Master's Thesis

OptiCork: A Segmentation-Based Computer Vision Approach for detecting anomalies on Cork Disk Surfaces

Debayan Sen

Examiner:	Prof. Dr. Thomas Brox
Second Examiner:	Prof. Dr. Matthias Teschner
Adviser:	David Joel Regina

Albert-Ludwigs-University Freiburg

Faculty of Engineering

Department of Computer Science

Chair for Pattern Recognition and Image Processing

September 1st, 2021

Writing period

01. 03. 2021 – 01. 09. 2021

Examiner

Prof. Dr. Thomas Brox

Second Examiner

Prof. Dr. Matthias Teschner

Adviser

David Joel Regina

Declaration

I hereby declare, that I am the sole author and composer of my thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work.
I hereby also declare, that my Thesis has not been prepared for another examination or assignment, either wholly or excerpts thereof.

_____ _____

Place, Date Signature

Abstract

Over the recent years deep neural networks have become the primary option for solving several image processing tasks. Convolutional Neural Networks (CNN) have seen recent success in many Computer Vision applications for automated optical quality inspection in the industries. This thesis presents an approach to automatically identify and predict anomalies on cork disk surfaces for the purpose of quality inspection and monitoring. To achieve this task, an image processing pipeline is developed which uses images captured from a high resolution camera to estimate the quality of cork and detect the anomalies present on the cork disk surface by segmenting those defects. This camera is planned to be mounted on the fast-moving conveyor belt of the production line to capture the cork disk images and make a real-time prediction on it. To get the most precise and accurate segmentation, two very popular frameworks, namely U-Net and Mask R-CNN are implemented. The predictions from these two models are then compared and subsequently combined using a weighted average ensemble method which can outperform both the models, thereby giving a robust prediction on the cork disk images. Both U-Net and Mask R-CNN have their own strengths and weaknesses and therefore the ensemble method can exploit the advantages from both the models and perform better than the individual model. The best model for the cork quality classification achieves an accuracy of 66%, while the best ensemble model for segmentation achieves a mean IoU score of 0.772 and a Dice score of 81.9% on the test dataset.

Contents

List of Figures

List of Tables

1 Introduction

Cork is an impermeable buoyant material obtained from the bark tissue that the cork tree produces around the trunk. The most important industrial application of cork is the production of cork stoppers for sealing champagnes, wines and other liquors. Some stopper models are made up of multiple parts in which a cork disk is attached to the bottom of the stopper. These cork disks are sometimes defective and have wood inclusions on their surfaces which come in contact with the liquid and contaminate it. Furthermore, inserting the stoppers into the bottle necks requires a decent amount of pressure that may also cause potential wood inclusions on the surface to damage the stopper leading to their breakage and contamination of the liquid with the broken particles.

The main motivation of this thesis is to predict whether a cork has wood inclusions on its surface without human intervention. With the recent success of deep learning and computer vision techniques, it is possible to extract a lot of relevant features from images which are sometimes difficult to identify with the naked eye. Furthermore, automating the quality control process can be economical since it can save human capital and it may even be more reliable showing better reproducibility as there aren't multiple people working on the task. It has the added advantage of running it without any halt, as it doesn't have the limitations like exhaustion, sleepiness or sickness. Hence, it is faster and more efficient than manual human inspection. Previous tests carried out for detecting the defects showed less success in differentiating wood inclusions from the cork based on spectral or intensity information. However, while

manually inspecting the cork disks under a microscope, the wood inclusions can in fact be very well and unambiguously recognized since the microscopic texture of wood differs significantly from cork and other irregularities.

Currently, there exists an X-Ray detection system that analyzes the cork disks based on their density. Since wood inclusions have higher density than cork, the system has a high detection rate for wood particles. However, wood inclusions only on the surface are prone to contaminate the liquid and those that are inside the cork disk should be ignored. Due to the nature of the X-Ray, the machine is unable to determine if inclusion is on the surface or inside the cork disk and thus generating a big amount of false positives. Hence, a computer vision model with an image processing pipeline can be integrated along with the X-Ray system or implemented as an independent system to give robust predictions.

The remainder of this document is organized as follows:

Chapter 2 presents the various background concepts and theories that are crucial for this work.

Chapter 3 explores the literature for similar problems implemented before and provides an overview of the approaches taken.

Chapter 4 explains the methodology used for this thesis and includes details on (1) cork quality classification, (2) anomaly detection and segmentation, and (3) results aggregation.

Chapter 5 demonstrates the results of different experiments conducted on the prediction models.

Chapter 6 provides an insight into the outcomes and proposes some thoughts on future scope and possible improvements.

Chapter 7 presents the conclusion of this thesis.

2 Background

This thesis was conducted at Fraunhofer IPM and the project was funded by Corticeira Amorim which is the world's largest producer of highest quality cork products. Amorim aims to automate the quality inspection and monitoring of cork products by replacing their old quality control system with a new one which will reduce their cost of operation and improve the efficiency.

This chapter provides an overview on deep learning and some other important concepts useful in understanding the approach undertaken to tackle the problem of anomaly detection. It reviews the state-of-the-art algorithms for pattern detection and image segmentation and the last section discusses the current scenario regarding the quality control system of cork products at Amorim and also introduces the intuition behind expressing the problem statement as a computer vision task.

2.1 Computer Vision and Deep Learning

Over the last few years deep learning methods have seen huge success in pattern detection and they have been shown to outperform various previous state-of-the-art machine learning techniques in several fields, with computer vision being one of the most used cases. In literature, various exciting researches prior to Deep Learning have been already done to emulate the human sensory responses such as speech and vision [1, 2, 3, 4]. Deep Learning, as a branch of Machine Learning, tries to process data

and imitate the thinking process of human brains. Deep Learning uses multiple layers in Artificial Neural Network (ANN) to process data and make predictions on them. The first layer in a network is called the input layer, while the last is called an output layer. All the layers between the two are referred to as hidden layers. The history of ANNs can be traced back to the Perceptron by F. Rosenblatt in 1958 [5]. The Perceptron is considered as the most basic unit in an artificial neural network which can perform supervised binary classification. Rosenblatt's single-layer perceptron is the simplest neural network which is composed of an artificial neuron that resembles the neurons in human brains. The neuron is associated with a weight vector and a bias term which are the learning parameters. This weight vector is multiplied by the input vector and added to the bias term to form a linear combination and then an activation function is applied to this linear combination to obtain a single output. The combination of multiple perceptrons in a series of layers gave birth to deep neural network and paved the way for the development of Deep Learning.

Many traditional Computer Vision techniques which had been undergoing progressive development in years prior to the rise of Deep Learning are getting obsolete [6], such as, histogram of oriented gradients (HOG) [7], scale-invariant feature transform (SIFT) [8] and bag of words (BoW) [9]. Among them, SIFT technique was the most used technique to identify objects within images, regardless of the image orientation, scale and rotation. However, a potential drawback of SIFT is that it is computationally heavy and not effective in low powered devices since it is based on the Histogram of Gradients which means the gradients of each pixel in the patch need to be computed and these computations cost time. A major breakthrough in the field of image processing with artificial neural networks came with the invention of the Convolutional Neural Networks (CNNs), which is a subclass of ANNs by LeCun in 1989 [10]. LeCun had built on the work done by Kunihiko Fukushima, a Japanese scientist who, a few years earlier, had invented the neocognitron [11], a very basic image recognition neural network. The early version of CNNs, called LeNet (after LeCun), could recognize handwritten digits. In 1998, an improved version of

LeNet known as LeNet-5 achieved an accuracy of 99% on the test dataset of MNIST Handwritten Digits Image Dataset [12]. But despite their ingenuity, CNNs remained on the sidelines of computer vision and artificial intelligence because they faced a serious problem of not being able to scale. CNNs needed a lot of data and compute resources to work efficiently for large images, and at that time, the technique was only applicable to images with low resolutions. However, in 2012, AlexNet [13] (named after its main creator, Alex Krizhevsky) brought a major breakthrough in the image recognition tasks by improving the recognition rate from 70% to more than 80% and also increasing the speed of the network by a significant margin. AlexNet also won the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [14] with an amazing 85 percent accuracy.

A convolutional neural network (CNN) is basically a type of artificial neural network that is specifically designed to process pixel data with the help of convolution which is a simple mathematical operation of multiplying pixel values by weights and summing them. The main idea behind convolution is to to reduce or downsample the image pixels into a form that is easier to process, without losing features that are critical for getting a good prediction. Convolutional neural networks are composed of multiple layers of artificial neurons. Artificial neurons are mathematical functions that calculate the weighted sum of multiple inputs and outputs an activation value. When an image is given as an input to the CNN, each layer generates several activation functions that are passed on to the next layer. The first layer usually extracts basic lower-level features such as edges and blobs. This output is passed on to the next layer which detects more complex features such as corners or combined edges. As we move deeper into the network it can identify even more complex higher-level features such as objects and faces. The most attractive feature of CNN is its ability to exploit spatial or temporal correlation in image data. Other notable attributes of CNN are hierarchical learning, automatic feature extraction, multi-tasking, and weight sharing [15, 16]. In the last few years, it has been proved that different levels of features, including both low and high-level, can be transferred to a generic recognition task

by exploiting the concept of Transfer Learning (TL) [17, 18]. Transfer Learning is perhaps another notable development in the field of Machine Learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem.

The success of CNNs has gone beyond the academic community. Companies such as Google, Microsoft, and Facebook have active research groups who are working for exploring new architectures of CNN. In recent years, most of the front-runners of image processing and Computer Vision (CV) competitions are employing deep CNN-based models.

2.2 CNN Architectures

For many years researchers are exploring the architectures of CNN models to improve the prediction accuracy and at the same time reduce the computation cost. As the introduction of AlexNet drew great attention in the field of Computer Vision, large tech companies joined the force of developing deep learning frameworks with deeper and more efficient architectures. Among them, Google open-sourced the famous TensorFlow [19] framework that is still the most popular deep learning framework in the Machine Learning field. The inventor of Caffe [20] joined Facebook and continued the release of Caffe2 and at the same time, Facebook AI Research (FAIR) team also released another popular framework PyTorch [21] which was based on the Torch framework but with the more popular Python APIs. Microsoft Research developed the CNTK [22] framework.

While developing new architectures, it was found that deeper networks work better than the shallower ones because they increase the model capacity, improve the generalization ability and also allow the network to learn features at different level of abstraction. However, deeper networks faced the problem of Vanishing Gradient [23]. This problem arises when the network gets bigger and the gradient of weights

becomes zero while back-propagating to the previous layers of the network. In 2015, ResNet [24] proposed by Kaiming He et al. solved the problem of Vanishing Gradient by introducing shortcut skip connections between the input and the output layers of every block known as Residual Block. ResNet is basically a stack of 'Residual Blocks' and hence the name Residual Network. ResNet was successful in setting a new record by further improving the accuracy of image classification. There are many other notable architectures that worked well in terms of accuracy and inference time and at the same time having less number of parameters, such as, MobileNet [25] and EfficientNet [26]. MobileNet uses depth-wise separable convolutions to build light weight deep neural networks whereas EfficientNet uses a compound scaling factor to uniformly scale network width, depth, and resolution and thus achieving higher accuracy with less parameters.

2.3 Region-based CNNs (R-CNNs)

The Region-based Convolutional Neural Networks (R-CNN) are the family of machine learning models for computer vision that were introduced specifically for the task of Object Detection by Girshick et al. in 2014 [27]. The R-CNN models mainly follow a two-stage process to perform object detection. The first stage is to identify the regions in the image that contain an object and these rectangular regions are called region proposals. R-CNN uses selective search to extract these bounding boxes from an image. It generates 2000 region proposals for each individual image. Next, the feature maps are extracted by a CNN model for the set of region proposals obtained from the selective search. These extracted feature maps are then used to classify and localize the objects in the image. To classify the detected objects, a pre-trained CNN model is used, which extracts the feature maps for each region of interest and then they are passed to a set of binary Support Vector Machine (SVM) models. Along with the classification, a regression model is also used to predict the coordinates of the bounding boxes by minimizing the localization errors.

The R-CNN model was followed by the Fast R-CNN model in 2015 [28]. This algorithm is similar to the R-CNN algorithm. However, instead of feeding the region proposals to the CNN, the input image is fed to a pre-trained CNN model to generate a single convolutional feature map. A selective search is then applied on this feature map to extract the region of interest for region proposals and RoI Pool layer resizes them to a fixed size to feed them into a fully connected layer for object classification and bounding box regression. So, the convolutional operation takes place only once instead of 2000 times which makes the Fast R-CNN more efficient than original R-CNN. The Fast R-CNN model was further succeeded by the Faster R-CNN model in 2016 [29], in which a CNN called Region Proposal Network (RPN) was used instead of the selective search algorithm in order to generate the region proposals. Selective search is slow and inefficient and therefore eliminating the selective search algorithm makes the model faster and more efficient in generating the region proposals. Mask R-CNN model by He et al. in 2017 [30] was the fourth improvement in the R-CNN model's family which extends the ability to also perform instance segmentation along with object detection. Mask R-CNN extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for object classification and bounding box regression.

There are two stages of Mask R-CNN similar to other members of the R-CNN family. First, it generates proposals about the regions where there might be an object based on the input image. Second, it predicts the class of the object, corrects the bounding box coordinates and generates a mask of the object for every detected region proposed by the first stage. In the first stage, the backbone CNN serves as a feature extractor to extract features from raw images. This backbone network is usually pre-trained on a large dataset. The feature map obtained from the backbone network is then used by the Region Proposal Network (RPN) to predict the possible regions where an object might be present. The RPN uses a sliding window approach to scan all over the feature map. At every position of the sliding window, a fixed set of rectangular boxes called anchors are defined with different scales and aspect ratios so that they

can fit almost every object with different shape and size. These anchors are passed to the RPN and then it generates two outputs for every anchor, the anchor class, and the bounding box coordinates for the anchor location. Positive classes are assigned to the anchors with a high probability of containing an object and negative classes are assigned for the anchors containing only the background. The top N anchors with the highest probability of containing an object are selected by the RPN. The network then applies non-max suppression to the overlapping anchors generated for the same object to remove the redundant anchors and keeps only the anchors with the highest confidence value. Finally, the RPN returns the bounding box coordinates for the detected objects.

At the second stage, a different neural network takes two inputs, (1) the regions proposed by the first stage and (2) the feature map from the backbone CNN. The output generates objects classes, bounding box coordinates and masks. During the training on the original Faster R-CNN architecture, the Mask R-CNN authors realized that the regions of the feature map processed by the RoI Pool layer was slightly misaligned from the regions of the original image. This issue was solved by replacing RoI Pool with RoI Align layer which removes the harsh quantization of RoI Pooling operation by using the bilinear interpolation technique to calculate every region of the resized feature map which properly aligns the extracted features with the input. Next, these aligned and resized anchor grids can be passed to the fully connected layers which further performs classification to predict the object classes and regression to predict the bounding box coordinates. It has a separate branch with fully convolutional layers for predicting the mask which accepts the output from the ROI Align layer and generates a binary mask for every detected object.

The main improvement of Mask R-CNN is the addition of a completely new branch for performing segmentation, which is missing in Fast/Faster R-CNN. Furthermore, Mask R-CNN is simple to implement and train with the initial Faster R-CNN framework, which facilitates a wide range of flexible architecture designs. Additionally, the

mask branch only adds a small computational overhead which is negligible for real-time applications. Mask R-CNN has been used in many well known applications of instance segmentation, for example, object detection and segmentation of satellite imagery, street-view image segmentation for autonomous vehicle, biomedical image segmentation and so on. Due to its recent success and similar usage, this framework can be considered a suitable choice for this project.

2.4 Semantic Segmentation with U-Net

The U-Net was developed by Olaf Ronneberger et al. in 2015 [31] for the purpose of semantic segmentation of biomedical images. It is a fully convolutional network (FCN) just like the mask branch of Mask R-CNN. It has a U-shaped architecture which contains two paths. The first path is the contraction path (also called as the encoder) which is used as a feature extractor to capture and encode the context in the image. The encoder is a stack of convolutional and max pooling layers which reduces the spatial dimensions in every layer and increases the channels. The second path is the symmetrical expansion path (also called as the decoder) which is used to decode the encoded feature map and perform segmentation by assigning classes to every pixel. The decoder increases the spatial dimensions while reducing the channels. At every step of the decoder, the feature maps are up-sampled by transposed convolutions and skip connections are used to concatenate the corresponding output from the encoder at the same level. Thus it is an end-to-end fully convolutional network (FCN), which means it only contains convolutional layers and does not contain any dense layer because of which it can accept images of any size.

The basic model of U-Net for the down-sampling path can also be replaced with a deeper network like ResNet [32] which allows the model to learn more complex features as the network depth can be greatly increased with residual blocks, thereby increasing the network's learning capacity. Another advantage of replacing the encoder with a

deep network is transfer learning in which pre-trained weights can be used to initialize the network, and therefore, the network is not required to be trained from scratch. This is especially important when the available dataset size is small.

Both Mask R-CNN and U-Net perform segmentation. However, a major difference between Mask R-CNN and U-Net is that Mask R-CNN produces instance segmentation while U-Net produces semantic segmentation. Instance segmentation generates a separate mask for every instance found in the image whereas semantic segmentation generates a single mask for multiple instances of the same class. For example, if an image has two dogs and three cats, then instance segmentation generates five different masks for every instance whereas semantic segmentation generates only two masks.

There are many applications of U-Net in biomedical image segmentation, such as brain image segmentation and liver image segmentation. Due to the recent success of U-Net in many computer vision applications for texture pattern segmentation, it can be considered a suitable choice for the work of this thesis.

2.5 Ensemble Learning

Ensemble learning is the process by which multiple models are trained individually and the outputs are subsequently combined to get a robust prediction which is better than the individual predictions by the different models. The individual models are called as "base learners" or "weak learners". The ensemble model can be created either by combining multiple algorithms or using different subsets of training datasets. The main purpose of ensemble method is to reduce the generalization error in the machine learning algorithms. The three main classes of ensemble learning methods are bagging, stacking, and boosting. Bagging involves fitting many decision trees on different subsets of the same dataset and averaging the predictions. Stacking involves fitting many different types of models on the same data and using another model to learn how to best combine the predictions. Boosting involves adding ensemble

members sequentially that correct the predictions made by prior models and outputs a weighted average of the predictions.

Ensemble methods are mostly used when the dataset size is small or the individual models are not efficient enough to give robust predictions. While ensemble learning is a very powerful tool, it also has some drawbacks. Ensemble methods are usually computationally expensive. Therefore, they add learning time and memory constraints to the problem. Ensemble models also suffer from lack of interpretability and explanations. It is hard to investigate the decisions taken by multiple models for a single task. In recent years, there is an increasing number of applications of ensemble learning which motivates the work of this thesis.

2.6 Problem Statement

The primary objective of this thesis is to automate the quality monitoring process of cork products. Anomaly detection and quality inspection of industrial products with the help of computer vision technique has been a topic of active research in the past few years. Currently, the quality inspection of cork products at Amorim is done with an X-Ray detection system which detects the wood inclusions present in the cork. However, the X-Ray is unable to predict whether the wood inclusion is present on the surface or inside the cork disk, and thus, generating a huge amount of false positives. Therefore, human supervision is required to select random samples of cork disks to check whether the samples show an expected distribution of anomalies. If not, all disks of the batch (>100k) are reintroduced to the sorting and quality inspection cycle. The entire system currently being employed at Amorim for quality inspection and monitoring is shown in Figure **1**.

The cork disks are only considered bad if the wood inclusion is on the surface because when they are used as a stopper in wine bottles, the wood inclusions on the cork surface can come in contact with the liquid and contaminate it. Furthermore, the

cork stopper might break under pressure when they are inserted into the bottle necks due to the presence of wood inclusions on the surface. Hence, it is very important to detect the wood inclusions that are present on the surface of the cork disk.

OptiCork, the name given to the image processing software at Fraunhofer IPM, mainly addresses two problems associated with the current quality inspection system. Firstly, the process involves human intervention from time to time. Secondly, it is inefficient due to the generation of large amount of false positives by the X-Ray detection system. To tackle these problems, a computer vision technique can be employed which will be faster and more efficient. The old X-Ray detection system can either be replaced with the new computer vision software or used in combination with it, although replacing the X-Ray requires the software to be quite efficient producing very few false positives.

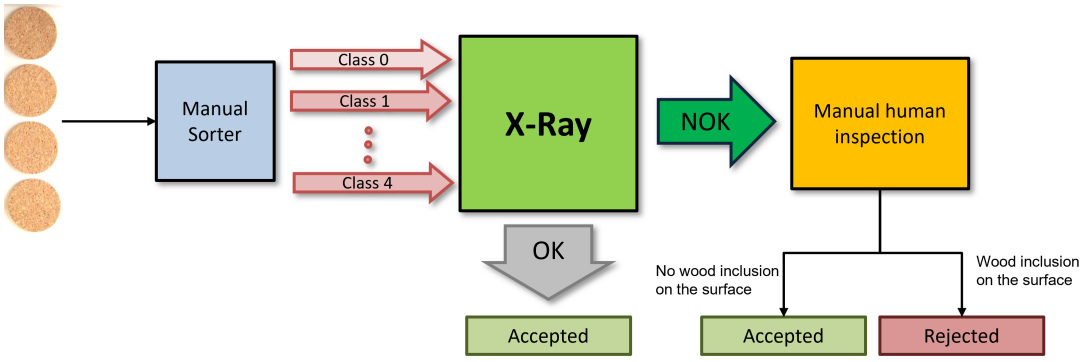


Figure 1: X-Ray detection system at Amorim to detect wood particles in the cork disks.

3 Related Work

Several different studies have been done in the recent years which attempt to solve the task of surface defect detection similar to the work done in this thesis. The task of automated visual inspection has always been a hot topic in the context of image processing and it has received some recent attention with the growing popularity of computer vision and deep learning models. Several works have already been done related to finding cork defects with image processing techniques. Prior works done on surface anomaly detection mostly involved the use of image processing techniques without the application of deep learning. Notable works in this area include [33], which uses morphological manipulation, convolution, binarization and clustering to extract the features of the cork and [34], which uses adaptive morphological filters to identify the contours or particular structures for defect detection. The authors of [35], [36] and [37] also provide interesting insights into the application of CNN (Convolutional Neural Network) models to detect surface defects. The work done in [38] and [39] in the recent years have also been successful in pattern detection to a great extent. In the upcoming sections of this chapter, a few of these previous studies are discussed in detail.

3.1 Quality Classification

The work done in [33] introduces an automatic vision system that uses a linear charged-coupled device (CCD) camera with a signal conditioning system to classify

the defects present on the surface of a cork-stopper. This system acquires and sends the image's RGB components to the computer. According to the red, blue and green components, the surface defects on the cork-stoppers are identified and classified with the help of image processing algorithms like morphological manipulation, convolution, binarization and clustering [40]. Another important work by Helena et al. in 1996 [34] also uses morphological filters to extract the features of the cork for the quality classification task. They use geometrical characteristics like shape, size and area to identify the contours and particular structures on the cork surface. Their algorithm achieves a high degree of accuracy on the classification task. Other notable works that have taken a similar approach in classifying corks include [41], [42] and [43]. Although the approaches discussed in these researches are not directly used in this thesis, they helped in better understanding of several image processing techniques that are experimented on before the rise of deep learning.

A recent study in [44] proposes a Deep Convolutional Neural Network (D-CNN) with transfer learning method and bottleneck features to perform texture classification on wood surfaces. There are five different classes of wood such as class I, class II, class III, class IV, and class V. The best quality of this wood is class I and the worst quality is class V. The manual human inspection could achieve only 55% of correct classification rate. However, the proposed method achieves an accuracy of 95.69%. This work is very similar to the task of quality classification on cork dataset done in this thesis. They use Xception model pre-trained on ImageNet [14] dataset whereas this thesis uses EfficientNet model pre-trained on ImageNet dataset.

3.2 Defect Detection and Localization

The research done in [38] proposes an automatic visual inspection system for the location and classification of defects on the wood surface. The work done in this thesis for the detection network is mostly inspired from this research. In this

research, they use a faster region-based convolutional neural network (Faster R-CNN) for the identification of defects on wood veneer surfaces. They also explore data augmentation and transfer learning using pre-trained AlexNet, VGG16, BNInception, and ResNet152 models. The results in this research demonstrate the applicability of data augmentation and transfer learning techniques for the identification of four classes of wood veneer surface defects. The best average accuracy was obtained using the pre-trained ResNet152 neural network model (80.6%), while by combining all the defect classes into one type, 96.1% accuracy in finding surface defects was achieved. The results show that this surface defect detection algorithm can be used for industrial wood processing applications. Furthermore, the method can also be adopted for detecting surface defects in other industrially processed materials like corks whose textures are similar to wood; hence, this thesis uses Faster R-CNN to detect anomalies similar to the work done in the research.

3.3 Anomaly Segmentation

A very recent work by Shi et al. in 2020 [39] explores the use of Mask R-CNN to detect wood veneer defects. In this paper, glance multiple channel mask region convolution neural network (R-CNN), was constructed to detect wood veneer defects, which included a glance network and a multiple channel Mask R-CNN. Firstly, 2838 wood veneer images were collected using data collection equipment developed in the laboratory and labeled by experienced workers from a wood company. Then, an integrated model, Neural Architecture Search (NAS) technology was used to automatically construct the glance network with the lowest number of floating-point operations to pick out potential defective images out of numerous original wood veneer images. A genetic algorithm was used to merge the intermediate features extracted by the glance network. Multi-Channel Mask R-CNN was then used to classify, locate and segment the defects. The experimental results showed that the proposed method achieved a 98.70% overall classification accuracy and a 95.31%

mean average precision, and only 2.5 s was needed to detect a batch of 50 standard images and 50 defective images. This paper has been a major inspiration behind the implementation of this thesis. In this thesis, Mask R-CNN is used without the use of NAS since it takes much time in training and requires huge computational resources.

3.4 Ensemble Learning

In recent years, ensemble learning has been used in many industrial applications and deep learning competitions due to the growing computational power, which allows training large ensemble models in a reasonable time-frame. Several works have already been done in the recent years which attempts to utilize the ensemble approach in image segmentation by combining various deep learning models. Kaggle's 2018 Data Science Bowl [45] presented a computer vision task for nuclei segmentation in a competition format. To achieve this task, a research was done in [46] which compares U-Net and Mask R-CNN and develops an ensemble model to combine their predictions that can outperform both the models by a significant margin. This ensemble model proved to produce better results than the predictions from the individual models. Another important research done in [47] also uses ensemble approach for tumor tissue segmentation in histopathology images by using multiple re-scaled images and different subsets of images for training different U-Net models and finally ensemble the outputs by linear combination. A very recent work by Kim et al. in January 2021 [48] proposes a novel approach using a stacking ensemble method to combine multiple heterogeneous deep-learning-based portrait segmentation models. Simple and weighted soft voting method is used to combine the models. The experiment results showed that the proposed ensemble approach can perform with higher accuracy and lower errors than single deep-learning-based portrait segmentation models. This research has been a major influence on the work of this thesis.

4 Methods

The main goal of this thesis is to create an end-to-end image processing system to acquire cork disk images on a running conveyor belt and then make use of deep learning models to (1) classify them into their quality classes and (2) detect wood inclusions on their surfaces along with segmenting those anomalies. The main objective is to automate the anomaly detection system based on a single image only without considering other physical attributes of the cork. Initially, the focus was only on detecting the location of the wood inclusions on the cork disk. However, detecting the location of wood inclusions does not solve the problem completely, since one also needs to know precisely the area of these anomalies for complete analysis and hence requires image segmentation. This chapter discusses in detail about all the approaches related to deep learning and computer vision that were undertaken to solve the problem of anomaly prediction.

The remainder of this chapter is organized as follows: Section **4.1** provides an overview of the image processing pipeline, Section **4.2** discusses all the implementations and the approaches used in this thesis for getting a robust prediction on cork disk datasets including dataset pre-processing and model training, followed by the data augmentation strategy used which is discussed in Section **4.3**. Section **4.4** provides details about the different performance metrics used to evaluate the training efficiency, and lastly the section **4.5** describes the experimental setup used for the experiments.

4.1 Image Processing Pipeline Overview

The entire system can be sketched as a multi-step image processing pipeline which is shown in Figure **2**. The task breakdown per step of the pipeline is as follows:

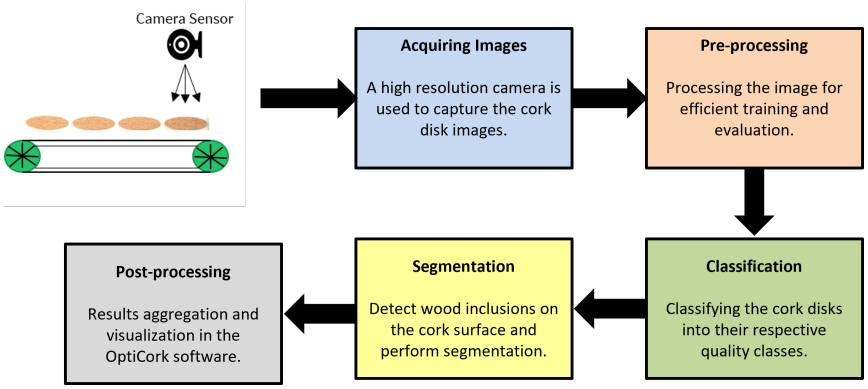


Figure 2: Steps in image processing pipeline related to quality inspection and monitoring of cork disks on a conveyor belt during manufacturing.

1. **Acquiring Data:** A high resolution camera is used to capture the cork disk surface images on a fast moving conveyor belt. This camera is mounted on top of the conveyor belt where the cork disks move in a line. The images are then sent to the software for evaluation and analysis, so that the cork disks that are defective can be sorted out.

2. **Preprocessing:** Before the images are sent to the neural networks, they go through a background subtraction method for eliminating the background from the cork disk. Images are then resized to one size (1024 x 1024 pixels) and it is modified according to the format expected as an input to the trained neural network. Aspect ratio is preserved while resizing. If an image is not square, then zero padding is added at the top/bottom or right/left.

3. **Quality Classification:** An EfficientNet classification model is used to perform multi-class classification. The model predicts the quality class of the cork disk along with the confidence score.

4. **Instance Segmentation:** A trained Mask R-CNN model is used to perform instance segmentation on the cork disk images. The model returns the bounding box coordinates and pixel masks for the detected wood inclusions in the input image.

5. **Post-processing:** The software aggregates all the results from the segmentation model and saves them in a JSON file. The results are visualized and analyzed with a tool that is implemented in the software. Certain parameters of threshold like confidence value, segmentation area and IoU can be changed and the output predictions can be displayed accordingly.

4.2 Workflow Description

This section discusses about all the deep learning implementations and research approaches undertaken to get a robust predictions for the cork disk images. It explains all the details starting from preparing the datasets to training several deep learning models to achieve a high efficiency anomaly detection system. An overview of the workflow is shown in Figure **3**. The details of every stage of the experiment is provided below.

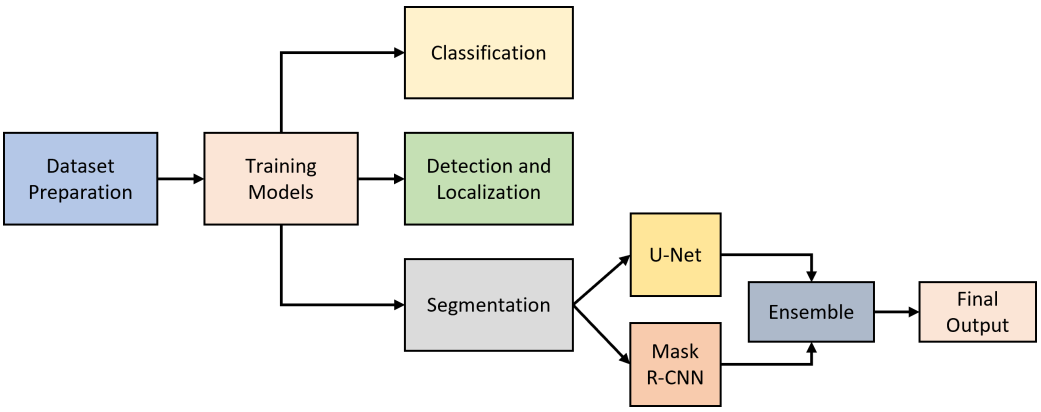


Figure 3: Description of the general workflow related to all the approaches and strategies implemented in this thesis.

4.2.1 Dataset Preparation

A dataset of 1000 cork disk images captured with a high resolution camera is used for the experiment. An example of an anomalous cork disk can be found in Figure **4**. For performing segmentation, accurate polygonal annotations of the wooden inclusions are required during training. This is different from the task of pattern detection, where only a bounding box around the object must be provided. In the beginning, the cork disks were physically inspected and labeled by the domain experts at Amorim. These labels were again recreated at Fraunhofer IPM with online annotation tools. There are several open-source image annotation tools available online which support a variety of image annotation formats to prepare a dataset for segmentation task. The tools used here for annotating the cork disk dataset are *VIA (**V**GG **I**mage **A**nnotator)* [49] and *MakeSense* [50]. An example of a cork disk being annotated with *MakeSense* annotation tool is shown in Figure **5**. The label *wood* is assigned to every wood inclusion found on the cork disk surface. A class is also assigned to every cork disk according to the texture, smoothness and some other physical properties of the cork. The final dataset consisted of around 1000 images of resolution 1924 x 1624 x 3, split equally among the five classes, where *Class 0* being the best quality and *Class 4* being the worst. Every class is again divided into two categories, *OK* and *NOK* (meaning not OK). The distribution of the dataset across the classes is shown in Figure **6**. A cork is acceptable and falls under the category of *OK* only if it follows certain quality standards, otherwise it falls under the category *NOK*. These quality standards are decided by the domain experts at Amorim. If the cork has wooden inclusions (>3 mm^2 and no holes) on it's surface, it falls under the category of *NOK*, otherwise it belongs to the class *OK*. Figure **7** gives a detailed analysis on the frequency of occurrence of wood inclusions per disk and the average surface area occupied by them based on the cork dataset quality classes and categories. The training, validation, and test split was kept at 80%, 10%, and 10% respectively.

Initially, training data consisted of all the quality classes in equal proportion and

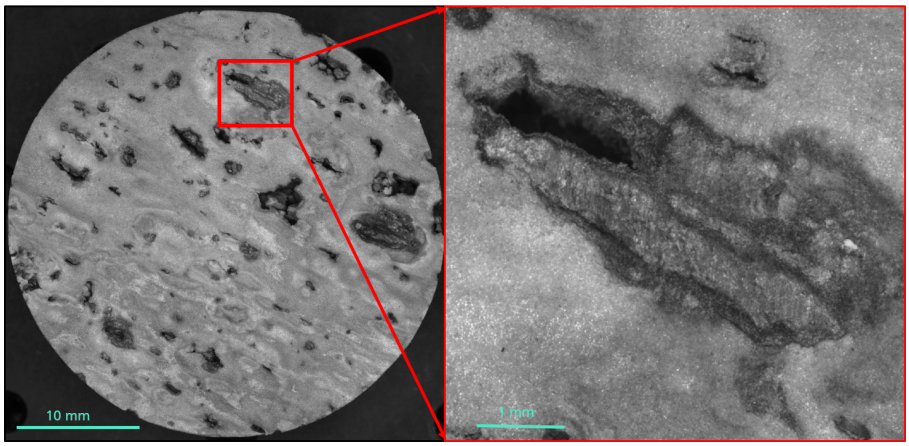


Figure 4: Example of a cork disk with a wood inclusion.

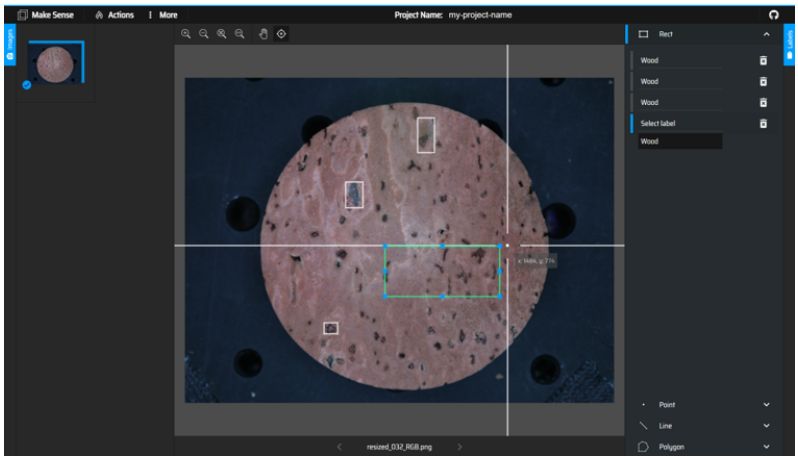


Figure 5: A browser view of the MakeSense online annotation tool.

this dataset group was named as Type I. Type II dataset was the one from which *Class 3* and *Class 4* cork disks were removed because of the poor quality of the cork disks which affects the training process. Also, the datasets belonging to *Class 3* and *Class 4* are noisy due to inaccurate and large amount of false annotations and so eliminating these from the training data was crucial for the training.

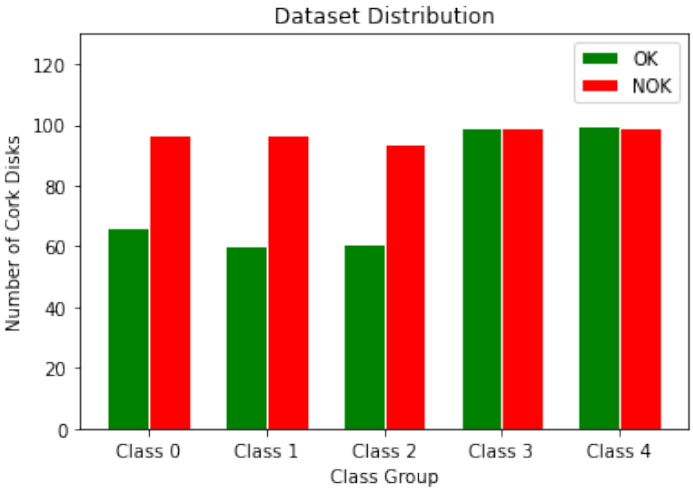


Figure 6: Visualizing the distribution of cork dataset across different classes.

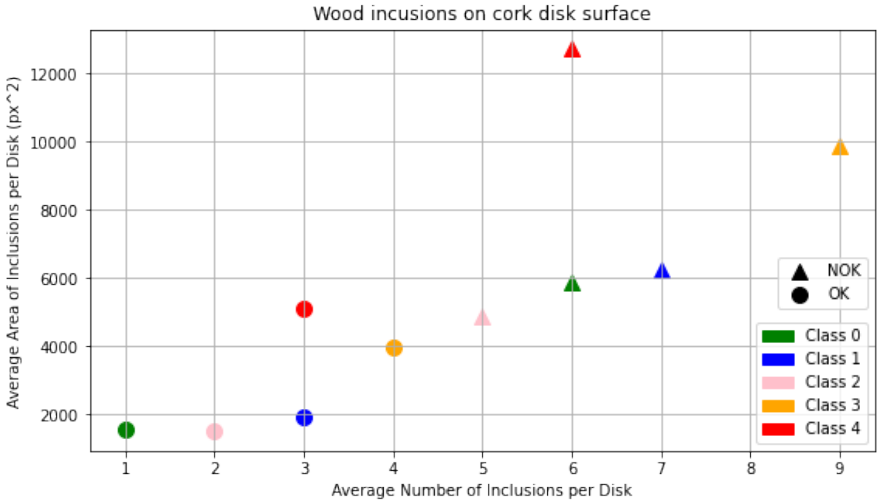


Figure 7: Analyzing the frequency of occurrence of wood inclusions per disk and the average surface area occupied by them based on the cork dataset quality classes (*Class 0,1,2,3,4*) and categories (*OK/NOK*).

Dataset preparation for Mask R-CNN

In this section, all the dataset preparation steps specifically related to Mask R-CNN are discussed. The dataset is preprocessed before it is used for training. A Python script is written which expects an RGB image and resizes it to one size (1024 x 1024

pixels) keeping the aspect ratio same. If an image is not square, then zero padding is added wherever necessary to preserve the aspect ratio of the original image and at the same time making the image square in shape. It then subtracts the mean pixel from the image for the purpose of normalization and converts it into float.

The Github Matterport [51] implementation also provides a preprocessing technique for optimizing mask in case the image resolution is high. Binary masks can get very large when training with high resolution images. For example, when training with 1024 x 1024 pixels images, the mask of a single instance requires 1MB of memory. If an image has 100 instances then that is 100MB of space for the mask alone. To improve training speed, masks are optimized by storing mask pixels that are inside the object bounding box, rather than the mask of the full image. Most objects are small compared to the image size, so a lot of space is saved by not storing a lot of zeros around the object. Mask is resized to a smaller size (e.g. 56 x 56 pixels). A slight loss in accuracy is involved for objects that are larger than the selected size. But most object annotations are not very accurate in itself, so this loss is negligible for most practical purposes.

Selecting the anchor scales and anchor ratios for the bounding boxes in the region proposal network is a crucial step before starting the training. The anchor scales and ratios should be such that it could fit the majority of objects in the data. The whole dataset is checked to find the maximum and minimum size of instances for selecting anchor scales. Five anchor scales are selected based on the possible size of the instances and also the diameter of the cork disk since a wood inclusion cannot be bigger than the diameter. Since the instances can be rectangular in shape and not necessarily square, three anchor ratios are selected corresponding to three different aspect ratios of the bounding boxes. Therefore, in total there are 15 (3x5) anchor boxes for each cell or unit in the image as can be seen in the Figure **8**.

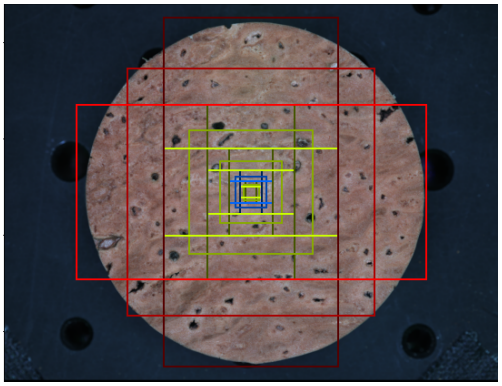


Figure 8: Visualizing the anchor scales and anchor ratios required for training.

4.2.2 Quality Classification

Every cork disk belongs to a quality class according to which it is sent to either different vendors or different manufacturing lines for various products. For example, a high quality cork is used for sealing expensive wine bottles to have an attractive look that a wine connoisseur might appreciate. Similarly, a low quality cork can be used in sport's accessories like baseballs, cricket balls, badminton and so on. A low quality cork can even be used in lower quality wines to make it cheaper and more affordable. Therefore, it is very important to have a separate pipeline to classify cork into their respective quality classes.

Keras provides an open-source implementation of the EfficientNet model architecture, along with pre-trained weights on the ImageNet Dataset, which have been used for this project. Before starting the training process, the cork dataset was compiled in a single folder and a python script was written to split the dataset into train, validation and test set. Dataset augmentation was applied with Keras in-built APIs. Random left-right flipping and random rotation were used for image augmentation. The EfficientNet-B0 model was trained for 60 epochs with a learning rate of 1e-4 and a batch size of 32 images. A dropout rate of 0.4 was used for the softmax classifier. Early stopping was implemented to stop the training when there was no further improvement seen in several epochs.

To improve the performance, a few experiments were conducted in which a different number of layers of the network were fine-tuned on the target dataset. Different variants of EfficientNet were also compared to find out the model capacity required for the cork dataset. For the purpose of building a strong classifier, EfficientNet models B0 through B3 were trained for several epochs. For comparing the prediction performance based on the effects of changing model capacity and the number of fine-tuned layers, different other model architectures were trained such as ResNet50 and VGG16. However, EfficientNet proved to have performed best in terms of accuracy and inference time. Finally, hyperparameter optimization was performed for finding the best values of hyperparameters during training.

4.2.3 Anomaly Detection and Localization

In the initial stages of the project, the primary goal was to only detect the wood inclusions on the cork surface and classify them as wood. Faster R-CNN was the initial approach used to detect the wood inclusions on the cork disk surface and predicting them as wood.

An open-source implementation of Faster R-CNN available from Kbardool [52] was used in this project to detect the wood inclusions with a rectangular bounding box around the detected pattern. The experiment was conducted with only 50 cork disk images captured with a normal digital camera since those were the only cork disks available at the time. This experiment was considered the starting point in this project, the results of which would decide if it is worth conducting a research on this project.

The dataset was annotated in *Pascal VOC (Visual Object Challenge)* [53] format which is an XML file format. In *Pascal VOC* format, an XML file is created for each image in the dataset having all the attributes and bounding box coordinates. The annotation was done with the help of a browser based annotation tool known as

MakeSense. The dataset was also resized such that their shorter side is 600 pixels keeping the aspect ratio same. Data augmentations like vertical flip, horizontal flip and rotation were added to the input images before they were used in training.

After the dataset preprocessing, the Faster R-CNN network was trained with ResNet50, ResNet101 and VGG16 backbone architecture. The model is trained for several epochs. Setting the parameters *ANCHOR_BOX_SCALES* and *ANCHOR_BOX_RATIOS* to (128, 256, 512) and [0.5, 1, 2] respectively generates 9 anchors per location to detect pattern for both larger and smaller scale. Even though the number of images were very less and the image resolution was low, the outcome was quite convincing to continue the research. The performance and outcome of this experiment are discussed in the next section.

4.2.4 Anomaly Segmentation

The results obtained from the Faster R-CNN model were quite motivating to conduct further research in this project. Detection of wood inclusions on the cork surface does not completely solve the problem of predicting a cork good or bad. One also needs to find the size and area of these wood inclusions to decide whether to accept or reject a cork disk. For segmenting the wood inclusions, two very popular segmentation frameworks have been used; U-Net for semantic segmentation and Mask R-CNN for instance segmentation. A comparative study was done between U-Net and Mask R-CNN based on the performance of their best performing models. To get the best out of both the models, a weighted average ensemble method was implemented to combine the predictions from both the models, since both U-Net and Mask R-CNN have their own strengths and weaknesses. Sometimes there are very good models that we wish to contribute more to an ensemble prediction, and perhaps less skillful models that may be useful but should contribute less to an ensemble prediction. Hence, a weighted average ensemble approach allows multiple models to contribute to a prediction in proportion to their trust or estimated performance. The detailed

process of training the two segmentation models as well as the ensemble method is discussed below:

U-Net Model

There is a wide range of open-source U-Net model architectures and pre-trained model weights available for the public use. An open-source implementation from qubvel [54] was used to build the U-Net network and train the cork dataset. The model implementation uses the deep learning library Tensorflow 2.4.1 with a Keras 2.4.3 high level API. The model comes with 25 backbone architectures, along with pre-trained weights on the ImageNet Dataset. A python script was written to integrate the already annotated dataset into the U-Net compatible format where the ground-truth mask is provided as an image rather than a JSON or XML file. The mask has a dimension of HxWxD where H and W are the height and width of the cork disk images respectively and D is the number of classes. In this case, there is only one class and that is 'Wood'. A 'Background' class can also be added to make it a problem of multi-class segmentation from binary segmentation.

The data is randomly cropped to a fixed size (640x640). Data augmentation was done with the help of *albumentations* [55], which is a Python library for image augmentation. Since the image was cropped to a smaller size, the training was run for several epochs to make sure that every part of the image was trained and not left out. Several augmentation strategies were used to increase the training data variance, although heavy augmentations did not give good results in this case.

The model was trained for 100 epochs with a batch size of 8. Sigmoid activation function was used for binary segmentation and Softmax was used for multi-class segmentation. The model was trained with different augmentation strategies and backbone architectures to get the best segmentation accuracy, the results of which are discussed in the next chapter.

Mask R-CNN Model

A very popular open-source Mask R-CNN implementation available from Matterport [51] was used in this thesis. This implementation was written in TensorFlow 1.13. To ensure compatibility with the current versions, a fork of this implementation which is upgraded to TensorFlow 2.4.1 was used for this work. The Mask R-CNN model with a ResNet101 Feature Pyramid Network (FPN) [56] backbone is trained and fine-tuned on the training dataset. The network backbone were pre-trained on the Microsoft Common Objects in Context (MS COCO) [57] dataset and ImageNet dataset. Besides transfer learning, Mask R-CNN is also trained from scratch to find out whether the pre-trained weights are making any difference. The Mask R-CNN has a lot of parameters that can be tuned to improve the overall model performance, and these are usually selected based on the target object type and possible real-world object sizes. The Region Proposal Network (RPN) generates anchor boxes which look for possible objects of interest in the images. Setting the parameters *RPN_ANCHOR_SCALES* and *RPN_ANCHOR_RATIOS* to (32, 64, 128, 256, 512) and [0.5, 1, 2] respectively generates a maximum of 15 anchors per location to detect object of both smaller and larger scale. In Figure **8** it is well understood while studying the data that anchor size bigger than 512 is not required as it covers almost the whole cork disk. A maximum of 100 ground truth instances was used for training. The training was conducted with a batch size of 2 and the learning rate was kept at 1e-2. The image size was resized to a fixed size (1024 x 1024 pixels) to remain within GPU memory constraints. The minimum detection confidence of the network was set at 0.7. The learning momentum was kept at 0.9 and the weight decay was set to 1e-3.

Initially, Mask R-CNN was trained with very few datasets to check if the model is overfitting and gradually learning to detect the wooden inclusions. Once the model started learning to detect and segment wooden inclusions, a couple of training strategies were adopted to improve the performance. In the beginning, only the last

layer was trained keeping all other layers frozen. This was done because the model is already pre-trained on other dataset and so the initial layers have already learned some low level features like shapes, textures and edges. Therefore, the final layer is only tuned to adjust its prediction with the cork dataset. However, fine-tuning more layers has shown improvement in the performance. To improve further the performance of Mask R-CNN model, different backbone architectures as feature extractor were experimented. The main reason behind experimenting with the backbone is to find out whether the cork dataset require shallow network or rather a deep network for detecting and segmenting the wooden inclusions on the cork surface. Performance metrics like Average Precision, Average Recall, mean IoU score and mAP were used to measure the model performance on the datasets. Finally, the model was trained with different data augmentation strategies and different backbones.

Ensemble Learning

After running several experiments on U-Net and Mask R-CNN, it is found that both the models have some strengths and failures. For example, Mask R-CNN has a good recall value whereas U-Net has a good precision value. Furthermore, Mask R-CNN produces more accurate segmentation than U-Net in terms of border shapes and segmentation size. But a major downside of Mask R-CNN is having large amount of false positives in the prediction which makes the model discard the good corks. Thus, the ensemble method is used to exploit the strengths of both the models.

Mask R-CNN and U-Net are trained extensively and the best models are chosen for the ensemble learning. As can be seen in the Figure **9**, the image of the cork disk is passed as input to both the U-Net and the Mask R-CNN model. To combine the model predictions, weighted soft voting method is used as a Meta-Learner of stacking ensemble. Stacking is the process of training individual segmentation models called First-Level learners and then combine them with an ensemble model called Second-Level learners or Meta-Learners. Mask R-CNN and U-Net are the First-Level learners

in which the input images from the cork dataset are fed and the soft segmented output masks are obtained. The final layer of the Fully Convolutional Network from both the models gives a probability matrix P of segmentation mask which is computed by pixel-wise soft-max activation over the final feature map. Since Mask R-CNN produces instance segmentation for every detected object 'z', they are combined in a single mask to get a semantic soft segmentation. The probability matrix is defined as:

$$P = [p_k(x)]_{mxn} \tag{1}$$

where x is the coordinate of each pixel in the image for the k-th channel and m,n is the size of the image. $p_k(x)$ is the probability value at the pixel position x, which is defined as follows:

$$p_k(x) = \exp(a_k(x))/(\sum_{c=1}^{K} \exp(a_c(x))) \tag{2}$$

where K is the number of classes and $p_k(x)$ is the approximated maximum-function whose value will be close to one for the k that has the maximum activation $a_k(x)$ and close to zero for all other k. The final output of the ensemble model function is defined as:

$$F = \sum_{i=1}^{M} w_i * P_i \tag{3}$$

in which the weights are required to be constrained, as given below:

$$0 \leq w_i \leq 1 \quad and \quad \sum_{i=1}^{M} w_i = 1 \tag{4}$$

Where F is the final output of probability matrix from the ensemble model, M represents the number of models, w_i represents the weight of the i-th model and P_i is the prediction probability map of i-th model. The weights w_i are optimized by grid-search and Bayesian optimization using Gaussian Processes. The optimizations are done with the help of a Python library known as scikit-optimize. After the final output from the ensemble learner, the mask is converted into binary segmentation by assigning pixel value of 1 if $p_k(x) \geq 0.5$ or else 0. Finally, for the purpose of post-processing and visualizations, the binary mask for the semantic segmentation is converted into instance segmentation with the help of Morphological Transformations available in OpenCV [58], which cuts out connected components in a binary mask and puts it on a separate mask layer. The performance of the ensemble method is discussed in the next chapter.

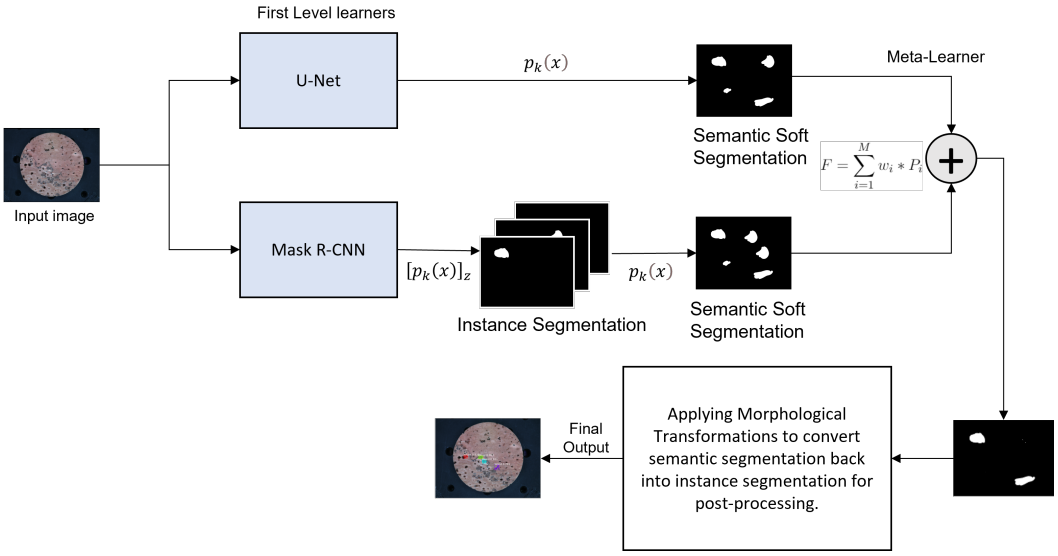


Figure 9: Ensemble learning workflow.

4.3 Data Augmentation

Data augmentation is an important step for training deep learning models to achieve the best performance. When a deep learning model is trained from scratch with a very limited amount of data, the model often memorizes the training examples and does not generalize well to the unseen data which results in overfitting. Data augmentation is one of the key techniques used in deep learning when the training dataset size is small because it provides a way to artificially expand the size of the training dataset. The dataset obtained from augmentation looks very similar to the training dataset with few changes that might occur in real life situations. Data augmentation is useful to improve performance and outcomes of deep learning models since the image's contrast, brightness and orientation may add some bias in the learning of the model. It actually prevents the model to learn on trivial features such as contrast, brightness, orientation, etc, and instead helps the model to identify other important patterns and shapes in the input data that are more relevant to the given problem. In short, it acts as a regularizer and helps reduce overfitting when training a machine learning model.

Besides TensorFlow and Keras having some built-in support for data augmentation, some third-party libraries also offer a collection of more diverse augmentation techniques which are very easy and flexible to integrate into the code. These modern data augmentation libraries are also very fast and easy to implement. For this work, the library *imgaug* [59] was used in Mask R-CNN and *albumentation* [60] was used in U-Net. They both provide a way to pile up multiple augmentation techniques and use them in a random order to the training data. This is called the augmentation strategy where multiple augmentation techniques are stacked together and randomly applied to the input image so that the network effectively receives a new and slightly modified image every time a training data is fetched from the dataset. While techniques like transfer learning have reduced the need for data variance, a dataset of sufficient quality, quantity and variety is still a requirement for efficient training. The following

augmentation techniques were used to augment the dataset in this work, few examples of which are provided in Figure **10**:

Horizontal and Vertical Flip: Flipping images left, right, up and down along the horizontal and vertical axis is applied during the loading of the training dataset. This was applied with a probability of 50% during the training.

Rotation: Rotating images clockwise and anti-clockwise by an angle theta about the center is another way to augment the image dataset. This was also applied with a probability of 50% during the training.

Shear Transformation: This technique makes the image distorted along a specific axis, mostly to create or rectify the perception angles. It is usually used to augment images so that the computers can see how humans see things from different angles.

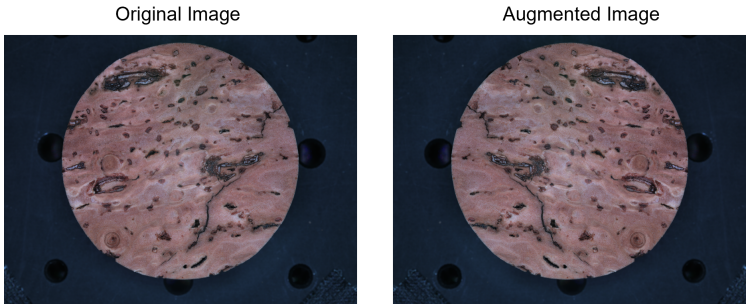
Random Crop: For training the U-Net model, randomly cropped images were used. The images were cropped to a fixed size of 640 x 640 pixels and they are trained along with other augmentation techniques.

Brightness, Saturation and Contrast: The cork dataset was mostly captured in good lighting conditions and in future it is going to be captured in the same way. But still some small random changes to the image brightness, saturation, and contrast were made to increase model robustness to these factors in case the lighting condition does not remain the same as the training dataset.

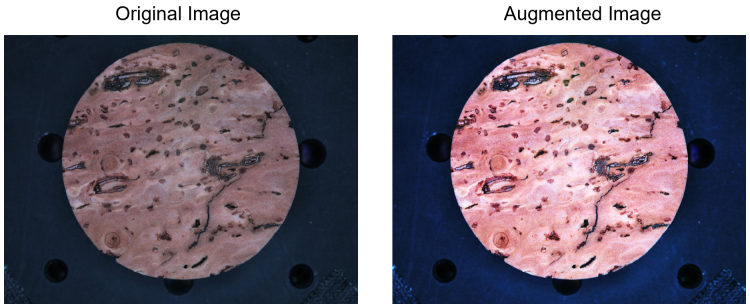
Gaussian Noise: Real world images captured with a high resolution camera may produce noisy images which is caused due to external factors like dust in the atmosphere or internal factors related to the camera. The addition of noise during the training of a deep learning model has a regularization effect and, in turn, improves the robustness of the model. Each time a training sample is exposed to the model, random Gaussian noise is added to the input image considering the fact that too

little noise has no effect on the training, whereas too much noise makes the learning process difficult.

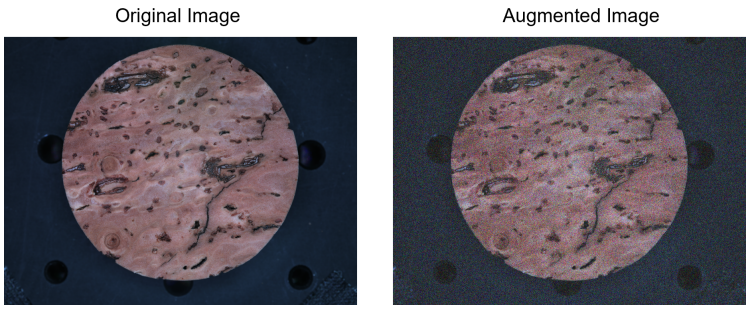
Smoothing and Blurring: Blurring is a type of image distortion that usually increases the network stability in the case of object detection and classification. Blurring artificially produces images where the object is out of focus. That is probably the reason why it usually increases stability because in real world applications the object is often times not perfectly in focus. Blurring technically means that each pixel in the image is mixed with its surrounding pixel intensities and this mixture of pixels creates the blur effect which causes the color transition in sharp image borders to become smooth. However, blurring did not work well for the cork dataset since the model needs good quality images to learn the texture and blurring is probably spoiling the texture details on the cork surface. Furthermore, the model will not have to deal with blurred images at test time.



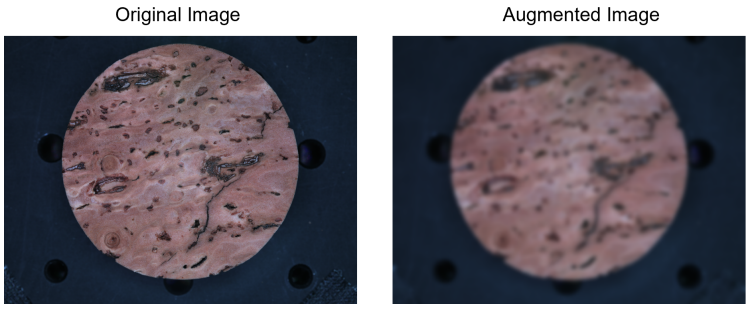
(a) Vertical Flip



(b) Brightness, Contrast and Saturation



(c) Gaussian Noise



(d) Smoothing and Blurring

Figure 10: An overview of the different augmentations applied to the cork disk images. Figure **10a** is an example of applying Vertical Flip, **10b** of applying random changes to Brightness, Contrast and Saturation, **10c** of applying Gaussian Noise, **10d** of applying Smoothing and Blurring during training time to the cork dataset images.

4.4 Performance Metrics

Choosing proper performance metrics is very important for efficient evaluation of a deep learning model. While data preparation and training a model are the key steps in the deep learning pipeline, it is equally important to measure the performance of this trained model. The model may give satisfying results when evaluated using a particular metric but may give poor results when evaluated against other metrics. Metrics are used to monitor and measure the performance of a model both during training and testing of the model. Although the training accuracy seems to be a suitable choice to measure the correctness of a model's predictions, it does not completely explain the efficiency of the model. It is, therefore, necessary to use other metrics which might help in judging the model performance better and improve the training. In this section, various metrics used in this work are discussed in detail.

Confusion Matrix: The confusion matrix is one of the most important metrics used for finding the correctness and accuracy of a trained model. It is used for the classification problems where the output can be of two or more types of classes. The confusion matrix is basically a table with two dimensions ("Actual" and "Predicted"), and each dimension has two classes, positive and negative. Every cell of the confusion matrix are explained as follows:

1. **True Positives (TP):** True positives are the cases when both the actual class and the predicted class of the data points are 1 (True).

2. **True Negatives (TN):** True negatives are the cases when both the actual class and the predicted class of the data points are 0 (False).

3. **False Positives (FP):** False positives are the cases when the actual class of the data points is 0 (False) but the predicted class for these data points is 1 (True).

4. **False Negatives (FN):** False negatives are the cases when the actual class of the data points is 1 (True) but the predicted class for these data points is 0 (False).

A false positive is like a false alarm in a test result of binary classification that incorrectly indicates the presence of a condition such as a disease when the disease is not present, while a false negative is the opposite case where the test result fails to identify the presence of a condition. To further evaluate the performance of a classification model, Precision and Recall metrics are used, which measure the model performance based on these four categories.

Accuracy: Accuracy is one of the most important metric for evaluating classification models. Informally, accuracy is the fraction of predictions that the model predicted correctly. Formally, accuracy has the following definition:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} \tag{5}$$

For binary classification, accuracy can also be calculated in terms of positives and negatives as follows:

$$Precision = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

Precision: Precision, also known as *Positive Predictive Value (PPV)*, measures the proportion of positive predictions that are correct. In other words, precision is the ratio of the number of predictions which were actually True to the total number of predictions labeled as True by the prediction model. Precision helps when the model is required to have predictions with less false positives since a high precision value implies less number of false positives. For instance, in the case of skin cancer detection, if the model has very low precision, then many patients will get a positive

report for cancer even if they do not have cancer. Mathematically, precision is defined as follows:

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

Recall: Recall, also known as *Sensitivity*, measures the percentage of predictions that are positive out of all the true observations. In other words, recall is the ratio of the number of predictions that were True to the sum of the number of predictions which were (1) True and labeled as True by the model, (2) True and labeled as False by the model. Recall helps when the model is required to have predictions with less false negatives since a high recall value implies less number of false negatives. For instance, in the case of nuclear missiles detection, if the model has very low recall, then the consequences can be devastating. Mathematically, recall is defined as follows:

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

To fully evaluate the effectiveness of a model, both precision and recall must be measured since they are inversely proportional to each other, and hence, improving precision typically reduces recall and vice versa. Every model generates some false negatives, some false positives, or possibly both. The model can be tuned to minimize one or the other, often facing a tradeoff, where a decrease in false negatives leads to an increase in false positives, or vice versa. The performance metrics should be optimized in such a way that suits the specific problem. While the metrics discussed above either consider false positive or false negative, F1 Score considers both false positive and false negative in evaluating the model performance.

F1 Score: F1 Score, also known as *Dice Score*, is the weighted average of Precision and Recall. In other words, it is defined as the harmonic mean of the model's Precision and Recall. This score considers both false positives and false negatives into account.

F1 is usually more useful than accuracy in the case of an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it is definitely better to consider both Precision and Recall. Mathematically, F1 Score is defined as follows:

$$F1\ Score = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{9}$$

Evaluating the performance of segmentation and object detection models require some additional metric. IoU (Intersection over Union) and mAP (mean Average Precision) are the important metrics used for evaluating model performance which helps in the prediction of segmentation masks and object detection (localization and classification) respectively.

Intersection over Union (IoU): Intersection over Union, also known as *Jaccard Index*, is a metric that evaluates how similar the predicted bounding box (or segmentation masks) is to the ground truth bounding box (or segmentation masks). The idea is to compare the ratio of the area where the two boxes (or masks) overlap to the total combined area of the two boxes (or masks). Mathematically, IoU metric can be expressed as:

$$IoU = \frac{prediction \cap target}{prediction \cup target} \tag{10}$$

Mean Average Precision (mAP): The mean Average Precision (mAP) is a popular metric used to measure the performance of models doing object detection tasks. AP (Average Precision) is defined as the area under the Precision/Recall curve obtained by interpolating the precision values at every recall value. Finally, the mAP for object detection is the average of the AP calculated for all the classes. Usually, a prediction with an IoU score of 0.5 and above is considered as a positive prediction. However,

calculations can also be done by thresholding the bounding box at different IoUs depending on how strict the predictions are required for the given problem.

All the metrics, that are discussed above, have been used in this work for classification, detection and segmentation of wooden anomalies on cork disk surfaces. Choice of proper metrics influences the performance of deep learning models and it is a key requirement to measure the model performance and compare different algorithms to understand which one works the best.

4.5 Experiment Environment

Google's free cloud platform "Google Colab" [61] and a local workstation is used for running all the experiments mostly related to training, evaluation and testing of the deep learning models. The local machine runs on an Ubuntu 18.04.5 LTS operating system. It is equipped with NVIDIA Quadro RTX 6000 (24 GB of memory) and NVIDIA Quadro K6000 (12 GB of memory) GPUs whereas Google Colab offers GPUs that vary over time and one may get access to any of the following Nvidia K80s, T4s, P4s, and P100s GPUs. Since there is also no guarantee that some specific amount of memory will be available every time, the local workstation is used for most of the experiments. To maintain the consistency of library versions and to separate the experimental setup from the installed softwares and packages of the local workstation, a customized docker file is built and created for running the experiments. For most part of the experiments, a docker with Jupyter notebook is used.

5 Results

The outcome of all the experiments conducted in this thesis are discussed in this chapter in detail. Each of the upcoming sections discusses the results for every experiment and gives an insight about the possible solutions for the problem.

5.1 Quality Classification

As discussed in Chapter **4.2.2**, the experiments with the cork disks dataset aim to find a trade-off between model capacity and the number of layers fine-tuned to find the best performing model with maximum accuracy, low training time and low evaluation time. As a starting point, EfficientNetB0 pre-trained on ImageNet was trained, fine-tuning only the last three layers to figure out whether the model is capable of learning the quality classes. In Figure **11** the loss and accuracy curves are shown respectively. It can be inferred from the figure that the network starts learning and converging even on datasets with low resolution (224 x 224), and therefore, training on higher resolution EfficientNet networks might produce better results.

To get the best performance, five variants of EfficientNet from B0 to B4 were trained. The loss and accuracy curves for the networks B0 through B4 are shown in Figure **12**. As can be seen from the figure, both EfficientNetB3 and EfficientNetB4 have similar performance with EfficientNetB4 performing slightly better on the validation accuracy. However, EfficientNetB3 was trained for a longer period of time and that

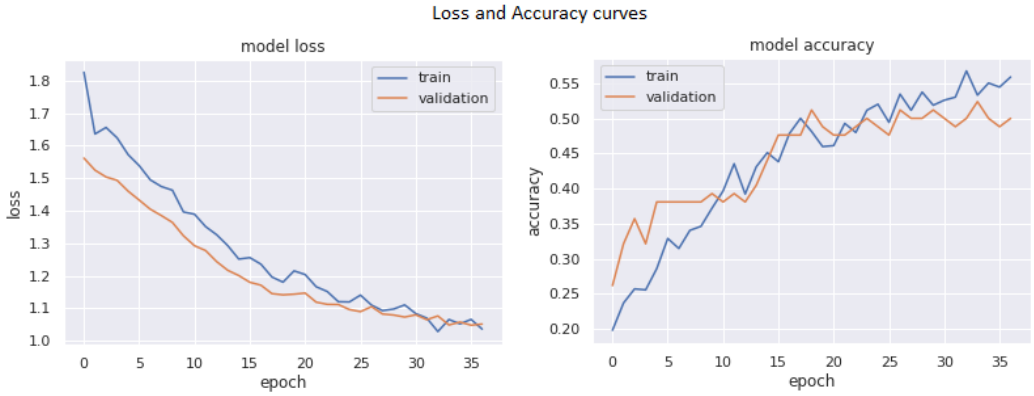


Figure 11: Loss and accuracy curves from the training and validation of the EfficientNet-B0 model fine-tuning only the last 3 layers.

might be the reason it performed well on the test dataset. Therefore, EfficientNetB3 was selected for further experiments due to its good performance on test dataset based on the accuracy and inference time. To better adapt to the network pre-trained on ImageNet, the initial layers were frozen and the last few layers were fine-tuned depending on how much capacity the model needs to predict the classes. This is an important step in transfer learning where the initial layers have already learned low-level features like edges and blobs, and only the last few layers are needed to be fine-tuned according to the current problem. As can be seen in Figure **13**, fine-tuning the last four blocks (B4-B7) along with the TOP layer gives the best performance.

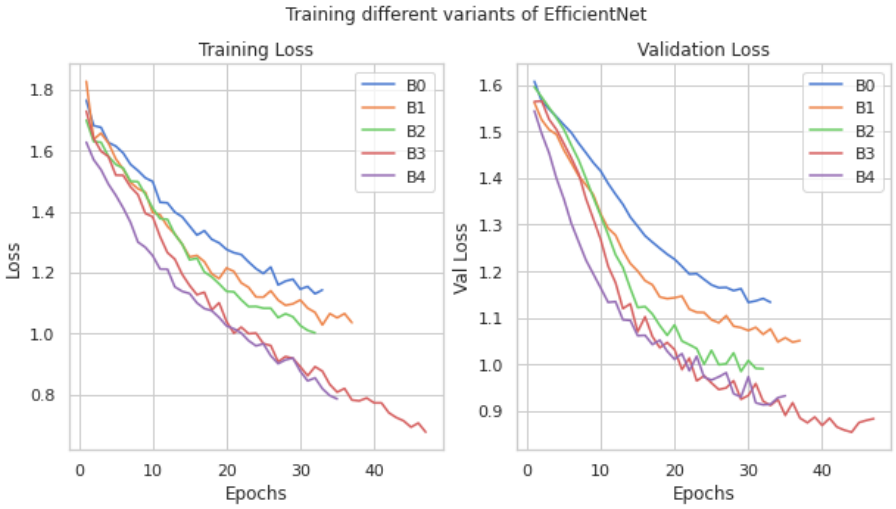
To further improve the model performance from the baseline values, hyperparameter optimization was done on the EfficientNet-B3 network. The hyperparameter space consisted of (1) learning rate, (2) number of blocks to fine-tune, (3) optimizer and a (4) dropout value in the final classification layer. The hyperparameters were searched based on Random Search technique. A total of 30 trials were run and for each trial there were three executions to validate the current combination of hyperparameters. Every trial took approximately 15 minutes to complete and the whole hyperparameter search took nine hours to finish. The algorithm returns the top-10 trials from the experiment, the best of which was used for the final training. The best validation

Class	Precision	Recall	F1 Score	Support
Class 0	0.76	0.64	0.69	17
Class 1	0.55	0.59	0.57	17
Class 2	0.57	0.44	0.49	16
Class 3	0.59	0.76	0.66	21
Class 4	0.83	0.86	0.84	21
accuracy			0.66	92
macro avg	0.66	0.66	0.65	92
weighted avg	0.66	0.66	0.65	92

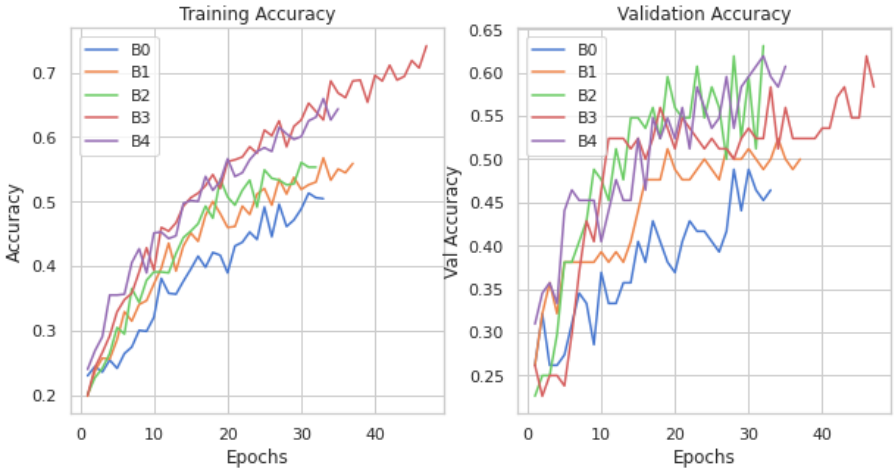
Table 1: Performance overview of the EfficientNet-B3 model on test data trained with the suggested hyperparameters from Random Search algorithm.

accuracy obtained was 0.7011. The hyperparameter values obtained from the Random Search were: (1) a learning rate of 9.21e-5, (2) four blocks to fine-tune along with the top layer (TOP + B7 + B6 + B5 + B4), (3) ADAM optimizer and (4) and a dropout rate of 0.4. The final model was trained with a batch size of 32 and for 60 epochs. There was a significant improvement seen in the accuracy, especially for the prediction of class 3 and class 4. There are certain other factors that also determine the relegation of cork disks to inferior classes like size of hidden wood inclusions present inside the cork disks and also physical features of cork material, which are hard to predict from the surface images only. This is where the model gets confused while training and it is difficult to get very accurate predictions for all the classes. However, the model learns to predict classes depending on the textures and patterns on the cork disk surfaces as can be seen in Figure **15**. The figure shows a Gradient-weighted Class Activation Mapping (Grad-CAM) technique in which the final layer of the network is visualized. It visualizes the parts of an image that lead to the highest activation in the network, so that we know where the model is looking for making the predictions.

The performance of the EfficientNet-B3 model on test data trained with the suggested hyperparameters from Random Search algorithm are shown in Table **1** and the confusion matrix for the test dataset is shown in Figure **14**.

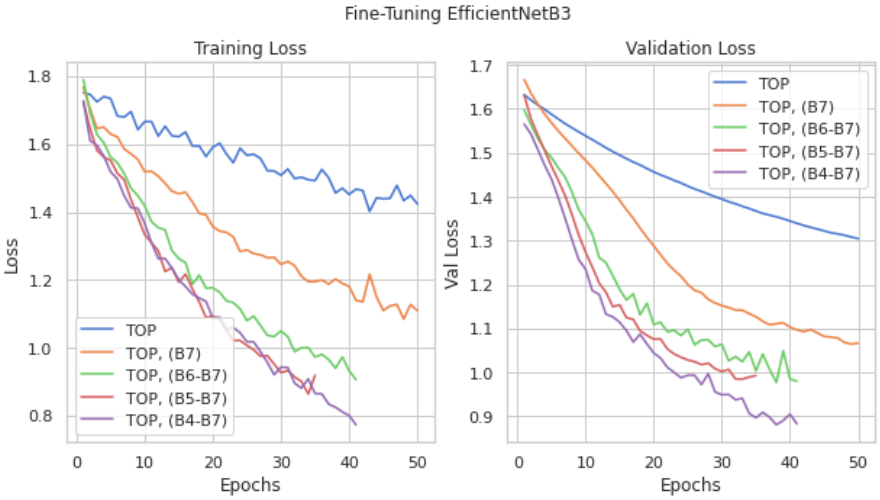


(a) Loss Curves

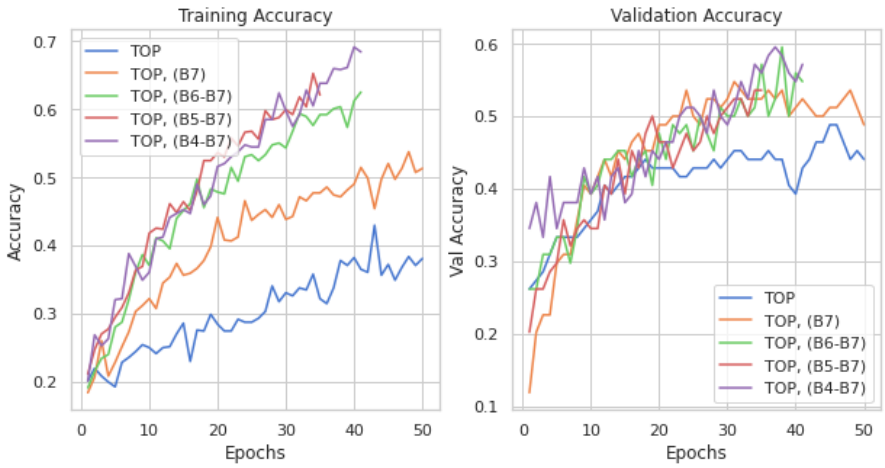


(b) Accuracy Curves

Figure 12: Loss and Accuracy curves for comparison of different variants in EfficientNet. Some curves cut-off earlier than 50 epochs because of Early Stopping.



(a) Loss Curves



(b) Accuracy Curves

Figure 13: Loss and Accuracy curves from the Training and Validation of EfficientNet-B3 model. The performance of the different number of layers fine-tuned during training is compared.

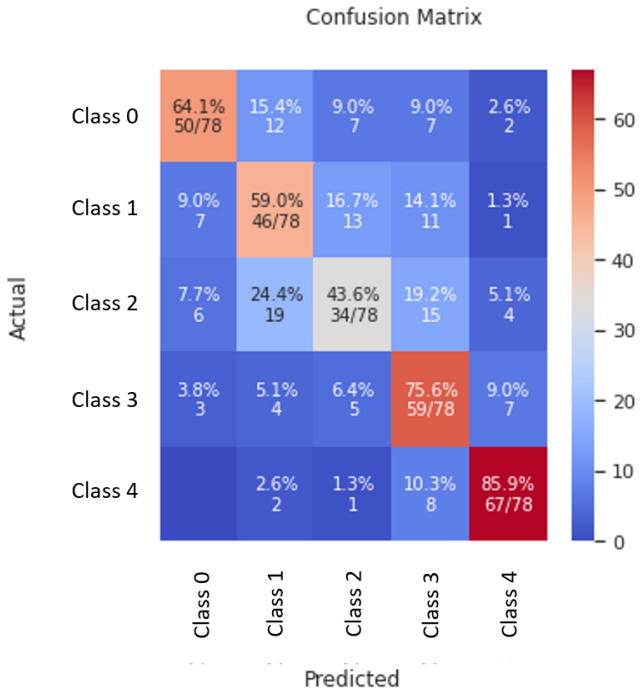
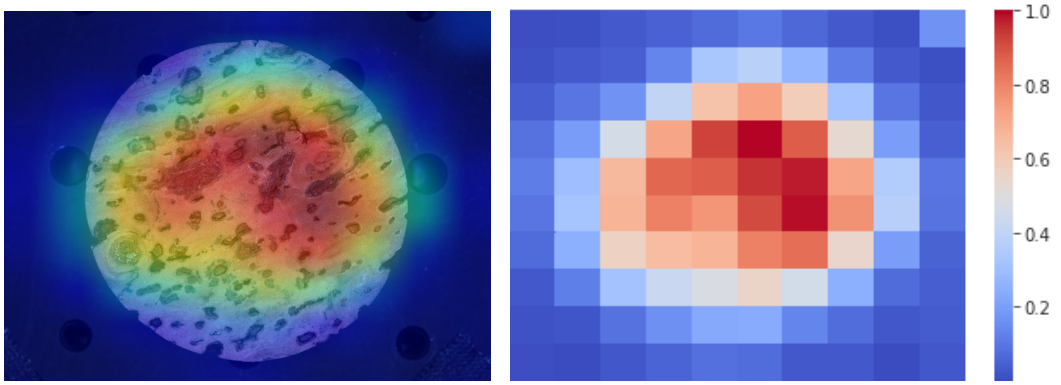


Figure 14: Confusion Matrix of EfficientNet-B3 model on the test dataset for five quality classes of cork disk dataset.



(a) Class 4 Cork Disk **(b)** Grad-CAM Heatmap for (a)

Figure 15: Visualizing predictions of EfficientNet-B3 model for a Class 4 cork disk using Grad-CAMs.

5.2 Anomaly Detection and Localization

Anomaly detection with Faster R-CNN was the first approach that was implemented as a baseline to deal with the problem of predicting wood inclusions on cork disk surfaces as discussed in Chapter **4.2.3**. In this section, the results of Faster R-CNN predictions are discussed in detail. Table **2** shows the accuracy for training Faster R-CNN with different backbone networks. An example of Faster R-CNN prediction is shown in Figure **16**. In spite of low availability of data and training Faster R-CNN only for 30 epochs, the predictions on test dataset were quite motivating to conduct further research in this area.

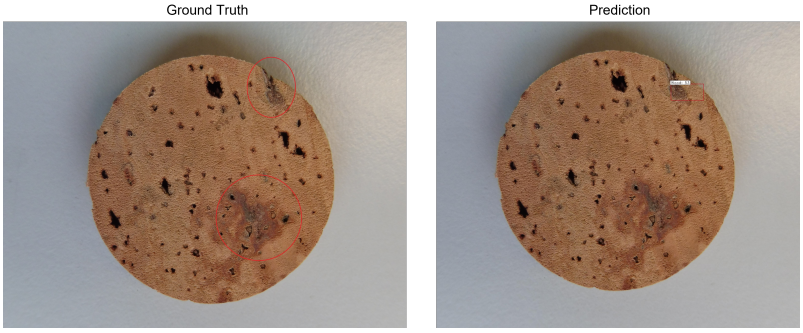


Figure 16: An example of Faster R-CNN prediction on a cork disk image captured with a digital camera.

Backbone	mAP	pTime (s)
ResNet50	0.193	1.44
VGG16	0.172	1.31
ResNet101	0.250	1.76

Table 2: Performance overview of Faster R-CNN on test data trained with different backbone models. Prediction time is marked with pTime.

5.3 Anomaly Segmentation

Segmentation is the most important task for analyzing the wood inclusion anomaly on the cork disks. This whole section discusses the results obtained from training

U-Net and Mask R-CNN with different strategies. Finally, the two frameworks were compared and an ensemble method was used to combine both the predictions in such a way that it outperforms both the models by a significant margin.

5.3.1 U-Net

In this section, the results for training U-Net with different deep learning networks are discussed. Since U-Net has an encoder-decoder architecture, it is very important to use a proper feature extractor for the encoder network. In this work, a couple of different models and their variants have been used to train the U-Net based on their performance on ImageNet dataset. EfficientNet models have stable performance which achieve both higher accuracy and better efficiency in general as compared to other deep learning models even with less number of parameters. EfficientNet reduces compute cost, battery usage, and also training and inference times. ResNet model was also chosen for this experiment since ResNet can scale well with deeper networks due to the residual blocks. Apart from EfficientNet and ResNet, other models have also been tried for experimental purposes such as ResNeXt and DenseNet. The training, validation and test accuracy are mentioned in Table **3**. As can be seen from the table, EfficientNet-B2 performed better than all the other models. In Figure **18** the loss and accuracy curve for training and validation of U-Net with different models are shown.

U-Net was also trained both as binary segmentation (with Sigmoid activation) and multi-class segmentation (with Softmax activation). For multi-class segmentation, the 'background' was considered as the second class. Table **4** shows that training the background worked better than training only the anomalous inclusions because it is important to learn the cork texture to differentiate it from the anomaly. Furthermore, training without augmentation performed better than training with augmentation as can be seen from the Table **5**. Augmentation was not very important in this case since in reality the lighting conditions and resolutions would be fixed and therefore

Model	Type	Train			Val			Test		
		Loss	IoU	Dice	Loss	IoU	Dice	Loss	IoU	Dice
EfficientNet	B0	0.12	0.81	0.87	0.23	0.78	0.84	0.21	0.73	0.80
	B1	0.11	0.80	0.88	0.21	0.77	0.82	0.21	0.71	0.81
	B2	0.12	0.82	0.88	0.23	0.79	0.84	**0.19**	**0.74**	0.80
	B3	0.15	0.78	0.85	0.25	0.75	0.80	0.22	0.71	0.78
ResNet	18	0.14	0.79	0.86	0.23	0.77	0.82	0.69	0.32	0.35
	34	0.17	0.76	0.83	0.25	0.73	0.78	0.65	0.30	0.37
	50	0.14	0.80	0.87	0.25	0.77	0.81	2.93	0.01	0.09
	101	0.17	0.75	0.83	0.26	0.71	0.77	1.0	0.05	0.06
	152	0.19	0.75	0.82	0.25	0.76	0.82	**0.24**	**0.69**	**0.76**
ResNeXt	50	0.65	0.28	0.36	0.73	0.28	0.34	0.71	0.24	0.31
	101	0.77	0.21	0.29	0.81	0.22	0.26	0.92	0.13	0.18
DenseNet	121	0.64	0.30	0.37	0.79	0.24	0.29	0.71	0.24	0.31
	169	0.61	0.33	0.35	0.77	0.23	0.28	0.71	0.25	0.31
	201	0.67	0.27	0.34	0.72	0.29	0.38	0.70	0.22	0.29

Table 3: Results obtained from training U-Net with different backbone models.

augmenting the images with noise and blur actually spoils the training process. An example of a prediction from the best performing U-Net model is shown in Figure **17**.

Experiment	Loss	IoU	Dice
Binary Segmentation	0.42	0.49	0.61
Multi-class Segmentation	0.22	0.71	0.78

Table 4: Experiments showing the training of U-Net as binary and multi-class segmentation.

Experiment	Loss	IoU	Dice
Augmentation	0.71	0.24	0.30
Without Augmentation	0.21	0.73	0.80

Table 5: Experiments showing the effect of data augmentation during the training of U-Net. These results are obtained from the performance on the test dataset.

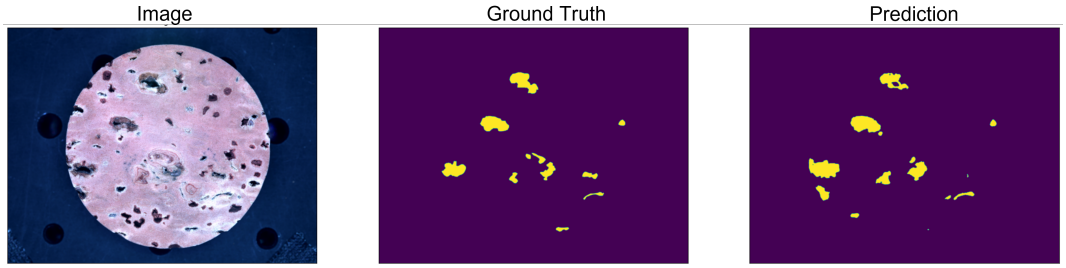
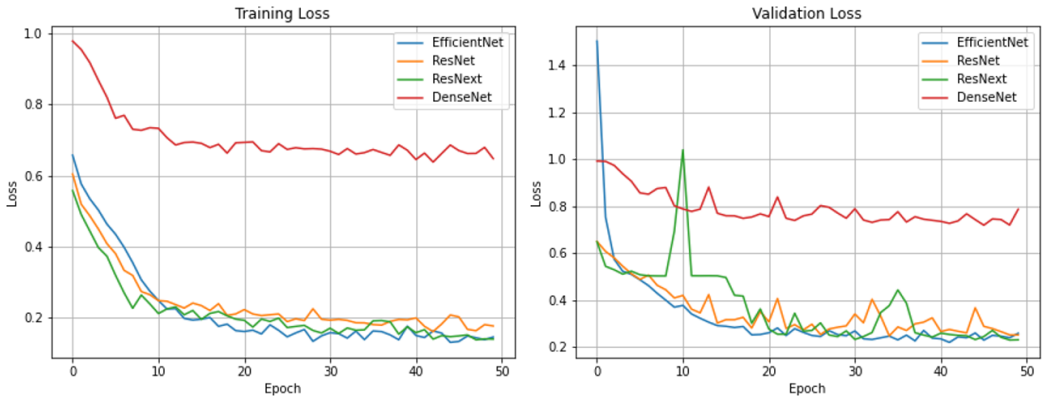
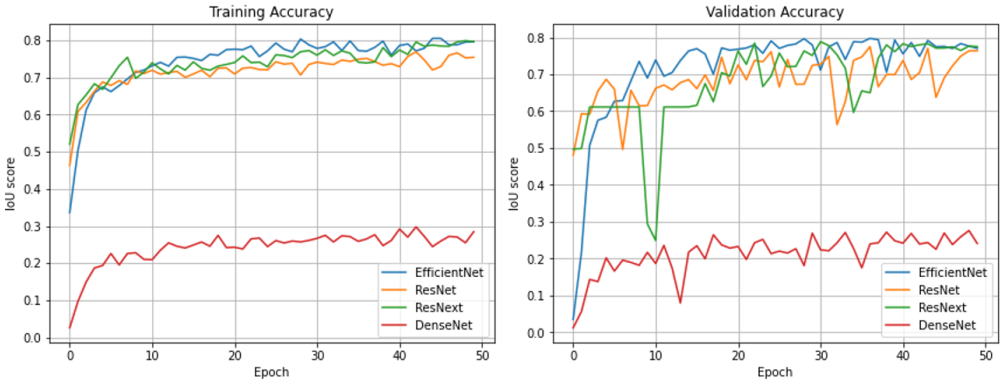


Figure 17: An example of a U-Net prediction on a Class 4 cork disk.



(a) Loss Curves



(b) Accuracy Curves

Figure 18: Loss and Accuracy curves for training U-Net with different backbone encoder models.

5.3.2 Mask R-CNN

In this section, the results for training Mask R-CNN with different strategies are discussed. Different approaches were taken to train Mask R-CNN keeping the hyperparameters fixed as discussed in Chapter **4.2.4**. The backbone layers of Mask R-CNN can be trained in different combinations keeping the initial layers frozen. They are denoted by: (1) *all* for training all the layers, (2) *3+*, (3) *4+* and (4) *5+* for training ResNet layer stages 3, 4, 5 and above, respectively, and (5) *heads* for training the last layer only. This notation has been used throughout the remainder of this section to denote the number of layers being trained in a pre-trained network. The overall loss curves for training Mask R-CNN with different layer combinations are shown in Figure **20a**, the classification loss curves are shown Figure **20b**, the bounding box loss curves are shown in Figure **20c** and the mask loss curves are shown in Figure **20d**. As can be seen in the loss curves, the model performs well with fine-tuning *3+* layers. Training all the layers also shows a similar performance for the validation dataset but it does not perform well on the test dataset.

To get the best out of Mask R-CNN, a couple of experiments were conducted to decide which strategy works the best. The results of the comparison of different experiments performed on Mask R-CNN are summarized in Table **6**. The highest values in the table for every set of experiments are highlighted in bold. The evaluation of Mask R-CNN was performed with the help of a few performance metrics like precision, recall, Intersection over Union (IoU) and mAP. In particular, IoU shows the accuracy of segmentation while precision, recall and mAP show the accuracy for detection. The first row of the table discusses the performance of ResNet from fine-tuning different layers as mentioned in the last paragraph. It shows that the model tuned with *Blocks 3+* performs relatively well. This explains that the network requires more capacity for training, although fine-tuning all the layers might not be beneficial since it can interfere with the already pre-trained weights for detecting low level features. This can be a probable reason why the performance drops when all the layers are

fine-tuned.

Regarding augmentation strategies, mainly two variations were experimented. Heavy augmentation consists of images with vertical/horizontal flips, rotation, translation, brightness, saturation, noise and blur. Moderate augmentation consists of only vertical/horizontal flips, rotation, scale and shear transform. The model was also trained without any augmentation. It can be well understood from the table that training the model with moderate augmentation has performed well. Heavy augmentation did not work well probably because too much augmentations like noise and blur could interfere with the learning process of wood inclusion patterns, since the model needs to learn the anomalous textures on the cork disk image instead of a particular shape, which is usually the case in other object detection scenario.

The model was also trained with two types of dataset split as discussed in Chapter **4.2.1**. Type I consists of training data having all the quality classes in equal proportion whereas Type II consists of training data where Class 3 and Class 4 were removed from the training dataset. Type II performed better than Type I dataset group probably because Type I dataset is noisy and has labeling issues.

As can be seen in the table, the model trained with pre-trained weights of MS COCO dataset worked better than the one trained with ImageNet dataset in terms of recall, IoU score and average precision. Furthermore, experiments were also conducted with different backbone architectures from shallow to deep network to figure out the model capacity required for the cork dataset. ResNet101 gave the best performance indicating that the cork dataset requires a deep network to learn the features.

Finally, to get the best outcome, the early stopping was implemented, and the model with the lowest loss was chosen to perform the evaluation on the test dataset. Thus, the final model was trained with *Blocks 3+* fine-tuned layers, moderate augmentation, Type II dataset, COCO dataset pre-trained weights and ResNet101 backbone. An example of Mask R-CNN prediction is shown in Figure **19**.

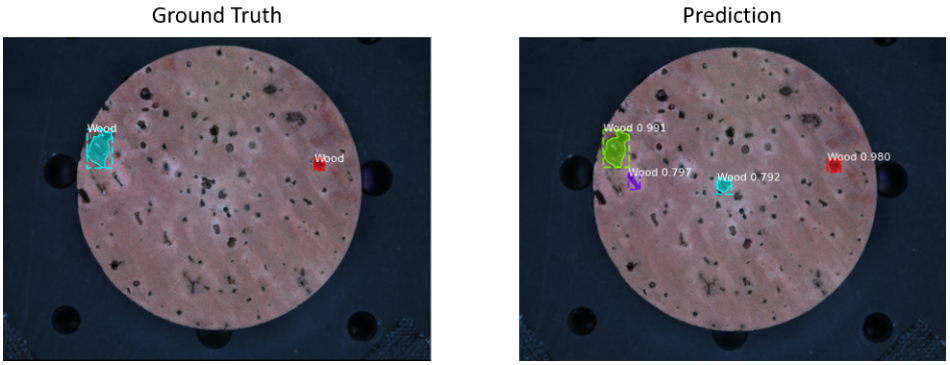
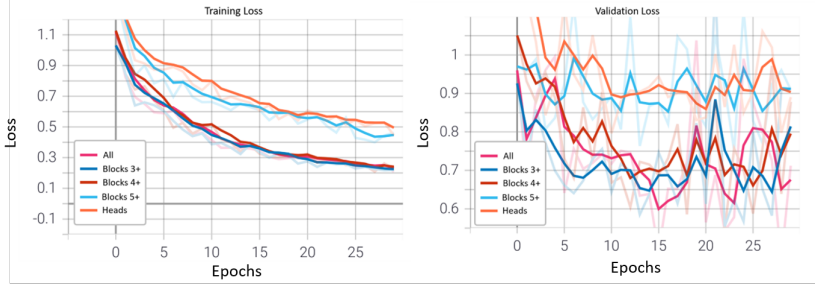


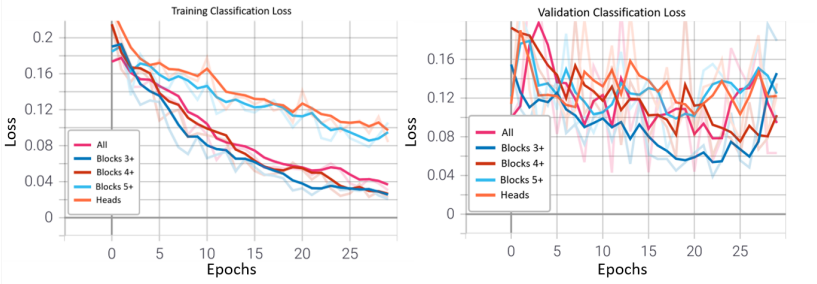
Figure 19: An example of Mask R-CNN predictions on a Class 2 cork disk.

Experiment	**Type**	Precision	Recall	IoU	mAP@0.5	mAP@0.5-0.95
Layer	Heads	0.330	0.771	0.451	0.501	0.240
	Blocks 5+	0.432	0.777	0.479	0.549	0.311
	Blocks 4+	0.486	0.805	0.498	0.559	0.320
	Blocks 3+	**0.617**	0.808	**0.551**	**0.626**	**0.402**
	All	0.479	**0.832**	0.543	0.611	0.391
Augmentation	Heavy	0.289	0.406	0.320	0.327	0.182
	Moderate	**0.489**	**0.859**	**0.563**	**0.658**	**0.418**
	None	0.479	0.832	0.543	0.611	0.391
Dataset Split	Type I	0.215	0.842	0.512	0.566	0.323
	Type II	**0.489**	**0.859**	**0.563**	**0.658**	**0.418**
Pre-trained Weights	ImageNet	**0.380**	0.822	0.591	0.620	0.335
	COCO	0.277	**0.876**	**0.645**	**0.639**	**0.368**
Backbone	MobileNetV1	0.056	0.193	0.019	0.024	0.004
	MobileNetV2	0.113	0.419	0.238	0.135	0.046
	ResNet50	**0.495**	0.772	0.540	0.592	0.331
	ResNet101	0.375	**0.918**	**0.608**	**0.678**	**0.424**

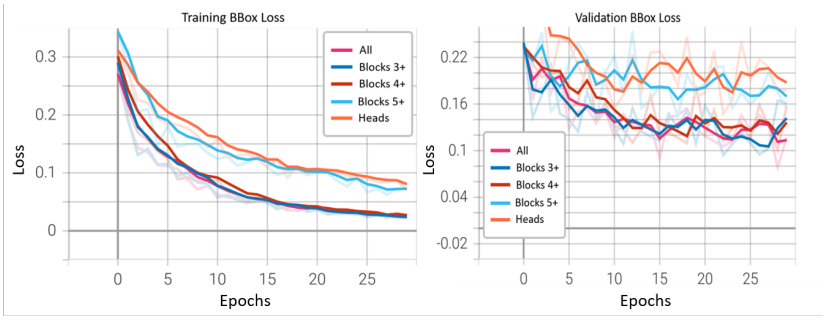
Table 6: Performance of Mask R-CNN on test dataset based on training with different strategies. The experiments were conducted row-wise from top to bottom as they are arranged in the table. In the first row, fine-tuning different layers of ResNet101 pre-trained on COCO datasets were done with Type II dataset and no augmentation for 30 epochs. The last two rows show experiments with increased epochs of 60 and 90 respectively.



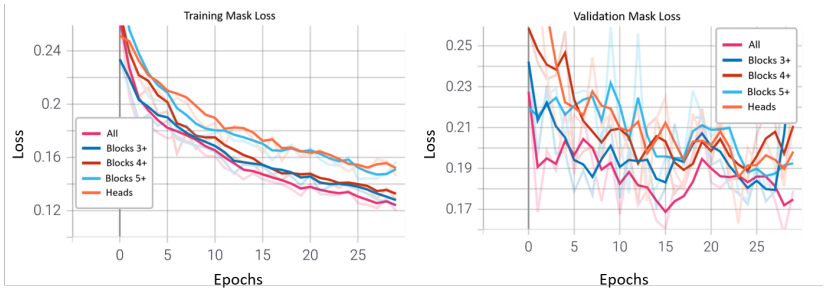
(a) Overall Loss



(b) Classification Loss



(c) Bounding Box Loss



(d) Mask Loss

Figure 20: Different loss curves for fine-tuning Mask R-CNN model on cork disk dataset.

5.3.3 Comparison

The results obtained from U-Net and Mask R-CNN were compared to decide which model works best for the cork dataset. While comparing, it was observed that U-Net had a good precision but apparently it failed to capture accurately the shape of the anomalies. On the other hand, Mask R-CNN had a good recall value and also the segmentation boundaries of the wooden inclusions proved to be more effective and accurate than that of the U-Net. But a downside of Mask R-CNN was that it predicted a lot of false positives which is bad for the prediction task since it will mark the good corks as bad corks. Another problem in the U-Net predictions was that the segmentation borders were not accurate and sharp like Mask R-CNN. To handle this situation, the ensemble method was experimented at the cost of evaluation time to exploit the strengths and avoid the failures of both the models.

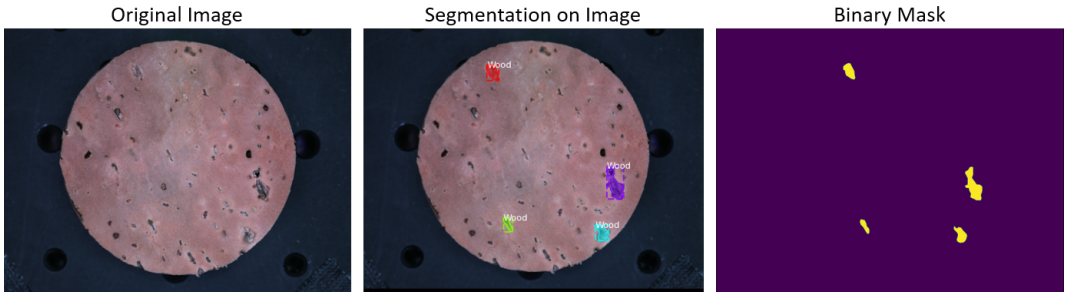
5.3.4 Ensemble Learning

In this section, the results obtained from ensemble learning are discussed. Table **7** shows the difference in the performance of U-Net, Mask R-CNN and ensemble learning. After running Grid Search and Bayesian Optimization on the weights of the ensemble model, the optimized weights were found to be 0.625 and 0.375 for U-Net and Mask R-CNN model respectively. It is observed that the two models ensemble using weighted soft voting shows a significant improvement in precision, IoU score and dice score. The improvement in precision is quite significant since a prediction with a low recall value does not hurt much but a low precision value is unacceptable as this might discard some good corks. Ensemble learning gave a better performance with the whole prediction process taking only about 400 ms approximately. Therefore, combining the U-Net and Mask R-CNN model gives the opportunity to improve the model accuracy at the cost of prediction time. Figure **21** shows the predictions of U-Net, Mask R-CNN and Ensemble for a Class 2 cork disk

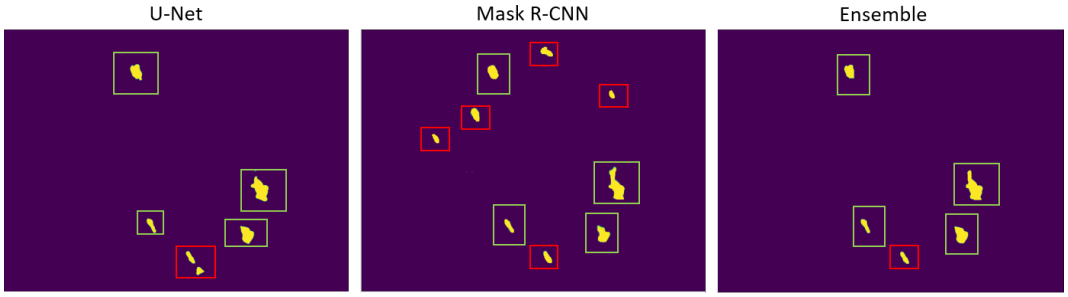
	Precision	Recall	IoU	Dice	pTime (s)
U-Net	0.401	0.554	0.740	0.803	0.11
Mask R-CNN	0.375	0.918	0.608	0.532	0.18
Ensemble	**0.462**	0.625	**0.772**	**0.819**	0.46

Table 7: Performance overview for U-Net, Mask R-CNN and Ensemble models at IoU of 0.5. Prediction time is marked with pTime.

image, which indicates that the ensemble prediction helps in reducing false positives and improves the segmentation quality. The analysis of cost efficiency shows that the ensemble of the two segmentation models typically increases the use of memory and computing power, but also shows that the ensemble method can perform better than a single DL-based segmentation model with less memory and computing power.



(a) Ground Truth



(b) Predictions

Figure 21: An example of a prediction obtained from weighted average ensemble of U-Net and Mask R-CNN models. In Figure **21b**, true positives are marked by green and false positives are marked by red.

6 Discussion

As discussed before, the new anomaly detection system based on computer vision can be used in combination with the old X-Ray detection system at Amorim or even without it. The results from the experiments without the X-Ray indicate that there are still chances of improvement and further research can be conducted in this project. The initial results from Faster R-CNN were quite promising which led to the further development of this project. The fact that a CNN network could identify and localize the wood inclusion patterns on the cork surface helped the project to research in more advanced frameworks and deep learning architectures. The results from the EfficientNet used for classifying the cork disks suggest that it is hard to differentiate between the good quality corks (Class 0 to Class 2) but the model can classify the bad quality corks (Class 3 and Class 4) very well. U-Net and Mask R-CNN were quite successful in detecting wood inclusions and segmenting those anomalies but at the cost of few false positives and inaccurate segmentation. However, the ensemble method helped to reduce some false positives and improve the segmentation quality.

There were certain limitations that caused a hindrance in this project and affected the results to a great extent. Firstly, the dataset size was too small to achieve high accuracy on the classification task. Secondly, to classify the cork disks, only the surface images were used to train the model and make the predictions. However, there were certain other factors that also determine the class of cork disks, such as, wooden inclusions inside the cork disks and physical properties of the cork material, which are hard to predict from the surface images only. In the future, the results from

the X-Ray could be combined with the predictions from the classification model to get a more robust prediction on the classes. Finally, a major problem of this research was the noisy dataset. The annotations for Class 3 and Class 4 dataset were often times inaccurate and had massive amounts of label noise. This led to the removal of Class 3 and Class 4 dataset from the training data, resulting in further reduction in the dataset size. The dataset size was too small to train a model from scratch and therefore transfer learning was employed in the training. However, due to the unavailability of a pre-trained model trained on datasets similar to cork, models pre-trained on COCO and ImageNet datasets were used since they have images with textures or patterns similar to the cork disk. Furthermore, there were also hardware limitations to train a model with a huge dataset from scratch which was also a reason to resize the dataset during training since the images with the original resolution would require more GPU resources.

As compared to the previous works done on texture analysis and pattern detection, the current problem for anomaly detection is hard to resolve completely since the model needs to learn the anomalous texture which looks very similar to the normal patterns and holes on the cork surface and also the dataset size is small. However, the initial results of the image processing pipeline look promising and further research in this area can be done in the future.

7 Conclusion

This research aimed to automate the quality inspection and monitoring system of cork products with the help of computer vision techniques. The main objective was to reduce the human effort, the operation cost and the evaluation time. Based on the quantitative and qualitative analysis of the prediction results, it can be concluded that the proposed image processing system is successful in learning the feature representations of cork disk images and their anomalies. The results indicate that these learned feature representations have captured the cork texture and anomaly pattern very well and they can also generalize well with all the classes of cork disks.

To predict the quality class of a cork disk, an EfficientNet model was used since it has achieved both higher accuracy and better efficiency over other existing CNNs on ImageNet dataset with reduced parameter size. After experimenting with the model architecture and comparing their results, hyperparameter optimization was done and the best model achieved an accuracy of 66% on the test dataset.

To detect and localize the wood inclusions on the cork disks, a Faster R-CNN model was used, the results of which concluded that a CNN can learn the feature representations of cork disk surface images. To estimate the area of wood inclusions, image segmentation was performed with U-Net and Mask R-CNN. Mask R-CNN achieved a mAP of 0.678 at an IoU value of 0.5, whereas U-Net achieved an average IoU score of 0.74 on the test dataset. Finally, the results from both the models were combined with the weighted average ensemble method and the final prediction

achieved an average IoU score of 0.772 and a Dice score of 81.9%. For the post-processing part, these results are stored in JSON files which are used by the software to visualize the predictions on the cork disks. The detailed study done on different model architectures of U-Net and Mask R-CNN illustrates the possibility of further improvement and research in this field. Additionally, the results obtained in this research can be useful to other surface defect detection researches based on wooden materials because they have common characteristics with the cork.

8 Acknowledgments

This project would not have been possible without the support of many people.

First and foremost, I would like to express my gratitude to Prof. Dr. Thomas Brox for being very flexible and approachable throughout this thesis.

I owe a great deal to my adviser David Joel, who has been very supportive and patient with me these past few months. My sincerest thanks to you for all the help and guidance you provided along the way.

I would also like to thank my colleagues at Fraunhofer IPM who kept the workplace alive with ideas, making it a wonderful place to work. Special thanks to Manav and Sharique for all the great discussions and talks that indeed proved very helpful.

A big thank you to my parents, for it was their kind words and encouragement that kept me going for this long. And lastly, I want to thank my girlfriend Sherry for being a constant support throughout my work.

Bibliography

[1] D. H. HUBEL and T. N. WIESEL, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of physiology*, vol. 160, no. 1, pp. 106–154, 1962.

[2] X. Wang, H. Gong, H. Zhang, B. Li, and Z. Zhuang, "Palmprint identification using boosting local binary pattern," in *18th International Conference on Pattern Recognition (ICPR'06)*, vol. 3, pp. 503–506, 2006.

[3] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, vol. 2, 06 2005.

[4] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," vol. 3951, pp. 404–417, 07 2006.

[5] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[6] N. O'Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. V. Hernandez, L. Krpalkova, D. Riordan, and J. Walsh, "Deep learning vs. traditional computer vision," in *Advances in Computer Vision* (K. Arai and S. Kapoor, eds.), (Cham), pp. 128–144, Springer International Publishing, 2020.

[7] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893 vol. 1, 2005.

[8] D. Lowe, "Object recognition from local scale-invariant features," vol. 2, pp. 1150 – 1157 vol.2, 02 1999.

[9] L. Wu, S. C. H. Hoi, and N. Yu, "Semantics-preserving bag-of-words models and applications," *IEEE Transactions on Image Processing*, vol. 19, no. 7, pp. 1908–1920, 2010.

[10] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Computation*, vol. 1, pp. 541–551, Dec. 1989.

[11] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[12] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov. 1998.

[13] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Neural Information Processing Systems*, vol. 25, 01 2012.

[14] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li, "Imagenet large scale visual recognition challenge," *CoRR*, vol. abs/1409.0575, 2014.

[15] T. Guo, J. Dong, H. Li, and Y. Gao, "Simple convolutional neural network on image classification," pp. 721–724, 03 2017.

[16] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," *CoRR*, vol. abs/1708.06519, 2017.

[17] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[18] M. Hussain, J. Bird, and D. Faria, "A study on cnn transfer learning for image classification," 06 2018.

[19] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[20] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *MM 2014 - Proceedings of the 2014 ACM Conference on Multimedia*, 06 2014.

[21] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *NeurIPS*, 2019.

[22] F. Seide and A. Agarwal, "Cntk: Microsoft's open-source deep-learning toolkit," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowl-*

edge Discovery and Data Mining, KDD '16, (New York, NY, USA), p. 2135, Association for Computing Machinery, 2016.

[23] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," 2001.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Los Alamitos, CA, USA), pp. 770–778, IEEE Computer Society, jun 2016.

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

[26] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 05 2019.

[27] 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.

[28] R. B. Girshick, "Fast R-CNN," *CoRR*, vol. abs/1504.08083, 2015.

[29] 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.

[30] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *CoRR*, vol. abs/1703.06870, 2017.

[31] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[33] J. Lima and P. Costa, "Real-time cork classification method: A colour image processing approach," *Int. J. Fact. Autom. Rob. Soft Comput.*, vol. 2, 01 2006.

[34] F. Lopes, H. Pereira, F. G. De Natale, F. Tintrup, D. D. Giusto, and G. Vernazza, "Cork pores and defects detection by morphological image analysis," in *1996 8th European Signal Processing Conference (EUSIPCO 1996)*, pp. 1–4, 1996.

[35] J.-K. Park, B.-K. Kwon, J.-H. Park, and D. Kang, "Machine learning-based imaging system for surface defect inspection," *International Journal of Precision Engineering and Manufacturing-Green Technology*, vol. 3, pp. 303–310, 07 2016.

[36] B. Staar, M. Lütjen, and M. Freitag, "Anomaly detection with convolutional neural networks for industrial surface inspection," vol. 79, 07 2018.

[37] T. He, Y. Liu, C. Xu, X. Zhou, Z. Hu, and J. Fan, "A fully convolutional neural network for wood defect location and identification," *IEEE Access*, vol. 7, pp. 123453–123462, 2019.

[38] A. Urbonas, V. Raudonis, R. Maskeliunas, and R. Damasevicius, "Automated identification of wood veneer surface defects using faster region-based convolutional neural network with data augmentation and transfer learning," *Applied Sciences*, vol. 9, p. 4898, 11 2019.

[39] J. Shi, L. Zhenye, T. Zhu, D. Wang, and C. Ni, "Defect detection of industry wood veneer based on nas and multi-channel mask r-cnn," *Sensors*, vol. 20, p. 4398, 08 2020.

[40] J. Lima and P. Costa, "A modular approach to real-time cork classification using image processing," vol. 2, pp. 8 pp. – 368, 10 2005.

[41] J. R. Gonzalez-Adrados and H. Pereira, "Classification of defects in cork planks using image analysis," *Wood Science and Technology*, vol. 30, pp. 207–215, 06 1996.

[42] A. Costa and H. Pereira, "Quality characterization of wine cork stoppers using computer vision," *Journal International des Sciences de la Vigne et du Vin*, vol. 39, pp. 209–218, 10 2005.

[43] V. Oliveira, S. Knapic, and H. Pereira, "Natural variability of surface porosity of wine cork stoppers of different commercial classes," *Journal International des Sciences de la Vigne et du Vin*, vol. 46, pp. 331–340, 10 2012.

[44] V. Ristiawanto, B. Irawan, and C. Setianingsih, "Wood classification with transfer learning method and bottleneck features," in *2019 International Conference on Information and Communications Technology (ICOIACT)*, pp. 111–116, 2019.

[45] "Kaggle: data science bowl:https://www. kaggle.com/c/data-science-bowl-2018," 2018.

[46] A. O. Vuola, S. U. Akram, and J. Kannala, "Mask-rcnn and u-net ensembled for nuclei segmentation," *CoRR*, vol. abs/1901.10170, 2019.

[47] Y. Li, X.-R. Huang, Y. Wang, Z. Xu, Y. Sun, , and Q. Zhang, "U-net ensemble model for segmentation inhistopathology images," 2019.

[48] Y.-W. Kim, Y.-C. Byun, and A. V. N. Krishna, "Portrait Segmentation Using Ensemble of Heterogeneous Deep-Learning Models," *Entropy*, vol. 23, p. 197, Feb. 2021.

[49] A. Dutta and A. Zisserman, "The VGG image annotator (VIA)," *CoRR*, vol. abs/1904.10699, 2019.

[50] P. Skalski, "Make Sense." `https://github.com/SkalskiP/make-sense/`, 2019.

[51] W. Abdulla, "Mask r-cnn for object detection and instance segmentation on keras and tensorflow." `https://github.com/matterport/Mask_RCNN`, 2017.

[52] K. Bardool, "keras-frcnn." `https://github.com/kbardool/keras-frcnn`, 2017.

[53] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *Int. J. Comput. Vision*, vol. 88, p. 303–338, June 2010.

[54] P. Yakubovskiy, "Segmentation models." `https://github.com/qubvel/segmentation_models`, 2019.

[55] A. V. Buslaev, A. Parinov, E. Khvedchenya, V. I. Iglovikov, and A. A. Kalinin, "Albumentations: fast and flexible image augmentations," *CoRR*, vol. abs/1809.06839, 2018.

[56] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," *CoRR*, vol. abs/1612.03144, 2016.

[57] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014.

[58] N. Mahamkali and V. Ayyasamy, "Opencv for computer vision applications," 03 2015.

[59] A. B. Jung, K. Wada, J. Crall, S. Tanaka, J. Graving, C. Reinders, S. Yadav, J. Banerjee, G. Vecsei, A. Kraft, Z. Rui, J. Borovec, C. Vallentin, S. Zhydenko, K. Pfeiffer, B. Cook, I. Fernández, F.-M. De Rainville, C.-H. Weng, A. Ayala-Acevedo, R. Meudec, M. Laporte, *et al.*, "imgaug." `https://github.com/aleju/imgaug`, 2020. Online; accessed 01-Feb-2020.

[60] A. V. Buslaev, A. Parinov, E. Khvedchenya, V. I. Iglovikov, and A. A. Kalinin, "Albumentations: fast and flexible image augmentations," *CoRR*, vol. abs/1809.06839, 2018.

[61] E. Bisong, *Google Colaboratory*, pp. 59–64. 09 2019.

Appendix

This appendix contains additional figures and plots that were not directly included in the main document. The GUI of the OptiCork software and some training plots for the different variants of EfficientNet classification network have been placed in the subsequent sections.

Figures

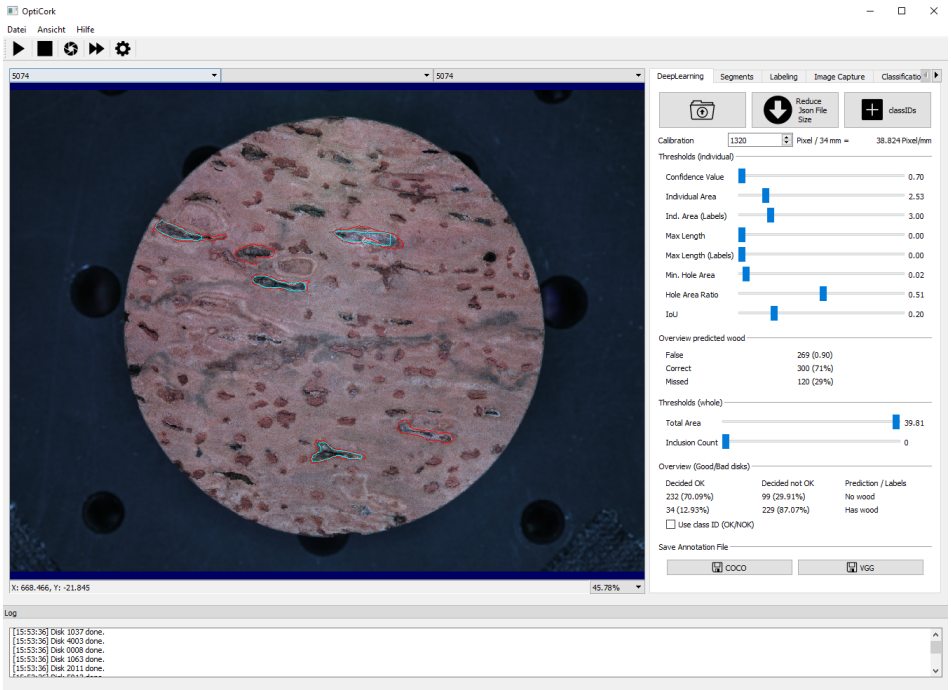
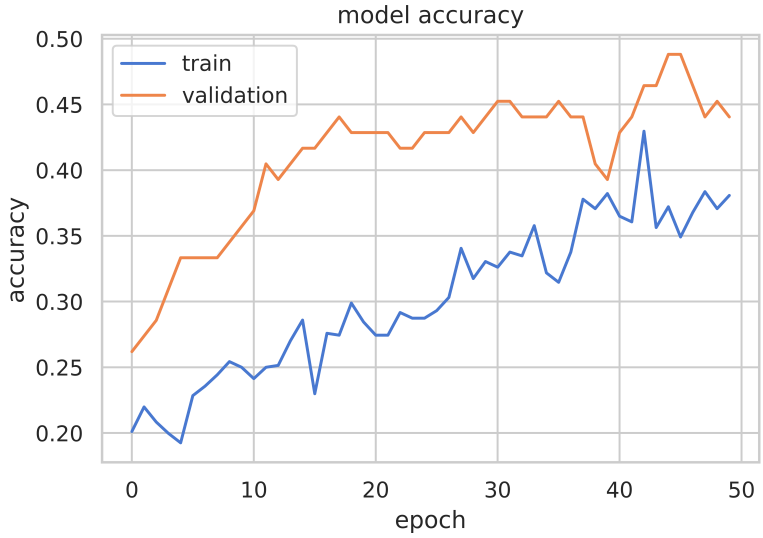


Figure 22: A view of the OptiCork software for visualizing the predictions.

Plots

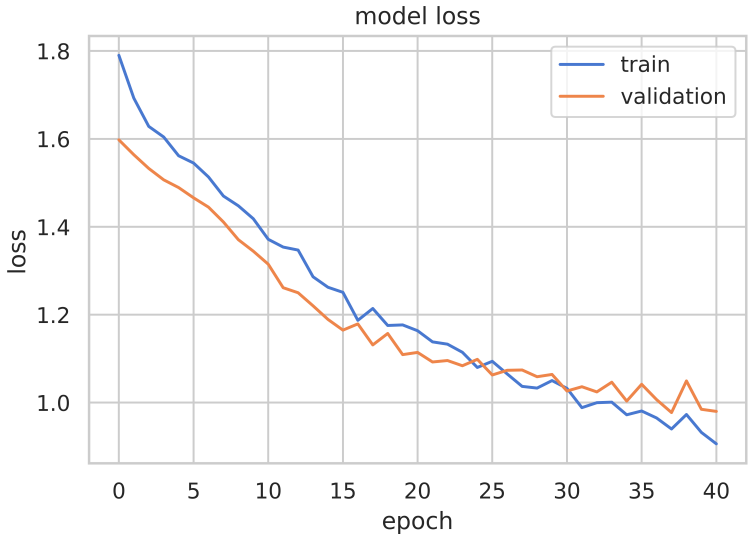
Additional Plots for the Quality Classification Model

(a) Loss curves

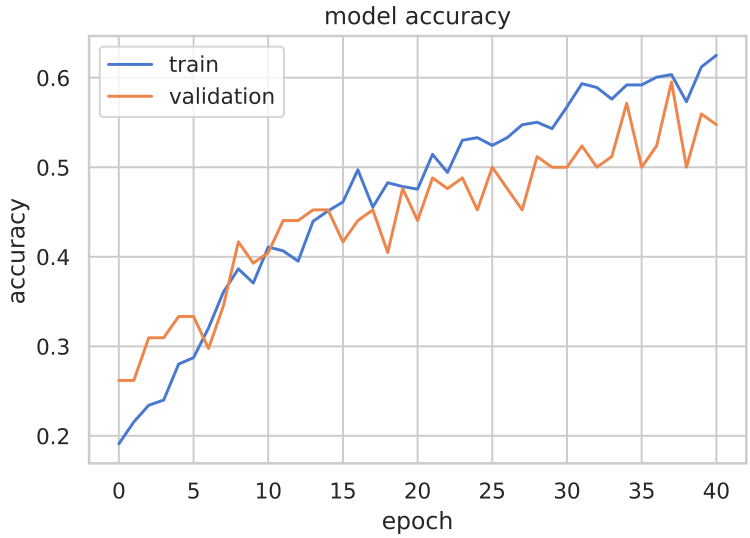


(b) Accuracy curves

Figure 23: Training and Validation plots for the EfficientNet-B1 classification model. Early stopping is implemented to halt the training when the model performance stops improving.

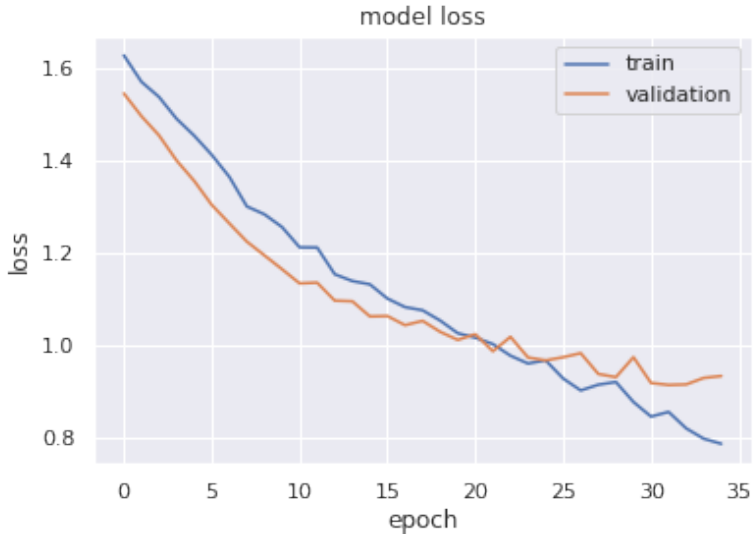


(a) Loss curves

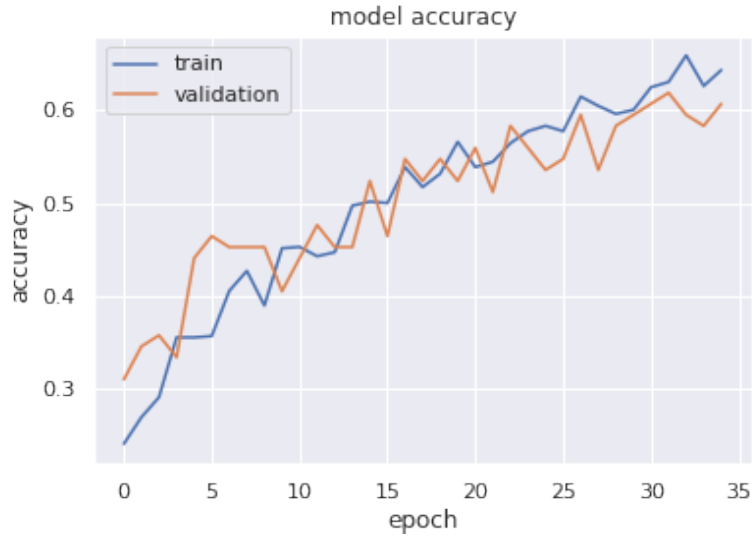


(b) Accuracy curves

Figure 24: Training and Validation plots for the EfficientNet-B2 classification model. Early stopping is implemented to halt the training when the model performance stops improving.

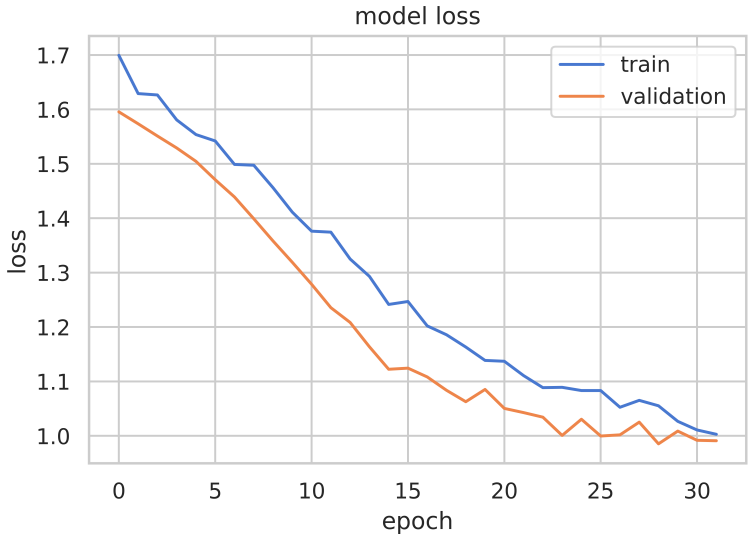


(a) Loss curves

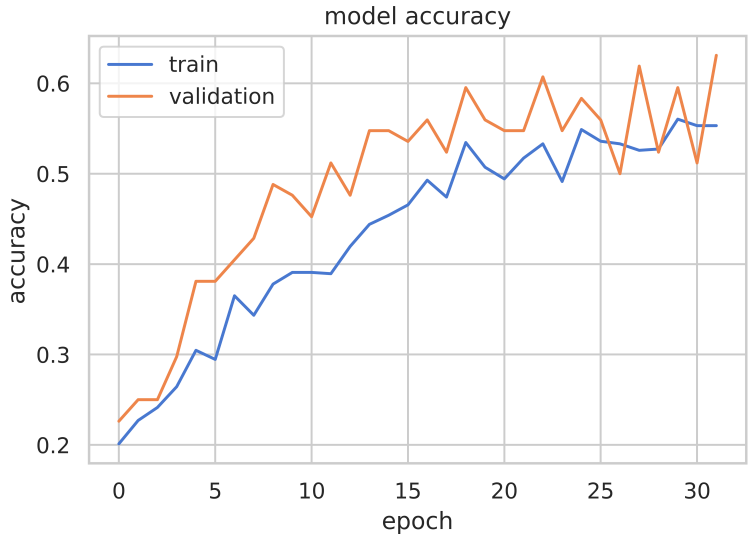


(b) Accuracy curves

Figure 25: Training and Validation plots for the EfficientNet-B3 classification model. Early stopping is implemented to halt the training when the model performance stops improving.



(a) Loss curves



(b) Accuracy curves

Figure 26: Training and Validation plots for the EfficientNet-B4 classification model. Early stopping is implemented to halt the training when the model performance stops improving.