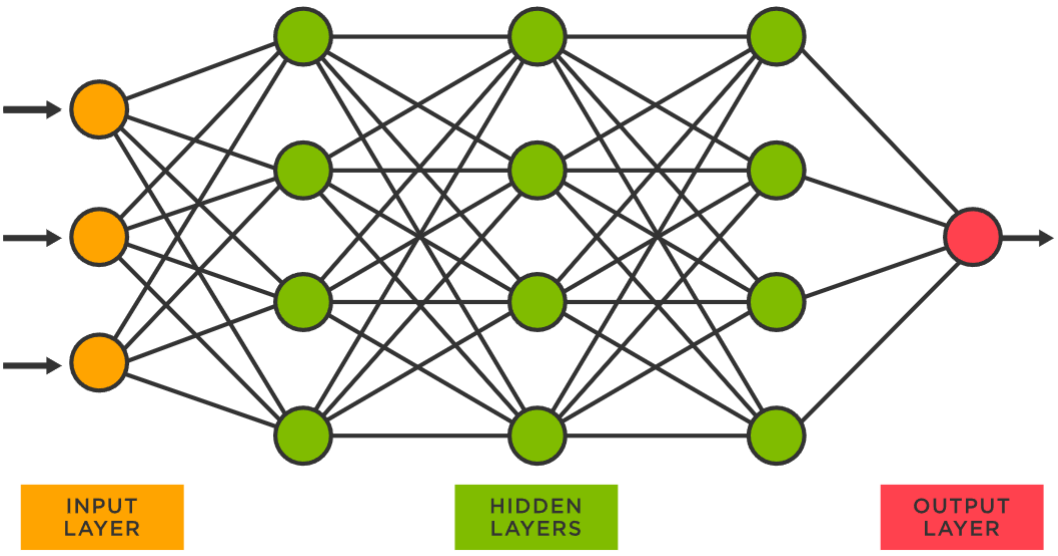


Figure 2.1 Structure of a neural network [7]

Prior to CNNs, manual, time-consuming feature extraction methods were used to identify objects in images. However, convolutional neural networks now provide a more scalable approach to image classification and object recognition tasks, leveraging principles from linear algebra, specifically matrix multiplication, to identify patterns within an image [8].

2.1 How does a convolutional neural network work ?

A convolutional neural network has three main layers, namely the convolution layer, the pooling layer, and a fully connected (FC) layer. The working of CNN can be explained in the following steps.

1. Convolution: Convolution is an effective way of feature extraction and small filters (kernels) are convolved over the input image to obtain the feature map.
2. Non-linear activation: A non-linear activation function (e.g. Leaky ReLU) is applied to introduce non-linearity in the feature maps obtained from the previous step.
3. Pooling: In this step, the size of feature maps is reduced to decrease computation and remove the noise.
4. Flattening: The feature maps after pooling is converted to a 1D vector to give as input to the fully connected neural network.
5. Output layer: This is the final layer that gives the output such as the class score or the label.
6. Loss function: is used to predict the difference between the prediction and the true label (e.g. Binary cross-entropy).
7. Back-propagation: The loss value is back-propagated and the weights and biases are updated to reduce the loss. All these steps are repeated until the loss is reduced to a minimum value.

In this project, we are using computer vision and image processing to capture the image and then classify it using deep neural networks.
In computer vision, there are mainly four approaches for classifying images.

- Image classification: is used to classify the image to a particular class (e.g. predicting if an image belongs to the cat or dog category).
- Object detection: can be used to detect multiple objects within an image. Each object is represented using a bounding box.
- Semantic segmentation: is used to recognize a collection of pixels that belongs to the same class. Using semantic segmentation, each individual item cannot be separately identified.
- Instance segmentation: This method treats multiple objects of the same class as distinct individual instances. This will provide both a bounding box and a mask representing the object pixels.

2.2 Mask-RCNN for instance segmentation

Mask region-based convolution neural network is a model developed by Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick as a part of Facebook AI Research (FAIR) [9]. Instance segmentation using Mask-RCNN is a state of the art model for obtaining the exact pixel locations for each object within the image. The Mask-RCNN model is built on top of Faster RCNN (Faster region-based convolutional network) (Figure 2.2). Faster-RCNN by itself can predict the bounding boxes. The second part of the Mask-RCNN is a Mask Head, which predicts the masks. The network is trained using a large number of labeled datasets. The labels include information about the bounding box, masks, etc. During the working of the Mask-RCNN model, it takes an image or video as an input (pre-recorded or live feed) and gives a set of bounding boxes, class labels, and masks as output. It has applications in various computer vision tasks, such as pose estimation, object tracking autonomous driving, etc.

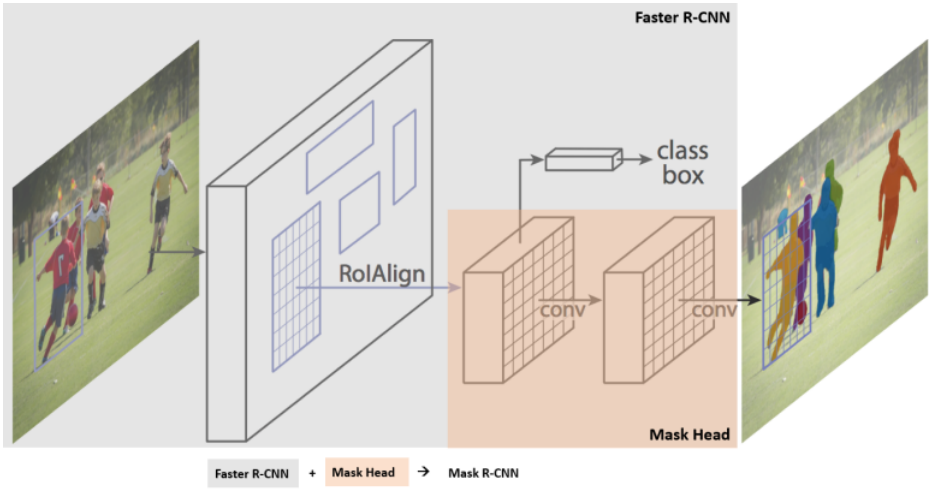


Figure 2.2 Structure of Mask-RCNN [9]

2.3 Working of Mask-RCNN

The working of Mask-RCNN can be explained in three stages (Figure.2.3):

- Stage 1: The first stage consists of two networks. A CNN part generates the feature map. The region proposal network (RPN) gives information about the bounding box and the object class. This step gives rough information about the location of different objects within the image.
- Stage 2: In this stage ROI (region of interest) pooling takes place. The ROI pooling layer divides each ROI into sub-regions. This new sub-region should have an equal size (e.g. 5x5). Various pooling operations such as maximum, minimum, average, etc are applied. This step is done to make the feature map independent

of size since each ROI have a different size and aspect ratio. The resulting feature map is passed to the next stage.

- Stage 3: In this stage, two things take place. Firstly, a fully connected layer (FC) predicts the classes and the bounding box. Secondly, a pair of convolution networks predict the mask. As a result, we get an output image with an instant segmented image along with its class and boundary box.

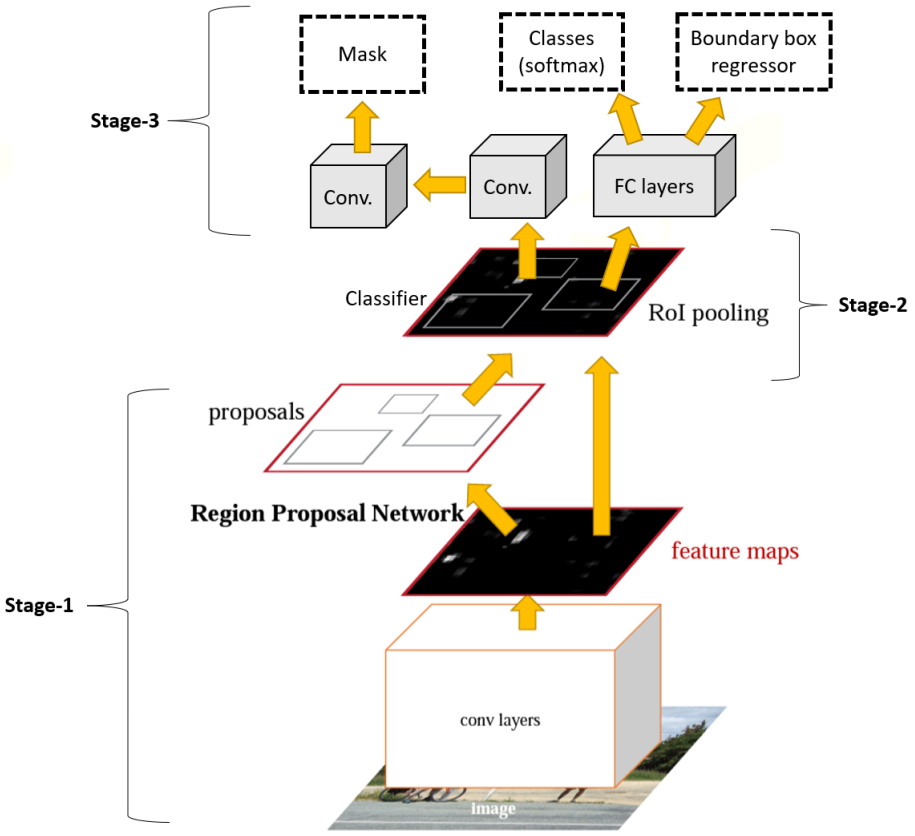


Figure 2.3 Working of Mask-RCNN [10]

2.4 Transfer learning

Transfer learning is a machine learning technique where a pre-trained model is fine-tuned with a new dataset [11]. This technique has multiple advantages. Transfer learning can be used when there is less training data or to reduce the training time. The main idea is to use the features from the original task as a starting point for the neural network instead of starting from scratch. In this project, pre-trained weights on the COCO (Common Objects in Context) are used. These weights were developed by the

matterpot team [12]. The COCO dataset contains over 330,000 images, each annotated with 80 object categories and 5 captions, and is commonly used as a benchmark for computer vision models.

3 Experimental Setup and Dataset Preparation

This chapter deals with the types of wood samples and how the data is named. It also explains the working of the data collecting apparatus, the labeling process and the preparation of training data for the Mask-RCNN model. Due to the limited number of A4 samples, wood classes A1, A2, and A3 are only being used.

3.1 Types of wood classes

As per German law, the waste wood categories are:

1. Category A1: Waste wood in its natural form or wood waste that is only mechanically processed, during use, which has the least amount of contaminants that are harmful to the wood.

2. Category A2: Coated, bonded, painted, lacquered, or otherwise treated waste wood with no halogen-organic compounds in the coating and with no wood preservatives.

3. Category A3: Waste wood with halogen-organic compounds in the coating without wood preservatives.

4. Category A4: Waste wood treated with wood preservatives, such as waste wood from damaged structures, vineyard poles, railway sleepers, utility poles, hop poles, as well as other waste wood that cannot be assigned to waste wood categories A1, A2, or A3 due to its pollutant load, with the exception of PCB waste wood [13].

3.2 Wood sample nomenclature

Wood samples for this project were provided by the post-consumer wood utilization plant "Stark GmbH" in Lindau/Bodensee. All the samples are labeled to their corresponding classes using visual inspection. Both sides of the samples are analysed and in some cases, each side of the same wood sample can belong to different classes. The naming of the sample is done in the following format.

Species Class Sample Surface (e.g. F AII S002 F1)

'Species' representations are Beech (B), Spruce (F), Larch (L), or blank (if species is not determined). 'Class' represent the specific wood class the sample belongs to and is written as AI, AII, AIII, or AIV (Roman numerals are used in the nomenclature whereas throughout this project, the naming convention A1, A2, A3, and A4 is being used). 'Sample' represents the specific count (e.g. S001). 'Surface' represents the face of the sample (e.g. F01, F02).

3.3 Experimental setup

The camera used for data collection is *Intel Realsense L515*. It is a RGB-LIDAR camera . A schematic representation of the camera system is shown in Figure 3.1. The camera has a Minimum depth distance (Min-Z) of approximately 25 cm and a Maximum depth distance (Max-Z) of 9 m [14]. Hence the depth sensor of the camera wouldn't work if the distance between the sample and the camera is less than 25 cm. In this project, the size of the wood samples is relatively small, and increasing the camera height will reduce the quality of the image. As shown in Figure 3.2, the height of the camera can be adjusted with the help of the adjustable frame. In this project, the camera was kept at a height of 18 cm, hence depth value was not used for training the model (only RGB data was used for training).
The *Intel Realsense L515* camera specifications are:

1. RGB frame resolution: 1920 × 1080
2. RGB frame rate: 30 fps
3. RGB sensor technology: Rolling Shutter
4. RGB sensor resolution: 2 MP

Wood samples are placed on a metallic surface. The camera is attached to an adjustable cantilever beam that is above the wood sample (Figure 3.2). Two overhead lambs are used as the light source. The working of the camera is controlled by a custom build python script. The images collected are named on the basis of their wood class. The data set has 416 images with 180 images from A1, 183 images from A2, and 53 images from A3. The images are stored in JPEG format and should undergo multiple processing steps before being fed into the neural network.

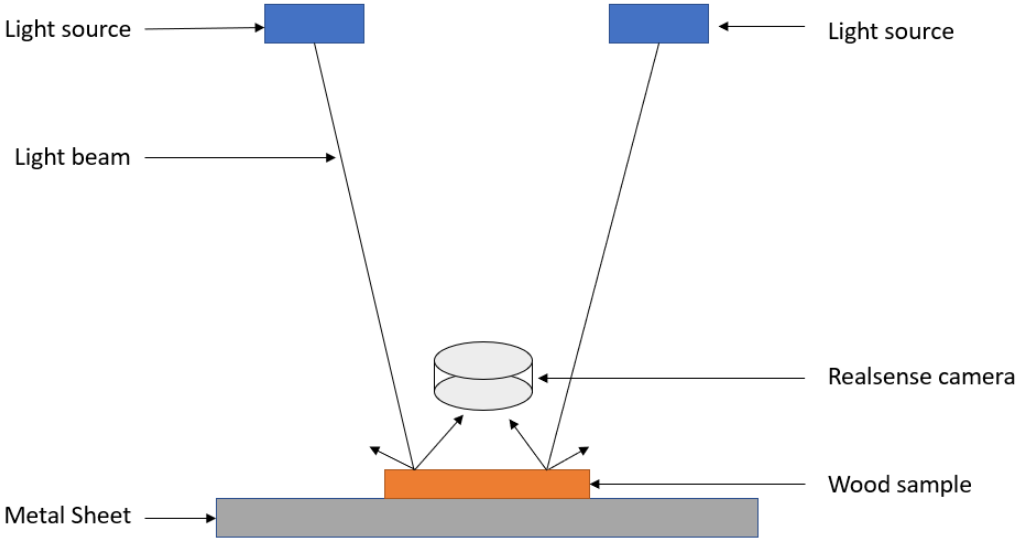


Figure 3.1 Schematic representation of the experimental setup

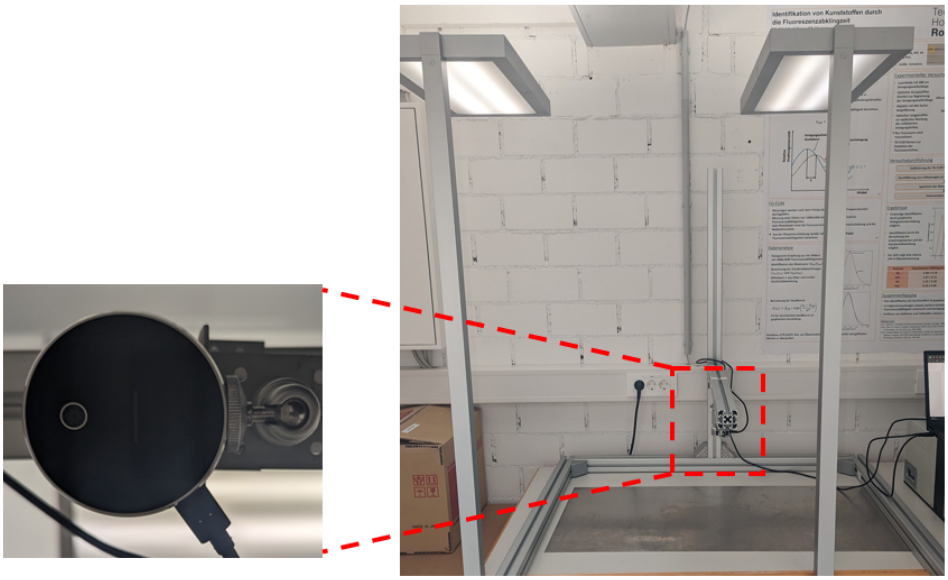


Figure 3.2 Experimental setup

4 Image processing and preparation of training data

The input data of the Mask-RCNN model includes an image file and a label file in JSON format which contain information about the size, shape, number of objects in the image, class of each object, coordinates of the mask of each object, and path of the image. There are two methods to generate the label. A manual method, which requires a large amount of time by using tools like *Labelme* or *Labelstudio*. The advantage of the manual labeling method is that the quality of the generated masks is high. The second one is an automated method using image-processing tools. This project uses the automated method for label generation. The algorithm used is Holistically-Nested Edge Detection (HED).

4.1 Holistically-Nested Edge Detection

Holistically-Nested Edge Detection is an algorithm based on deep learning and was developed in the year 2015. It uses CNN to predict the absence or presence of edges at each pixel in an image. It is better than conventional image processing tools like canny edge detection because unlike the Canny edge detector, which requires preprocessing steps, manual tuning of parameters, and often does not perform well on images captured using varying lighting conditions, Holistically-Nested Edge Detection seeks to create an end-to-end deep learning edge detector [15]. A pre-trained HED model developed by *OpenCV* is being used in this project. The goal is to generate a blob from the input image and feed it to the HED model. It will generate the edges of the input image. Then a binary mask is generated and multiple steps of averaging are done to smoothen out the mask. The steps involved are explained in the diagram Figure.4.1.

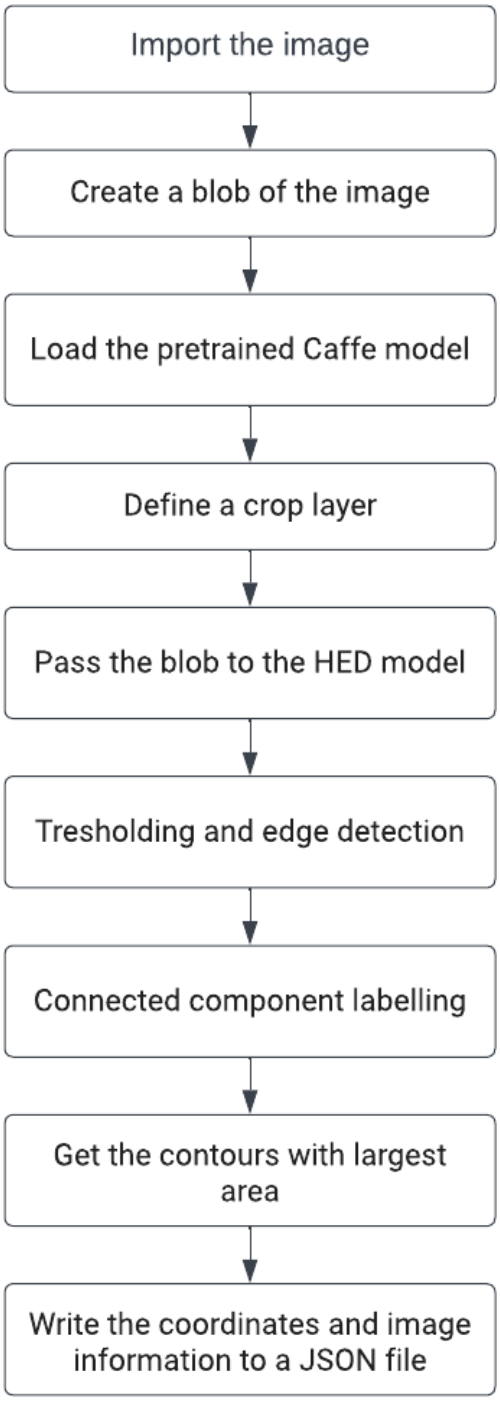
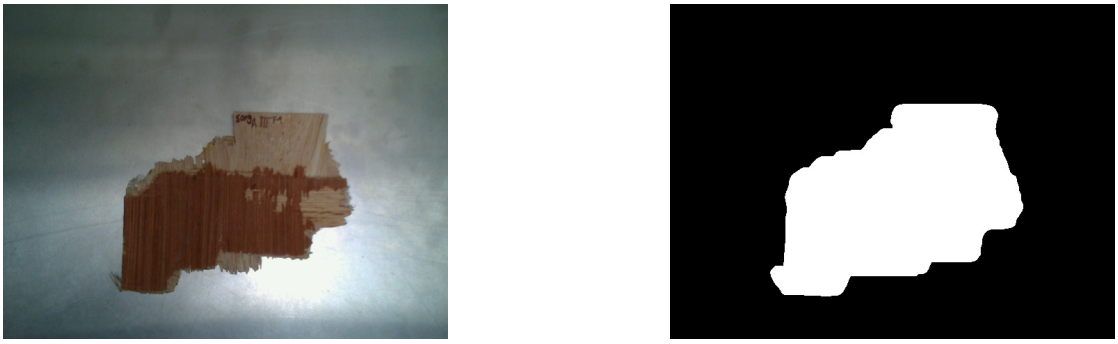


Figure 4.1 Steps involved in generating labels using HED



(a) Input image **(b)** Binary mask of the input image

Figure 4.2 Input image and output mask of HED model

Figure.4.2 shows how a binary mask is generated using the HED model. The HED model predicted the mask with an accuracy of 95 % for the whole wood dataset. This method is extremely fast and can generate labels of hundreds of wood samples within minutes. The disadvantage of this method is that the quality of the mask is low as compared to the manual labeling approach. Another disadvantage is that this particular model is used for wood samples, for a different sample set (e.g. plastic or food) the model has to be modified.

4.2 Data Augmentation

This is a method used for increasing the dataset size by artificially modifying the existing data. The main goal should be to avoid overfitting. This project uses a powerful tool called *Albumentation* for augmentation. The different augmentations done are listed below.

1. Horizontal Flipping: Flips the original image horizontally.
2. Vertical flipping: Flips the original image vertically.
3. Random Brightness: Randomly adjust the image brightness. A probability value is defined, which predicts how often the brightness should be adjusted.
4. Blurring: The images are blurred. The amount of blurring can be adjusted by defining a probability function.

The existing dataset was increased five times with augmentation techniques. During the augmentation process, both the original image and its label file are modified. This can be achieved by generating a bitmask (Figure.4.2b) with the help of the original label file. The mask, original image, and the coordinates for the bounding box are given to *Albumentation* pipeline as the input. The output generated is the new image, mask, and coordinates of the bounding box. The output of the *Albumentation* pipeline is written to a new JSON file and the augmented label file is generated.

5 Training and evaluation

This chapter discusses the training and evaluation process, involving splitting the data into training, validation, and testing sets. The data set was split in the ratio 60:20:20 for training, testing, and validation respectively. The training dataset is used to fit the model parameters, the validation dataset is used for hyperparameter tuning, and the testing data is used for model evaluation.

5.1 Training the Mask-RCNN model

The first step of training is to import the basic libraries. Some of the major libraries used are *keras* from *Tensorflow*, *pandas*, *numpy*, *matplotlib*, etc. The next step is to load the COCO dataset (explained in section 2.4). After this step the model architecture is created, here the pre-trained model architecture is used. The next step is to compile the model. The activation function used is *ReLU* (Rectified Linear Unit), because of its higher computational efficiency. *ReLU* is a non-linear activation function. It converts negative values to zero and positive values to themselves. As a result, the network can learn complex non-linear relationships between outputs and inputs. The optimizer used for the model is *Stochastic Gradient Descent (SGD)*. The function of the optimizer is to reduce the loss function. In *SGD*, the model parameters are adjusted after each mini-batch of data.

The number of epochs is a hyper-parameter that defines the number of times that the learning algorithm will see the entire training dataset. During each epoch, the parameters in the model are adjusted to minimize the loss function. The number of epochs has a significant role in the model performance. If the model is underfitting, the number of epochs is increased, and decreasing the number of epochs can prevent overfitting. The learning rate is the next parameter that should be defined. The learning rate determines the speed at which a model learns. If the learning rate is very high, the model may overshoot the optimal values. If the learning rate is very low, the training time will increase. The learning rate is initially set to 0.001.

5.2 Performance parameters

The confusion matrix is a table used to evaluate the performance of a model. It compares the actual value with the predicted value. The confusion matrix can be used to calculate several metrics to evaluate the model performance. This includes accuracy,

precision, recall, and F1-score. We get these values by interpreting the confusion chart, which is plotted based on the True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) values. The schematic diagram of the confusion matrix for a binary classification problem is shown in Figure 5.1.

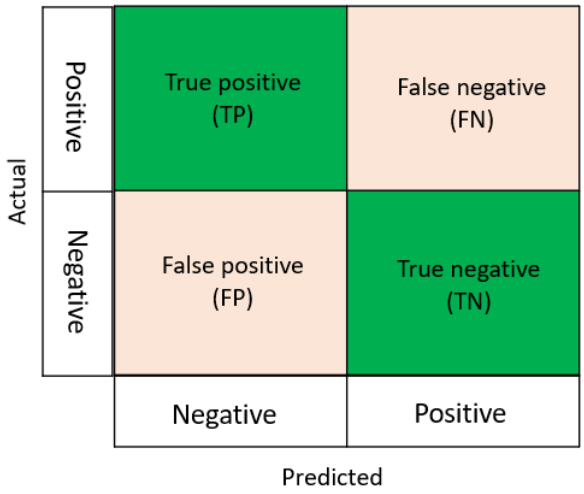


Figure 5.1 Confusion matrix

- True Positives (TP): The number of correct positive predictions made by the model.
- False Positives (FP): The number of incorrect positive predictions made by the model.
- True Negatives (TN): The number of correct negative predictions made by the model.
- False Negatives (FN): The number of incorrect negative predictions made by the model.

Accuracy is the ratio of correct predictions to total predictions. It is calculated by using Equation 5.1.

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{5.1}$$

Precision is the ratio of true positive predictions to total positive predictions and is calculated by Equation 5.2.

$$Precision = \frac{TP}{(TP + FP)} \tag{5.2}$$

Recall (Sensitivity or True Positive Rate) is the ratio of true positive predictions to total positive cases. It is calculated by the Equation 5.3.

$$Recall = \frac{TP}{(TP + FN)} \tag{5.3}$$

F1-score is the harmonic mean of precision and recall and is calculated as (Equation 5.4),

$$F1score = \frac{2 \times Precision \times Recall}{(Precision + Recall)} \tag{5.4}$$

Mean Average Precision (mAP) is the most common metric in computer vision tasks, such as instance segmentation, object detection, and image classification [16]. mAP is measured by taking the average of Average Precision (AP) over all detected classes. AP is calculated by computing the precision-recall curve and taking the area under the precision-recall curve. In the Equation 5.5, R refers to the total number of ground truth positives, n refers to the total number of documents you are interested in, P_k is the precision at cut-off, k is the ranked list of results and rel_k is a relevance function. The value $rel_k = 1$, if item k is relevant and $rel_k = 0$, if item k is not relevant.

$$AP = \frac{1}{R} \times \sum_{K}^{n} (P_k \times rel_k) \tag{5.5}$$

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{5.6}$$

The calculation of mAP is shown in Equation 5.6. N represents the total number of queries or samples, and AP_i is the average precision for the i'th query or sample. To sum up, mAP summarizes the performance of an information retrieval system by averaging the recall values across all queries, taking into account the number of relevant items retrieved by the system.
Mean Average Recall (mAR) is a performance parameter used for evaluating the ability of a system to retrieve a large fraction of the relevant items in the dataset. It is calculated by taking the average of the recall values retrieved by the system. The relationship between mAR, mAP, and F1-score is shown in Equation 5.7. mAR (Equation 5.8) is calculated by transforming the Equation 5.7.

$$F1score = \frac{2 \times mAP \times mAP}{mAP + mAR} \tag{5.7}$$

$$mAR = \frac{F1score \times mAP}{2mAP - F1score} \tag{5.8}$$

5.3 Performance of the augmented dataset

The dataset includes 656 augmented images for training and 150 for testing. After evaluating the model, mAP for training is 24% and mAP for testing is 22% and 28% for out of sample prediction. The overall performance with augmented data was really low. Even though augmentation is supposed to improve the accuracy of the model, for this particular case, it showed negative results. This is because the automatically generated masks had lower quality. The augmentation led to a cumulative decrease in quality. For further training, augmented images are not used.

5.4 Performance of the non-augmented data

The model was trained with the original dataset. 262 samples were used for training and 72 for testing. The mAP for training is 55% and mAP for testing is 56% and 54% for out of sample prediction. This dataset showed greater performance than augmented data. Further tuning is done on this dataset to increase the performance of the model.

5.5 Performance of the model at 50 epochs

Evaluating the model by increasing the number of epochs to 50 showed good results. From Table 5.1, we can see a significant increase in the mean average precision, recall values, and the F1-score.

Table 5.1 Performance parameters at 50 epochs

Dataset Type	mAP (%)	mAR (%)	F1score (%)
Training	60	81	69
Testing	47	74	58
Out of sample prediction	53	86	66

From the loss curve (see Figure 5.2), it is clear that the validation loss is steadily decreasing as the number of epochs increases. The shape of the curve suggests that the value of the learning rate is adequate. For further hyperparameter tuning, the learning rate is kept constant at 0.001.

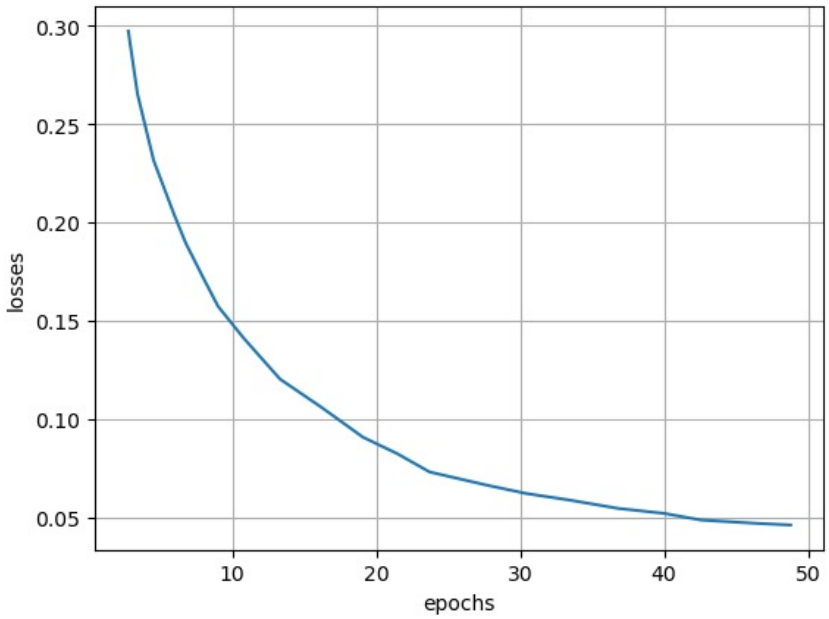


Figure 5.2 Validation loss curve for 50 epochs

5.6 Performance of the model at 70 epochs

The model is trained for 70 epochs. Values of mean average precision for training, validation, and out-of-sample prediction is improved but there is a significant reduction in recall values and the F1-score. Table.5.2 shows that out of sample prediction has increased from 53% to 64%. The validation loss curve shows a steadily decreasing trend (Figure 5.2).

Table 5.2 Performance parameters at 70 epochs

Dataset Type	mAP (%)	mAR (%)	F1score (%)
Training	69	39	49
Testing	65	32	43
Out of sample prediction	64	27	38

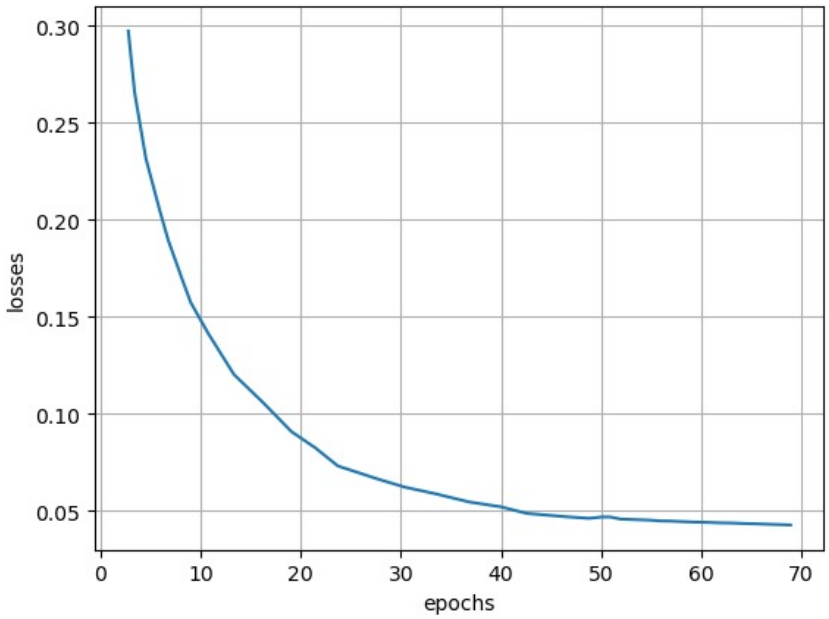


Figure 5.3 Validation loss curve for 70 epochs

5.7 Performance of the model at 90 epochs

From Table 5.3 it is evident that the mAP value has increased to 67%. This is a relatively good result for the Mask-RCNN model. The values of mAR and F1-score have also increased to 84% and 74% respectively. Evaluation of the loss curve shows no sign of overfitting (Figure 5.4).

Table 5.3 Performance parameters at 90 epochs

Dataset Type	mAP (%)	mAR (%)	F1score (%)
Training	73	85	79
Testing	65	81	72
Out of sample prediction	67	84	74

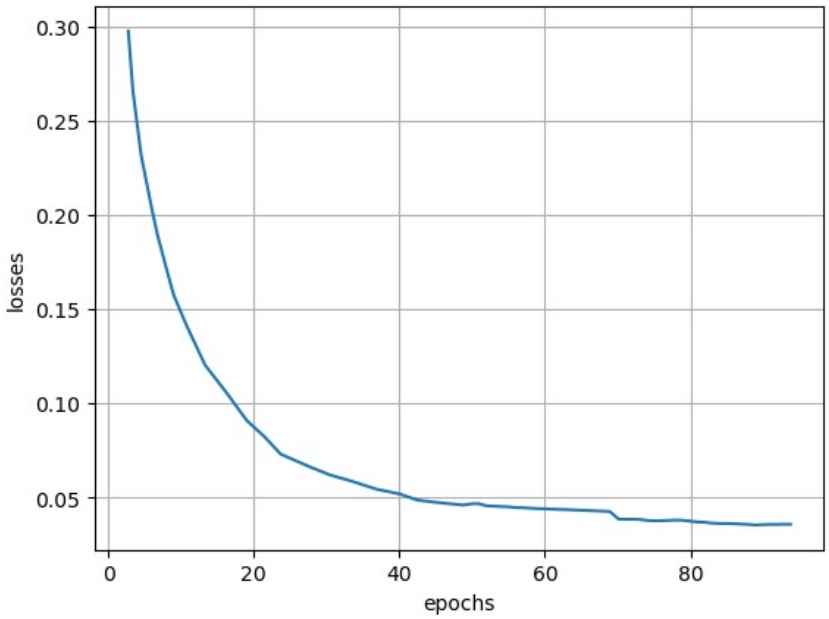


Figure 5.4 Validation loss curve for 90 epochs

Figure 5.5 An A2 sample accurately predicted as A2

An example of out of sample prediction is shown in Figure 5.5. At 90 epochs, the model is able to develop an accurate mask of the object along with its bounding box. In the example, the object class prediction is also accurate. Observation of the confusion matrix shows that 27 out of 35 A1 samples, 30 out of 36 A2 samples, and 5 out of 10 A3 samples were accurately predicted. According to the confusion matrix (Figure 5.6), A1 samples are predicted at an accuracy of 77.14%.

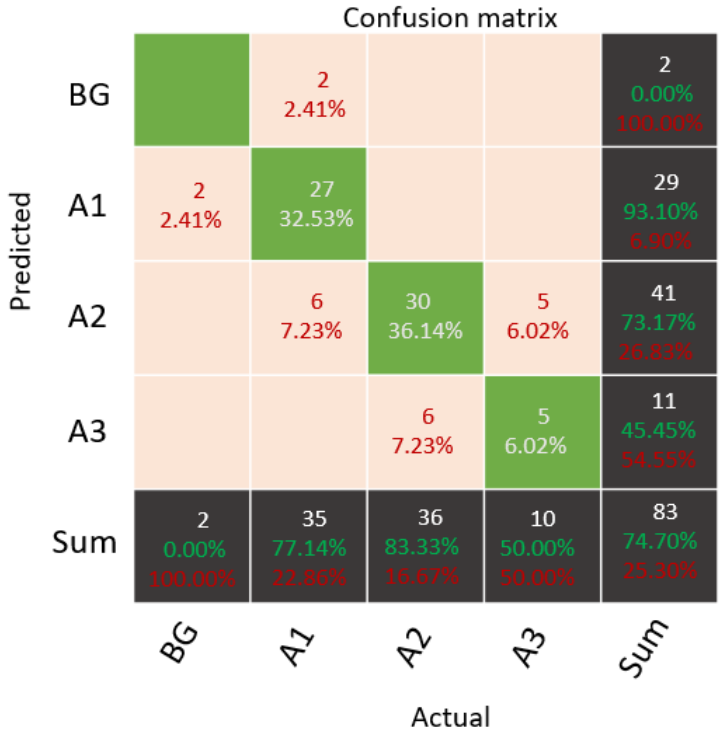


Figure 5.6 Confusion matrix for 90 epochs

5.8 Performance of the model at 95 epochs

At this stage, the number of epochs is increased to 95. It is clear from Table 5.4 that the mAP has decreased from 67% to 63%. The mAR and F1-score values don't have much change compared to the model at 90 epochs.

Table 5.4 Performance parameters at 95 epochs

Dataset Type	mAP (%)	mAR (%)	F1score (%)
Training	66	87	75
Testing	58	82	68
Out of sample prediction	63	85	72

The Figure.5.7 shows that the validation loss increases after 90 epochs. The confusion matrix for out of sample prediction is shown in Figure.5.8.

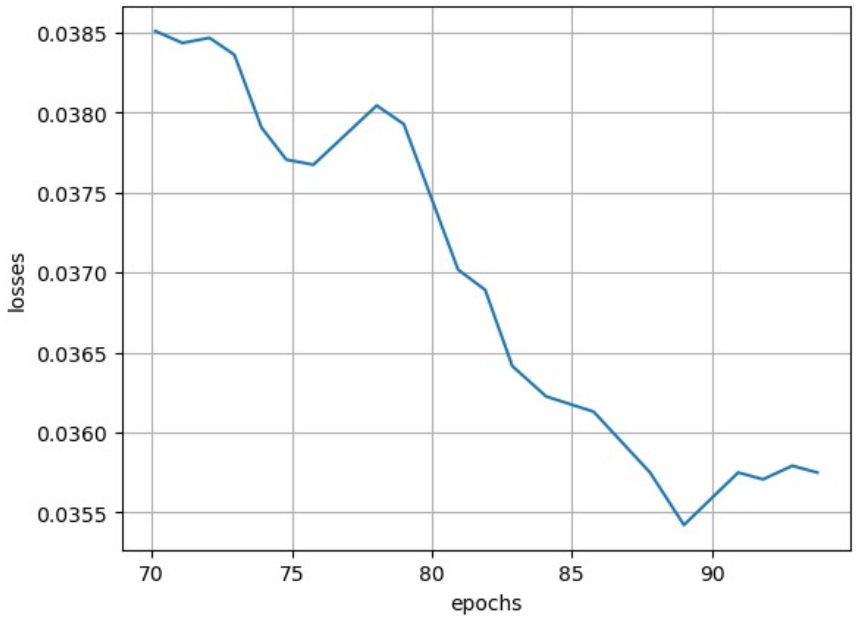


Figure 5.7 Validation loss curve for 95 epochs

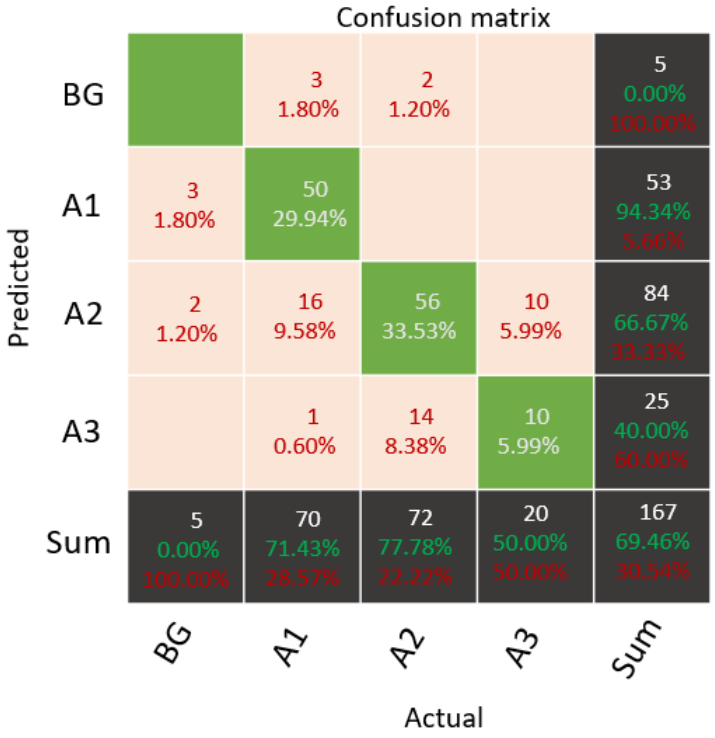


Figure 5.8 Confusion matrix for 95 epochs

5.9 Results

Analyzing the scaled version of the validation loss curve in Figure 5.7, there is an increase in loss value after 90 epochs. This shows signs of overfitting. This can be prevented by decreasing the number of epochs. From the confusion matrix (Figure 5.8), out of 70 A1 samples, the model can predict 50 samples as A1, 3 samples as background, 16 samples as A2, and 1 sample as A3. Accuracy for predicting the A1 samples has decreased from 77.14% at 90 epochs to 71.43% at 95 epochs.

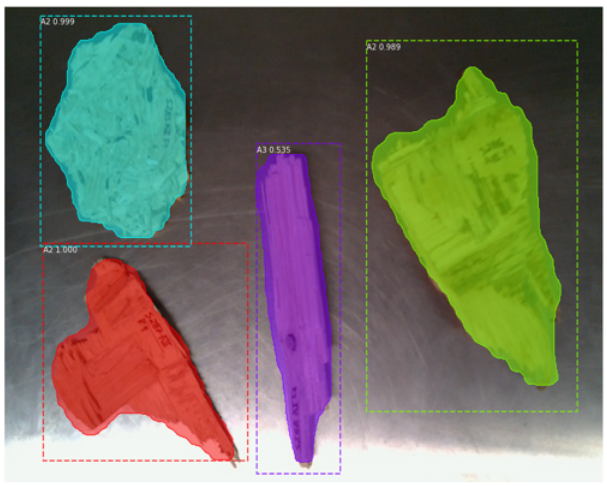


Figure 5.9 Instance segmentation of multiple wood samples using Mask-RCNN

After comparing the mAP, mAR, F1-score, validation loss curve, and the confusion matrix at 90 and 95 epochs, it is clear that the model at the 90 epochs is best in terms of all parameters. So this model is finalized for future use. Figure 5.9 is generated by testing the final model with multiple wood samples. The Mask-RCNN model was able to successfully identify multiple wood classes within the image. Out of 4 samples, 3 A2 samples were accurately classified as A2 and one A1 sample was misclassified as A2.

6 Conclusion

The primary goal of this project was to develop a neural network to classify A1, A2, and A3 waste wood classes using the RGB-LIDAR camera. This goal was achieved with the help of the Mask-RCNN model. The model showed the best accuracy at 90 epochs with an mAP of 65%, mAR of 81%, and F1-score of 74%. The Mask-RCNN model was able to predict AI samples at an accuracy of 77.14%.

The label generation using the HED model was significantly faster compared to the manual labeling process. HED model showed an accuracy of 95% but the quality of the generated masks was low. Augmentation of data didn't provide satisfying results, with an mAP of 24% for out of sample prediction. Training of the model was done using transfer learning. The pre-trained COCO weights were used for this. As a result, the training time was reduced.

To conclude, the Mask-RCNN model can be used for the classification of wood types A1, A2, and A3. However, the prediction by this model is not completely accurate. So further processing should be done using techniques like FD-FLIM to confirm the class of the wood. This model can be used as a pre-sorting tool. Combining both techniques will make the wood sorting process faster and more accurate.

Bibliography

[1] Food and Agriculture Organization of the United Nations *"Wood Energy"*, https://www.fao.org/forestry/energy/en/ (accessed on 7th February).

[2] European Environment Agency *"Trees help tackle climate change"*, https://www.eea.europa.eu/articles/forests-health-and-climate-change/key-facts/trees-help-tackle-climate-change(accessed on 7th February).

[3] Bundesministerium für Ernährung und Landwirtschaft *"Renewable raw material wood"*, https://www.bmel.de/DE/themen/wald/holz/nachwachsender-rohstoff-holz.html (accessed on 7th February).

[4] Federal association of waste wood processors and recyclers *"Wood waste volume"*, https://altholzverband.de/ (accessed on 10th February).

[5] Peter Meinlschmidt *"Technology for Wood and Natural Fiber-Based Materials"*, https://www.wki.fraunhofer.de/en/departments/hnt/profile/research-projects/recycling-of-waste-wood.html (accessed on 4th February).

[6] Aromal Somarajan Rajan, Maximilian Dietlmeier *"Development of a neural network for the identification of different substances using fluorescence lifetime imaging microscopy (FLIM)"*, Master project, TH Rosenheim, 2022.

[7] *"What is a Neural Network"*, https://www.tibco.com/reference-center/what-is-a-neural-network (accessed on 9th February).

[8] IBM *"Convolutional Neural Networks"*, https://www.ibm.com/topics/convolutional-neural-networks (accessed on 5th January).

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick *"Computer Vision and Pattern Recognition"*, https://doi.org/10.48550/arXiv.1703.06870.

[10] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun *"Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks"*, https://doi.org/10.48550/arXiv.1703.06870.

[11] *"Transfer learning"*, https://en.wikipedia.org/wiki/Transfer_learning (accessed on 9th February).

[12] *"Mask R-CNN for Object Detection and Segmentation"*, https://github.com/matterport/Mask_RCNN (accessed on 9th February).

[13] Magdalena Borzecka *"Recycling of Waste Wood"*, https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e5bf1792ce&appId=PPGMS.

[14] "Intel RealSense LiDAR Camera L515", https://www.intelrealsense.com/lidar-camera-l515/ (accessed on 4th February).

[15] Adrian Rosebrock, "Holistically-Nested Edge Detection with OpenCV and Deep Learning", https://pyimagesearch.com/2019/03/04/holistically-nested-edge-detection-with-opencv-and-deep-learning/ (accessed on 4th February).

[16] Shivy Yohanandan *"What is Mean Average Precision"*, https://towardsdatascience.com (accessed on 5th February).

A Appendix

Table A.1 Packages used for the project

Package Name	Version
Cython	0.29.23
dataclasses	0.8
google-cloud-core	1.7.1
google-cloud-storage	1.41.0
googleapis-common-protos	1.53.0
grpcio	1.48.1
h5py	2.10.0
imagesize	1.4.1
imgaug	0.4.0
imgviz	1.6.2
immutables	0.19
importlib-metadata	4.8.3
importlib-resources	5.4.0
ipykernel	5.5.6
ipyparallel	8.2.1
ipython	7.16.3
ipython-genutils	0.2.0
ipywidgets	7.7.2
jupyter-client	7.1.2
jupyter-core	4.9.2
jupyterlab-pygments	0.1.2
jupyterlab-widgets	1.1.1
Keras	2.1.4
Keras-Applications	1.0.8
Keras-Preprocessing	1.1.2
kiwisolver	1.3.1
labelme	5.1.1

Table A.2 Packages used for the project

Markdown	3.3.7
MarkupSafe	2.0.1
mask-rcnn	2.1
matplotlib	3.3.4
olefile	0.46
opencv-python	4.6.0.66
opt-einsum	3.3.0
packaging	21.3
pandas	1.1.5
pbr	3.1.1
pickleshare	0.7.5
Pillow	8.4.0
pip	21.2.2
python-dateutil	2.8.2
python-speech-features	0.6
python-utils	2.5.6
scikit-image	0.17.2
scikit-learn	0.24.2
scipy	1.5.4
tensorboard	1.14.0
tensorboard-data-server	0.6.1
tensorboard-plugin-wit	1.8.1
tensorflow	1.14.0
tensorflow-estimator	1.14.0
torch	1.8.2+cpu
torchaudio	0.8.2
torchvision	0.9.2+cpu
tornado	6.1
utils	1.0.1
visualization	1.0.0
zipp	3.6.0

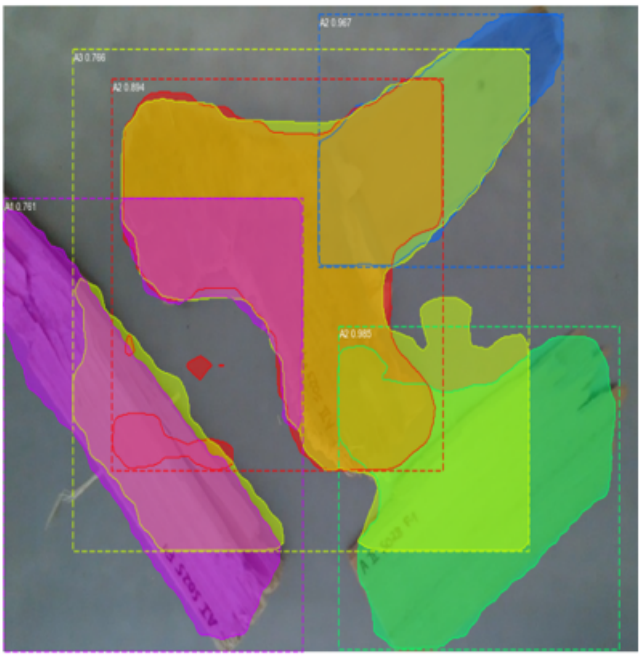


Figure A.1 Mask-RCNN prediction with overlapping masks

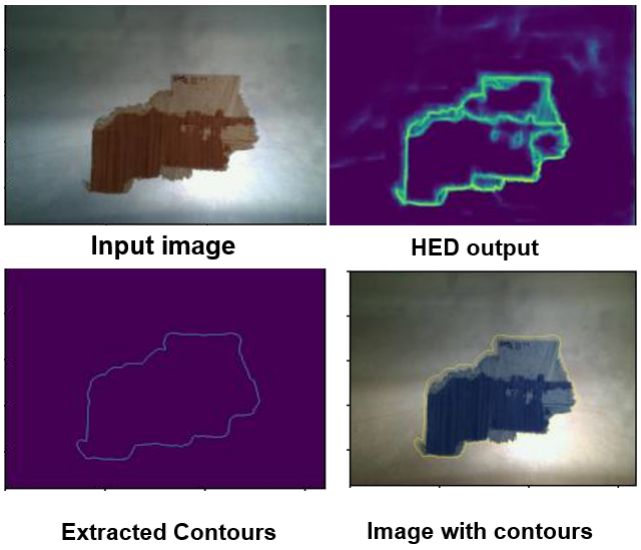


Figure A.2 Multiple stages of Holistically-Nested Edge detection

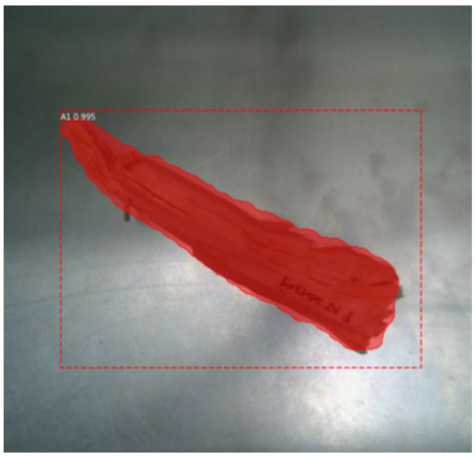


Figure A.3 Sample Tested on model with transfer learning

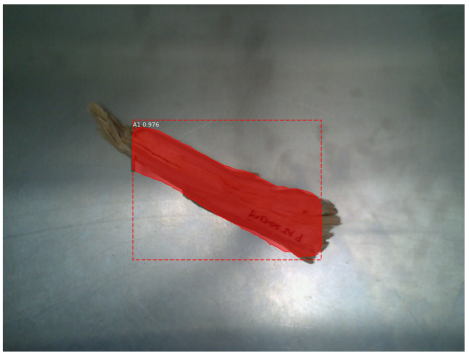


Figure A.4 Sample Tested on model without transfer learning