Università degli Studi di Padova

DEPARTMENT OF INFORMATION ENGINEERING

Automatic detection of knots and wood logs classification using machine learning

Advisors

Prof. Pietro Zanuttigh

Candidate :

Marian Ashaq

Company Tutor :

Enrico Ursella

Academic Year: 2022 / 2023

Acknowledgement

Firstly, all praise to God, the absolute source of knowledge and wisdom. Secondly, I would like to express my deepest gratitude to my academic supervisor Prof. Pietro Zanuttigh and my supervisor from Microtec Company Enrico Ursella, for their direct supervision, continuous help, constant guidance and valuable Advice that have greatly contributed in achieving these results. Thirdly, I also want to express my gratitude to my family especially my husband and my parents for their continuous encouragement and patience throughout my master studies at Padua University. Finally, many thanks to teaching assistants and professors at Padua University who helped us since our first days at Padova University till graduation.

Abstract

These days the world is suffering shortage in some natural resources such as wood. There are two possible solutions to solve this problem. One possible solution is to maximize the usage and the other is to reduce the rejection losses. The process to maximize the value of wood is mainly depending on defects detection and then grading based on type and severity of these defects. For example, these defects could be knots, cracks or mildew.

One widely studied topic in machine vision applications is timber defects detection. To illustrate, the number and the dimensions of knots in each wood log determine its quality and consequently its price. In this project, we propose a method for detection of knots and classification of wood logs based on the number and dimensions of defects detected in each wood log. Firstly, we read all information about knots that were detected in each wood log using CT Log scanner. Then, this information was used to compute the distance between each knot and the surface. Thirdly, Cartesian coordinates were converted into polar coordinates. After that, all images were cropped into several crops in order to artificially enlarge our dataset. As a fifth step, scores were given to each wood log crop based on the number and the dimensions (radius) of knots found in each wood log crop. Then, crops were classified into two classes before processing by the deep learning algorithm where class zero represents crops with almost no knots whereas class one represents crops with many knots. Finally, crops were categorized using a classification threshold.

The proposed algorithm was evaluated using accuracy, precision and recall. The experimental results showed that our method achieved a precision of 0.96 and 0.90, and a recall of 0.98 and 0.94 for crops from both classes (zero and one) respectively.

In conclusion, by using this improved deep CNN model, we achieved an overall accuracy of 0.96, 0.94, 0.95 on training, test and validation sets respectively; that is, only 1.20 s was needed for both detection (image pre-processing and identification) and wood log crops classification. These results showed that the proposed CNN model could recognize knots and classify wood logs more accurately and effectively than conventional methods while using CT Log images.

List of Figures

List of abbreviations

CT	Computerized Tomography	**MLP**	Multilayer Perceptron
1D	One Dimension	**SVM**	Support Vector Machine
2D	Two Dimension	**ML**	Machine Learning
3D	Three Dimension	**DL**	Deep Learning
DKB	Dead Knot Border	**FPR**	False Positive Rate
RGB	Red Green Blue	**AUC**	Area Under the Curve
Info	Information	**VGG**	Visual Geomtry Group
CT	Computed Tomography	**ResNet**	Residual Network
GPR	Ground Penetrating Radar	**ReLU**	Rectified Linear Unit
LDA	Linear Discriminant Analysis	**GUI**	Graphical User Interface
LoG	La Placian Gaussian	**FCN**	Fully Convolutional Network
RCNN	Region based Convolutional Neural Network	**DCNN**	Deep Convolutional Neural Network
ML	Machine Learning	**SGD-M**	stochastic gradient descent with momentum
NN	Neural Network	**MAR**	Missing Alarm Rate
DR	Detection Rate	**FAR**	False Alarm Rate
RFID	Radio Frequency Identification	**PSO**	particle swarm optimization
WFP	Wood Failure Percentage	**MLP**	Multilayer Perceptron

List of Tables

TABLE OF CONTENTS

Chapter 1

Introduction

1.1 Microtec

This thesis is the result of an internship at Microtec Srl Gmbh, a leading company in intelligent wood characteristics recognition for optimizing the use in the wood processing industry and has been setting the standards in this market since 1980. The company develops state-of-the-art scanners that are able to detect and recognize defects within the wood, using this information to optimize cutting and thus improve production yields.

Microtec produces systems for internal defects detection such as Computed tomography. Using this scanner, it is possible to precisely describe the dimensions and the location of internal wood defects such as knots in a three-dimensional way. Furthermore, it is possible to figure out the external appearance, quality, and the resistance of the final product and the effect of these defects on the final product. Hence, the cutting solutions can be optimized based on the previous information, which helps produce a product with high quality and maximize sales profit.

In addition to that, Microtec has LogEye 300, which contains next generation cameras, sensors, and x-ray technology that is able to detect defects within the wood log. Using LogEye 300 multi-sensor quality scanner with modules such as color scanner and x-ray scan, it is easy to analyze logs before sawing.

1.2 Project Description

In today's world, stringent requirements are applied in the manufacture of wood products for surface processing (Zhongkang et al., 2020). For example, the comprehensive rate for wood usage is 90% in developed countries like Finland and Sweden compared to 60% in China due to waste of resources. To illustrate, the difficulty to cover the rapid growth demand in some countries all over the world is due to the processing level and the storage capacity of wood [19]. This leads to the surge in the need for methods to inspect the quality of wood logs to improve the quality of the yield products and consequently the usage rate.

Furthermore, the recent progress in measurement technologies have led to defects growth detection and process optimization in sawmill industries [1]. To illustrate, these days, there are various techniques for non-destructive scanning of wood logs prior to sawing such as ground penetrating Radar [3], laser testing [5-6], ultrasonic testing [7-8], acoustic emission technology [9, 10] that can help increase productivity and provide high value lumber [3] and commonly high recognition rates [11, 12].

Lumber production is highly dependent on finding out internal defects detection such as knots, decays and embedded metals. In other words, productivity and yield of wood with high values are basically depending on discovering knots and decays because the presence of embedded metals can have bad impact on saw blade which in fact leads to increase cost of maintenance of the sawmill and significantly increase the down time [3]. Nowadays, saw mills discard a vast number of logs or produce low value wood due to some internal defects like knots that are impossible to avoid while using the whole log.

Knots distribution has a significant role in sawmill industry. In other words, the quality of the final product and its appearance is highly dependent on it [11]. To illustrate, controlling the location of individual knots (especially knots on either of the faces) in each wood log can lead to optimization of sawing process of the resulting boards [4]. There are various methods for doing this like using sensors that can provide specific details about the internal structure of each wood log such as computer tomography scanners [4] or magnetic resonance imaging [1]. These methods are helpful to gain useful information about the internal log structure prior to sawing. On the other hand, these scanners have high cost and slow performance, so they are not suitable for usage in online integration in sawmill operations. There are other approaches, for example laser range scanners, that are fast but works with external log information [1].

In this project, a method to detect the knots and classify wood log crops based on knots dimensions and their distance from the surface will be presented.

1.3 Problem definition

Nowadays, despite the problem of shortage in wood resources in different parts of the world, a vast number of logs are discarded due to some internal defects like knots that are impossible to avoid while using the whole log.

There are many deep learning algorithms for non-destructive testing of wood logs as mentioned before but these algorithms still have problems like complex computation consequently high cost, slow performance, and inaccurate defect detection process.

To solve this problem a simple algorithm for internal defects detection with fast computation is needed for detection of knots and classification of wood logs based on the number and dimension of knots in each wood log crop.

1.4 Objectives

- Detecting knots in each wood log crop, which will result in optimization of sawing process.
- Wood log crops classification into two classes based on the number and dimensions of knots found in each crop. This will lead to alleviating number for discarded crops.
- Improving utilization rate of processed wood through internal wood features detection. Hence, make the best use of the forest resources.
- Simple model with fast computation.

1.5 Methodology

1. Get the 14 descriptors describing knots in each wood log from CT data
2. Using some columns from these descriptors, to know the distance between each knot and surface of the border, in addition to the radius of each knot
3. Convert Cartesian coordinates into polar coordinates
4. Crop each CT image into a number of crops (Data Augmentation)
5. Finally compute score for each wood log crop based on radius and number of knots in each region of this wood log crop
6. Save score for each wood log crop
7. Set a classification threshold (to distinguish between class zero and class one crops)
8. Training of the model using CNN
9. Evaluate the model
10. Predict on new data

1.6 Report Organization

The report is organized as follow:

A brief introduction to the project in Chapter1, which includes the problem definition, Objectives and Methodology.

Chapter 2 contains General literature review about wood defect recognition techniques, Machine learning for wood classification task and deep learning for features extraction.

Chapter 3 includes General theory of CNN, layers, architectures, loss function, and some data augmentation techniques and some evaluation metrics.

Chapter 4 illustrates information about the CT Log scanner dataset, acquisition machine, pre-processing steps such as cropping of the images, computing score for each wood log crop and setting the classification threshold between crops with almost no knots and those with many knots.

Chapter 5 demonstrates some information about the LogEye dataset, acquisition machine, pre-processing steps such as Getting knot location on coloured image, crop the RGB image, split knot diameter on pixels of the coloured image, save the score for each wood log crop, set depth and classification thresholds to discriminate between class one and class zero crops and finally the problems faced while using this dataset.

Chapter 6 contains Data augmentation, Training the CNN, Architecture of the model, Regularization techniques applied, hyperparameters used and loss function.

Chapter 7 demonstrates with figures the results achieved for classification and model evaluation.

In chapter 8, there is the conclusion.

Chapter 2

Related Work

2.1 Wood defect recognition techniques

In today's world, the fast development in information technology along with artificial intelligence techniques coupled with image processing has led to a significant improvement in wood defect detection process. To illustrate, few years ago image segmentation techniques were used for the purposes of detection and classification internal defects in wood logs but the accuracy was low [22].

That is why many scientific researches were carried out for the purposes of detection and classification of internal wood logs and wood boards features.

2.1.1 Classification and edge detection

In 2016, Zhang et al. [28] proposed a compressed sensing classifier using direction property of dual tree complex wavelet, which can express complex information of wood surface. The classification method introduced board grading with very high accuracy values without the need for image segmentation. The types of wood surfaces were radial and tangential textures, live and dead knot. Classification process was done on three main steps, which were:

1. Applying three-level dual-tree complex wavelet decomposition $\Longrightarrow$ 40-dimensional feature vectors were detected
2. For the purposes of feature selection and dimensionality reduction Particle swarm optimization algorithm was used $\Longrightarrow$ 11 key features were chosen;
3. All surface types were identified by the classifier based on the features optimized by Particle swarm optimization.

The obtained accuracy values for the given surface types were 100, 86.7, 96.7 and 86.7 % respectively.

To solve the damage occurred on upper surface during edge cutting process of rough laminated wood, Han el al. [29] proposed an algorithm for edge detection based on machine vision. This technique managed to regulate the position deviation of rough wood log.

2.1.2 Region extraction and image segmentation

Nowadays, machine vision techniques are applied to various industrial processes. In other words, to effectively apply these techniques a myriad of elements should be modified as each wood log has its own appearance, and internal features characteristics, which vary in each piece of wood. After image acquisition step, many mathematical algorithms can be used for detection and classification of the targeted features. In conclusion, if we have the representation of our target feature in one application, this does not mean that it could work with other application [30].

Image segmentation is a crucial step in defects detection and features extraction on images of wood surface. To explain, segmentation algorithms are divided in to two main classes based on the use of similarities or discontinuities in the image. Approaches dealing with adjacent pixels (similarities) are called region-based approaches, while others working with discontinuities are called edge based and these major classes are classified based on other sub-classes. In addition to that, algorithms can deal with colour or grayscale data, may work with windows of different sizes, and operate on either an individual pixel basis (global) or a neighbourhood of pixels (local).

Although there are variety of algorithms for segmentation of feature on wood log surface, it is hard to select the best approach among all. According to a research study conducted by a group of researchers [30], region-based approach that was the result of usage of clustering algorithm along with region-growing techniques have better performance compared to other approaches (Zhong et al.,2002).

Recently, there are many algorithms of both families different from edges and regions e.g. clustering, graph-based, etc. To explain, graph-based clustering transforms the data into a graph representation where vertices are the data points to be clustered and edges are weighted based on similarity between data points [69]. Furthermore, graph-clustering technique divides nodes into clusters such that connections among nodes in a cluster are dense while connections between nodes in different clusters are sparse. As a result, this approach provides the possibility to find clusters in a large graph, which is essential to understand the relationships between the nodes [69].

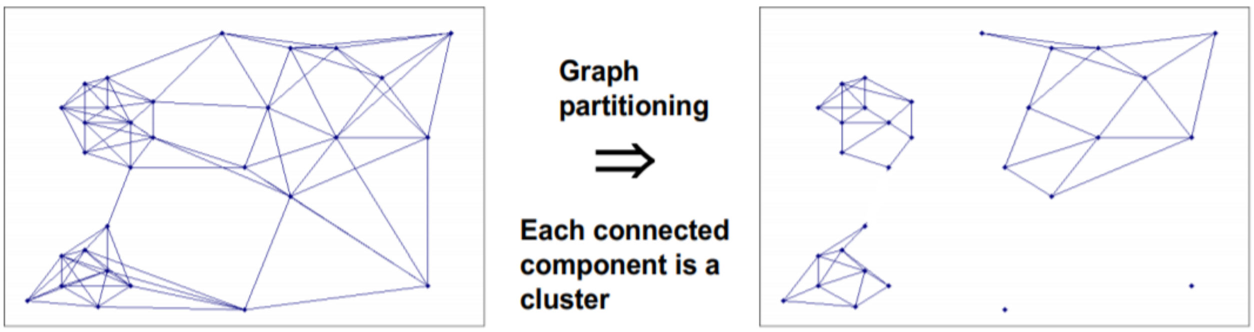


Fig. 2.1: Graph partitioning [69]

2.2 Machine learning for wood classification task

Machine learning methods are generally characterized into two major groups based on type of learning: supervised and unsupervised. The supervised learning method uses labelled data to train an algorithm to compute output variables [29]. Unsupervised learning methods, on the other hand, use unlabelled data in training an algorithm to identify hidden patterns [30]. The strength of applying machine learning to wood science and engineering is its ability to analyze any type of data, such as images [31], anatomy data [32], infrared spectra [12], etc. Furthermore, the larger the dataset that is used to train an algorithm, the better the accuracy and predictive power. The big data-based machine learning industry is growing and finding applications in various parts of the wood industry for genotype discrimination [35,36], species identification [8,12,23,37], and wood moisture content prediction [38,39].

Some machine learning approaches that were applied in the field of wood defects recognition and classification will be presented in the next few lines.

2.2.1 Unsupervised learning methods

I. K-means clustering method in wood failure detection

Currently, wood failure percentage is a significant metric while evaluating the strength of the bond in plywood. In the last decades, wood failure percentage was monitored based on visual inspection, which resulted in alleviating efficiency values.

As a solution for efficiency problems, researchers developed an algorithm based on thresholding and k-means clustering unsupervised learning algorithm [31].

Dataset used composed of two kinds of widely used adhesive (PF, UF) and two kinds of veneer (Poplar spp. and Eucalyptus spp.) were used for the manufacture of four kinds of three-layered plywood. The thickness of the veneer was 1.8 mm with a moisture content (MC) of 8–10 %. Parameters of hot pressing were 140 C, 1 min/mm, 1 MPa for PF and 100 C, 1 min/mm, 1 MPa for UF. After the panels had been prepared, specimens for bond strength test were cut and bond strength tests were carried out according to the Voluntary Product Standard PS1-95 for construction and industrial plywood. Wood failure percentage for each specimen was inspected and recorded by experienced personnel from Chinese National Center for Quality Supervision and Testing of Wood and Bamboo Products (Wang et al, 2015). A series of thirty plywood shear specimens were tested for each kind of plywood. The tested specimens were scanned (HP Scanjet G3010) with a resolution of 200 dpi. Scanned RGB images were used in the image processing process to realize the automatic detection of wood failure [31].

Results: Thresholding and k-means methods were applied and mean absolute error was calculated for both of them, the first one was significantly affected by adhesive colour and veneer colour (PF-Eucalyptus: 15.77 %, UF-Eucalyptus: 30.55 %, UF-Poplar: 21.48 %) while k-means clustering there were no huge differences in mean absolute error (PF-Eucalyptus: 11.07 %, UF-Eucalyptus: 14.77 %; UF-Poplar: 8.50 %) [31].

In conclusion: From the results it could be reiterated that k-means clustering provided a remarkable compatibility in wood failure detection while dealing with different wood adhesive and types (Fu et al,2015).

II. Principal component analysis

Li et al. [32] introduced the use of PCA dimensionality reduction technique with compressed sensing in defects detection from wood log images. The proposed algorithm assured the effectiveness of PCA in reducing data redundancy and dimensions of features while the role of compressed sensing was to develop the identification accuracy as a classifier.

PCA feature fusion steps are shown in the block diagram figure 2.2:

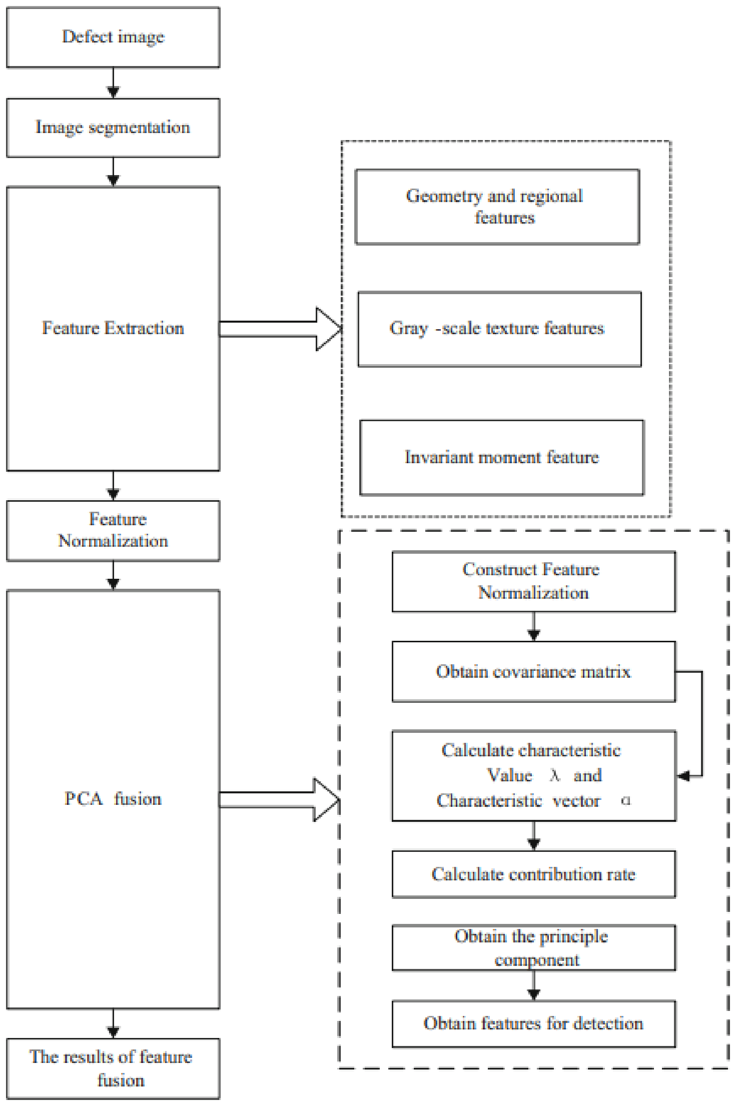

Defect image
Image segmentation
Feature Extraction
Geometry and regional features
Gray -scale texture features
Invariant moment feature
Feature Normalization
PCA fusion
Construct Feature Normalization
Obtain covariance matrix
Calculate characteristic Value λ and Characteristic vector α
Calculate contribution rate
Obtain the principle component
Obtain features for detection
The results of feature fusion

Fig. 2.2: PCA feature fusion steps [32]

Twenty-five features including geometry, texture, region features etc. were extracted from wood log images. After that, these features were integrated with the help of PCA and eight principal components were selected for expressing defects. As soon as fusion process was done, the features were used to create a data dictionary, which was used for calculating the optimal solution for classification based on least square method.

Materials and Methods: size of boards that were used in this experiment was 40 × 20 × 2 cm, wood species was Xylosma. The experiment took place-using MatlabR2012 and a platform with a 64-bit PC, Oscar F810C IRF camera was used to obtain the images. To increase the brightness of the images, two parallel LEDs were used for illumination. Moreover, 500 images 8-bit gray scale were used for the training (200 with live knots, 200 with dead knots and the rest with cracks) and 200 for testing.

Results: The usage of PCA for feature fusion with compressed sensing as a classifier scored 0.2015 and 0.7125, which was the fastest detection time, and accuracy value was 92% compared with 87% with the usage of the SOM (Self Organizing Map) neural network [32].

2.2.2 Supervised learning methods

Supervised learning occurs when your model is trained using labelled training set. In other words, if we have correctly classified data, we can use supervised learning in this case for making predictions on a new data.

I. Support Vector Machine

SVM is one of the most common supervised learning methods that are used for classification and regression tasks. To explain, for linear SVM points that are located on the same side belong the same category and vice versa.

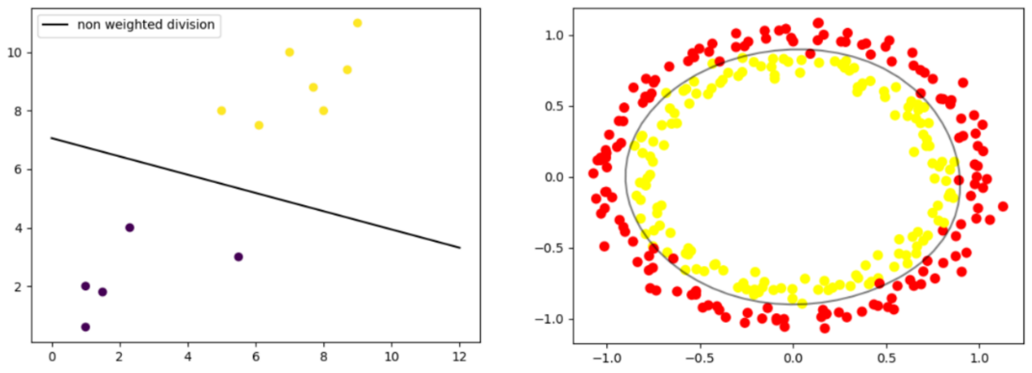


Fig. 2.3: a- Represents Linear SVM b-and Kernel SVM

Figure 2.3 represents the two SVM types where the image **a** represents simple SVM which is commonly used for regression and classification problems while image **b** represents kernel SVM which is flexible while dealing with non-linear data by adding multiple features instead of two in linear SVM.

The number and type of defects have a significant effect on the quality of wood. To illustrate, if we have a piece of wood log with many knots or cracks, this piece of log will be sold with cheap price [36].

In 2015, D. Sidibe et al. [36] used SVM classifier for detection of wood log defects. The detection was done in several steps. To begin with, the dictionary of words was obtained by LBP or SURF or might be both. After that, regions with potential defects were detected with the help of image processing pipeline including filtering, enhancement of contrast and maximization of entropy. Finally, SVM classifier was used to detect two types of defects, which were knots and cracks.

<u>Datasets:</u> Two datasets representing two different wood species with different characteristics. The first dataset contains 100 images of Epicea wood, with size of 640 × 4500 pixels. The dataset was provided with manual ground-truth annotation for each image showing the position of the defects. This dataset contains only knots as defects. The second dataset was composed of 100 images of Pine wood and contains cracks as defects. Both datasets were acquired locally using a scanner of the Luxcan Company, which allows acquisition of different wood characteristics.

<u>Results:</u> 0.92 and 0.91 precision and 0.94 and 0.96 recall for the two datasets respectively with multiple features dictionary (M. M. Hittawe, 2015).

Note:

1-Precision: the ratio between the numbers of *Positive* samples correctly classified to the total number of samples classified as *Positive* (either correctly or incorrectly). The precision measures the model's accuracy in classifying a sample as positive

$$Precision = \frac{True_{Positive}}{True_{Positive}+False_{Positive}} \quad (2.1)$$

2-Recall: the ratio between the numbers of *Positive* samples correctly classified as *Positive* to the total number of *Positive* samples. The recall measures the model's ability to detect *Positive* samples. The higher the recall, the more positive samples detected

$$Recall = \frac{True_{Positive}}{True_{Positive}+False_{negative}} \quad (2.2)$$

II. Linear Discriminant Analysis

LDA is a dimensionality reduction technique, which is commonly used in supervised learning classification problems. In other words, it projects features from higher dimension space into a lower dimension space.

In 2017, a method for wood defects detection based on linear discriminant analysis and the use of compressed sensor images was proposed [11]. Firstly, images for wood surface were captured using F810C IRF camera. Then defect features were extracted from wood logs images after segmentation. LDA algorithm was used for integrating features, alleviating their dimensions and minimizing the time for processing. Finally, a data dictionary was constructed directly from features fusion and a compressed sensor was designed to detect types of defects in wood logs. Three major defect types knots, dead knots and cracks that were used to ensure the effectiveness of this method. The average duration for classification and feature fusion was about 0.446 Ms with accuracy 94% [11].

Dataset:

The research was mainly concerned about three types of wood board defects including dead knots, live knots, and wood cracks. The species of wood included Fraxinus mandshurica, Xylosma racemosum, Korean Pine, and Oak. The samples received a series of treatments, including drying and polishing before the experiment was carried out. The size of the boards was 40 cm × 920 cm × 92 cm (Liang et al., 2017).

Method:

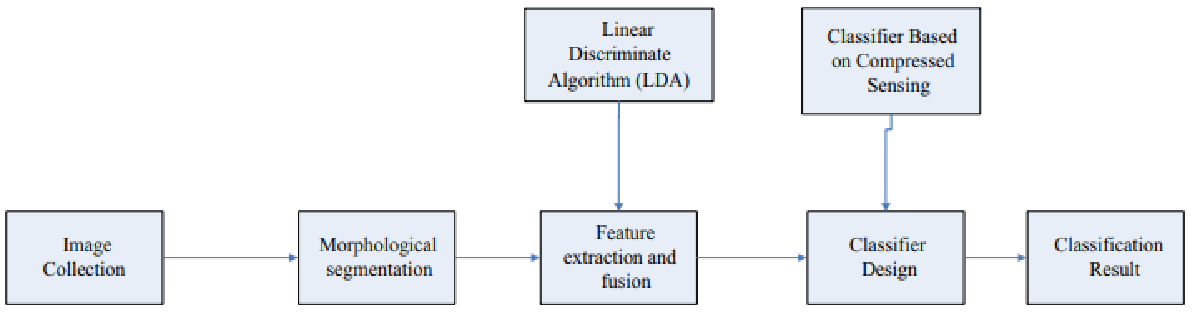

Linear Discriminate Algorithm (LDA)
Classifier Based on Compressed Sensing
Image Collection
Morphological segmentation
Feature extraction and fusion
Classifier Design
Classification Result

Fig. 2.4: The process of online Classification method **[11]**

Figure 2.4 is a block diagram explaining the five main steps for the process of online classification as follow:

Firstly, image collection was done using integral projection method to find the border of the wood log. Secondly, morphological segmentation image processing technique was applied to have continuous image skeleton, fast and exact image segmentation [13]. Thirdly, feature extraction and fusion where 25 features of three main types were detected [14]. LDA theory was used to reduce the dimension of the original pattern, maximize the distribution of samples between classes, and reduce the distribution of sample within the class. Fourthly, the classifier was designed using several equations. Finally, Classifier result: testing samples of live knot, dead knot and cracks were classified in this test. This research paper provides recognition accuracy of live knot, dead knot, and crack 90, 95,100% respectively [11]. The recognition time was 44.199, 49.059, 44.268 ms respectively. In addition to that, this method introduced very high values for defect detection recognition rates with fast computation time, which was sufficient for on-line board sorting.

Conclusion: Having manifested the aforementioned steps, it could be reiterated that the use of LDA fusion with compressed sensing method improved recognition rate by 26%, 12 %, and 7% compared with non-selection method, deviation method and SOM neural network respectively. It also managed to reduce computation time by 0.67 ms, 0.042 ms compared with non-selective and deviation method [11]. In addition to reducing complexity of computation, LDA has visual space that can be intuitively classified. Finally, the proposed methods showed effectiveness in soft measurement of wood defect detection.

III. Modelling Internal knot distribution using external knot features

Zolotarev et al. [1] proposed an innovative technique to estimate the internal structure of logs using information about external features that were collected from laser range scanner data. The proposed method was done in various steps. Firstly, estimation of log centreline depending on surface height map generation. Then knot segmentation and volumetric reconstruction of knots in order to obtain a log model. This model is useful for virtual sawing and estimating knot location in the resulting boards. Virtual sawing is essential to actively search for the optimal sawing parameters before the actual sawing which helps in optimization of sawing process.

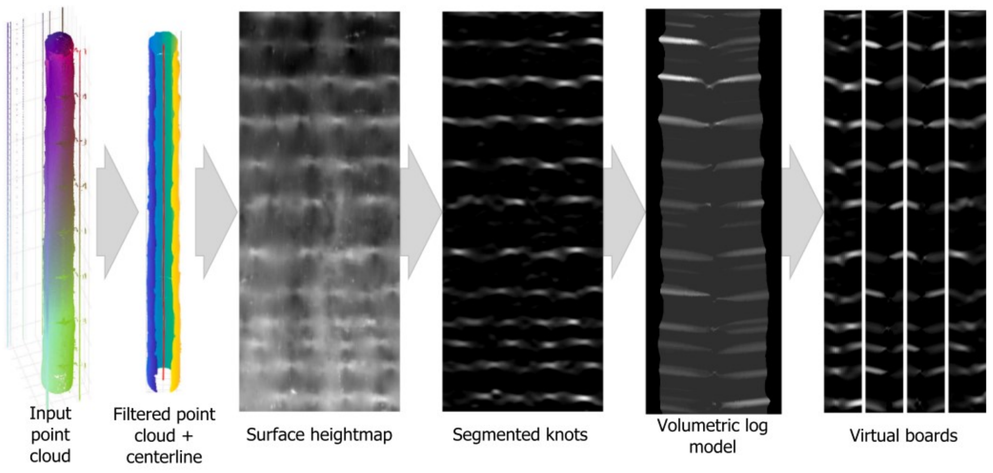


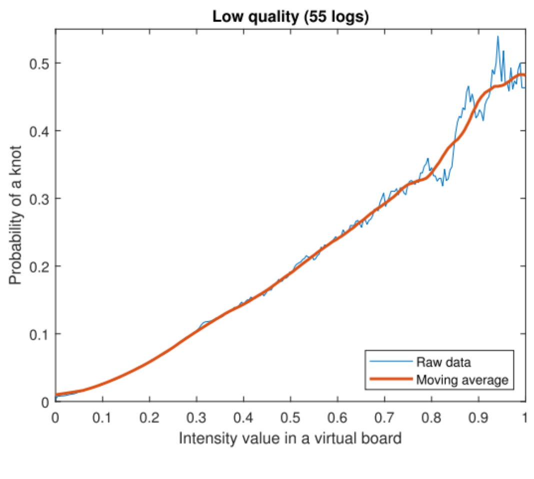
Fig. 2.5: Schematic representation of the main steps of the proposed method [1]

The proposed method was divided into five crucial steps as we can see in figure 2.5: Point cloud filtering and centreline estimation where DBSCAN was used to cluster layer points where the densest area corresponds to the log surface cross section which was assumed to have circle shape or even part of it (Ester et al., 1996). Then, log surface height map was generated using equation to find knot location on the log (Nguyen et al., 2016). Thirdly, Knot segmentation was done given the fact that knots can be seen as slight bumps (Lindeberg, 1993) from computer vision point of view, so they were aligned with the area of a specific elliptical shape and size with high values. Blob detection cannot only be done using SIFT (Lowe, 1999) but also using SURF (Bay et al., 2006) and la placian and Gaussian filter. Volumetric reconstruction of knot took place after knot segmentation; we get an image containing filter activations that refer to the knots' location in each wood log. The places where activations are stronger represent the knots' location (Duchateau et al., 2013). Finally, virtual sawing was applied in order to check the quality of the final product.

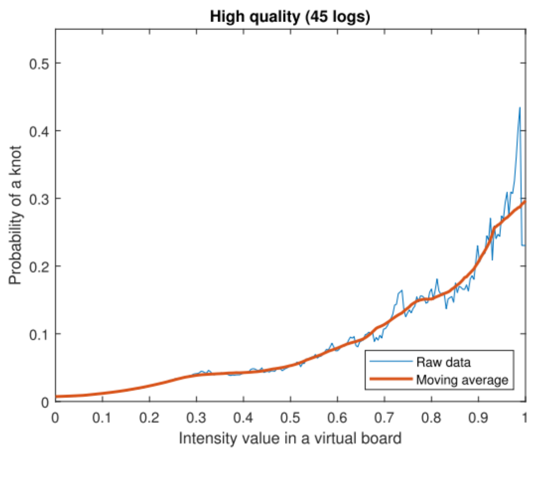
Dataset: 100 debarked softwood logs collected in two subsets each of 50 in separate sessions, images were collected using laser range scanners 270 main yield logs, age of each wood log ranges between 60 to 80 years, they were given numbers from 1 to 100.

Logs are automatically graded with the help of an existing grading system in the sawmill into five grades depending number of knots in each wood log and the diameter of each wood log. A, B, C are wood logs with high quality and D, E are wood logs with bad quality.

Results:



(a)



(b)

Fig. 2.6: Estimated dependence of the knot probabilities on the pixel intensities of the virtual images for the low and high quality logs [1]

Figure 2.6 explains the relation between the probabilities for knot presence and pixel intensities on different log categories. As we can, notice in the plots that for low quality log the correlation is linear between the two coefficients, unlike high quality logs that experience a high spike for higher range intensities. It can also be noticed that high quality logs with limited number of knots were from the category of bottom logs that were collected from the lower part of a tree. This type of logs is characterised by knots that did not grow enough to be detected from outside (Zolotareva et al., 2020).

Conclusion: This paper proposed an effective method for virtual sawing of wood logs using laser range data with the possibility of indicating the probabilities of possible knot locations on generated images corresponding to those on the board sides. The pixel intensities of these images were aligned with the probabilities of knots location on the real log. The data collection step was fast and cheap as it was done using laser range scanner and can be used online.

2.3 Deep learning for features extraction

These days have witnessed great development in computer hardware technology and upgrade of GPU that led to the improvement and the efficiency of computing and image processing. Among these methods, the convolutional neural network (CNN) which represents deep learning. Practical application scenarios are image recognition, natural language processing, and speech recognition. Image recognition using deep learning does not need to carry out the complex process of manual feature extraction but can completely carry out feature extraction and recognition from the input image end to end, which greatly liberates work force [20, 21]. Furthermore, the feature of autonomous learning is no less than that of manual feature extraction by experience, and the recognition effect is better [9].

In 2020, a linear array CCD camera was used by Zhao et al. [22] to obtain images for the surface of wood and introduced a new hybrid Mix-FCN for detecting and locating wood defects. Unfortunately, it was complex in computation due to deep depth. (Guan et al., Wang et al. 2020) used a combination of feature extraction algorithms along with R-CNN for detection of defects in wood logs but both the algorithms and the model were complex.

In the next few lines, several deep learning approaches in the field of detection, classification, and features extraction in wood logs will be presented.

2.3.1 Hopfield neural network

There are many neural network models for learning and information retrieval such as Hopfield model and Boltzmann machine. These models are mainly used in information retrieval, classification, and feature extraction tasks. In other words, if we have a generative model, they can effectively learn it from the observed data and thus providing metrics to statistical learning.

In 2010, Qi et al. [36, 37] used Matlab to modify Hopfield neural network for the purposes of detecting wood defects boundary. The experiment resulted in remarkable minimization in the cost function. The optimization process steps were done as follow: Canny algorithm was used for initial boundary estimation, then the state of Hopfield neural network was determined by gray pixel value, finally the first two steps were repeated many times until the minimum value for the cost function was reached.

Results: The proposed cost function proved that activated neurons were located at points where there were great changes in gray value, noiseless and vivid boundary better than those done by traditional methods [37].

2.3.2 Deep Convolutional neural network

To solve the aforementioned problems of imprecise defects location and complex computation, an application of deep convolutional neural network on features extraction and wood defects detection was proposed by Zhongkang Hu et al. [22]

In 2020 [22], a learning method for automatic detection and classification of internal defects in wood images that were captured using laser range scanner was proposed through DCNN. For the training of the neural network, TensorFlow was used which was composed of: four convolutional layers, four max-pooling layers, three fully connected layers, a softmax layer and an output layer. To reduce the risk of overfitting, data augmentation, regularization L2 and dropout were used.

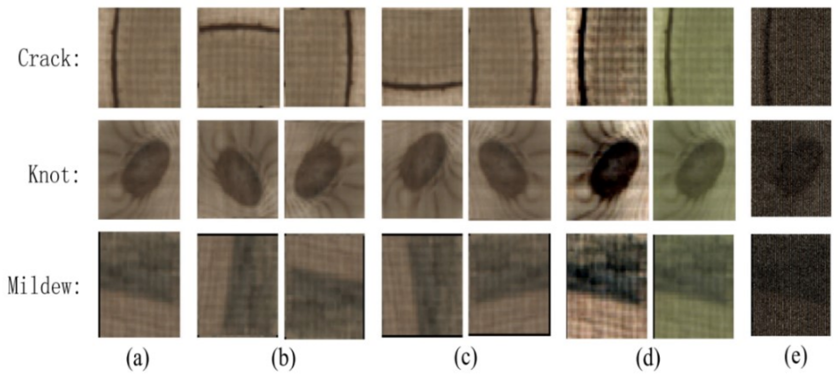
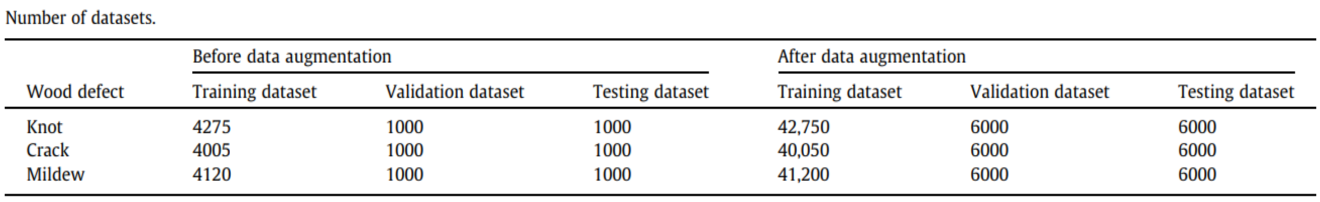


Figure 2.7: Represents the three types of internal defects, which were crack, knot and mildew. As we can see in the figure the images were subjected to rotation by 90 and 180 degrees, flipping diagonally and horizontally, contrast increase by three and Gaussian noise addition [22].

Dataset used:



Number of datasets.

	Before data augmentation			After data augmentation		
Wood defect	Training dataset	Validation dataset	Testing dataset	Training dataset	Validation dataset	Testing dataset
Knot	4275	1000	1000	42,750	6000	6000
Crack	4005	1000	1000	40,050	6000	6000
Mildew	4120	1000	1000	41,200	6000	6000

Fig 2.8: Dataset used in DCNN

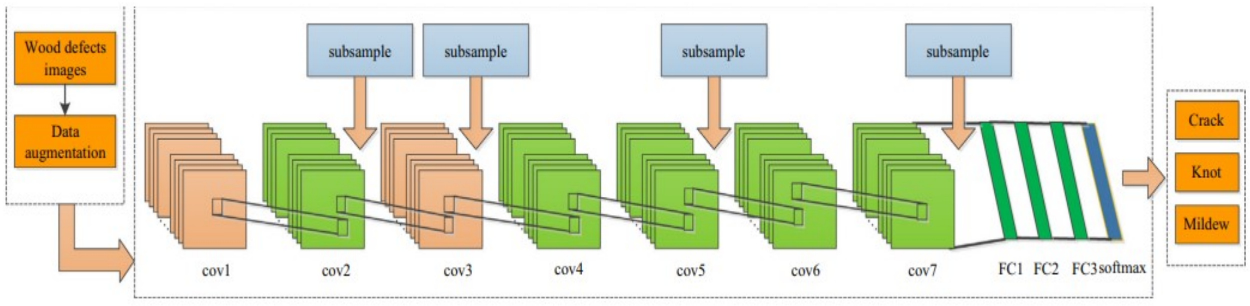
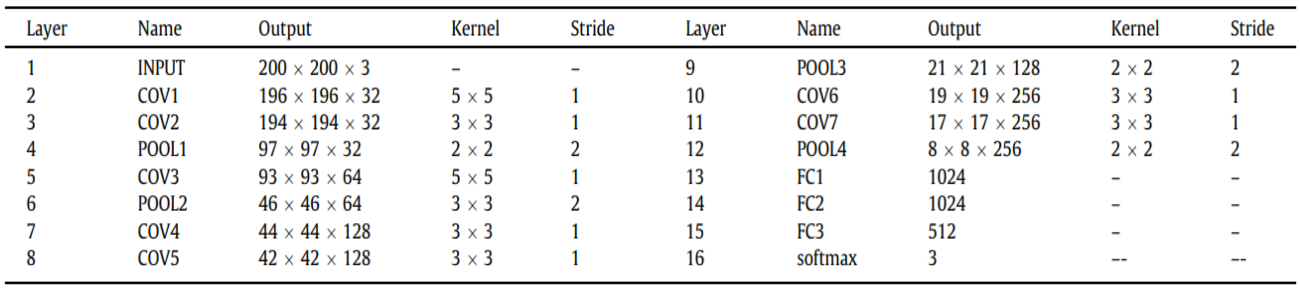


Fig. 2.9: Structure of wood defect recognition on the TensorFlow framework [22]

Figure 2.9 represents the Architecture of the improved DCNN used for wood defects recognition and classification.



Layer	Name	Output	Kernel	Stride	Layer	Name	Output	Kernel	Stride
1	INPUT	200 × 200 × 3	–	–	9	POOL3	21 × 21 × 128	2 × 2	2
2	COV1	196 × 196 × 32	5 × 5	1	10	COV6	19 × 19 × 256	3 × 3	1
3	COV2	194 × 194 × 32	3 × 3	1	11	COV7	17 × 17 × 256	3 × 3	1
4	POOL1	97 × 97 × 32	2 × 2	2	12	POOL4	8 × 8 × 256	2 × 2	2
5	COV3	93 × 93 × 64	5 × 5	1	13	FC1	1024	–	–
6	POOL2	46 × 46 × 64	3 × 3	2	14	FC2	1024	–	–
7	COV4	44 × 44 × 128	3 × 3	1	15	FC3	512	–	–
8	COV5	42 × 42 × 128	3 × 3	1	16	softmax	3	--	--

Fig 2.10 Represents parameters of the network structure [22]

As we can see in figure 2.10 DCNN architecture was used [25, 26] which had 16 trainable layers. These layers include: input layer 200 * 200 pixels of coloured images, seven convolutional layers each was followed by a ReLU activation function, four Max-Pooling layers used for subsampling to alleviate the memory, number of parameters usage, three fully connected layers where Kernels and strides for all the 16 layers were given in the figure. The output had three ways for the purpose of detection and classifying knots, cracks and mildew stains.

$$\mathrm{Softmax}(Z)_i = \frac{\exp(Z_i)}{\sum_j \exp(Z_j)} \quad (2.3)$$

This output of a classifier for the purpose of representation the probability distribution over different classes is usually computed using equation 2.3.

Results: The trained model was tested using validation set and testing set. The accuracy was 99.21% with validation set and 99.13% for the testing set.

Conclusion: The results reached using this research proved the effectiveness of DCNN in wood defects recognition and classification. The proposed model scored 98.76%, 98.73% accuracy values when applied to both validation set and testing set respectively (**Liu et al., 2020**).

Chapter 3

General Theory

3.1 Introduction to deep learning

Deep learning is a part of machine learning that imitates the human brain in the process of information processing as in figure 3.1. For example, DL has the ability of taking huge amount of data and classifying them with the corresponding labels. DL has several layers such as artificial neural networks, each layer has a different insight on the data that has been given to it [39, 40].

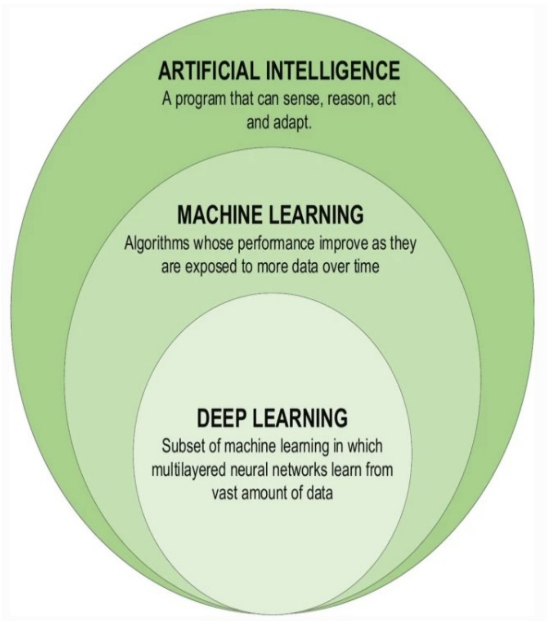

ARTIFICIAL INTELLIGENCE
A program that can sense, reason, act and adapt.
MACHINE LEARNING
Algorithms whose performance improve as they are exposed to more data over time
DEEP LEARNING
Subset of machine learning in which multilayered neural networks learn from vast amount of data

Fig. 3.1: Artificial Intelligence, Machine Learning and Deep Learning [47]

As we can see in figure 3.2, using ML for classification tasks requires a myriad of steps such as pre-processing, features extraction and features selection, learning and finally the classification. In addition to that, ML may have the risk of incorrect class classification due to biased feature selection. On the contrary, DL achieves learning and classification in a single step by enabling automatic features set learning for many tasks [39, 41]. More importantly, DL has become one of the most well-known types of ML algorithms especially since the evolution of big data because of its outstanding performance in various fields. Moreover, DL has noticed a remarkable development in several ML tasks such as object detection [42, 43], image recognition [45, 46], and image-super resolution [44]. DL managed to not only improve accuracy of ML classification task but also exceed human performance in tasks like image classification.

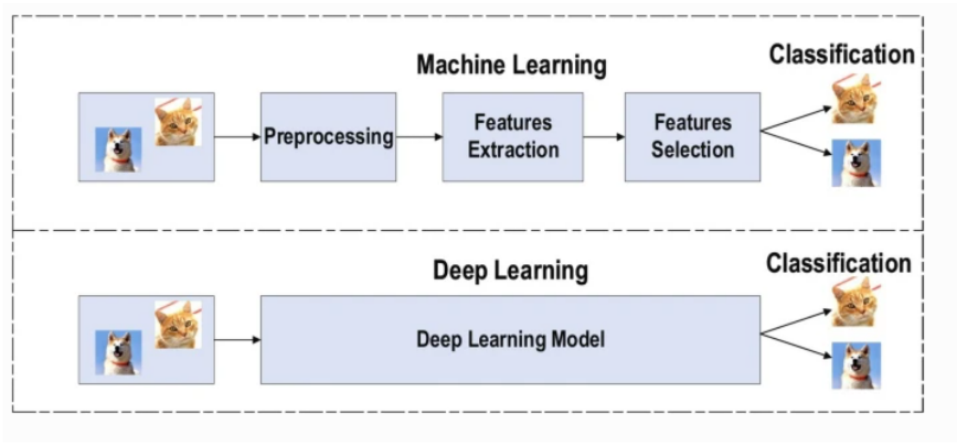


Fig. 3.2: Differences between ML and DL [47]

DL algorithms are segregated into three main categories which are supervised, semi-supervised, RL which is considered as semi-supervised, and unsupervised learning techniques. Unlike deep supervised learning, unsupervised learning has the ability to develop learning process despite the unavailability of labelled data. Some examples of supervised DL algorithms are RNNs, CNNs and DNNs while generative networks, dimensionality reduction techniques (PCA) and clustering are considered unsupervised learning techniques. Thanks to the use of restricted Boltzmann machines, auto encoders and GANs techniques, deep learning proved high levels of efficiency when dealing with non-linear dimensionality reduction and clustering tasks [47].

The learning process in semi-supervised DL is based on semi-labelled datasets. To elaborate, this technique uses both supervised and unsupervised learning techniques. Firstly, manually labelling a small portion of the unlabelled data. Then, train the model with the small portion of the labelled dataset. Thirdly, use the model to predict on the remaining unlabelled portion of data. After labelling of the unlabelled portion of data with one of the possible semi-sup approaches such as pseudo labelling, then the model is trained with the full dataset.

3.2 Types and applications of deep learning

Some common types of DL networks are RNNs (Recurrent Neural Networks), RvNNs (Recursive Neural Networks), and CNNs (Convolutional Neural Networks). CNN will be explained in deep due to its efficient performance in many applications.

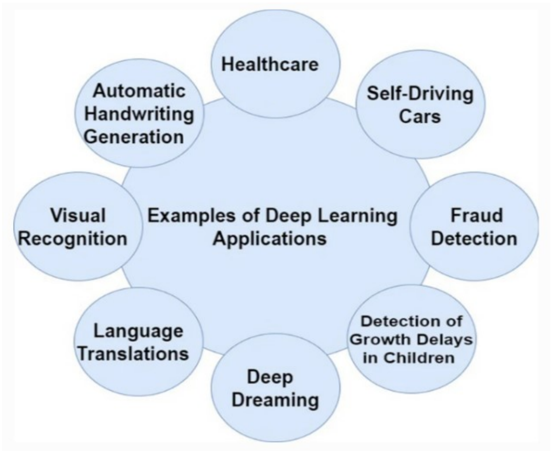

Healthcare
Self-Driving Cars
Fraud Detection
Detection of Growth Delays in Children
Deep Dreaming
Language Translations
Visual Recognition
Automatic Handwriting Generation
Examples of Deep Learning Applications

Fig. 3.3: DL Applications [47]

As we can see in figure 3.3, there is a huge number of DL applications. For example, DL applications in computer vision field are divided into five major categories: localization, segmentation, detection, classification and registration. Furthermore, the significant role of DL in healthcare could not be neglected especially in image analysis field.

3.3 Convolutional neural network

3.3.1 CNN Overview

Convolutional Neural Network is popularly used for analysing images, data analysis and classification tasks. To elaborate, CNN has some type of specialization to detect patterns, edges, corners, etc. that have significant role in image analysis. Convolutional layers receive input then transform the input in some way and then output the transformed input to the next layer. Each convolutional layer has a number of filters (that we need to specify) such that these filters can detect patterns, edges, corners, circles, faces etc. By using filters in later layers, we can detect not only simple edges but also more details like eyes, ears, hair, etc. CNN layers, loss function, regularization data augmentation techniques and some metrics for evaluation of CNN performance will be demonstrated in the next few lines.

3.3.2 CNN layers

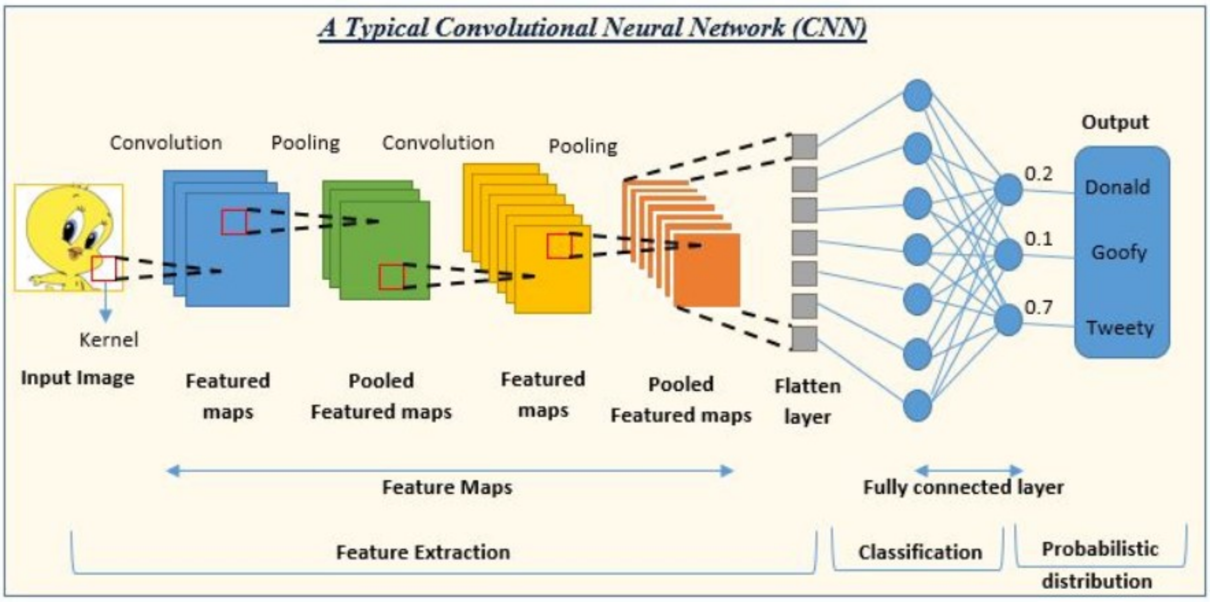


Fig. 3.4: Convolutional Neural Network Layers

As we can see in the figure 3.4 the CNN contains several layers, which are called also building blocks, and each layer in it has its own role, which will be demonstrated in details below.

(1) **Convolutional Layer**: The most significant layer in CNN is convolutional layer [47]. To explain, this layer contains a number of convolutional filters that are called kernels. As shown in figure 3.4 the input image is convolved with these filters to produce the output feature map.

Kernel: kernel can be defined as a grid of discrete values that are called kernel weights. These weights are given random numbers at the start of the CNN training process. During training process, these weights are updated that is why kernel learns to extract crucial features.

Convolution Operation Steps: (a) The kernel passes by the whole image top, bottom, right and left. (b) Dot product takes place between the kernel and the corresponding values in the matrix of the input image. (C) Summation to get a scalar value. (d)The process is repeated several times until the end of the input image matrix values. (e) The result from this process is called feature map.

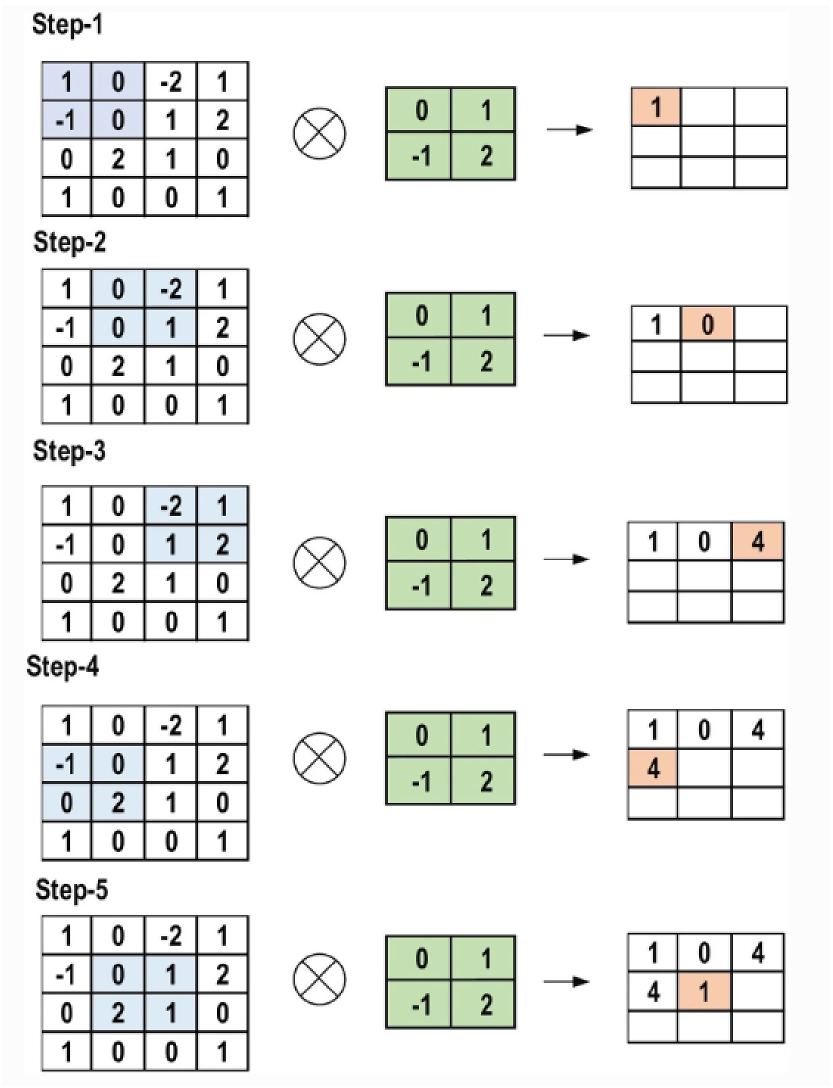


Fig. 3.5: The primary calculations executed at each step of convolutional layer [47]

As we can see in figure 3.5, the green colour represents the kernel weights that are initialized (2×2) that is moving with stride =1 , the blue colour represents the corresponding pixels of the image that are of the same size as the kernel , while orange colour represents the resulted feature map(as a result of multiplication and summation).

CNN is better than fully connected network for two main reasons [47]:

Table 3.1: Represents two main differences between FC neural networks and CNN

FC neural network	CNN
Each neuron in FC neural networks is connected to all other neuron s in the following layer. Thus, number of weights is large which needs large memory space to store.	Only few weights that are between two neighbouring layers. Consequently, number of weights is small which requires a small memory space to save.
No sharing of weights	Sharing of weights which leads to alleviating training time and several costs

(2) **Pooling layers:** The target of pooling layers is sub-sampling of the feature maps that are produced from convolution operation. To demonstrate, these layers mainly decrease the size of feature maps by keeping only the main features in every pooling step. There is a myriad of types for pooling such as global average pooling, min pooling, max pooling, tree pooling, gated pooling, average pooling and global max pooling.

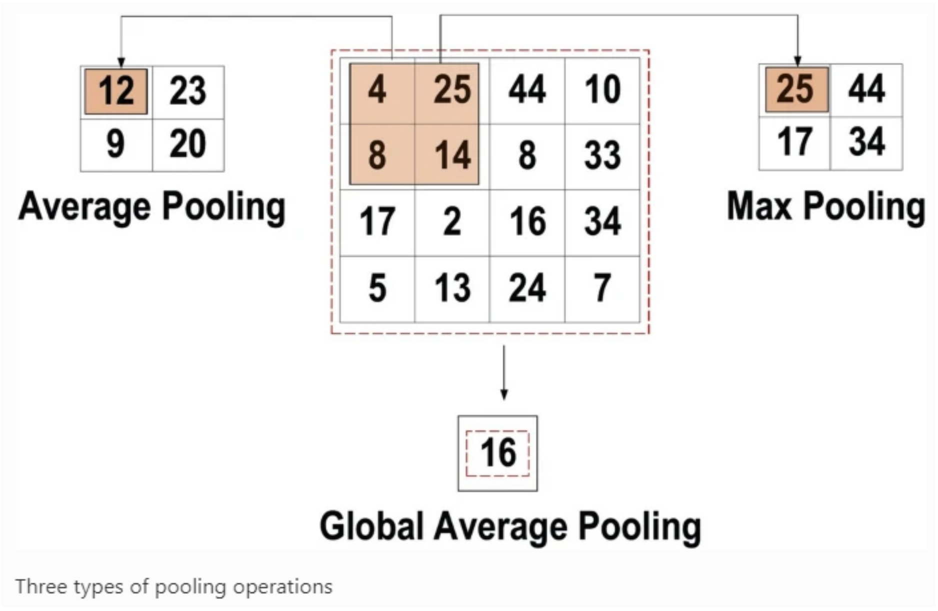


Fig. 3.6: Represents the most common pooling types [47]

Figure 3.6 illustrates the method of computation of the three most common pooling types, which are Max pooling, Global average pooling and Average pooling. Pooling layers provide the CNN with crucial information like whether a certain feature is found in the image or no , in addition to the exact location of the feature if found [47]. Thus, the presence of pooling layers has the power of improving the overall performance of the CNN and vice versa.

(3) **Activation Function**: These non-linear are always applied after all layers that have weights such as FC and convolutional layers (L. Alzubaidi et al., 2021). The main goal of these layers is to convert the input weighted sum with bias into a number between some upper and some lower limit. Furthermore, activation functions have the power to decide whether to fire a specific neuron or not depending on its input. Thanks to error backpropagation that is used to train the neural network, activation function can detect the most significant feature. There are some common activation functions that are widely used in CNN and other deep learning networks:

a- Sigmoid activation function: Its output value ranges between zero and one. There are three main possibilities for sigmoid output, which are completely based on the input values. Firstly, if the input is negative, it will be transformed to a number, which is very close to zero. Secondly, if the input is very close to zero, then it will be transformed to a number between zero and one. Finally, if the input is positive, it will be translated into a number, which is very close to one.

$$f(x)_{sigm} = \frac{1}{1+e^{-x}} \qquad (3.1)$$

b- Tanh activation function: The output of Tanh activation function ranges between -1 and 1. Its equation is expressed as follow:

$$f(x)_{Tanh} = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \qquad (3.2)$$

c- ReLU activation function: It is the most commonly used activation function in CNN due to its lower computational load [47]. If the input is less than zero, it will be translated into zero but if the input is greater than zero, it will be transformed to the same input. Its equation is represented as follow:

$$f(x)_{ReLU} = \max(0, x) \qquad (3.3)$$

d- Leaky ReLU: It is used to solve the problem of neglecting negative values while using ReLU.

$$f(x)_{Leaky\ ReLU} = \begin{Bmatrix} x\ , & if\ x > 0 \\ mx\ , & if\ x \leq 0 \end{Bmatrix} \qquad (3.4)$$

Where **m** is leak factor, which is usually a small value like 0.001

e- Parametric Linear Units: Instead of using leak factor of value 0.001 like in Leaky ReLU, in parametric linear units the leak factor is updated during the training process [47].

$$f(x)_{Parametric\ Linear} = \begin{Bmatrix} x\ , & if\ x > 0 \\ ax\ , & if\ x \leq 0 \end{Bmatrix} \qquad (3.5)$$

f- Noisy ReLU: It is made by applying Gaussian noise to ReLU activation function and it is mathematically expressed as follow:

$$f(x)_{Noisy\ ReLU} = \max(x+Y)\ where\ Y \sim N\big(0, \sigma(x)\big) \ (3.6)$$

(4) **<u>Fully connected layers:</u>** These layers are located at the end of the CNN architecture and acts as a classifier in CNN. From its name, we can understand the fact that each neuron in this layer is connected to all neurons in the previous layer. The input of FC layer is the output from last pooling or convolutional layer, which is usually, a vector made from the resulted feature maps after flattening. The output of this layer represents the CNN output as we can see in figure 3.7.

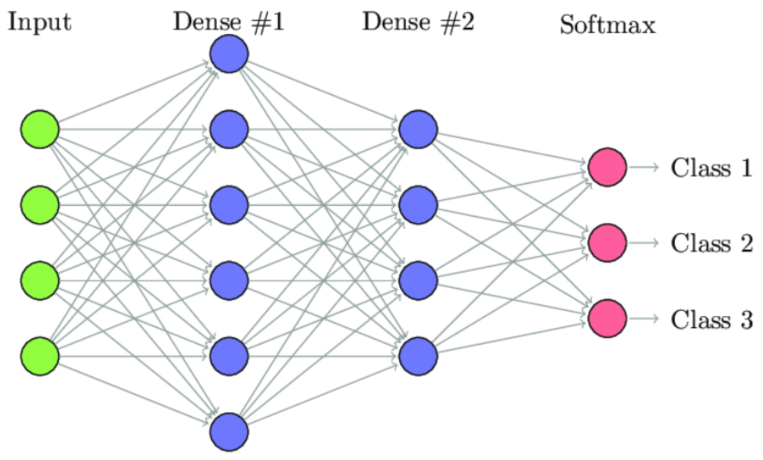


Fig. 3.7: Fully connected layers with input and output [54]

3.3.3 Loss function

Loss functions are categorized in to two main classes: Regression and classification loss functions. For regression loss functions, input is given to the neural network and the target of the model is to predict the output value e.g. Mean squared error and mean absolute error. In case of classification loss function, input is given to the neural network; the neural network produces set of probabilities where the output is the category with the highest probability among all e.g. Binary cross entropy and categorical cross entropy.

Several loss functions that are used to evaluate CNN performance will be explained in the next few lines.

a. Cross-Entropy: Its output is represented as $p \in \{0,1\}$ and commonly used in multiclass classification tasks. Softmax activation in the output layer is used to generate the output [47].

The output class probability is computed using the following equation:

$$P_i = \frac{e^{a_i}}{\sum_{K=1}^{N} e_k^a} \qquad (3.7)$$

The mathematical formula to compute cross entropy is:

$$H(p,y) = -\sum_i y_i \, Log(p_i) where \; i \; \in [1,N] \quad (3.8)$$

e^{a_i} : Non-normalized output from the previous layer

b. Mean Square Loss: It is commonly used in regression tasks and calculated using the equation:

$$H(p,y) = \frac{1}{2N}\sum_{i=1}^{N}(p_i - y_i)^2 \qquad (3.9)$$

C. Hinge Loss: it is usually used in binary classification problems especially with SVM tasks, which depends on maximum margin-based classification [47].

$$H(p,y) = \sum_{i=1}^{N} \max(0, m - (2y_i - 1)\, p_i\,) \qquad (3.10)$$

m: margin usually set to 1

Note: p_i represents the predicted output, y_i represents the actual/real output, N: number of neurons in the output layer

3.3.4 Regularization to CNN

There are two main problems affecting CNN performance, which are overfitting, and underfitting issues [47]. To illustrate, overfitting occurs when the validation metrics are considerably worse than training metrics or the training data metrics is good and when we use the model to predict on test data the results are not accurate. Unlike overfitting, underfitting takes place when the model is not able to predict on the training data. While the balanced model takes place when the model works well with both training and test data. Regularization is a technique that helps reduce overfitting by penalizing the complexity of the model, which makes it unable to generalize well as we can see in figure 3.8. Several techniques are proposed to get a model with balanced performance and help regularization to alleviate overfitting:

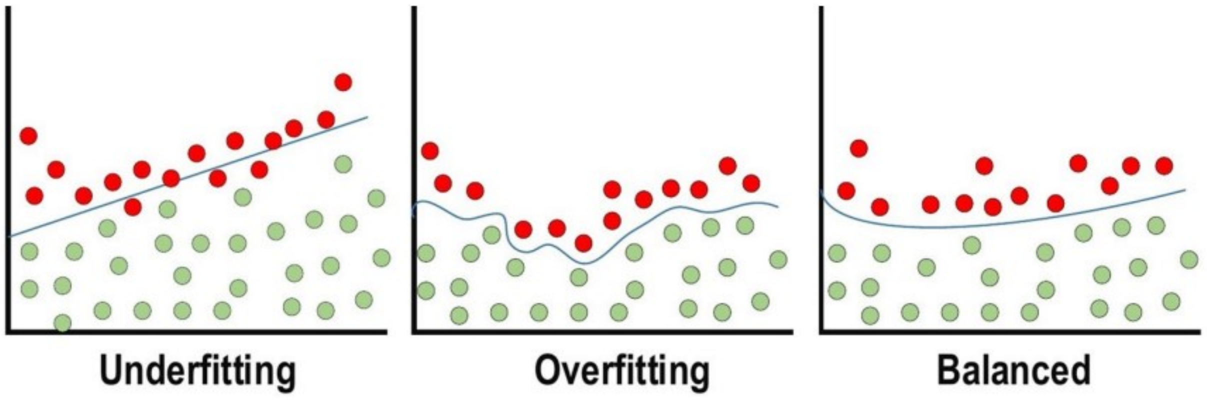


Fig. 3.8: Represents overfitting, underfitting and balanced issues [47]

a. **Dropout:** if we add dropout to our model, it will randomly ignore some subset of nodes in a given layer during training. Consequently, it prevents these dropped out nodes from participating or making prediction on the data. Values of dropout in hidden layer usually ranges between 0.5 and 0.8 in input layers.

b. **L2 Regularization:** In this technique, the sum of the square of all parameters is added to the squared difference between the real output and the predictions [47]. The higher the value of lambda, the lower the parameters will be due to penalization by L2.

   This technique would help set weights close to zero, reduce the impact of some layers, and reduce the complexity of the model. Hence, reduce the risk of overfitting.

$$L(x,y) = \sum_{i=1}^{n}(y_i - h_\theta(x_i))^2 + \lambda \sum_{i=1}^{n}(\theta_i)^2 \quad (3.11)$$

c. **Data Augmentation:** Several techniques can be applied to artificially expand the size of the training set as a solution to overfitting problem that will be discussed in the next section.

d. **Batch Normalization:** This is part of pre-processing of the data to make it ready for training. The main purpose of batch normalization is standardization by transforming data to put all the data points on the same scale because non-normalized and unbalanced data are harder to train and decrease the training speed. In addition to this, batch normalization is an efficient solution to many problems such as exploding gradient problem or vanishing gradient problem and poor weight initialization [51].

Steps for batch normalization:

1-normalize the output from the activation function

$z = \frac{x-m}{s}$ Where m: mean and s: standard deviation (3.12)

2-multiply the output by arbitrary parameter **g**

3- Add another arbitrary parameter **b** to resulting product

$$(z * g) + b \tag{3.13}$$

3.3.5 CNN Architectures

During the last decade, a myriad of researchers introduced several CNN architectures [55, 56]. The main idea is that developing the CNN architecture means improving its performance as well. To illustrate, many modifications were done in some CNN aspects such as structural, reformulation, regularization and parameter optimization [47]. However, such modifications have small effect on the performance of the CNN. On the other hand, it was noticed that the development of novel blocks and the processing-unit reorganization have significant role in upgrading CNN performance. There are several CNN architectures such as AlexNet in 2012, Network-in-Network, ZefNet, Visual geometry group (VGG), GoogLeNet, Highway Network, ResNet, Inception: ResNet and inception-V3/4, DenseNet, WideResNet, Pyramidal Net, Xception, Residual attention neural network, Convolutional block attention module, Concurrent spatial and channel excitation mechanism, CapsuleNet and High-resolution network (HRNet) [47]. In this part, some of the popular CNN architectures will be demonstrated.

1- AlexNet

It is one of the most famous CNN architectures as it achieved significant results in the world of image classification and recognition. The proposed AlexNet architecture [45] has shown a significant positive change in CNN learning ability by applying various parameter optimization strategies and changes in its depth.

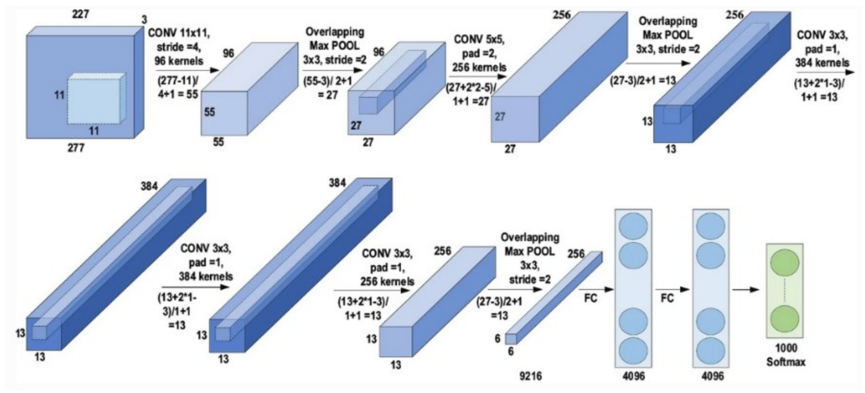

227
3
CONV 11x11, stride =4, 96 kernels
(277-11)/ 4+1 = 55
11
11
277
96
55
55
Overlapping Max POOL 3x3, stride =2
(55-3)/ 2+1 = 27
96
27
27
CONV 5x5, pad =2, 256 kernels
(27+2*2-5)/ 1+1 =27
256
27
27
Overlapping Max POOL 3x3, stride =2
(27-3)/2+1 =13
256
13
13
CONV 3x3, pad =1, 384 kernels
(13+2*1-3)/ 1+1 =13
384
13
13
CONV 3x3, pad =1, 384 kernels
(13+2*1-3)/1+1 =13
384
13
13
CONV 3x3, pad =1, 256 kernels
(13+2*1-3)/ 1+1 =13
256
13
13
Overlapping Max POOL 3x3, stride =2
(27-3)/2+1 =13
256
6
6
9216
FC
4096
FC
4096
1000 Softmax

Fig. 3.9: Represents AlexNet Architecture [47]

Figure 3.9 shows the structure of AlexNet. As we can see, there are seven feature extraction stages instead of five in LeNet, which boosted the capability of CNN while using different categories of images. In addition to this, in order to make sure that the algorithm is reliable, it was passed through many transformational units during training stage, which helped in tackling vanishing gradient problem. Furthermore, ReLU activation function [60] and filters with large sizes were used in earlier layers to provide high convergence rates [61] and improve the overall network performance.

Despite its effectiveness of depth in generalization, there was overfitting problem that was tackled by using Hinton's idea [58, 59]. Moreover, overlapping subsampling and local response normalization were used also for better generalization to overfitting.

In conclusion, AlexNet represents a powerful innovation in CNN generations as well as CNN applications.

2- Visual geometry group (VGG)

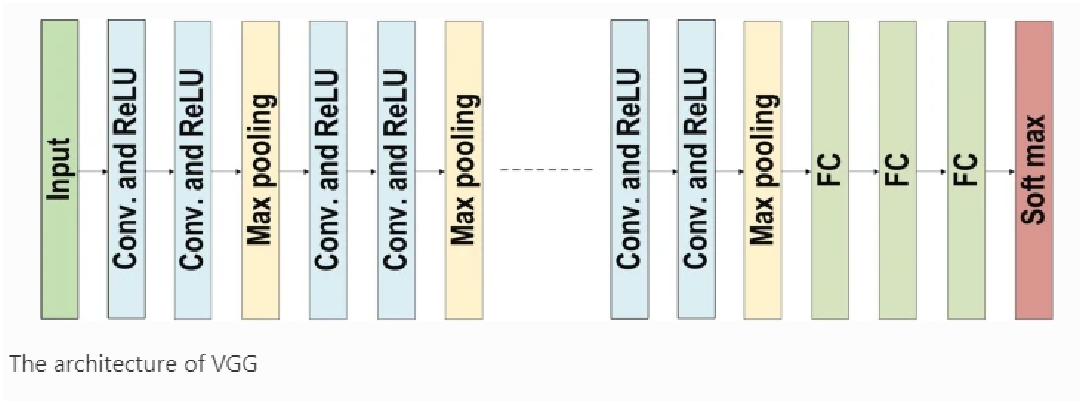


Fig. 3.10: VGG Architecture [47]

A multilayer model [62] with nineteen layers more than in ZefNet [63] and AlexNet [45] was introduced by Simonyan and Zisserman, which was called VGG Net. This model assured its efficiency by using 3 × 33 × 3 filters instead of 5 × 55 × 5 and 11 × 11 filters that were previously used by ZefNet. To illustrate, the use of small sized filters in VGG introduced new architecture with simple computation compared with previous models. As we can see in the figure 3.10 the architecture started with convolutional layer and the max pooling layers were added [64] and padding was considered to keep the spatial resolution, in order not to lose important information that are on the edges.

To summarize, VGG was the best choice at that time for localization problems and image classification tasks [47]. However, the computational cost that used about 140 million parameters is the main drawback for VGG. Consequently, nowadays it is no more among the best.

3-High-resolution network (HRNet)

This network represents a robust backbone in computer vision Position-sensitive tasks like semantic segmentation, object detection and estimation of human pose [47]. Nowadays, images are converted from high-resolution representation to lower ones using networks such as VGGNet and ResNet and then these images can be retrieved in higher representation resolution again. On the other hand, HRNet plays a fundamental role in keeping the images in high-resolution representation during the whole process [65, 66].

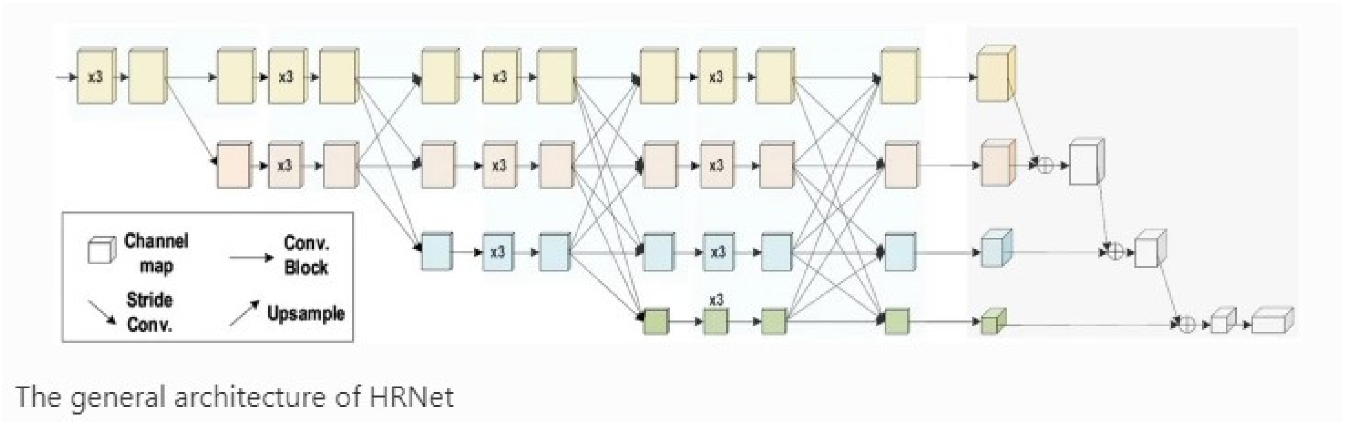


Fig. 3.11: General architecture of HRNet [47]

As we can see in figure 3.11, there is a parallel connection between high-to-low convolution series, which increases the speed of exchanging information between two resolutions and the accuracy in spatial domain.

3.3.6 Data Augmentation techniques

Data augmentation is one of the effective solutions to tackle overfitting problem by improving attributes and size of the available dataset [50]. Data augmentation techniques will be demonstrated in the next lines.

1-Cropping: Random Crop is one of the most common techniques that are used in data augmentation. This technique is simply done by cropping a dominant patch from each image in the dataset. To illustrate, if we want our model to learn about objects that are not clearly seen in our images, we should first train the model to generalize better.

2-Rotation: This technique can be applied by rotating images to the left or right directions in range from 0 to 360 degrees. The angel of rotation is usually decided based on the required task. For instance, in some cases like digit recognition tasks, it is better to use small angle of rotation (from 0 to 20 degrees) in order not to lose the data label while using large rotation angle [47].

3-Flipping: It is the simplest data augmentation technique among all. Vertical flipping techniques is less popular than horizontal one. The use of this technique on ImageNet and CIFAR-10 datasets helped in achieving the desired results. It is recommended to use flipping technique in text recognition tasks, as it is not affecting label information.

4-Translation: Shifting the image up, down, left, right is an effective transformation to alleviate positional bias problems (J. Zhang, 2021). Putting into consideration that when translating initial images in a specific direction, Gaussian or random noise should be added to the residual space. Using padding, we can keep the spatial dimensions of image after augmentation.

5-Noise injection: This technique can be achieved by applying a matrix of arbitrary values from Gaussian distribution to our dataset. According to a research study conducted by Moreno et al., injecting noise to images in nine datasets from UCI repository helps CNN to learn more robust features [52-53].

3.3.7 Evaluation Metrics

Deep Learning algorithm evaluation is the key to the success of any project. For instance, if you want to have an optimized classifier, you have to go through two major steps, which are training and testing [48]. There are many evaluation metrics for ML algorithms like accuracy, sensitivity or recall, specificity, precision, F1-score, J Score, False Positive Rate (FPR), Area under the ROC curve [47]. Some of these metrics with equations will be explained in the next lines.

1- Accuracy: It calculated as the ratio between the correct predicted classes and the total number of evaluated samples.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (3.14)$$

2- Sensitivity or Recall: It calculated as the fraction of correctly classified positive patterns.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (3.15)$$

3- Specificity: It calculated as the fraction of correctly classified Negative patterns.

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \quad (3.16)$$

4- Precision: It is computed by using positive patterns that are correctly predicted by all positive class prediction patterns.

$$Precision = \frac{TP}{TP + FP} \quad (3.17)$$

5- F1-score: It calculated as the average between precision and recall.

$$2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3.18)$$

6-J Score: It is calculated as the sum of sensitivity and specificity with subtraction of one.

$$\text{J}_{\text{Score}} = \text{Sensitivity} + \text{Specificity} - 1 \quad (3.19)$$

7-False Positive Rate (FPR): It is calculated as the probability of false alarm ratio.

$$\text{FPR} = 1 - \text{Specificity} \quad (3.20)$$

8- Area under the ROC Curve: It is calculated by using the following formula [49].

$$\frac{\text{S}_\text{P} - \text{n}_\text{P}(\text{n}_\text{n} + 1)/2}{\text{n}_\text{P}\text{n}_\text{n}} \quad (3.21)$$

S_P : Sum of positive ranked samples
n_P : number of positive samples
n_n: number of negative samples

Note: all equations about evaluation metrics are from reference [47]

Chapter 4

CT Dataset

4.1 CT Log Scanner Overview

Microtec provided the images that were used to train and test the model. The dataset includes **962** wood log photos with same height but different lengths, extending horizontally to the right edge of the Image. The images were coming from CT Log scanner. The images were classified into two classes wood logs with almost no knots and wood logs with many large knots.

It has always been the dream of saw millers to know what is inside the log before sawing. Now this could be done through the 360° x-ray computed tomography [4]. We can get full digital 3D reconstruction up to 180 m/min (590 ft /min). Bucking optimization, which helps in internal defect detection e.g. pith, sound knots, dead knots, splits, resign pockets [4]. In addition to that, breakdown optimization, which helps to correct skewing, find optimum rotation, choose the best cutting pattern. As a result, up to 8% average value increase per log, as we can know the value of the final product before sawing. The main advantage of using CT Log (optimized sawing) over the traditional log breakdown is the maximization in the overall resale value.

Some of the information that we can get from CT Log will be explained with real examples in the next heading, where each single image from CT Log contains an array of keys and each key has number of slices. Each key represents a specific measurement in a log.

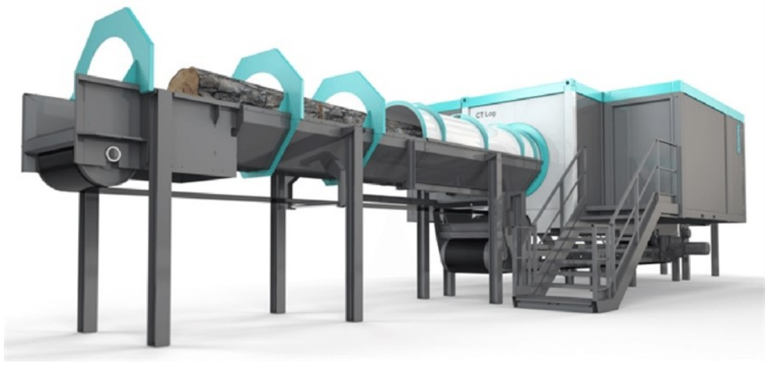


Fig. 4.1: CT Log scanner

4.2 Wood log information

4.2.1 Borders

I. Introduction

The log consists of a myriad of slices for every slice **Z**; there is a border which is usually measured and saved in **key 1**. This border is formed as a collection of **x** and corresponding **y** values. These **x** and **y** values forming border usually have circle shape. Hence, the length of the log is equal to the total number of slices.

Note: slice **Z** is 1cm, so by every move of 1 cm in the log, there is a border.

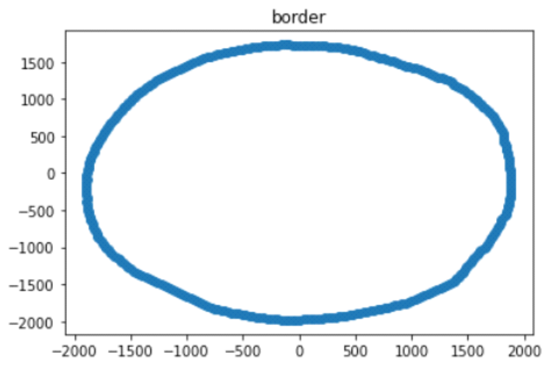


Fig. 4.2: Real border and scattered plot represents border in slice 325

Figure 4.2, the image on the left shows shape of a border form wood truck while the other represents border in slice 325 in key 1 of the CT image for the wood with ID: **154113**. As we can see that the border has a circle shape and it is formed as an array of **x** and corresponding **y** values. The shape of the border represents the total number of points (**x**, **y**) forming the border. Here in this figure, there are 360 points (Discrete values for **x** and **y**) forming this circle, this means also that we have 360 angles.

For each **x** and corresponding **y** points in the 2D array, forming the borders there is a corresponding angle that can be calculated using the following equation:

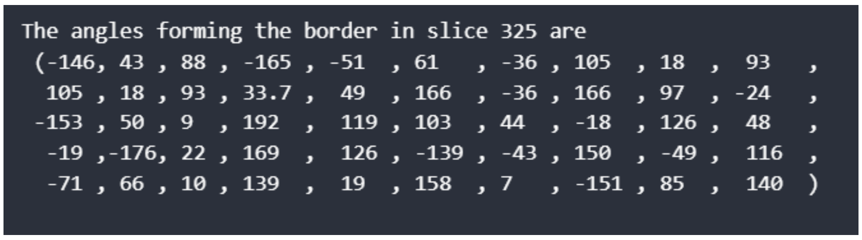
$$tan^{-1} = \frac{y}{x} \quad (4.1)$$

For example, in figure 4.2:

We have at a point (x = **30**, corresponding **y** = **20 coordinate)** by using equation 4.1 we can get the value for the angle which is in this case = **33.7** degrees.

To conclude, in order to get the values for the 360 angles, all **x** and corresponding **y** coordinates that are saved in the 2D array-forming border in slice 325 in key 1 (CT data) should be used to compute the angle at each **x** and the corresponding **y** coordinate.

As we can see in figure 4.3, which includes the first 50 angles forming the border in slice 325 and by the same way, we can get all the 360 angles forming the border.



```
The angles forming the border in slice 325 are
 (-146, 43 , 88 , -165 , -51   , 61    , -36 , 105  , 18  ,  93    ,
  105 , 18 , 93 , 33.7 ,  49   , 166   , -36 , 166  , 97  , -24    ,
 -153 , 50 , 9  , 192  ,  119 , 103   , 44   , -18  , 126 ,  48    ,
  -19 ,-176, 22 , 169  ,  126 , -139  , -43  , 150  , -49 ,  116   ,
  -71 , 66 , 10 , 139  ,  19   , 158   , 7    , -151 , 85  ,  140  )
```

Fig. 4.3: First 50 angles forming borders in slice 325

II.comparison

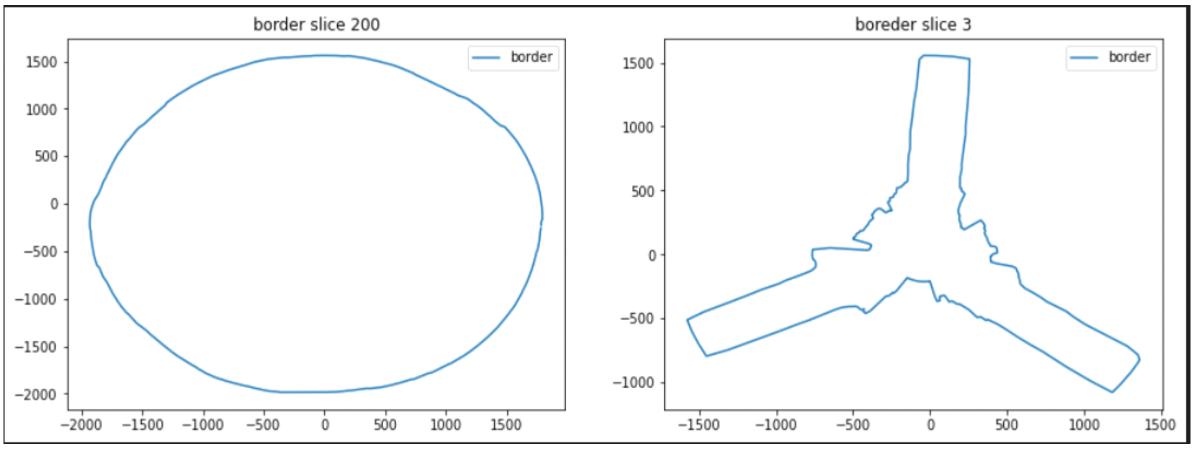


Fig. 4.4: Scattered plot represents border in slice 200 and slice 3

Figure 4.4 represents a simple comparison between two slices in the CT image with ID: **154113** where one of them is located at the beginning of the log, which is slice three, and the other is located at the end of the log which is slice two hundred. We can notice that slices found at the starting point of the log do not have the expected border shape, which is supposed to be circle shaped. On the other hand, we can find that border at slice two hundred has the expected border shape. Hence, using this comparison we can detect the reason why slices at the beginning of the log are always neglected in measurements.

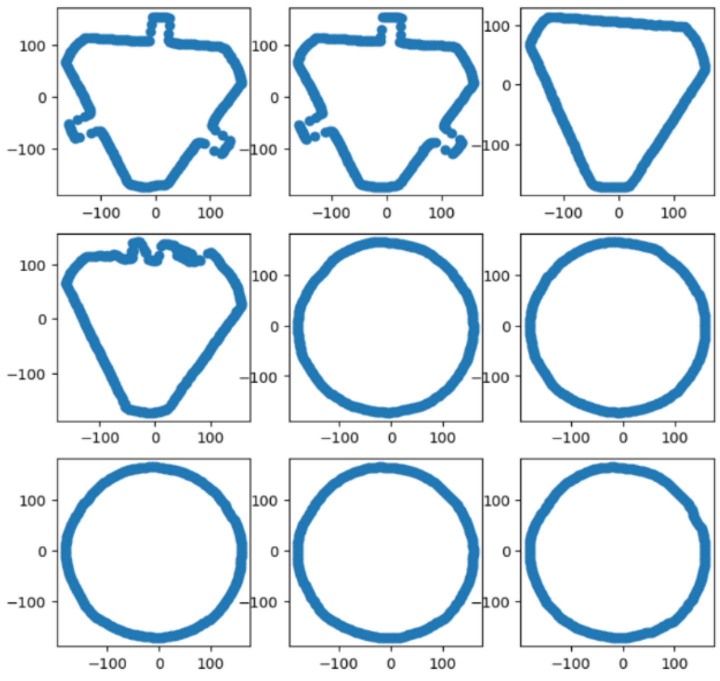


Fig. 4.5: Represents shapes of some borders

Figure 4.5 shows some examples of borders of some slices in the CT image with ID: **154113** from the same wood log. Borders that are located in the first row and the one in the beginning of the second row represents the unwanted slices whereas others with circle shape are exactly the ones that we are interested in.

Note: some causes for irregular borders shape will be explained in the next few lines.

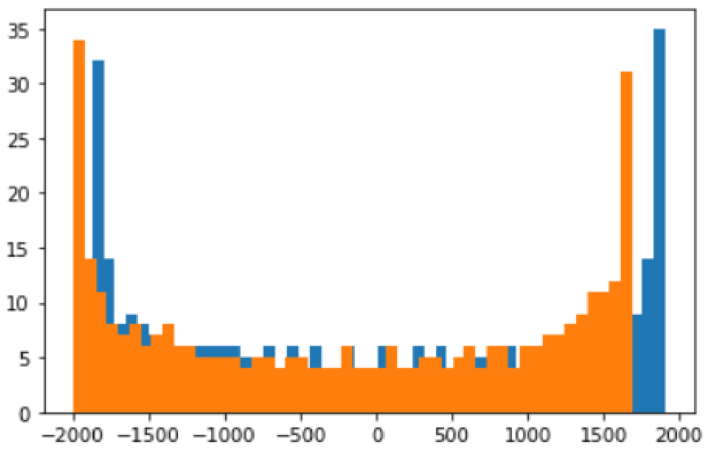


Fig. 4.6 Histogram plot represents border in slice 300

Figure 4.6 is a histogram plot for the 2D array of **x** and corresponding **y** values forming borders in slice 300 in key 1 in the CT image with ID: **154113** where the **blue colour** shows the values of **x** and **orange colour** represents **y** values.

III. Appearance of irregular borders shape

There are two main reasons for irregular borders shape that are mainly found in the first or last ten slices of a wood log (first and last 10 cm). First reason is the possibility to have some errors from the machine at the beginning of image acquisition process. Second possibility is that the branch was cut while the tree is still growing. To illustrate, if a knot was removed by man or accidentally broken, this would affect its end point and consequently irregularities in the process and the end of the branch is assumed to have an irregular shape known as scar.

4.2.2 Centroids and pith

I. Introduction

Each slice (1 cm) of a log has a border and for each border there is a centroid which is a 1D array allocating the centroid on **x** and **y** and a pith which is center or the origin of the log and it is also represented by 1D array allocating it on **x** and **y**. The measurement of centroid is usually saved in key 2 while the measurements for pith in each slice are usually saved in key 3 of an image.

II. A closer look to CT images of logs

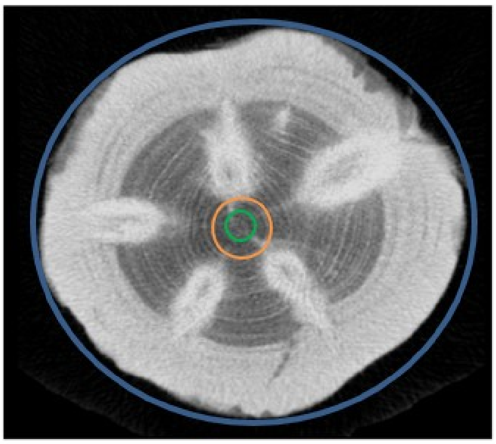


Fig. 4.7: A CT slice from a Scots pine log

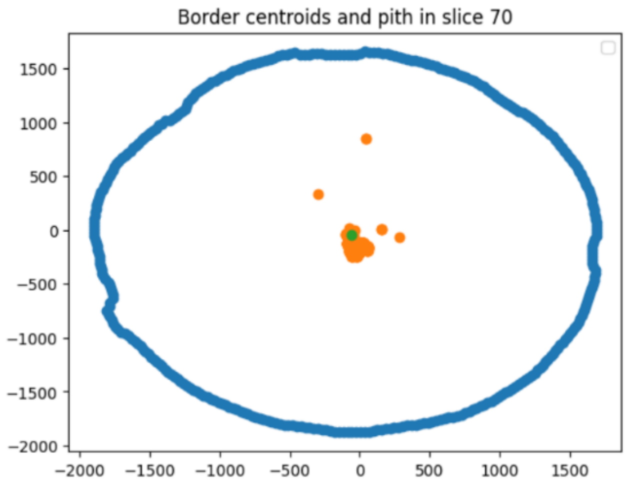


Fig. 4.8: Scattered plot for borders centroids and pith in slice 70

Some of the methods and algorithms available in the literature and dealing with knots work on the full three-dimensional log tomography, while others just analyze small portions of consecutive 2D slices. To explain, log tomographies can be considered as a set of bi-dimensional images stacked along the 16 longitudinal directions of the stem, which are also referred to as "slices". The resolution of a single slice (in the order of 1mm x 1mm) is usually quite higher than the one between consecutive slices (up to the order of the centimetre); therefore, nearly all the algorithms presented in the literature were designed to work on slices rather than longitudinally. The slice reported in Figure 4.7 was taken from the scan of a pine log performed with CT Log by Microtec [4], the scanner employed for this thesis.

Figure 4.7 and 4.8 show the borders (key 1), centroids (key 2) and pith (key 3) in slice 70 in the CT image with ID: **154113.** The **blue circle** is the border, the **orange points** are the centroids and the **green point** is the pith.

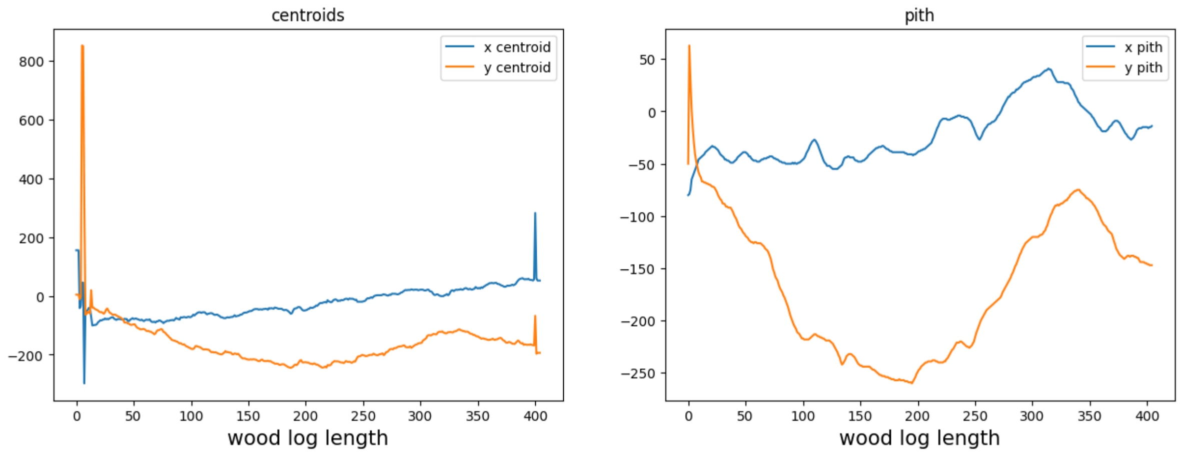


Fig. 4.9: Represents centroids and pith

Figure 4.9 represents centroids and pith where the centroids and pith measurement are saved in key 2 and key 3 in the CT image with ID: **154113** respectively. In this figure there are 405 points (values of x and y) forming the figure. These points representing the 1D array of **x** and **y** values allocating centroid and pith in each slice of a log. We can also notice that centroids and piths values are close to each other at some points and intersect at other points. As each border has one centroid and one pith, therefore 405 is the number of slices so it is the length of the log.

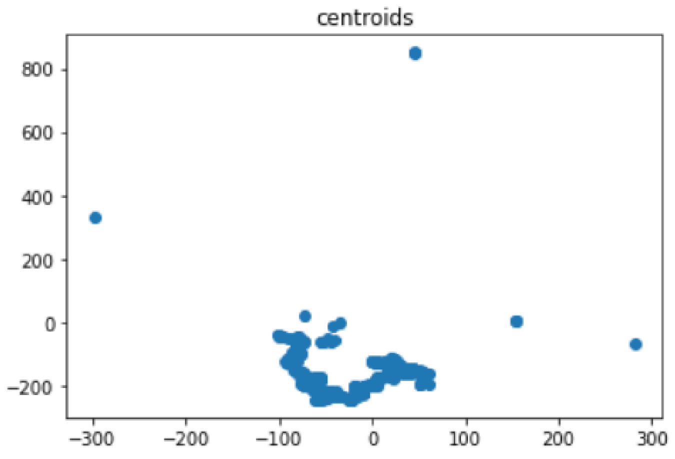


Fig. 4.10: Scattered plot for centroids

Figure 4.10 shows a scattered plot for centroids (key 2 in the CT image with ID: **154113**) which is 1D array of **x** and **y** values. Each slice **z** has one centroid, so each point in this plot represents a centroid of a certain slice of the log.

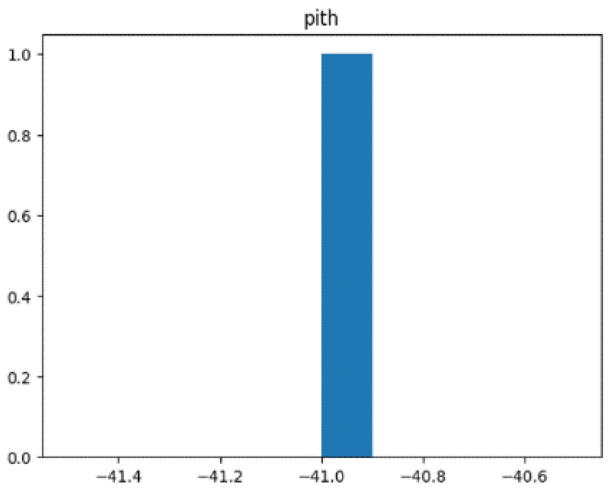


Fig. 4.11: Histogram plot for pith

Figure 4.11 illustrates the pith of the log (key 3 in the CT image with ID: **154113**) which is 1D array of **x** and **y** values allocating the pith on **x** and **y** coordinates. Pith is usually the point at which the knot starts.

4.2.3 Knots

I. Introduction

Knots can be described as imperfections from branches that can lead to the growing of living wood grain around them. These imperfections are taking part not only in making wood a beautiful material, but also can contribute to possible defects in structural strength for construction lumber. Dead branches drop off healthy, living trees all the time, therefore wood knots appear in the trunk where branches died.

II. What are Knots?

Knots in wood are referred to be located where the base of a tree branch met the tree trunk, interrupting the flow of the grain pattern. The knot is roughly found where the branch was cut off. Knots usually display round dark areas. Some of them have solid shape while others fall out partially or completely.

Knots are also defined as broken off/cut limbs or sprout branches that reveal exposed wood, either sound or rotten. In other words, these are common blemishes in trees, which often cause holes or lumps within a specific trunk of the tree.

III. How are knots formed on tree trunks?

Knots are formed because of a plethora of reasons. To begin with, knots can be formed due to natural growth of tree. In other words, while the tree is growing, its lower branches are about to die and their bases may become overgrown and then enclosed by subsequent layer of truck wood. As a result of this process, an imperfection appears in the tree trunk, which we call a knot. We can now conclude that knots appear in places where branches once were. Therefore, wood in the knot is very tough and even harder than the wood surrounding it. That is why, it forms (knot) a hard bulge around the branch emerging from the center, and sometimes forms holes in the wood.

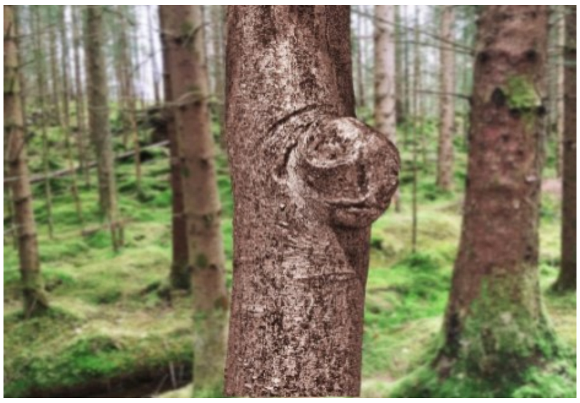


Fig. 4.12: Knots formed due to natural growth of tree

Second reason for the appearance of knots could be injuries to the tree itself. This type of knots is called "loose knots “or fungal infection that can easily spread to other trees. Third reason for knot formation in trees is "Black knot” disease. This disease causes rampant knot formation in some trees, which is harmful for trees.

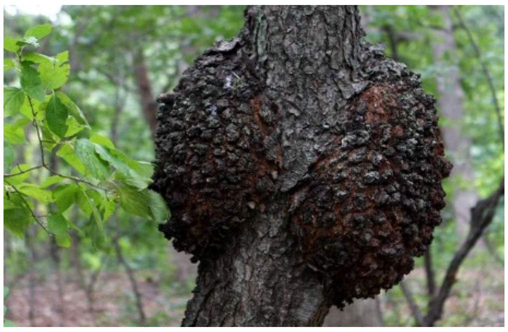
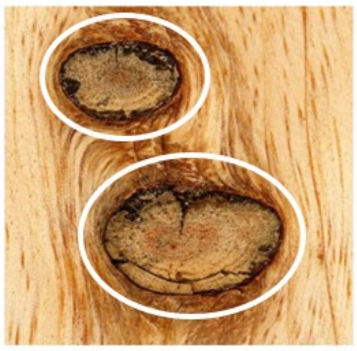


Fig. 4.13: Knots formed due to Black-Knot disease

IV. Effects of knots

Besides their contribution in making wood shape attractive and creating beauty, knots have a negative impact on technical properties of wood. To illustrate, they are not only hard to cut, but also leads to reduction in the strength of the surrounding wood. In addition to this, they do not affect the stiffness of structural timber because strength and stiffness rely more on wood that sounds good than upon localized defects.



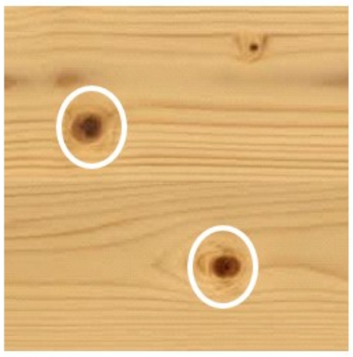


Fig. 4.14: Knots on a piece of log and knots on a piece of wood board

On the other hand, knots are exploited for visual effects. To explain, knots on trunks add to the aesthetic appeal of the planks that are sawn from those trees. For example, if want to maintain the natural beauty of wood piece and there is a knot present, there are two choices either to fill the knot or to leave it open. The decision in this case is based on whether we need the piece to be smooth or has a rustic look. For rustic look of the wood piece, we will go for leaving the knot, as it is (open). For the other choice, if we have a dining table or bedroom dresser, in this case we will need to fill it to achieve smooth touch of this piece of wood.

Fig 4.15: Table with knots not filled

Fig. 4.16: Table with all knots filled

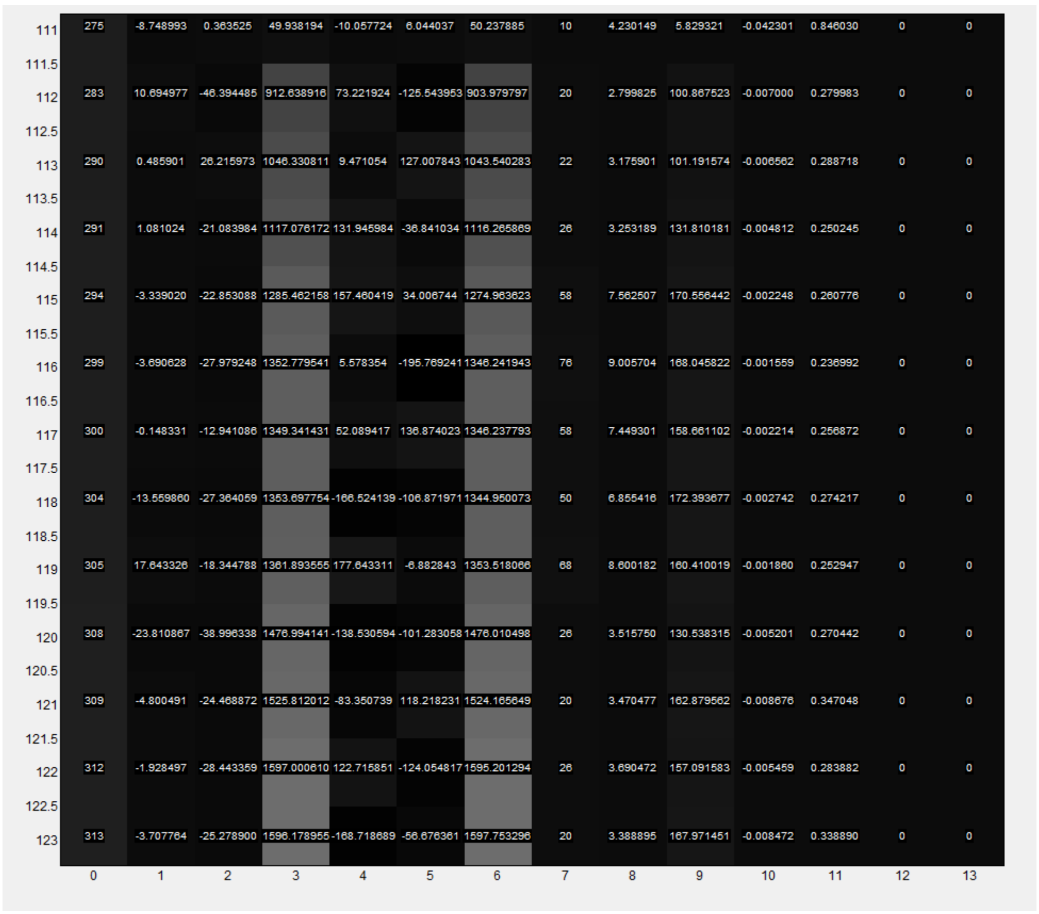
It is always recommended to fill knot holes that are found on horizontal surfaces like table tops and dressers for many reasons:

- creating a unique design.
- Keeping surface of the object smooth.
- Preventing food particles or spilled liquids from collecting in the holes (knots) which may lead to bacterial growth.
- Easy to clean as it is not exposed to building up of dust.

V. Knots modelling and measurement

1. Knots representation using 14 descriptors saved in key 4

Each slice (**Z = 1cm**) in each wood log may have a myriad of knots and each knot is characterized by **14** descriptors describing the knot characteristics. Each row represents a knot, so the total number of rows represents total number of knots in a specific slice of a log. Each column represents one of the descriptors for a knot. Here there is a description of columns with names: **column 0=Knot-ID, column 1=x-start, column 2=y-start, column 3=z-start, column 4=x-end, column 5=y-end, column 6=z-end, column 7=DKB (Dead Knot Border), column 8=Radius, column 9=Length, column 10=A, column 11=Column 12=Column 13=D**



	0	1	2	3	4	5	6	7	8	9	10	11	12	13
111	275	-8.748993	0.363525	49.938194	-10.057724	6.044037	50.237885	10	4.230149	5.829321	-0.042301	0.846030	0	0
111.5														
112	283	10.694977	-46.394485	912.638916	73.221924	-125.543953	903.979797	20	2.799825	100.867523	-0.007000	0.279983	0	0
112.5														
113	290	0.485901	26.215973	1046.330811	9.471054	127.007843	1043.540283	22	3.175901	101.191574	-0.006562	0.288718	0	0
113.5														
114	291	1.081024	-21.083984	1117.076172	131.945984	-36.841034	1116.265869	26	3.253189	131.810181	-0.004812	0.250245	0	0
114.5														
115	294	-3.339020	-22.853088	1285.462158	157.460419	34.006744	1274.963623	58	7.562507	170.556442	-0.002248	0.260776	0	0
115.5														
116	299	-3.690628	-27.979248	1352.779541	5.578354	-195.769241	1346.241943	76	9.005704	168.045822	-0.001559	0.236992	0	0
116.5														
117	300	-0.148331	-12.941086	1349.341431	52.089417	136.874023	1346.237793	58	7.449301	158.661102	-0.002214	0.256872	0	0
117.5														
118	304	-13.559860	-27.364059	1353.697754	-166.524139	-106.871971	1344.950073	50	6.855416	172.393677	-0.002742	0.274217	0	0
118.5														
119	305	17.643326	-18.344788	1361.893555	177.643311	-6.882843	1353.518066	68	8.600182	160.410019	-0.001860	0.252947	0	0
119.5														
120	308	-23.810867	-38.996338	1476.994141	-138.530594	-101.283058	1476.010498	26	3.515750	130.538315	-0.005201	0.270442	0	0
120.5														
121	309	-4.800491	-24.468872	1525.812012	-83.350739	118.218231	1524.165649	20	3.470477	162.879562	-0.008676	0.347048	0	0
121.5														
122	312	-1.928497	-28.443359	1597.000610	122.715851	-124.054817	1595.201294	26	3.690472	157.091583	-0.005459	0.283882	0	0
122.5														
123	313	-3.707764	-25.278900	1596.178955	-168.718689	-56.676361	1597.753296	20	3.388895	167.971451	-0.008472	0.338890	0	0

Fig. 4.17: Some knots representation with the 14 descriptors

Each row in figure 4.17 represents a knot from the CT image with wood log ID: **154113**. As we can see in the figure, we have knots numbers starting from 111 to 123. For each knot number (row), there are the 14 descriptors, which provide information about the ID, DKB, starting, and ending coordinates, radius and length of this knot.

2. The Descriptors explanation

In order to properly estimate the impact of knots on the grading of logs resulting from sawing, according to a chosen cutting pattern, it is necessary to know their size at a given position. This can be done via a proper modelling of knots by defining some parameters.

Before proceeding with a brief description of the most important knot parameters having a role in knot modelling, it is necessary to define a reference system for log and knots.

From here on, the longitudinal (**l**) direction of the log will be the vertical one following tree growth, the radial direction (**r**) will be the one running parallel to log radius (i.e. perpendicular to knot radius), while transversal (**t**) direction will be perpendicular to it.

Some of the most important knot parameters are:

- Starting point (column 1= x-start, column 2 = y-start, column 3 = z-start):
  It is the point where the knot originates, the innermost and closest to log center. Since it most often resides in the pith, it is itself known by "pith". When dealing with knot models it is most often chosen as the origin point for the reference system, thus meaning that all its **r**, **t** and **l** coordinates are equal to zero.

- Knot end (column 4 = x-end, column 5 = y-end, column 6 = z-end):
  As the name suggests, it is the ending point of the knot. If the knot dies at a certain point, it is most likely to occur inside the log, while it is on log surface for sound knots or knots, which stopped growing in its close proximity (i.e. its part reaching log edge is made up of dead wood)

- Length (column 9 = Length):
  It coincides with the radial distance of knot end from the pith. For a knot reaching the edge of the trunk, it indeed coincides with log radius along the direction marked by knot growth. In some cases, a knot may happen to be pruned by man or accidentally broken: this truncation affects its end point (and therefore its length) and has a detrimental effect with respect to the accuracy of knot detection in tomographic images since, because of irregularities in the process; branch end assumes an irregular shape known as scar.

- Diameter (column 8 = 2×Radius):
  At a given radial coordinate, it is the maximum width of the knot in the **radial-transversal**-plane. When referring to knot diameter, it is the maximum diameter along the radial direction.

- Dead Knot Border (column 7 = DKB):
  Radial coordinate at which the transition from sound to black knot takes place. Knots reaching log border without changing state (therefore globally labelled as sound) do not indeed exhibit one.

The following Figure 4.19 shows the projection of a knot on a single cross-section along the **l** axis and reports a representation of the parameters described above. DKB and (knot) diameter are represented in two separate lines for the sake of clarity, even though many models make them coincide.

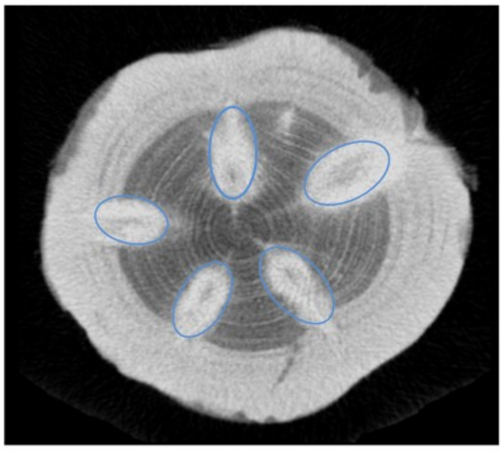


Fig. 4.18: A CT slice from pine log

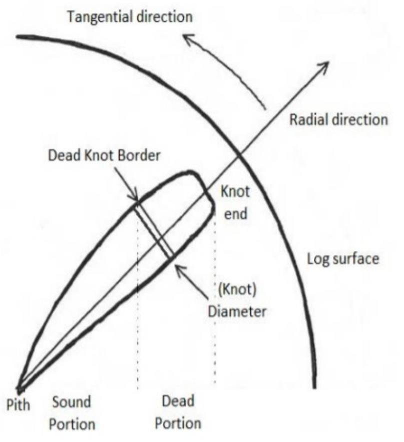


Fig. 4.19: Representation of principal knot parameters [70]

Figure 4.18, where it is also possible to notice five knots, clearly distinguishable as lighter zones (**inside the blue circles**) with an approximately elliptical shape around which age rings slightly bend towards the outside of the log.

Notes:
(1) x-start, y-start represent allocation of the start of the knot on x-axis and y-axis, while x-end, y-end represent allocation of the end of the knot on x-axis and y-axis
(2) z-start and z-end of the knot representing more or less index of the slice where the knot is located in
(3) Knot radius is how big the knot is so it will be used to decide on the weight to give to this knot

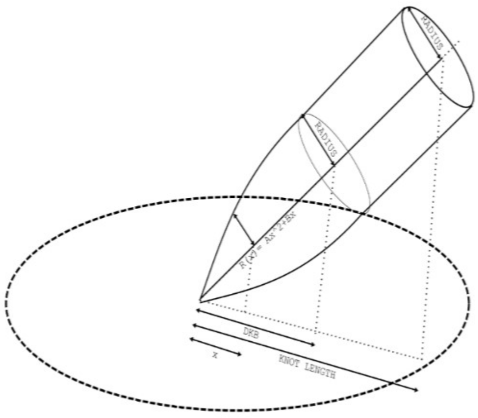


Fig. 4.20: Represents shape of normal knots

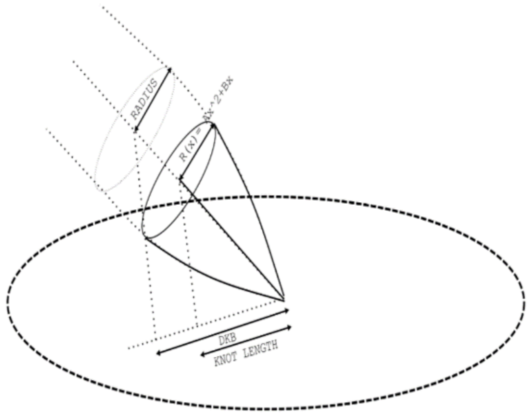


Fig. 4.21: Represents shape of other type of knots

The figures 4.20 and 4.21 above show two types of knots, which are normal type knots and other types of knots. As the aforementioned figures 4.20 and 4.21 illustrate, if the knot length (**column 9**) is larger than dead knot border (**column 7**), this knot will be of normal type knot. Else, if the knot length is smaller than dead knot border, in this case this knot will belong to other types but not normal type knot. The ways for computing volume are different for both types.

In knots whose type is **normal** as in Figure 4.20:

$$\text{Volume of Cone} = \text{Dead knot Border} \times \text{knot radius} \times \text{knot radius} \times \text{pi} \quad (4.2)$$

$$\text{Volume of Cylinder} = (\text{knot length-Dead Knot Border}) \times \text{knot radius} \times \text{knot radius} \times \text{pi} \quad (4.3)$$

Volume_Normal_Knot = Volume of Cone+ Volume of Cylinder (4.4)

Whereas in Knots that are of other types as in figure 4.21:

Volume_other_types = Knot length × knot radius × knot radius × pi (4.5)

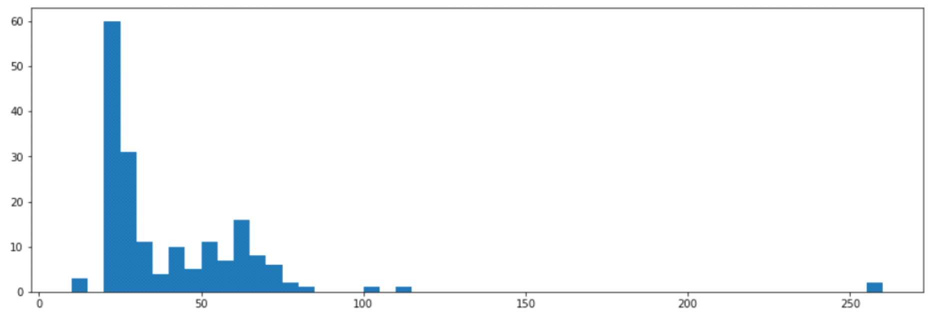


Fig. 4.22: Histogram plot for dead knot border (col: seven in the 14 descriptors)

Figure 4.22 represents a histogram plot for Average values of Dead knot Border (DKB), which is located in column number 7. This is a simple representation of all average values for all DKBs of all knots found in this slice of wood in the CT image with ID: **154113**. As we can notice also that knots with largest dead knot borders are located in the first rows, while others with smaller values are located in the middle and final rows.

4.3 Pre-processing

Before being processed by the CNN, the images undergo several pre-processing steps.

After getting all the information related to knots from the 14 descriptors as explained in 4.2, now we can use them in the following pre-processing steps:

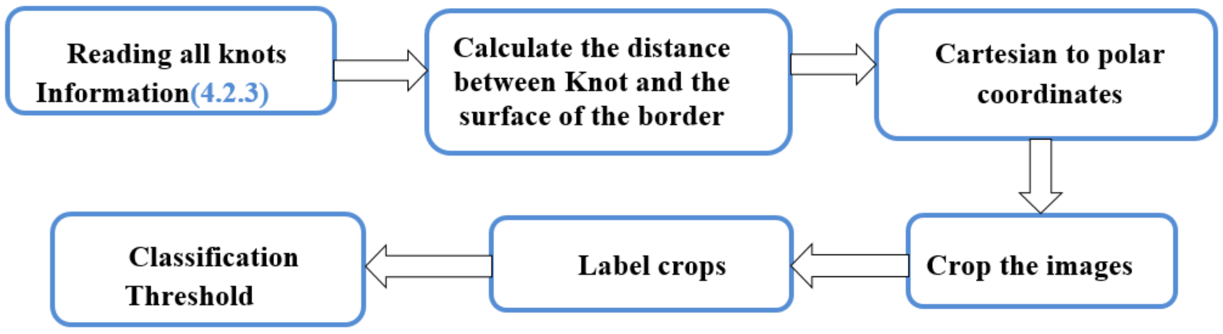

Reading all knots Information(4.2.3)
Calculate the distance between Knot and the surface of the border
Cartesian to polar coordinates
Crop the images
Label crops
Classification Threshold

Fig. 4.23: Block diagram for pre-processing steps on CT log images

4.3.1 Finding whether knot is on the surface of the border or away

I. Type of knots that we are interested in

To better understand and visualize what do we mean for the knot to be on the surface of a border or away from it. Here there are two figures explaining the difference between knots that we are interested in and the others that are discarded.

Figure 4.24 shows a knot that is away from the surface of a border, which is knot number 10 and slice 53 in this case, as we can see the **red point**, which represents the end of the knot, is away from the surface of the border in the CT image with ID: **154113**. On the other hand, we can see on figure 4.25 that the **red point** which represents end of the knot is on the surface in knot number 50 in slice 209 and this is exactly the type of knots that we are interested in (knots that are close to or on the surface) of a border of slice **(Z)**.

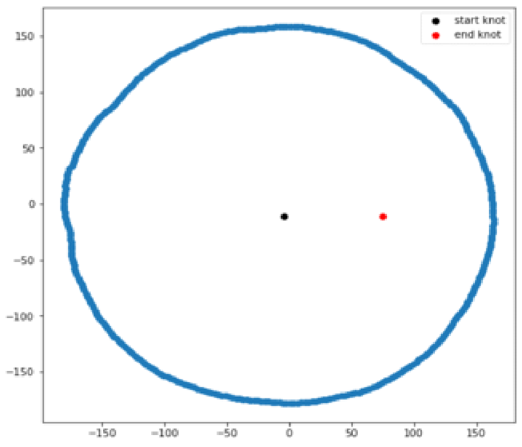


Fig. 4.24: Knot away from surface

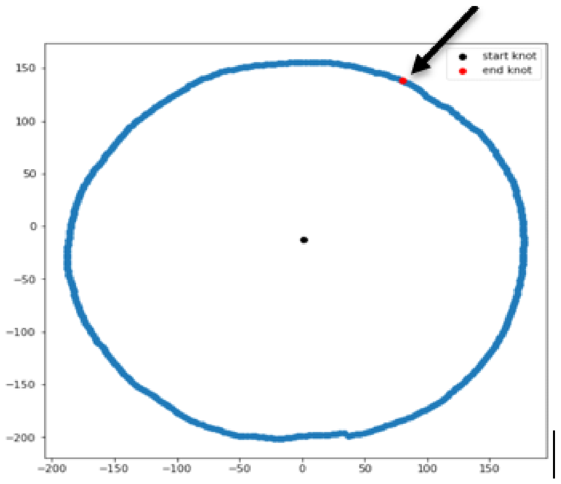


Fig. 4.25: Knot on the surface

II. Calculating distance between end of knot and the surface of the border

This step has been done to know whether to keep or discard the knot based on its distance from the surface (Depth threshold). In our project the depth threshold is **37** mm so if the knot is more than **37** mm away from the surface of the border, it will be discarded and will not be taken into consideration while computing the scores (labels).

- Firstly, get the start and end coordinates of the knot which are in the columns 1 and 2 for the start and columns 4 and 5 for the end of the knot that should be equal to the knot length in column 9 when converting start and end (**x** and **y** values) into Cartesian coordinates.
- Secondly, get index of the slice where the knot is located in from the 14 descriptors column 6 (z-end).
- Thirdly, as we know that (z-end) is index of the slice where the knot is located in. We now want to get the point of intersection with the border which we can easily get by the following equation:

$$tan^{-1} = \frac{x2-x1}{y2-y1} \qquad \textbf{(4.6)}$$

- Fourthly, by using the calculated angle on the border, we can select the local points, which are the projection of the knot length points on the border (from start of the knot to the intersection point with the border).
- Finally, use one of the following formulas for computation:

$$Difference\ in\ Distance = \sqrt{(xLocal - xend)^2 + (yLocal - yend)^2}\ \text{Or}$$

$$Difference\ in\ Distance = \sqrt{(xLocal - xstart)^2 + (yLocal - ystart)^2} - \sqrt{(xend - xstart)^2 + (yend - ystart)^2} \qquad (4.7)$$

- Knowing that the aforementioned equations give exactly same results which ensure the accuracy and certainty of the answer.

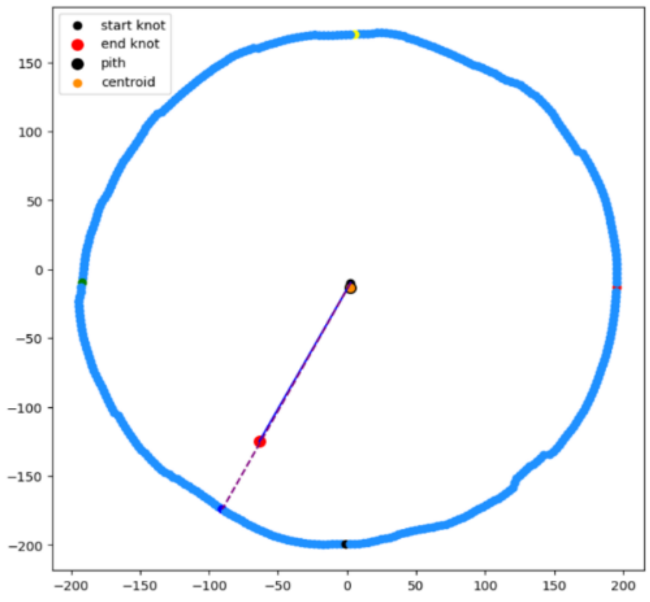


Fig. 4.26: Visualization of distance between knot end and the surface

Figure 4.26 shows a single slice of a wood log in the CT image with ID: **154113**. The circle shape, which has the blue colour, represents the border of this slice. There are two lines one of them is dashed line and the other is solid line. The dashed line represents the distance from the start of the knot to the **blue point**, which represents projection of knot length on the border. The solid line represents the exact knot length from the start of the knot to the end of the knot. By calculating the difference between the point of intersection with the border and the end of the knot, we can know if the knot is far or close to the surface. This calculation could be done using the above-mentioned equations.

III. Visualization of knots in 3D

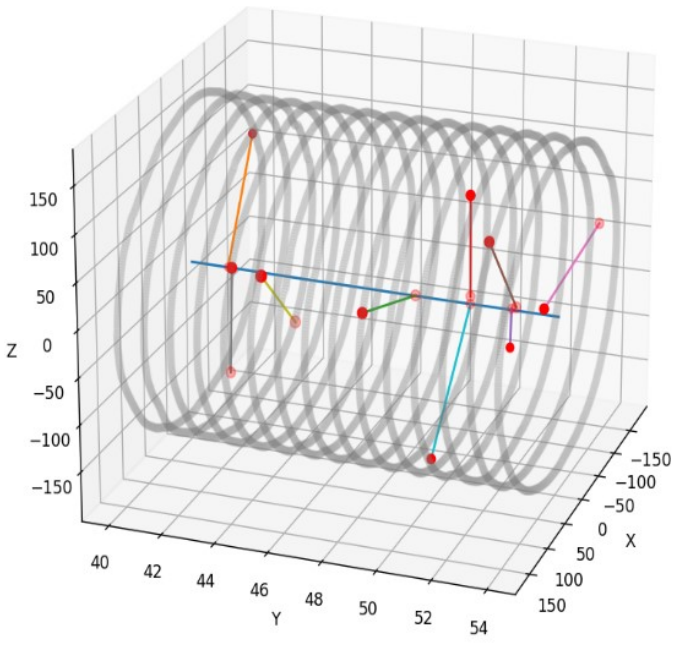


Fig. 4.27: 3D plot representing knots and their distance from borders

Figure 4.27 is a 3D plot from the CT image with ID: **154113** where the circles in **gray colour** are borders and the **blue line** passing through the middle of these borders represents the pith. The small circles in **red** on the **blue line** (pith) represent the start point for each knot (x-start, y-start). The line from the start of each knot to its end is called knot length (column 9 in the 14 descriptors).

Note: we care only about knots that are close to the borders and this can be controlled by setting certain depth threshold.

4.3.2 Convert Cartesian coordinates into polar coordinates

All images in the dataset were converted from Cartesian coordinates into gray scale polar coordinates. To illustrate, polar coordinates provide an alternate definition of complex numbers, which reveals that they are profoundly related to rotation [38]. Furthermore, if you want to describe different areas, polar coordinates are the best choice for integration. For example, it is quite easy to integrate a wall of a cylinder in polar coordinates. On the other hand, it is very difficult to do the same task in Cartesian coordinates [38].

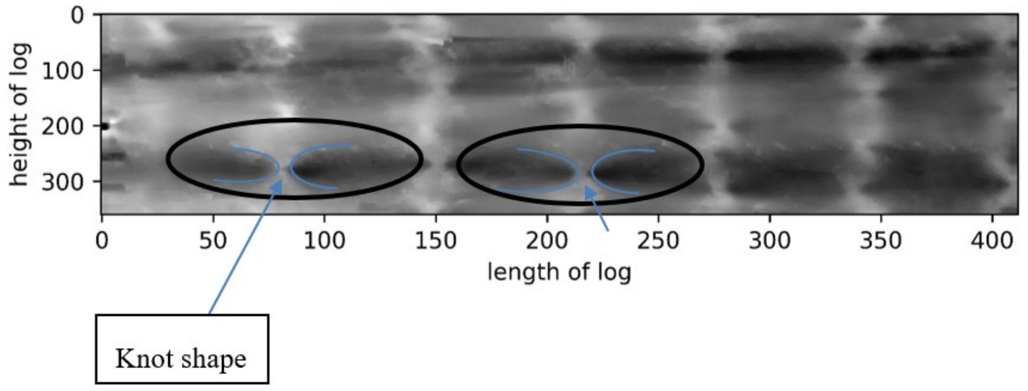


Fig. 4.28: Wood log with ID **407810730**

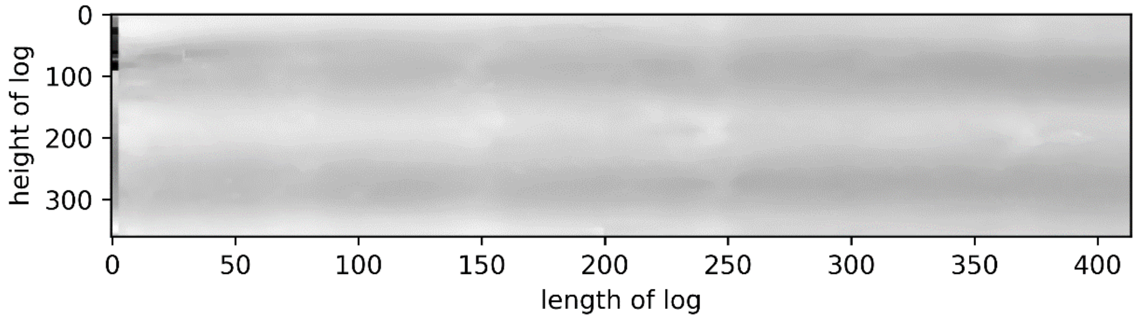


Fig. 4.29: Wood log with ID **407808210**

Figure 4.28 and 4.29 are for two images from our dataset after conversion from Cartesian coordinates to polar coordinates. To demonstrate, figure 4.28 represents a wood log with length about 4.15 meters and contains many knots “class one” where the knots have the butterfly shape. Figure 4.29 shows a wood log with almost no knots (no butterflies shape) “class 0”.

4.3.3 Crop the images

Splitting image into crops of same size is a crucial part while working with Machine Learning models in order to artificially enlarge our dataset. Moreover, the main goal of our project is not to classify the whole log but to classify each crop in each wood log in our dataset. Each wood log image was cropped into 360 cm × 100 cm for each crop. While generating crops the first and last 10 cm were neglected (removed) in order to avoid any unreliable parts of the images, **(reasons for removal were mentioned in 4.2.1 (III. Appearance of irregular borders shape))**.

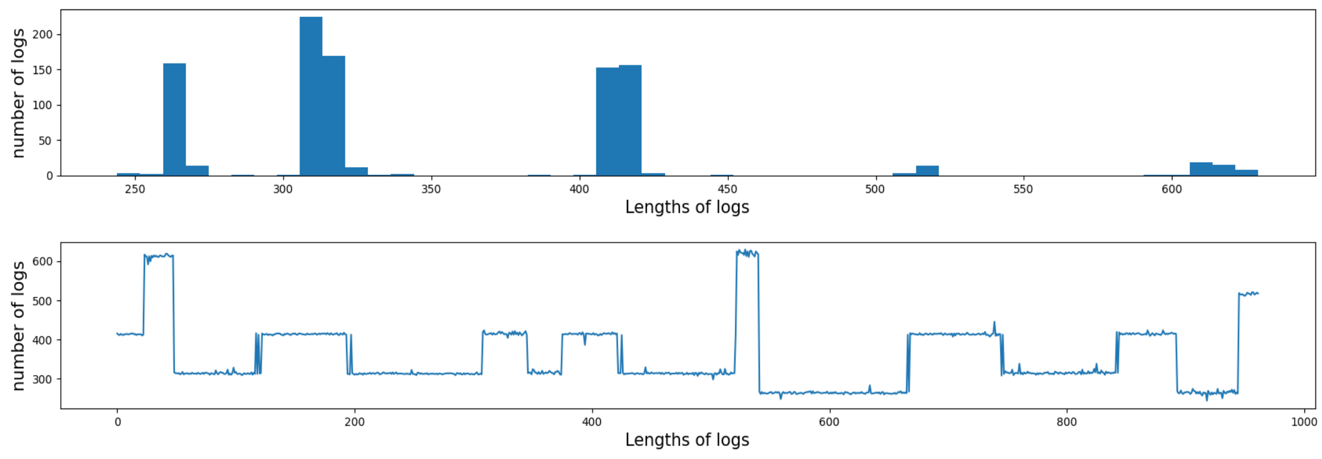


Fig. 4.30: Histogram representation for the length of wood logs in the dataset

AS we can see in figure 4.30, the horizontal axis represents the lengths of wood logs and the vertical axis represents the number of wood logs that fall in this category of length. This explains why we did not have the same number of crops for each wood log. For example, for a wood log with length 220 cm, we will have two crops with 100 cm each and the first and last 10 cm were neglected. Whereas, for a wood log with length 620 cm, we will have six crops with 100 cm each and 10 cm were removed from the start and the end of this log.

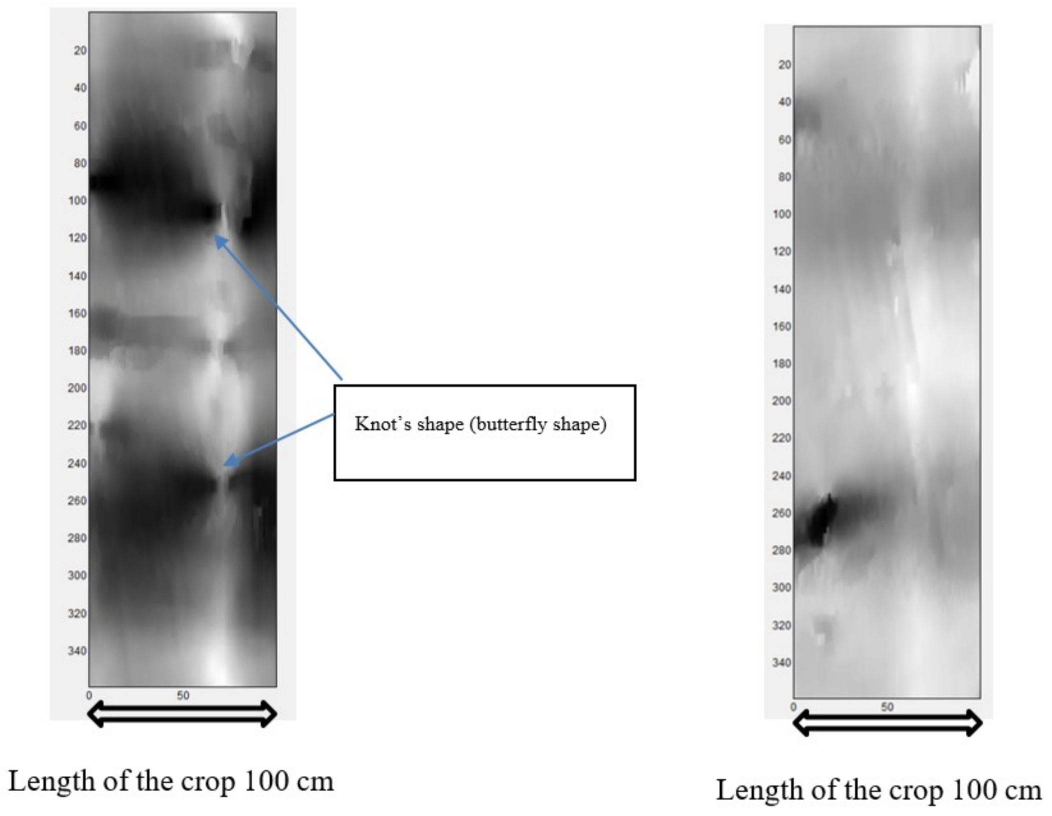

Knot's shape (butterfly shape)
Length of the crop 100 cm
Length of the crop 100 cm

Fig. 4.31: Crops from wood log with ID **407808500** and **407808890**

Figure 4.31 shows two crops from two different wood logs, where the dimensions are 360 cm×100 cm for each of them. The figure on the left can be classified as a crop with many knots" butterfly shapes", while the one on the right can be categorized as a crop with almost no knots "class 0 no butterflies shape".

4.3.4 Label crops

I. Depth threshold issues

Depth threshold: is a specific depth, which was set to discriminate between deep knots (away from the surface of the border), and knots that are close to the surface of the border (target knots). To begin with, distance between each knot found in each wood log crop and the surface was calculated using equation (**4.7**). By this way, we can get the distance between each knot and the surface of the border. So, we used these information (butterfly shape on wood log crop image and distance between each knot and the surface) to compare between the number of knots (butterflies) that we can clearly see on wood log crop image and the number of peaks (radius of knots) detected below this depth threshold. This step was done for several depth thresholds 3 mm, 5 mm, 7 mm, 17 mm, 20 mm, 35 mm, 37 mm, 40 mm and 70 mm. To demonstrate, when the number of peaks detected below the depth threshold is more or less corresponding to the visible knots on a wood log crop image, this means that it is the right depth threshold. Based on several trials with the aforementioned thresholds, we found out that at depth threshold 37 mm, number of knots (butterflies) that were visible on each wood log crop image was almost corresponding to the number of peaks (radius of knots that were detected below this depth threshold).

Note: The tested values for depth threshold were selected based on the most repeated values for distances that we get between each knot and the surface of the border (equation **4.7**).

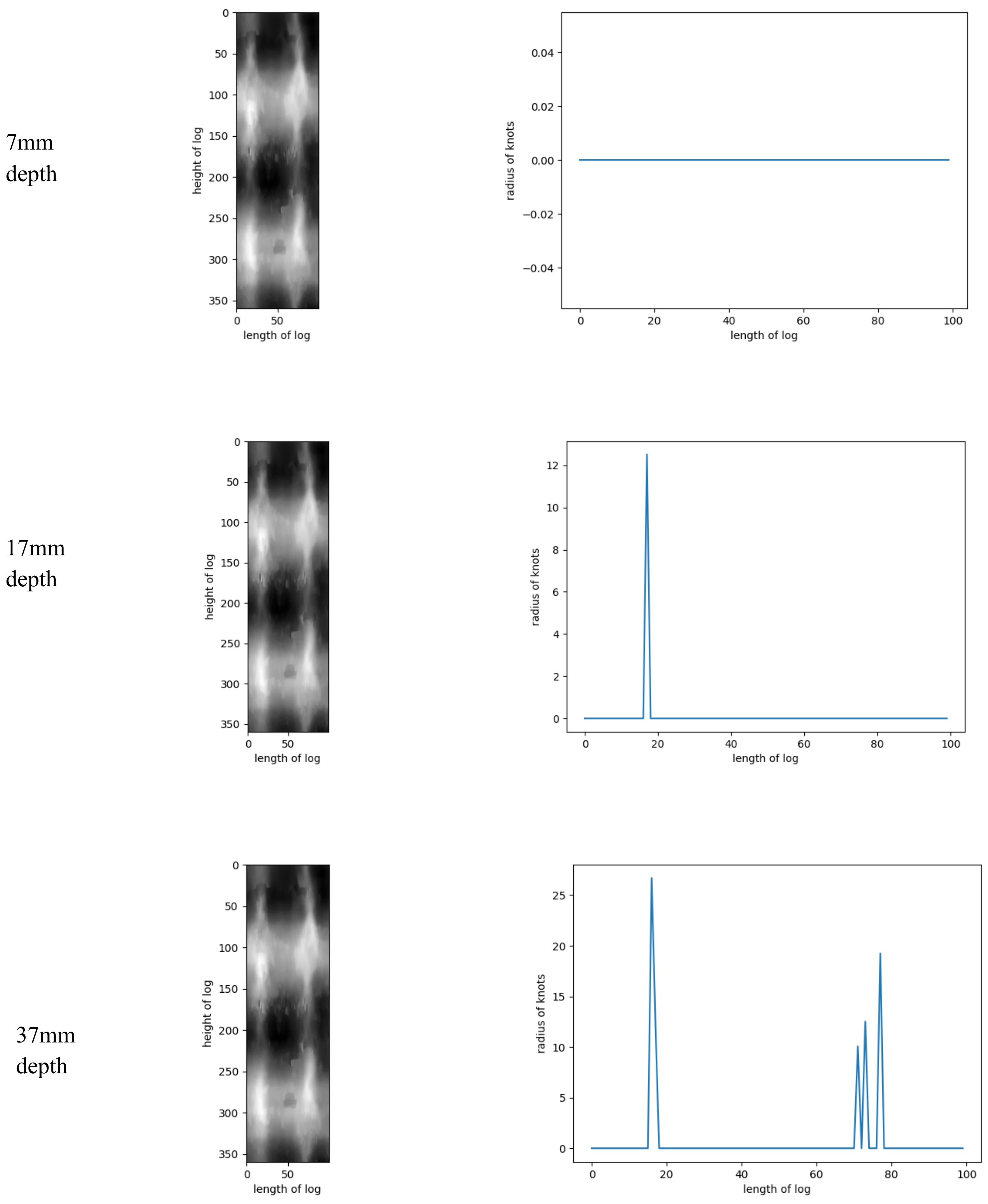


Fig. 4.32: Depth threshold issues

Figure 4.32 consists of three images for the same wood log crop, so the left side images are the same. On the other hand, the right-side images are different as they represent the detected knots below each depth threshold. For the image on the top, the depth threshold was set to be 7 mm and no knots (**blue peaks**) were detected below this depth threshold. The image in the middle represents depth threshold 17 mm and, in this case, one knot (**blue peak**) was found below this depth threshold. While on the last image, we can see four knots (**blue peaks**) that were detected below depth threshold 37 mm. Moreover, this number corresponds to the number of knots that we can detect by eyes on the grayscale image. That is why in our project we decided to consider only knots that are not more than 37 mm away from the surface of the border.

II. How labels (scores) are given to each wood log crop

Labels were given to each wood log crop based on the number and the dimensions of knots that were found in this wood log crop. The radius for each knot can be retrieved from column 8 in the 14 descriptors and the location of each knot can be retrieved from column 6 (**Z2 (or z-end)**) in the 14 descriptors. The score of each wood log crop was computed as the sum of the radius of all knots found in this wood log crop. After that, the scores for kept and discarded crops were saved in text files.

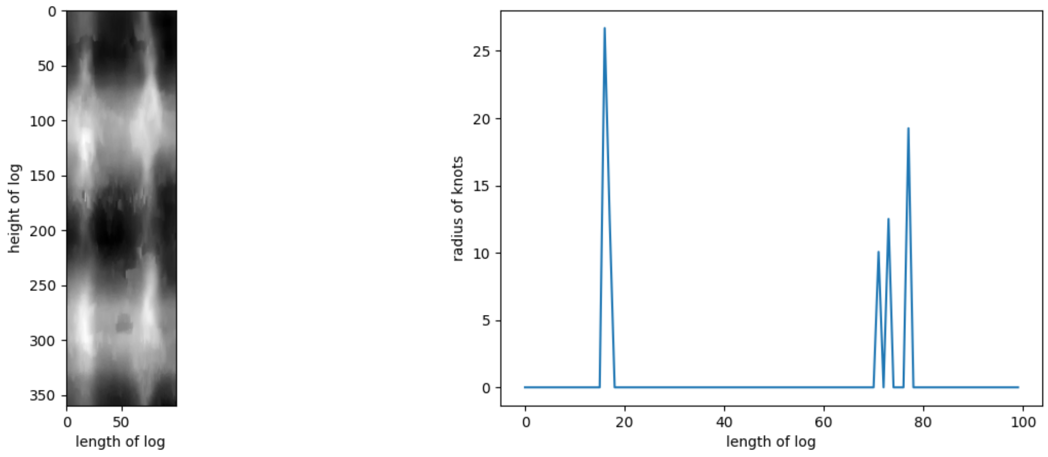


Fig. 4.33: Wood log crop and knots representation

The image on the left in figure 4.33 represents the second crop from wood log with ID: **407808220** with dimensions 360×100 cm. For the image on the right, the x-axis represents the length of the log and the y-axis (**blue peaks)** represents the dimensions (radius) of knots that were detected in this region. Consequently, the score for this wood log crop is 68.51, which is the sum of the radii of knots found in this wood log crop (26.69+12.51+19.25+10.06= 68.51).

III. Differentiate between labels and discarded labels

Labels (scores) are considered only if they are below the depth threshold. To illustrate, in this project the target is to detect knots and classify wood logs based on the number and dimensions of knot that are not more than **37** mm away from the surface of the border. Consequently, all knots that did not satisfy this depth condition were discarded (not considered in the classification).

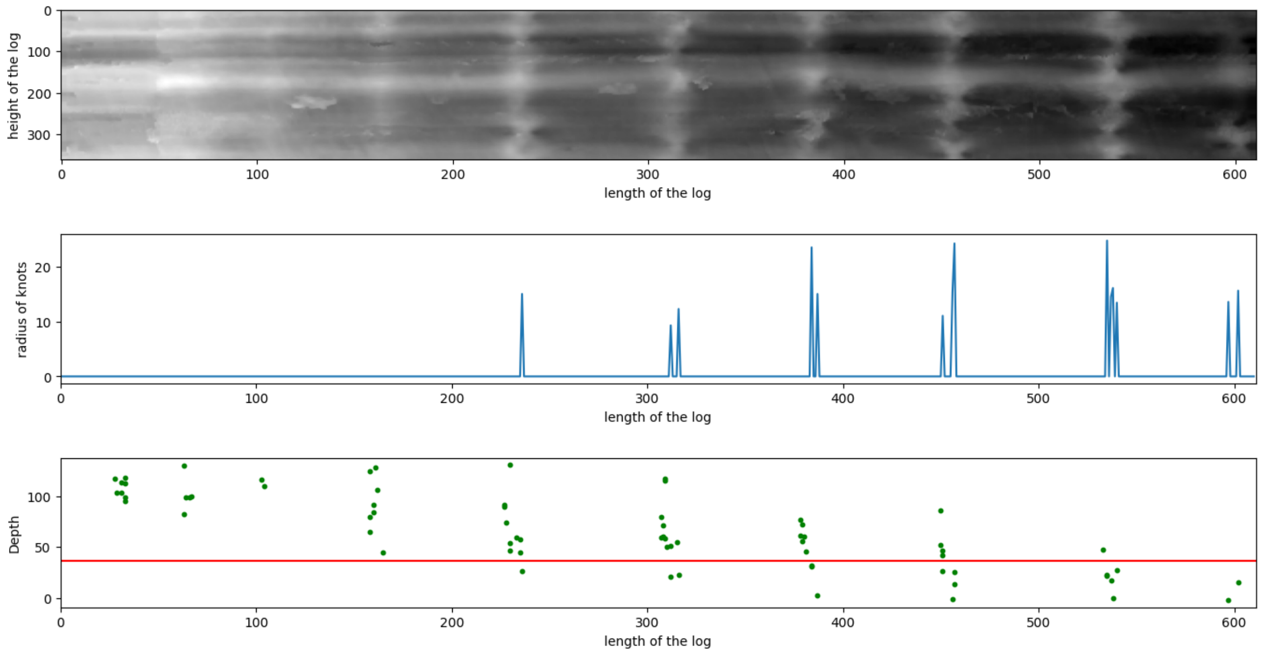


Fig. 4.34: Differentiate between kept and discarded knots

The image at the bottom in figure 4.34 explains the idea of the depth threshold. The **red line** is the depth threshold, which is **37** mm. To illustrate, the knots that will be considered in our calculations are those with **green dots** below the **red line** (depth threshold). While the others that were above the depth threshold will be discarded (not considered during computation). The x-axis in the image in the middle represents the length of the log while the **blue peaks** values on y-axis represents the radius of each knot that were detected in each region satisfying the depth threshold condition.

Table 4.1: Big knots distribution in our dataset

ID of the log	Area in mm^2
407808170	51.33 ,50.69
407808180	51.48
411306690	57.79,63.82,73.67
407812830	97.69,97.34
411306250	97
407808260	102.885
409440260	102.50
407813110	100.65

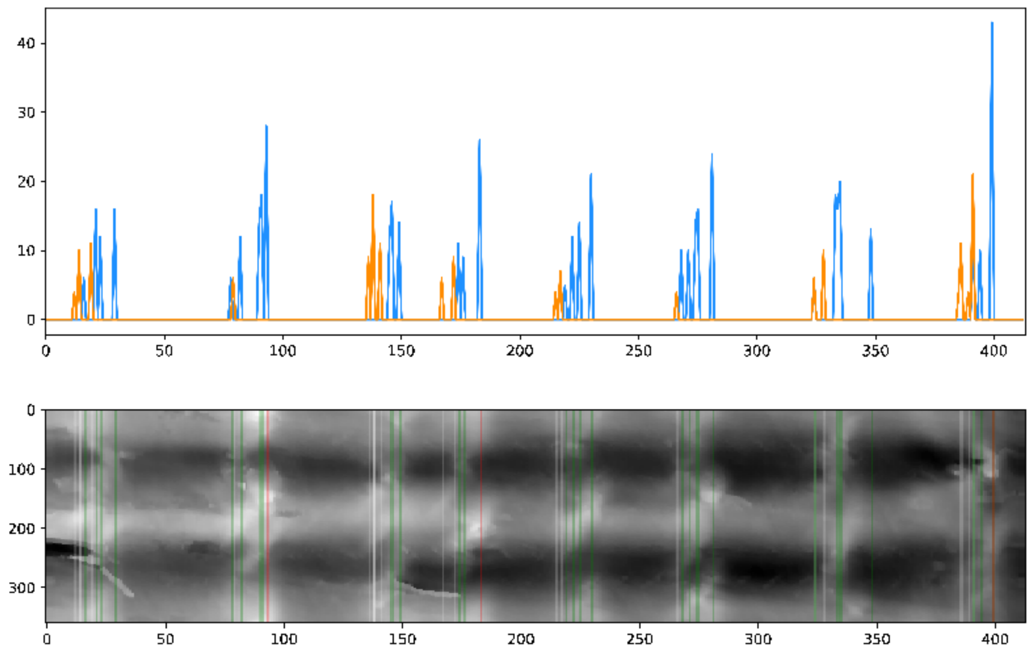


Fig. 4.35: Distinguish between big knots and small knots

As we can see in figure 4.35, the figure at the bottom represents a full wood log with length 4.15 meters classified as class 1 “many knots”. The **red**, **green** and **white** lines represent the presence of a knot in this region. The **red lines** are for the knots that were big with area more than 50 $mm^2$and below the depth threshold. While the **green lines** are knots with area less than 50 $mm^2$and below the depth threshold. On the other hand, the **white lines** are for knots that are above the depth threshold that are already discarded. In conclusion, small and big knots that were below the depth threshold were considered during classification. However, knots that were detected above the depth threshold were not considered during classification, even if they were big knots.

For the image on the top, the x-axis represents the length of the wood log and y-axis represents the radius of knots that were detected in each region. **Blue peaks** are for the knots that were considered during classification and **orange peaks** are for knots that were discarded (above the depth threshold).

4.3.5 Classification Threshold

I. Differentiate between class one and class zero crops

It is a threshold to differentiate between “class one crops (full of knots/bad)” and “class 0 crops (almost no knots/good)”. In other words, the classification threshold was chosen based on the sum of the radii of knots in each wood log crop. To illustrate, the classification threshold that was used is **60**. Therefore, if the sum of the radii of knots in a wood log crop is less than **60**, it will be classified as class 0 crop. Else if, the sum of the radii of knots in a wood log crop is more than **60**, it will be classified as class 1 crop.

1. **First Approach**
   Using only a threshold to differentiate between class 1 and class 0 crops.

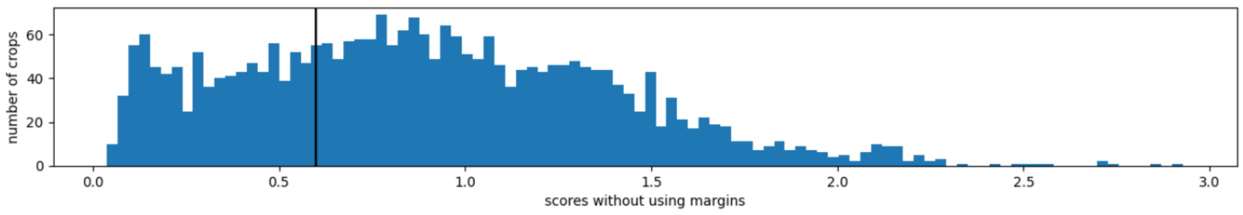


Fig. 4.36: Classification threshold without margins

As we can see figure 4.36 which explains the idea of the classification threshold. All patches with scores before the black line were classified as class 0 “almost no knots “while other crops after the black line were classified as class one “many knots”.

2. **Second Approach**

   Removal of crops closer to the classification threshold using margins.

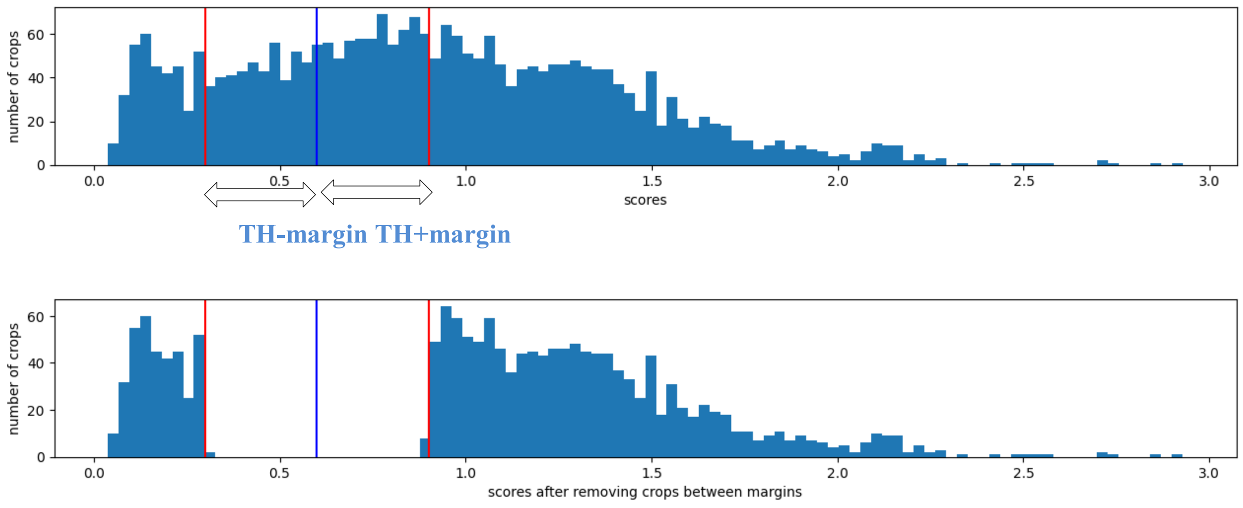


Fig. 4.37: Classification threshold with margins

Figure 4.37 explains the idea of the classification threshold; the **blue line** represents the classification threshold. To illustrate, all the wood log crops that have scores less than the value of the **blue line** were classified as class zero crops “almost no knots”. While the crops whose scores were greater than the value of the **blue line** were classified as class one crops “many knots”. The crops with scores between the **red line** and the **blue line** were discarded from both classes. The x-axis represents the saved scores for each wood log crop and y-axis represents the number of crops that fall in this score category.

Table 4.2: Total number of used and discarded crops first and second approach

First approach: Using only classification threshold to distinguish between class 1 and class 0

Table 4.2-a: First approach

Total number of crops	3171 wood log crops
Number of class 1 crops “many knots”	1871
Number of class 0 crops “no knots “	1300
Discarded	0

Second approach: Using classification threshold and margins

Table 4.2-b: Second approach

Total number of crops	3171 wood log crops
kept crops from both classes	2090
Number of class 1 crops “many knots”	1256
Number of class 0 crops “no knots “	834
Discarded	1081

Note: 1- The third approach will be explained in chapter 6 in section 6.1
2- All values for scores were divided by one hundred in the histogram figure 4.36, 4.37

II. The discarded crops in the second approach

Crops were discarded for three main reasons:

- First reason is that the target of the company at this moment was to find crops with almost no knots and crops with many knots. That was main reason for adding margins to the threshold.

- Second reason is that knots found in those crops were deep (away from the surface of the border).

- Third reason: for training of the CNN at the beginning, the model was trained with all crops and classification threshold without the margins, and for the sake of better classification results, we used the aforementioned margins.

Note: The discarded crops did not have different knot shapes but instead of having many knots (butterflies) like class one or having no knots as class zero, they have few small knots.

Chapter 5

LogEye Dataset

5.1 LogEye machine

Microtec provided the coloured images from LogEye scanner and the corresponding CT images. The number of pairs that were found was 203 pairs. Therefore, we have the CT image that contains all the information that we need and the corresponding image from log eye.

LogEye 301 allows the acquisition of colour images of the sides of wood logs (left, top and right). The coloured images that we worked on were the result of stitching (right, left and top) views for the wood log. Image stitching is the process of combining multiple photographic images with overlapping fields of view to produce a segmented panorama or high-resolution image [2]. In computer vision, there are a plethora of old and commonly used algorithms for aligning images and stitching them into seamless photo mosaics. These generated photo-mosaics have high resolution; as a result, they are used to produce today's digital maps and satellite photos [2].

In this case we will need for each coloured image from LogEye, the corresponding CT image that contains all information such as borders, centroids, pith and all knots information that are included in the 14 descriptors.

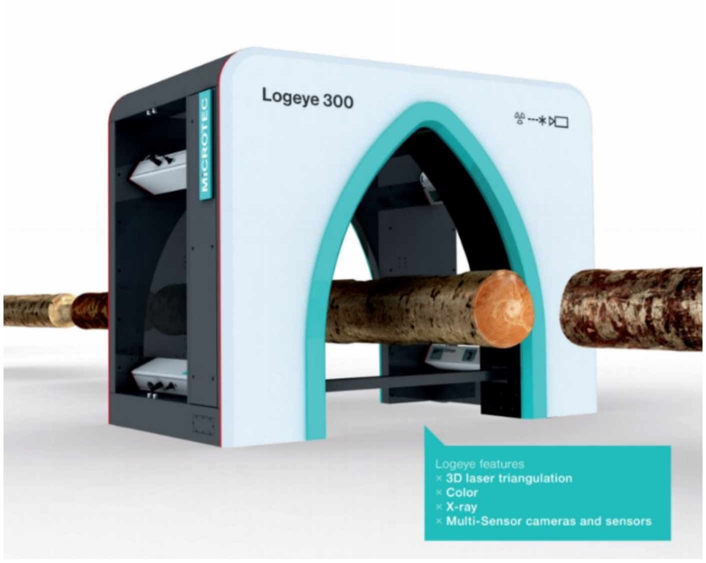


Fig 5.1: LogEye scanner

5.2 Pre-processing steps

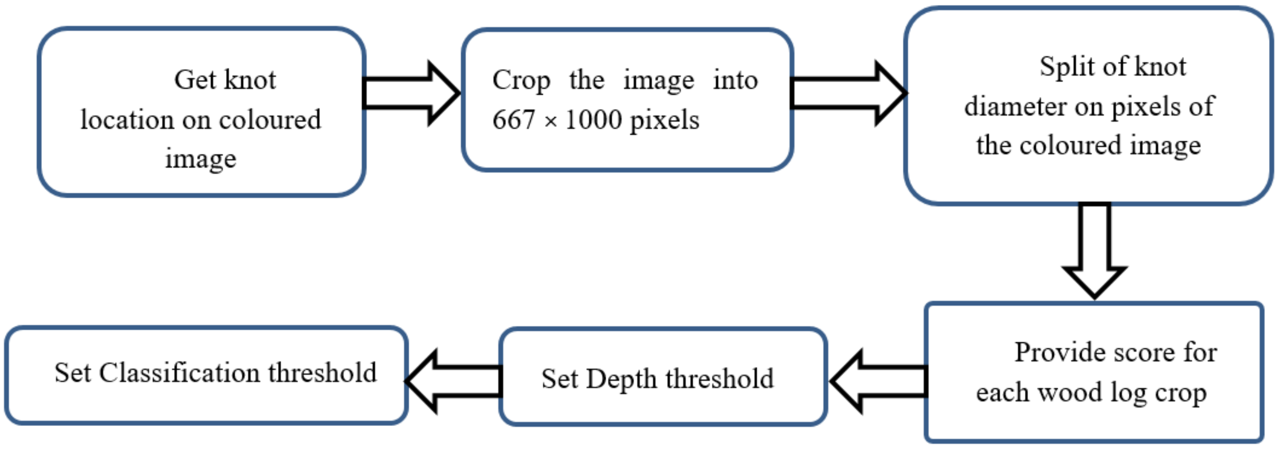

Get knot location on coloured image
Crop the image into 667 × 1000 pixels
Split of knot diameter on pixels of the coloured image
Provide score for each wood log crop
Set Depth threshold
Set Classification threshold

Fig. 5.2: Block diagram for pre-processing steps on LogEye images

5.2.1 Get knot location on coloured image

Compute knot location on coloured image (from LogEye) using z-end (column 6 in the 14 descriptors) from the corresponding CT image (from CT Log scanner).

Coloured image ID: **691523**, **6643** pixels (Length in pixels), Corresponding CT image ID: **596686210**, **428** slices (Length of the log in cm)

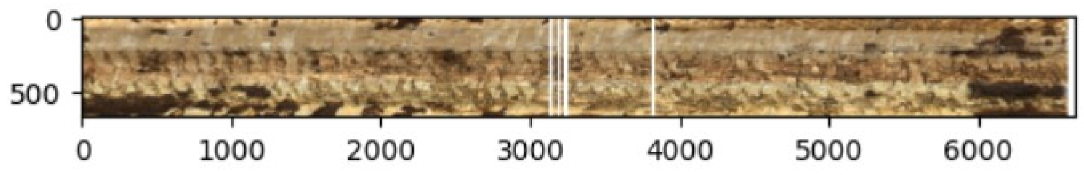


Fig. 5.3: Coloured image from Log Eye with allocation of some knots

Figure 5.3 represents an example of a coloured image from Log eye and the white lines on the image are placement of knots on the wood log. The exact location of knots on coloured images can be detected by using z-end found in column 6 in the 14 descriptors from the corresponding CT image (index of the slice where the knot is located) and the width of the image from log eye (number of pixels). Therefore, using this simple equation, we can allocate the knot on the coloured image of the wood log.

$$\text{Knot location on colored image} = \frac{\text{z−end} * \text{number of pixels}}{\text{Length of the log in cm}} \quad (5.1)$$

Where: z-end = index of the slice where the knot ends (from the corresponding CT image) number of pixels = width of the coloured image which is actually the length of the log (LogEye) Length of the log = number of slices or width of image in cm (CT image).

As we can see in the figure 5.3 which is a coloured image from log eye the first knot (first white line) is located in slice number 31 and around 3133 pixels on the coloured image, the second knot (second white line) in slice 32 and around 3136 pixels, the third knot (third white line) in slice 30 and around 3186 pixels, the fourth knot (fourth white line) in slice 34 and roughly 3231 pixels and the fifth white line is in slice 33 and around 3237 pixels. The last one is in slice 35 and 3816 pixels.

5.2.2 Crop RGB images

Coloured image ID: **691523**

Splitting images into crops of the same size is a crucial part while working with Machine Learning models. To explain, some ML models could not process high-resolution images that is why we need to divide them into smaller parts. While generating crops the first and last 100 pixels were neglected (removed) in order to avoid any unreliable parts of the coloured images.

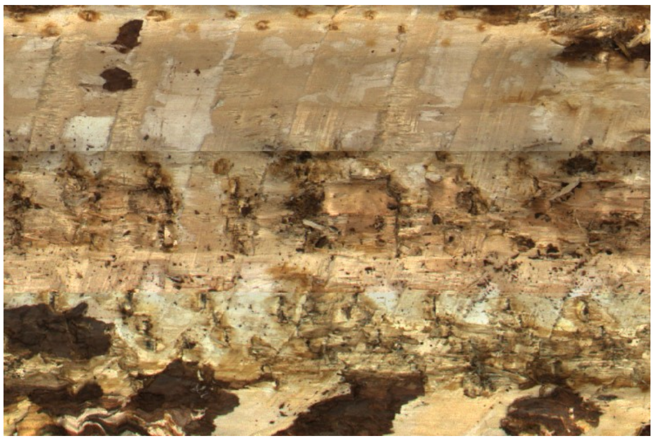


Fig. 5.4: Crop number 1 from LogEye image with ID: **691523**

Figure 5.4 represents one of the crops resulted from cropping the wood log with dimensions 1000×667 pixels for width and height respectively.

Note: All Crops have the same width and height 1000 × 667pixels.

5.2.3 Knot diameter split on pixels of coloured images

In Figure 5.5 let us assume that the image in **blue** is the CT image that contains the knots profile (14 descriptors). The image in **red** represents the coloured image from LogEye, using equation number 5.2 we can easily get the location of knot in pixels on the coloured image (New z2). As we know the radius of the knot from column 8 in the 14 descriptors from the CT image, it can be easily computed on the coloured image from LogEye using the following equation:

$$\text{Knot location on colored image} = \frac{\text{z−end} \times \text{number of pixels}}{\text{Length of the log in cm}} \quad (5.2)$$

$$\text{Knot radius in RGB pixels} = \frac{\text{knot radius} \times \text{length of log in pixels}}{\text{number of slices}} \quad (5.3)$$

In order to give knot scores, it is better to split the weight of the knot (diameter=two×radius) as shown in the figure 5.6 on pixels of the coloured image. In other words, instead of projecting knot weight (radius) only on one pixel value which is the knot location on coloured image (New z2), weights of the knots (diameter in this case) will be projected in the region from first split passing through New z2 to second split where first and second split were calculated as follow:

$$\text{First split} = \text{New z2 - Knot radius in RGB pixels} \quad (5.4)$$

$$\text{Second split} = \text{New z2} + \text{Knot radius in RGB pixels}+1 \quad (5.5)$$

Where new z2 is knot location on coloured image (5.2) and Knot radius in RGB pixels (5.3)

Using this method, it would be easier to provide score for each log crop as: the score for each pixel containing a knot is one computed with the following equation:

$$\text{knot weight in pixels} = \frac{2*r+1}{2*r+1} = 1 \quad (5.6)$$

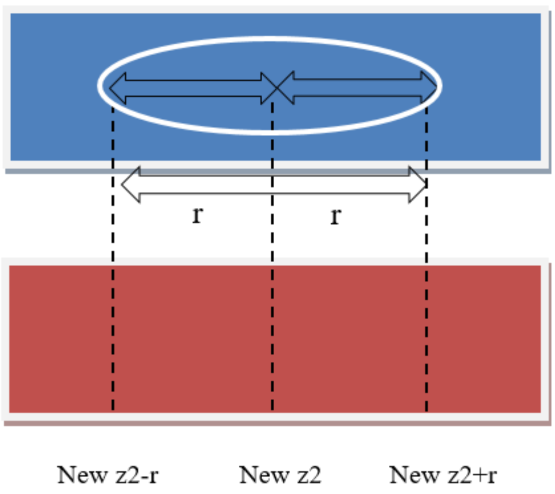


Fig. 5.5: Illustrates the split of knot diameter on pixels of coloured image

5.2.4 Save the score for each wood log crop

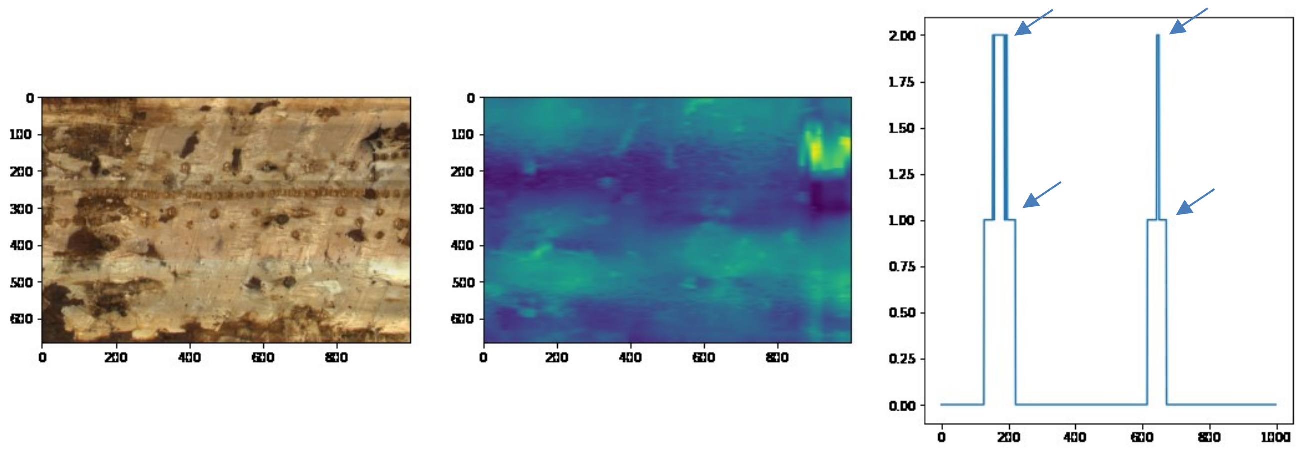


Fig. 5.6: Crop from coloured and the score for this crop

Scores were given to each wood log crop based on the sum of the radius of the knots where each pixel containing a knot has a score of one.

As we can see in figure 5.6, each **blue peak** represents the presence of a knot in this region. Here there are four peaks; the width of each peak represents the radius of the knot in this region.

5.2.5 Depth threshold

The main idea behind this step was to decide which knots to keep or to discard. Therefore, knots that were above the depth threshold were discarded, while other knots were considered.

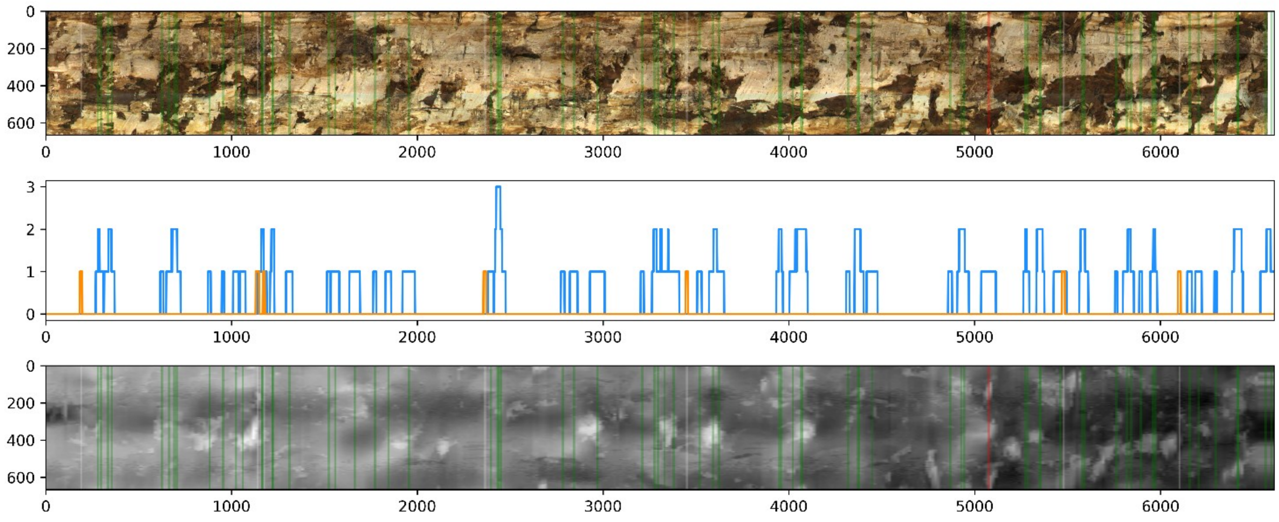


Fig. 5.7: Visualization of knots on coloured image and shape

As we can see in the figure 5.7 that the **green lines** represent the knots that are not more than 7 mm from the surface (close to the surface), while the **white lines** represent knots that are away from the surface. The **blue peaks** in the second image represent the knots that are close to the surface while the ones with **orange colour** the discarded ones and the width of each peak (**blue** and **orange**) represents the radius of the knot. The y-axis in the second image represents the number of knots that were found in a specific region. The **red line** in the last image (wood log shape in gray scale) indicates the presence of a big knot with area greater than 50 mm^2.

5.2.6 Classification threshold

This classification threshold was set to discriminate between class 1"Many knots" and class 0 crops "almost no knots ". To demonstrate, if a crop has a score (sum of radii of knots in this crop) less than the classification threshold, it would be classified as class 0; else it would be classified as class 1.

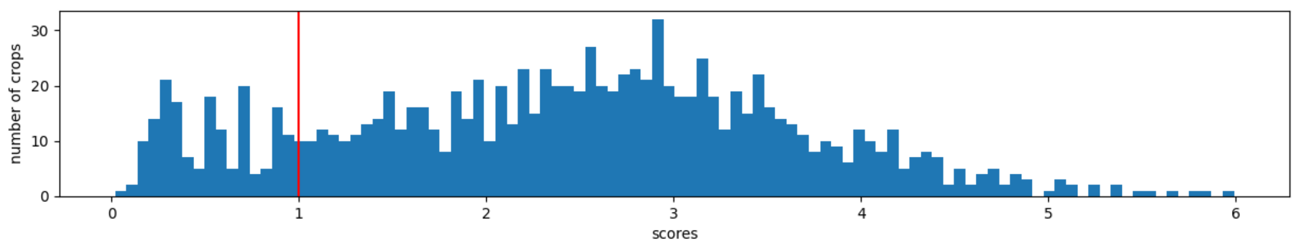


Fig. 5.8: Classification threshold in LogEye Dataset

As we can see in figure 5.8, where x-axis represents the score of wood log crops, (LogEye coloured images) and the y-axis represents (**blue peaks**) the number of crops, which fall in this score category. The **red line** is the classification threshold where all wood log crops that have scores less than (before) this threshold were considered class 0 crops “almost no knots”, while crops with scores greater than (after) the **red line** were considered class 1 crops.

5.2.7 Problems with this dataset

We faced several issues with this dataset:

1. Limited number of images from LogEye (203 images).
2. By examining the coloured wood log images manually, we found out that the knots were not visible on even 10 or 20% of the full dataset. Therefore, for a human it was not easy to detect a knot by eyes from the coloured image, consequently it would be difficult for a machine to do so.

All these reasons lead to stop working with LogEye dataset.

Note: All the work done after that (Training the CNN) was using CT Dataset, which was explained in chapter 4.

Chapter 6

Training the CNN

6.1 Dataset Manual Validation

I. Graphical User Interface for Manual Validation (Third approach)

Crops for each class were manually validated using a GUI interface to make sure that class 1 crops have already many visible knots in the images and class 0 crops roughly do not have any butterflies shape, which represent the presence of knots. During the manual validation for the dataset, some crops from both classes were discarded for two main reasons. First reason is that the crop has many knots but classified as class 0 and vice versa. Second reason is that, the image was not clear due to blurring, very low contrast, etc.

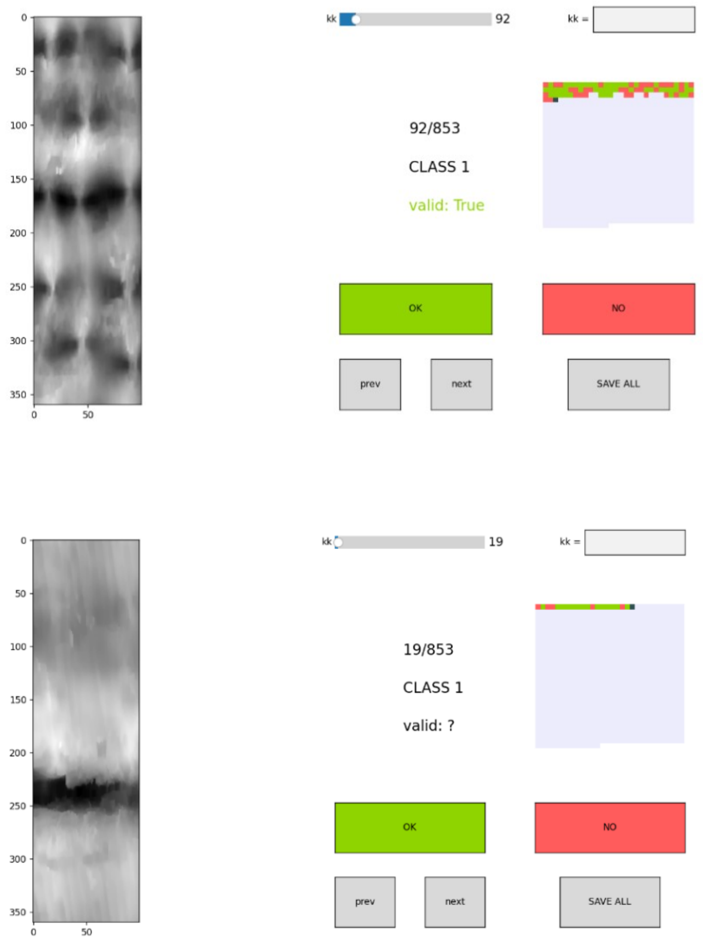


Fig. 6.1: GUI interface for Manual validation of the dataset

Figure 6.1 represents that GUI interface that was created in order to check whether the label for each crop (class 0 or class 1) is corresponding to the number of knots that were recognized while looking at the image.

6.2 Data Augmentation

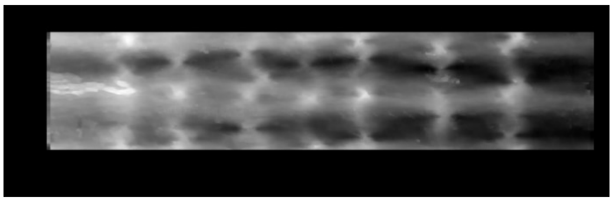
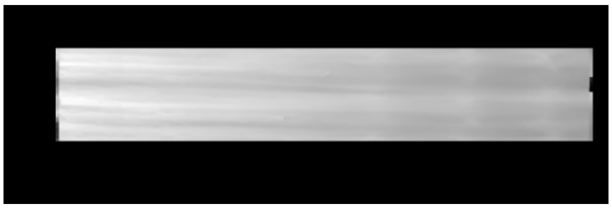
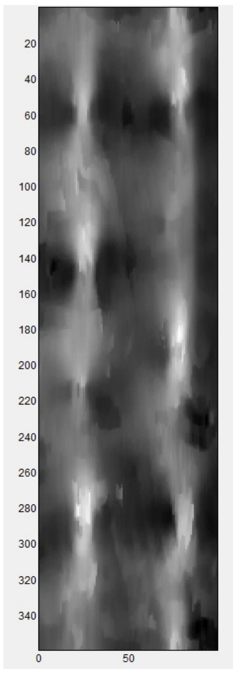
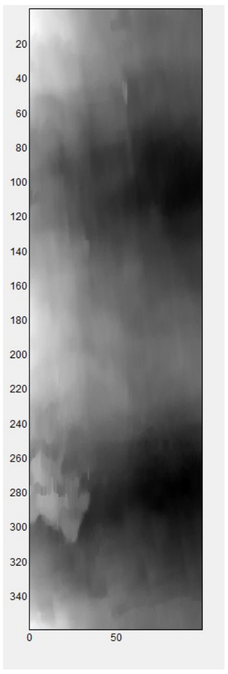
To solve the problem of limited training data, we applied the artificial expansion method to enlarge the dataset. To explain, if the number of images in the dataset is not sufficient, the ML model will not be able to generalize on new data that it has never seen before and overfitting will occur. In this project, data augmentation was applied by cropping each wood log in our dataset into several crops and then random rotation and random zoom using keras pre-processing layers. Thus, improving the generalization ability of the model and tackling the problem of overfitting.

I. Cropping

962 wood logs $\Longrightarrow$ **3171 Patches in gray scale**

Number of patches for each wood log is depending on its length. Therefore, in our dataset, number of crops is different for each wood log. For example, wood log with length 620 cm will have six crops each with length 100. While for wood logs with length 220 cm, there are only two patches.

Table 6.1: Original dataset after cropping

Original dataset	After data augmentation	
962 images from CT Log	3171 Patches	
Examples for full wood logs from the two classes:	Class 1 crops	Class 0 crops
	1871 images	1300 images
 Class 1 image “Many Knots” 360 × 415 cm  Class 0 image “almost no knots” 360 × 415 cm	 360cm×100 cm	 360 cm×100 cm

II. RandomRotation and RandomZoom

Keras pre-processing layers. To begin with, RandomRotation is a layer, which is responsible for rotating images during training. This layer will apply random rotations to each image, filling empty space according to fill mode. By default, random rotations are only applied during training [71]. While for RandomZoom layer, it randomly zooms images during training.

III. Number of patches for each approach as a result of data augmentation

Table 6.2: Total number of kept and discarded crops for each approach

	First approach	Second approach	Third approach
Total number of crops	3171	3171	2090
Used crops	3171	2090	1156
Class 1 patches	1871	1256	625
Class 0 patches	1300	835	531
Discarded crops	0	1080	934

Note: this manual validation (**Third approach**) was done only for crops resulted from the second approach

6.3 Overall Model Architecture

I. Training, validation and test sets

Table 6.3: Training, Validation and test sets

Training Set	Validation Set	Test Set
70%	20 %	10%

Our proposed model was trained on the crops generated from each approach (first, second and third), using the same split that was mentioned in table 6.3.

Note: the exact numbers used for training, test and validation sets in each approach will be mentioned later in table 6.7 in section 6.4.2

II. The CNN Model

The model that was used to detect knots and classify wood logs is a typical deep convolutional neural network as we can see in figure 6.2-a, b, c. The architecture of the model consists of an input layer, followed by RandomRotation and RandomZoom, four convolutional layers, and one global average-pooling layer, three fully connected layers, a softmax layer and an output layer. The input was 360×100×1 gray scale images. Each of the convolutional layers had kernels that were 3×3 and strides of two and followed by a batch normalization layer, which was followed by a Relu layer. Batch normalization layer was used in our case in order to accelerate the speed of the training and provide stability to the model by normalization of the layers' inputs through re-centring and rescaling. In addition to that, the fourth convolutional block was followed by global average pooling which helped in reducing feature map size, number of training parameters, memory usage and faster computation. The first fully connected layer had 1024 neurons, the second had 512 neurons and the third had 256 neurons. The fully connected layers were followed by batch normalization and Relu activation function. Finally, the final layer was two-way output for class 1 and class 0 crops.

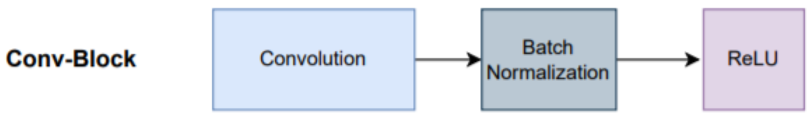


Fig. 6.2 –a : Convolution Block

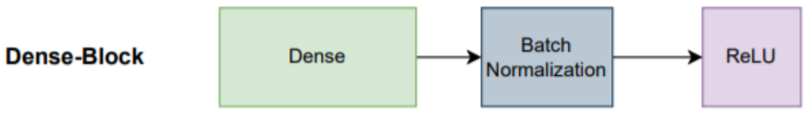


Fig. 6.2 –b : Dense Block

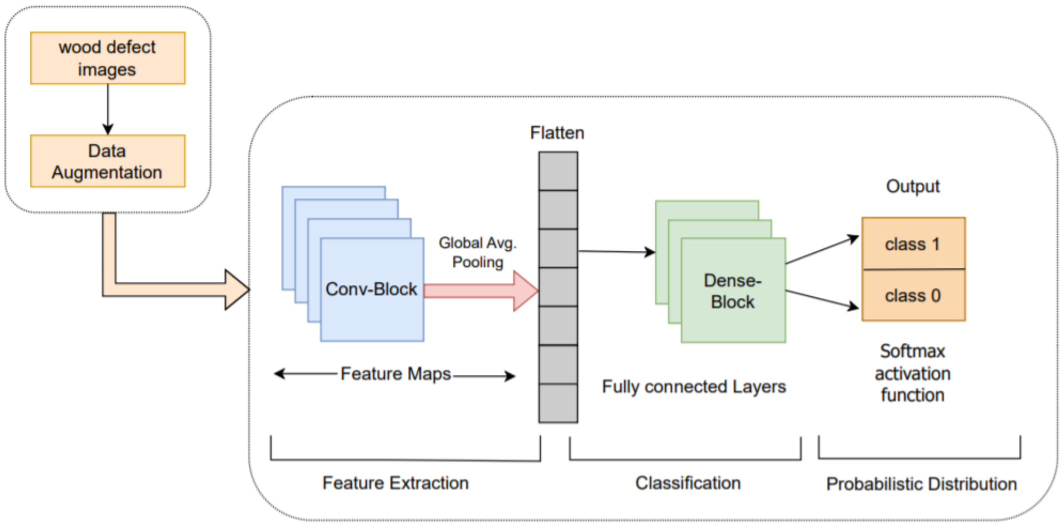


Fig. 6.2 – C: Network Scheme

As shown in figure 6.2-a ,our model has 4 convolutional blocks where each Conv-Block has a convolutional layer, batch normalization layer and ReLU activation layer. In addition to, three dense blocks where each Dense-Block figure 6.2-b has a dense layer, batch normalization layer and a ReLU layer. The convolutional layers along with pooling layers are responsible for extraction of features and generating feature maps whereas fully connected layers are responsible for classification.

Table 6.4: The DCNN Model

Layer	Output	Kernel	Stride	Parameters
INPUT	(None,360,100,1)	-	-	0
Random-Rotation	(None,360,100,1)	-	-	0

Random-Zoom	(None,360,100,1)	-	-	0
Convolutional	(None,180,50,8)	3*3	2	80
Batch Normalization	(None,180,50,8)	-	-	32
Convolutional	(None,90,25,16)	3*3	2	1168
Batch Normalization	(None,90,25,16)	-	-	64
Convolutional	(None,45,13,32)	3*3	2	4640
Batch Normalization	(None,45,13,32)	-	-	128
Convolutional	(None,23,7,64)	3*3	2	18496
Batch Normalization	(None,23,7,64)	-	-	256
GlobalAveragePooling	(None , 64)	3*3	2	0
Flatten	(None,64)	-	-	0
Dense	(None,1024)	-	-	665660
Batch Normalization	(None,1024)	-	-	4096
Dense	(None,512)	-	-	524800
Batch Normalization	(None,512)	-	-	2048
Dense	(None,256)	-	-	131328
Batch Normalization	(None,256)	-	-	1024
Output	(None,2)	-	-	514

Total Parameters	755,234
Trainable parameters	751,410
Non-trainable parameters	3824

III. Deep CNN for feature extraction and classification tasks

In many Computer Vision applications neural networks and in particular their deep counterpart has become the most popular choice thanks to the impressive improvements achieved in recent years. Another strength of these models is their great flexibility that allows them to be used to solve different problems, since their functionality is often independent of the nature of the subjects depicted in the images.

In addition to that, the impressive gain in performance obtained using deep neural networks (DNN) for various tasks encouraged us to apply DNN for our image classification task. We have used a variant of DNN for feature extraction and image classification Called DCNN. Neural networks can be used for classification as well as for feature extraction. Our whole work can be better seen as two complementary tasks. In the first task, our deep network was used for feature extraction. In the second task, the fully connected layers were used for the classification of the extracted features along with softmax activation that was used to provide the probability distribution over the two classes. Various configurations of DCNN were used for our experimental studies. Among different architectures that we have considered, the architecture with 4 levels of convolutional layers and Average pooling, followed by 3 fully connected layers was used for feature extraction and classification.

In 2020, Liu et al. [22] used DCNN for feature extraction and classification of three type of defects on wood boards. The model showed significant results in the field of recognition of wood defects images that were collected using laser range scanners. That was one of the reasons why we decided to use a deep network in our project while dealing with CT images of wood logs.

Moreover, comparing this model with previous work done in detection and classification of defects in wood as in [22, 33], we found out that our proposed model provided effective results in the world of detection and classification of knots in wood logs. Knowing that the proposed model in [22] was for detecting defects in wood boards, which is in fact easier than dealing with wood logs as in this project.

We tried avoiding to oversize the neural network. To explain, one must keep in mind that this system also has applications in scenarios where the time taken by the algorithm is a critical factor, while usage in real time to send the logs in the right direction. It is therefore necessary for the system to detect knots and classify wood logs very quickly, so the size of the neural network may be an important factor to consider.

6.3.1 Activation function

I. Rectified Linear Unit

ReLU was selected as an activation function as it can tackle the vanishing gradient problem, unlike sigmoid and hyperbolic tangent activation functions and characterized by the following equation [67].

$$F(x) = \max(0, x) \qquad (6.1)$$

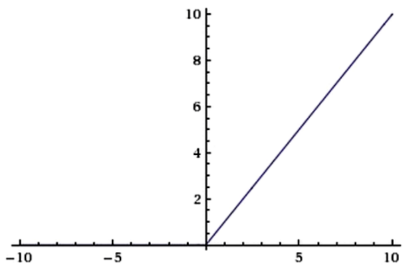


Fig. 6.3: ReLU activation function graph

II. Softmax

This function is often used as the output of a classifier to represent the probability distribution over different classes and characterized by the following equation:

$$\mathrm{softmax}(z)_i = \frac{\exp(z_i)}{\sum_j \exp(z_i)} \quad (6.2)$$

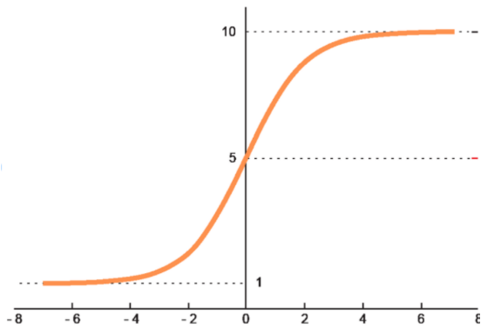


Fig. 6.4: Softmax activation function

6.4 Training

6.4.1 Details of training

Table 6.5: Parameters that are related to training

Related Parameter	Value	Meaning
Batch Size	8	Number of samples that will be passed through to the network at a time or number of images per training
Learning Rate	0.0001	Initial learning rate
Weight Decay	0.0001	Regularization technique which is used mainly to alleviate model complexity and reduce the risk of overfitting
Learning steps/epochs	300	Number of iterations

Reducing overfitting

Overfitting is one of the most serious problems that affects the CNN performance. In order to tackle this problem cropping, random rotation and random zoom were applied as data augmentation techniques and weight decay, batch normalization layers as regularization techniques were used.

6.4.2 Training process

I. What does it mean for a model to learn?

The model learns by passing the same input over and over again, updating weights in order to minimize loss. To demonstrate, the process of updating weights leads to get closer from the optimal value and hence decreasing loss.

$$\text{Gradient} = \frac{d(loss)}{d(weigth)} * learning\ rate \quad (6.3)$$

In fact, most practitioners use AdamW optimization, which is a stochastic gradient descent technique that is based on adaptive estimation of first-order and second-order moments with an added method to decay weights by (Loshchilov, Hutter et al., 2019). According to a research study conducted by a group of researchers in 2014, Adam method is efficient from computation point of view, less requirements in terms of memory, suitable especially for tasks with large number of parameters and big datasets (Kingma et al., 2014).

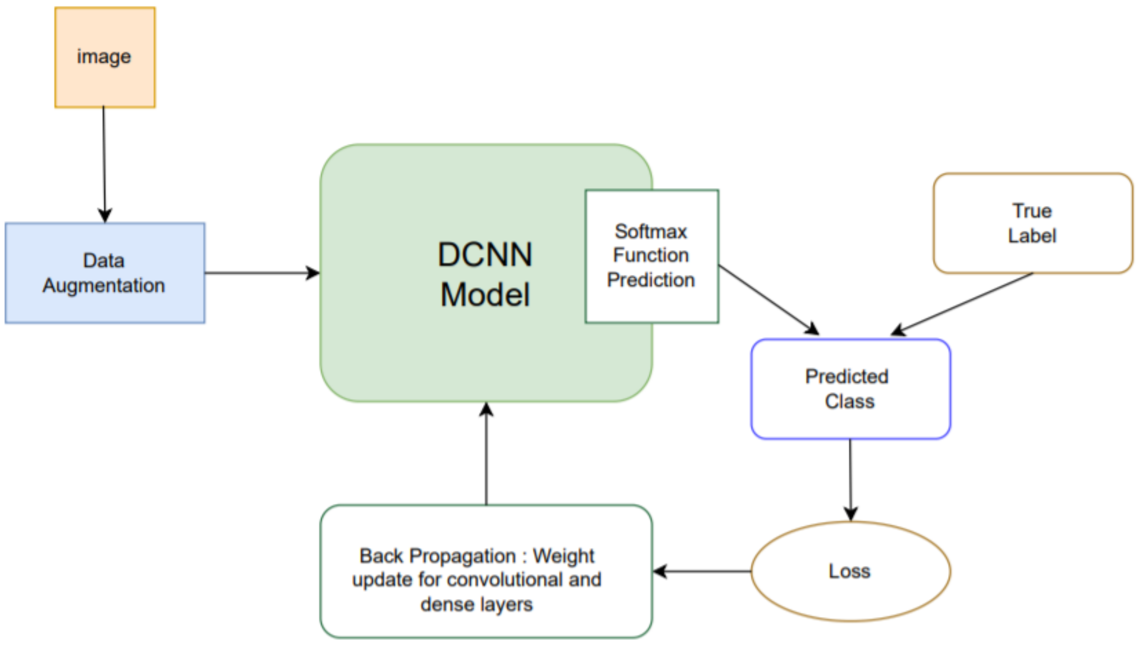


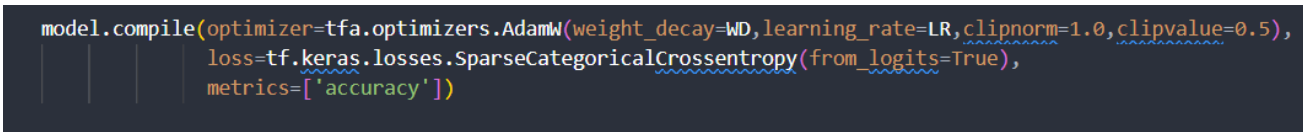
image
Data Augmentation
DCNN Model
Softmax Function Prediction
True Label
Predicted Class
Loss
Back Propagation : Weight update for convolutional and dense layers

Fig. 6.5: Weight update for convolutional and dense layers

In this project AdamW optimizer was used with batch size 8. The learning rate was initiated at 0.0001. The model was trained up to 300 epochs. In addition to this, weight decay was used with value 0.0001 as mentioned in table 6.5, which includes the parameters that are related to training.

Table 6.6: AdamW optimizer parameters

Argument	Value	Meaning
clipnorm	1	The gradient of each weight is individually clipped so that its norm is no higher than this value [68].
clipvalue	0.5	The gradient of each weight is clipped to be no higher than this value [68].



```
model.compile(optimizer=tfa.optimizers.AdamW(weight_decay=WD,learning_rate=LR,clipnorm=1.0,clipvalue=0.5),
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

Fig. 6.6: Compile the model

Figure 6.6 explains the activity that takes place in the middle stage immediately after having all the statements for the model and directly before the start of the training. The main target of this step is to check any errors that are available in the format, decide on loss function, the metric needed for model evaluation.

II. Training, test and validation sets

Our dataset in the model is broken to training test and validation set as we can see in table 6.7:

Table 6.7: Number of crops in training, test and validation sets

		First Approach	Second Approach	Third Approach
Training set	Labelled set Set of data used for training our model. At each epoch, our model continues to know more features about this data.	2220	1463	809
Validation set	Labelled set Set of data, which is used to validate our model and helps to adjust hyperparameters and detect overfitting.	634	418	209
Test set	Unlabelled set This set is used to test our model after training. Moreover, this set is different from both training and validation sets.	317	209	116

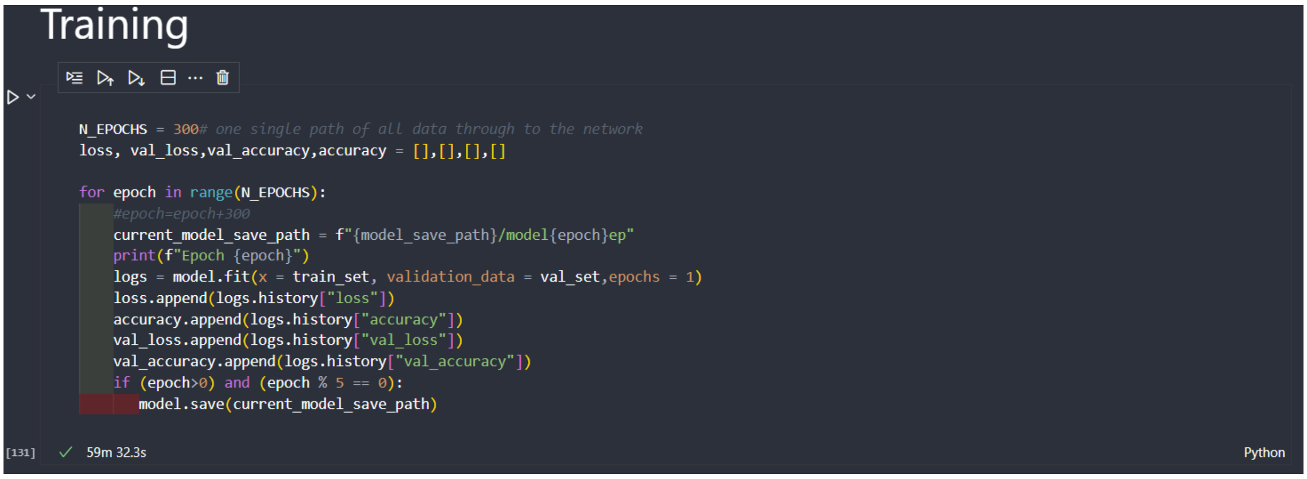
III. Train the model

Training CNN models requires the following steps:

1-Feed the training data to the model. In our project, the training data is in the train_set and the labels for each wood log crop (class one “many knots”, class zero “almost no knots”)

2- Using training data with their labels, the model learns to label images.

To start the training, we called model. Fit, which “fits” the model to the training data as we can see in figure 6.7.


Training

```python
N_EPOCHS = 300# one single path of all data through to the network
loss, val_loss,val_accuracy,accuracy = [],[],[],[]

for epoch in range(N_EPOCHS):
    #epoch=epoch+300
    current_model_save_path = f"{model_save_path}/model{epoch}ep"
    print(f"Epoch {epoch}")
    logs = model.fit(x = train_set, validation_data = val_set,epochs = 1)
    loss.append(logs.history["loss"])
    accuracy.append(logs.history["accuracy"])
    val_loss.append(logs.history["val_loss"])
    val_accuracy.append(logs.history["val_accuracy"])
    if (epoch>0) and (epoch % 5 == 0):
        model.save(current_model_save_path)
```

```
[131]  ✓  59m 32.3s                                                    Python
```

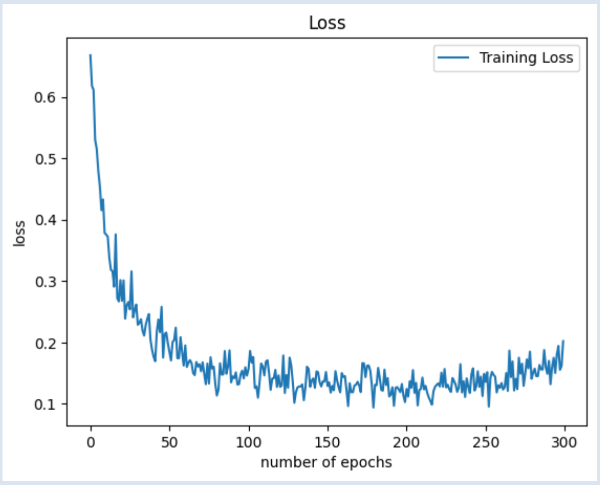
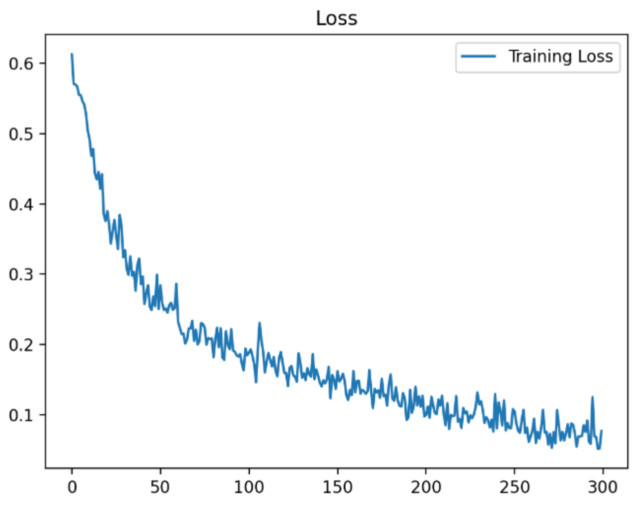
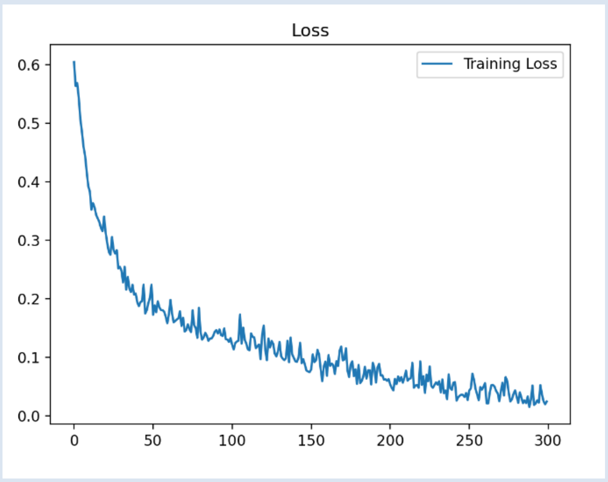
Fig. 6.7: Train the model

6.4.3 Loss function

After identifying the hyperparameters, the model was trained on the whole training set for 300 epochs for each approach. The target while working with deep learning tasks is to minimize the cost function. In other words, the lower the loss is the better the model is performing.

In this task, we used sparse categorical loss function as we can see in figure 6.5

Table 6.8: Training loss for each approach

Approach	**Loss**
First Approach	
Second Approach	
Third Approach	

Cross entropy was used as the loss function, which is characterized by the following equation:

$$H(p, q) = -\sum_{x} P(x) log q(x) \qquad (6.4)$$

Where p is the probability and q is the probability of the predicted value.

Loss function is located in the output layer to compute the difference between the actual (real) output and the predicted output, which will be optimized through CNN learning process. There are two main parameters that are used by loss function to get the error value. The first one is the prediction and the second parameter is the real label [47] as explained with figure 6.5.

As we can see in table 6.8 which shows the loss curve on the training set over 300 epochs (x-axis), that the loss is decreasing (y-axis) especially in the second and third approach which means that the model is learning gradually from the training set.

Note: For the first approach, we used the first 250 epochs where the training loss curve is decreasing.

Chapter 7
Results

7.1 Experimental setup

The whole developing and testing phases of the proposed approaches, as well as time measurements (when reported) were performed on the same computer, running Windows 10 on an Intel Core i7-7820HQ, 3 GHz processor and equipped with an NVIDIA Quadro M1200 GPU. All the parts concerning neural networks were developed in python 3.8.16, by using the Keras 2.8.0 library on a TensorFlow 2.8.0 backend.

7.2 Features extracted by CNN

Grad-CAM technique was used to visualize what CNN was actually looking at. In other words, Gradient weighted class Activation Map produces a heat map that highlights the significant regions of an image by using the gradients of the target of the last convolutional layer.

The class activation map for a specific class is the activation map of the ReLU layer that follows the final convolutional layer, weighted by how much each activation contributes to the final score of that class. Those weights are in fact equal to the weights of the final fully connected layer of the network for that class.

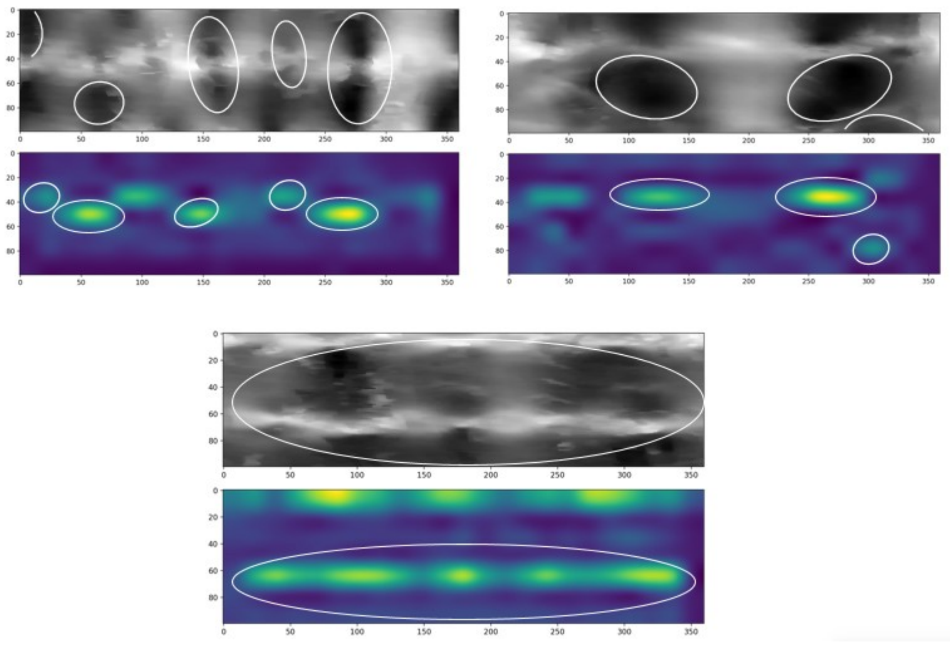


Fig. 7.1: Grad-CAM class activation output

As we can see in figure 7.1, by using the feature map of the final convolutional layer in our model, the places where knots' shape were located as a ground truth on the three different pine no bark gray scale images, were identified. Consequently, affecting positively the classification made by the fully connected layers. Moreover, the places where the gradient is large represent the places where the final score for each class depends on.

7.3 Model Evaluation

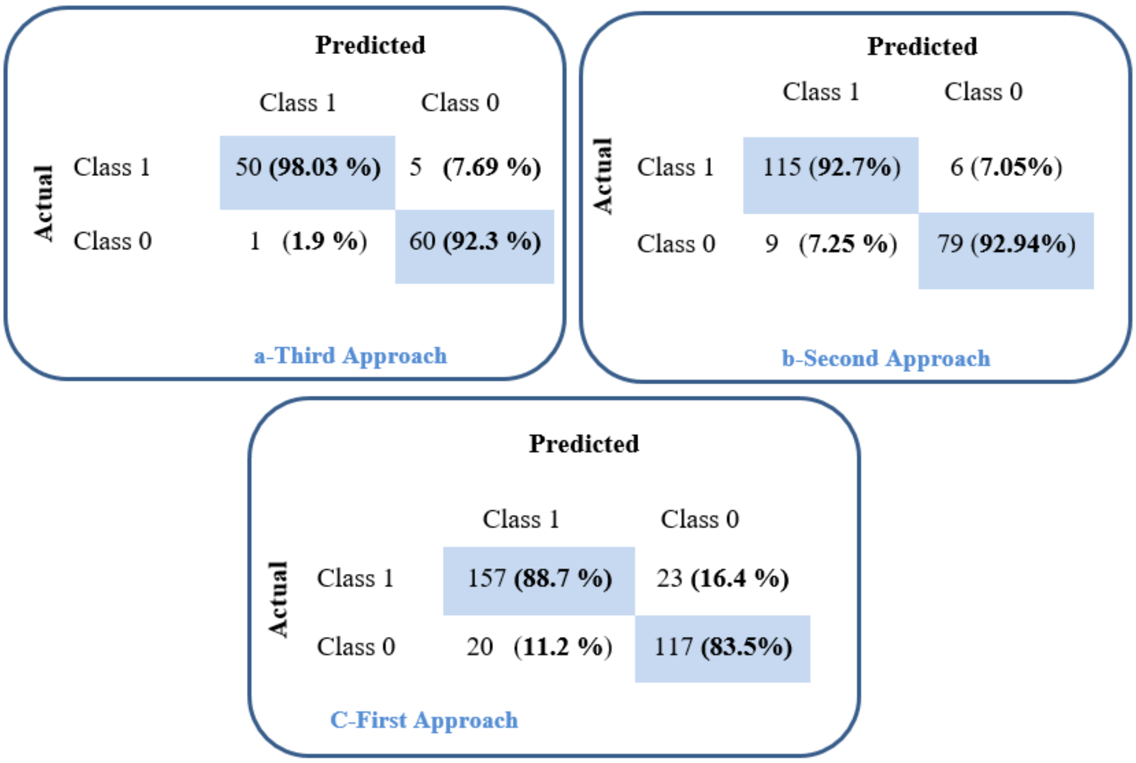
7.3.1 Accuracy

Table 7.1: Accuracy values for the applied approaches

Accuracy	**Training**	**Validation**	**Test Set**
First Approach	0.88	0.87	0.857
Second Approach	0.94	0.93	0.925
Third Approach	0.9634	0.9541	0.9460

Table 7.1 presents the accuracy values scored by each approach over training, validation and test sets. The approach with highest accuracy among all was the third approach with overall accuracy 96.34%, 95.41%, 94.6 % on training, validation and test sets respectively.

Table 7.2: Confusion Matrix for accuracy over the test set using the applied approaches



		Predicted	
		Class 1	Class 0
Actual	Class 1	50 **(98.03 %)**	5 **(7.69 %)**
	Class 0	1 **(1.9 %)**	60 **(92.3 %)**

a-Third Approach

		Predicted	
		Class 1	Class 0
Actual	Class 1	115 **(92.7%)**	6 **(7.05%)**
	Class 0	9 **(7.25 %)**	79 **(92.94%)**

b-Second Approach

		Predicted	
		Class 1	Class 0
Actual	Class 1	157 **(88.7 %)**	23 **(16.4 %)**
	Class 0	20 **(11.2 %)**	117 **(83.5%)**

C-First Approach

A closer inspection to these results reveals that the decrease in performance of the first approach with respect to the second and third approaches is due to the lower accuracy brought due to several reasons. First possibility is due to the presence of some pieces of bark, which in some instances are closer to the shape of some knots. That was main reason why some crops from class zero were misclassified as class one. Second possibility that may cause confusion to the neural network was the blurring that was clear in some images, which made it difficult for even a human brain to classify properly. Third reason could be the presence of only one butterfly shape, which represents the presence of a knot and the neural network classifies it as class zero because all the region around this knot seems to be clean, but as a ground truth this crop belongs to class 1 due to large radius of the detected knot even it was only one.

As we can see in table 7.2-a, b, c, which illustrates the predictions made by the model on the test set for each approach. The test set in the first approach contains 317 patches 274 crops were correctly classified from both classes. While for the second approach, 194 were categorized properly out of 209 patches in the test set. Finally, for the last approach the correctly classified samples were 110 over the 116 crops in this test set.

The significant improvement in the performance of the model in the second and the third approach is due to several reasons. For the second approach, we applied the classification threshold along with margins. To explain, the crops between the classification threshold and the margins were discarded. This may be at some instances made the difference between crops from both classes easier for the model. In addition to that, in the final approach (third) manual validation using a GUI interface was applied to the patches that were generated as the result of applying the second approach. To demonstrate, in this approach some images with low contrast, blurring issues, which may lead to misclassification, were removed. For example, images that contain pieces of bark, which were classified wrongly as class 1, but in fact, they were without knots.

7.3.2 Examples for the detected errors

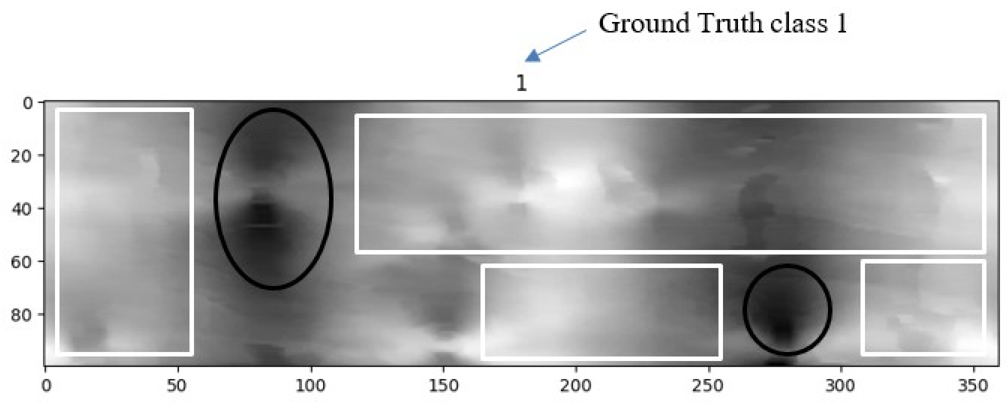


Fig. 7.2: Class 1 image misclassified as class 0

As we can see figure 7.2 is an example of the misclassified samples (ground truth class 1, classified as class 0). In this case, it seems that it is not easy to classify this wood log by eye as full of knots “class 1“ as maybe we can see only one or maximum two butterflies shape which represent knot's shape and all the area around seems to be clean "without knots". Regions that are inside the **white rectangle** represent almost clean area "without knots "while regions inside the **black circles** are the suspected areas "knot shape (butterfly)".

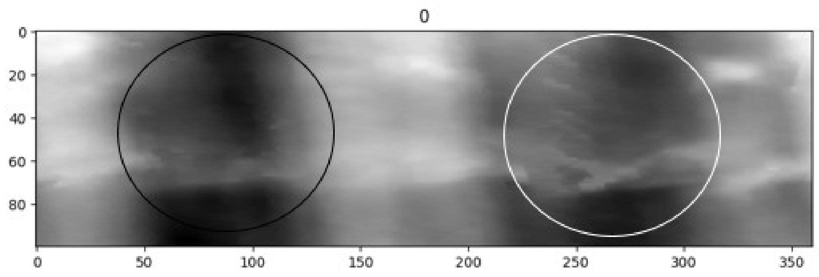


Fig. **7.3:** Class 0 image misclassified as class 1

Figure 7.3 is an instance of class 0 image (ground truth) which was classified as class 1 due to two main reasons. The first is due to the presence of a piece of bark (**black circle**) and the other is due to blurring (**white circle**).

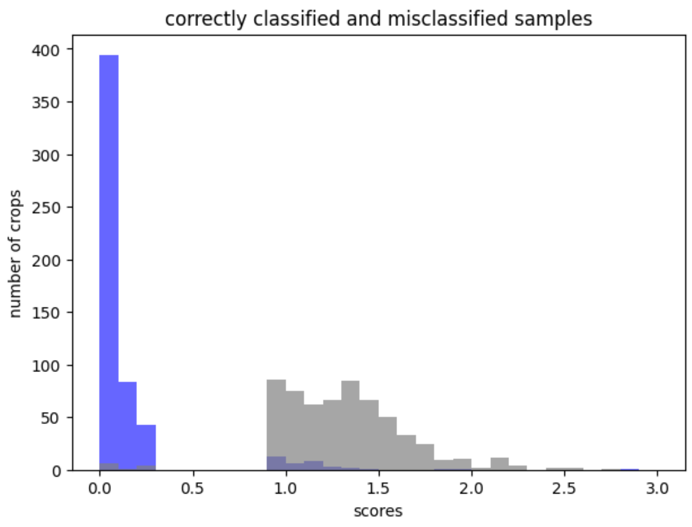


Fig. **7.4:** Correctly and misclassified samples in the test set of the third approach

Figure 7.4 explains the difference between the correctly classified and the misclassified samples. The **blue colour** represents the correctly classified samples from class 0 and the **gray colour** is for the correctly classified samples from class 1. The **blue colour** in the **gray region** is for the misclassified samples as class 0 where they are class 1 as a ground truth and vice versa.

7.3.3 Precision and Recall

The model was evaluated using other metrics such as precision and recall in addition to the aforementioned accuracy:

Table 7.3: Precision and Recall

	First Approach		Second Approach		Third Approach	
Precision	Class 1	Class 0	Class 1	Class 0	Class 1	Class 0
	0.80	0.83	0.905	0.897	0.909	0.956
	Overall: 0.80		Overall: 0.891		Overall: 0.9218	
Recall	Class 1	Class 0	Class 1	Class 0	Class 1	Class 0
	87.2	85.4	0.950	0.904	0.9424	0.9792
	Overall: 0.863		Overall: 0.928		Overall: 0.959	

7.4 Modifications to the model

Some modifications were added to the model for the purpose of improving its performance and reducing the risk of overfitting.

We added zero padding after each convolutional block in order to keep features that exist at the edges of the matrix and control the size of the output feature map. Adding padding to an image processed by a CNN allows for a more accurate analysis of images. Moreover, L2 regularization was added to the model as a regularization technique.

Note: Batch Normalization layers were removed.

Comparing results before and after modifications:

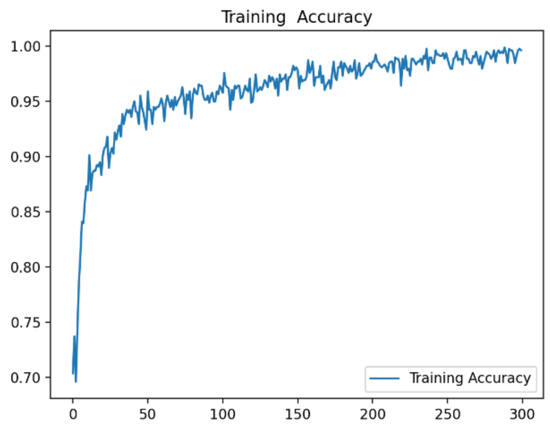


Fig. 7.5-a: training accuracy curve after modifications

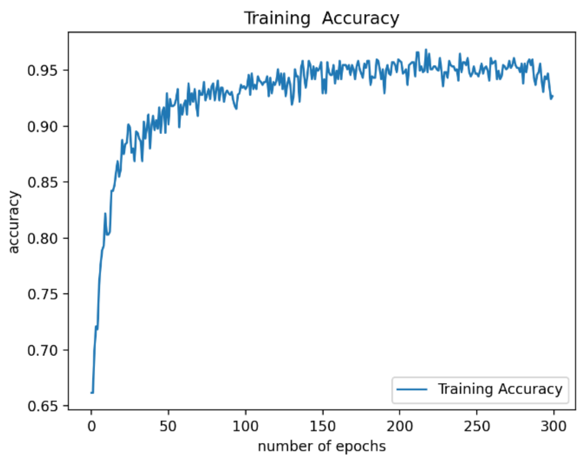


Fig. 7.5-b: training accuracy before modifications

Note: Training accuracy curves before and after modification in figure 7.5-a, b are for the third approach

As it is clear from the training accuracy curve, using zero padding and L2 regularization increases the training accuracy value by 2% higher than the original model. However, the accuracy on the test set in case of the three applied approaches slightly changed.

Table 7.4: Accuracy values for the three approaches after adding zero padding and L2 Regularization

Accuracy	Training	Validation	Test Set
First Approach	0.89	0.88	0.87
Second Approach	0.96	0.95	0.93
Third Approach	0.9835	0.96	0.955

7.5 Classification Results

In this section, classification results for patches from class one and class zero using the original (explained in section 6.3) and modified model (section 7.4) will be presented in table 7.5.

Table 7.5: Classification results for all patches with the original model and the modified one

		Original Model		Modified Model	
		Class 0	Class 1	Class 0	Class 1
First Approach	Total number of patches	1871	1300	1871	1300
	Correctly Classified	1559	1155	1608	1161
	Errors	312	145	263	138
Second Approach	Total number of patches	835	1256	835	1256
	Correctly Classified	762	1169	770	1180
	Errors	73	87	65	76
Third Approach	Total number of patches	531	625	531	625
	Correctly Classified	520	589	520	611
	Errors	11	36	11	14

Table 7.6: Comparison between the proposed model and some reported algorithms

Algorithm	Target	Results
SOM (Self Organizing Map) Neural network Lampinen et al. [72] The method was based on clustering of the high dimensional measurements into a small number of features with self-organizing maps.	A method for constructing classication features with unsupervised learning. Colour image recognition for selection of optimal reproduction parameters in printing press, and defect classication in wood surfaces.	1- Performance of the wood defect classication system was evaluated with a set of 400-knot images from Pinewood dataset with 7 classes from spruce boards. The recognition rate was about 85% with only gray level images, giving about 90% accuracy for the final board grading, to be compared to 75 - 80% accuracy that can be maintained by a human inspector [72]. 2-Computationally, the self-organizing feature mapping was rather heavy [72].
PCA for feature fusion with compressed sensing Li et al. [32] (**Section 2.2.1**)	Detect wood defects from wood plate images.	1- Twenty-five features were extracted, including geometry and regional features, gray-scale texture features, and invariant moment features, from

PCA was applied to reduce data redundancy and feature dimensions. Compressed sensing was used as a classifier.	Improve identification accuracy.	wood board images. 2- One hundred and fifty Xylosma samples of live knots, dead knots, and cracks were tested. The average detection time with PCA feature fusion and without were 0.2015 and 0.7125 ms respectively [32]. 3- Classification Accuracy over Xylosma wood boards: 92% [32].
Linear Discriminant analysis with compressed sensing Zhang et al. [11] (**Section 2.2.2**) LDA algorithm was used to integrate features of defects and reduce their dimensions.	Detection of three types of defects in wood, which were live knots, dead knots and cracks. Reduce processing time.	1-Fifty images for each defect type were used to test the effects of this method. The average time for feature fusion and classification was 0.446 ms with the classification accuracy of 94% [11]. 2-Classification Accuracy values over species of wood including Fraxinus mandshurica, Xylosma racemosum, Korean Pine, and Oak were 90% for Live knots, 95% Dead knots and 100% for crack [11].
Our Proposed model Using Deep CNN Three approaches were applied: 1-using classification threshold 2-using classification threshold and margins 3-Dataset manual validation (**Section 6.3**)	Detect knots with respect to certain depth threshold. Classify wood log crops into two classes (Class 1 and Class 0) based on the number and dimensions of knots detected in each crop.	1-The model was tested using test set of 209 and 116 patches for second and first approaches respectively. 2- Accuracy values over the pine no bark wood log crops test sets of the applied approaches were 92.5% and 95% for second and third approaches respectively. 3- Only 1.20s needed for both detection (image pre-processing and identification) and wood log crops classification.

Looking at the aforementioned evaluation metrics along with some of the reported algorithms and approaches, it could be reiterated that deep networks such as variants of DCNN could extract features of defects in wood and automatically classify wood log images that were collected using a CT Log scanner more accurately and efficiently than conventional methods.

Furthermore, the second and third approaches that were applied with our dataset proved their efficiency using metrics as accuracy, precision and recall.

Chapter 8

Conclusion

The aim of this thesis was to detect knots and classify wood log crops based on the number and the dimensions of knots that were found in each wood log crop. After reviewing some previous solutions in the literature, the research was expanded outside the context of extracting features of defects in wood like knots by classifying wood log crops into two categories “many knots“ and “almost no knots”. Thanks to the flexibility of deep learning algorithms, the DCNN model was easily adapted to the task of the project.

The deep CNN was trained on pine no bark wood log crops to extract common features in wood log crops that have many knots and others with almost no knots. The number of wood logs were 962 that were cropped into smaller crops for both categories. After that, labels were saved for wood log crops with knots below the depth threshold (close to the surface). While those knots above the depth threshold (deep knots) were discarded and not considered in computations.

Before the training, each wood log crop was given a score based on the number and dimensions of knots. Then, the crops were classified into Class 1 and Class 0 based on classification threshold. In other words, Wood logs with large score were categorized as class 1 “many knots”; while others were classified as class zero “almost no knots”. Three approaches were applied. In the first one, we used only classification threshold to distinguish between the two classes. For the second one, a classification threshold along with margins were used where those crops that were detected with scores in between the threshold and the margins were removed from the dataset. In the third approach, we used a GUI interface to manually validate crops from both classes.

The Performance of the model was evaluated by the accuracy on training, test and validation sets. The accuracy values for the model were 0.96, 0.94, 0.95 on training, test and validation sets respectively. The experimental results showed that our method achieved a precision of 0.96 and 0.90, and a recall of 0.98 and 0.94 for crops from both classes (zero and one respectively).

The results obtained from evaluating the model using accuracy, precision and recall assured that DCNN model could recognize wood defects more accurately and effectively than conventional methods when dealing with CT Log images.

References

[1] F. Zolotareva, T. Eerolaa, L. Lensua, H. Kälviäinena, T.Helina and H. Haarioa. “Modelling internal knot distribution using external log features,” Computers and Electronics in Agriculture, Vol.179, December 2020. **https://doi.org/10.1016/j.compag.2020.105795**

[2] J. D. Mehta, S. G. Bhirud, “Image Stitching Techniques,” Department of Computer Technology, VJTI, Mumbai, India, pp. 74–80, ISBN: 978-81-8489-989-4,2011. **https://doi.org/10.1007/978-81-8489-989-4_13**

[3] D. Devaru, U. B. Halabe, B. Gopalakrishnan, S. Agrawal and S. Grushecky, ”Algorithm for detecting defects in wooden logs using ground penetrating radar”. Proceedings of SPIE - The International Society for Optical Engineering, Vol.5999, November 2005. **https://doi.org/10.1117/12.630835**

[4] P. Li, J. Hi, A. L. Abbott, D. L. Schmoldt, ” Labelling Defects in CT Images of Hardwood Logs With Species-Dependent and Species-Independent Classifiers”, January 1996.

[5] Q. Qiu and D. Lau, “Grain effect on the accuracy of defect detection in wood structure by using acoustic-laser technique”, Non-destructive Characterization and Monitoring of Advanced Materials, Aerospace, Civil Infrastructure, and Transportation, Vol. 10971, 2019. **https:// doi:10.1117/12.2514285.**

[6] P. Siekański, K. Magda, K. Malowany, J. Rutkiewicz, A. Styk, J. Krzesłowski, T. Kowaluk, A. Zagórski. “On-Line Laser Triangulation Scanner for Wood Logs Surface Geometry Measurement”, Sensors (Basel) .Vol. 1074, March 2019. **https://doi: 10.3390/s19051074**

[7] L. Espinosa, F. Prieto, L. Brancheriau, P. Lasaygues. “Effect of wood anisotropy in ultrasonic wave propagation: A ray-tracing approach“. Ultrasonics, PP.242-251, Vol.91, 2019. **https://DOI: 10.1016/j.ultras.2018.07.015**

[8] H. Yang, L. Yu. “Feature extraction of wood-hole defects using wavelet-based ultrasonic testing”. Journal of Forestry Research. PP.395-402. Vol.28. June 2016. **https://doi:10.1007/s11676-016-0297-z**

[9] M. Lukomski, M. Strojecki, B. Pretzel, N. Blades., V. L. Beltran, A. Freeman. “Acoustic emission monitoring of micro-damage in wooden art objects to assess climate management strategies”. Insight – Non Destructive Testing and condition Monitoring, PP.256–264, Vol. 59.May 2017. **https://doi.org/10.1784/insi.2017.59.5.256**

[10] F. J. Rescalvo, I. Valverde-Palacios, E. Suarez,A. Roldan, A. Gallego. “Monitoring of carbon fibre-reinforced old timber beams via strain and multiresonant acoustic emission sensors”. Ultrasonic Sensors 2018, Vol. 18, PP.1224. **https://doi.org/10.3390/s18041224**

[11] C. Li., Y. Zhang, W. Tu, C. Jun, H. Liang, H. Yu. “Soft measurement of wood defects based on LDA feature fusion and compressed sensor images”. J. For. Res., Vol.28, PP.1285–1292, April 2017.

[12] J. Dai, Y. Li, K. He, J. Sun.” R-FCN: Object detection via region-based fully convolutional networks”. NIPs’16: Proceedings of the 30th International Conference on Neural Information processing systems. PP.379-387, December 2016.

[13] Y. Zhang, S. Liu, J. Cao et al.”A rapid, automated flaw segmentation method using morphological reconstruction to grade wood flooring. Journal of forestry research, Vol. 25, PP.959-964, December 2014. https://doi.org/10.1007/s11676-014-0543-1

[14] Y. Z. Zhang, J. Cao, L. Xu and H. L. Yu. “Wood floor defects segmentation and recognition based on morphological and SOM”. Electric Machines and Control, Vol. 17, PP. 116-120, April 2013.

[15] M. Ester, H. P. Kriegel, J. Sander, X. Xu. “A density-based algorithm for discovering clusters in large spatial databases with noise”. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. AAAI Press, pp. 226–231, 1996.

[16] V. T. Nguyen, B. Kerautret, I. Debled-Rennesson, F. Colin, A. Piboule, T. Constant. “Segmentation of defects on log surface from terrestrial lidar data”. In: 2016 23rd International Conference on Pattern Recognition (ICPR), IEEE. pp. 3168–3173, December 2016. https://doi.org/10.1109/ICPR.2016.7900122

[17] E. Duchateau, F. Longuetaud, F. Mothe, C. Ung, D. Auty, A. Achim. “Modelling knot morphology as a function of external tree and branch attributes”. Canadian Journal of Forest Research, Vol. 43, PP. 266–277, January 2013. https://doi.org/10.1139/cjfr-2012-0365

[18] T. Lindeberg. “Detecting salient blob-like image structures and their scales with a scale-space primal sketch: a method for focus-of-attention”. International Journal of Computer Vision, Vol. 11, PP. 283–318, December 1993. https://doi.org/10.1007/BF01469346

[19] Y. Yang, X. Zhou, Y. Liu, Z. Hu, F. Ding. “Wood Defect Detection Based on Depth Extreme Learning Machine”. Applied Sciences, Vol. 10, PP. 7488. , October 2020. https://doi.org/10.3390/app10217488

[20] C. Szegedy, W. Liu., Y. Q. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich. “Going Deeper with Convolutions”. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, PP. 1–9, June 2015. https://doi.org/ 10.1109/CVPR.2015.7298594

[21] Y. LeCun, Y. Bengio, G. Hinton. ”Deep learning”. Nature, Vol. 521, PP.436–444, May 2015. https://doi.org/10.1038/nature14539

[22] T. He, Y. Liu, Y. B. Yu, Q. Zhao, Z. K. Hu. “Application of deep convolutional neural network on feature extraction and detection of wood defects”. Measurement, Vol.152, February 2020. https://doi.org/10.1016/j.measurement.2019.107357

[23] K. Hu, J. B. Wang, Y. Shen, J. R. Guan, Y. Cai. “Defect identification method for poplar veneer based on progressive growing generated adversarial network and MASK R-CNN Model”. BioResources, Vol.15, PP. 3041– 3052, https://doi:10.15376/biores.15.2.3041-3052

[24] J. Shi, Z. Li, T. Zhu, D. Wang, C. Ni. " Defect detection of industry wood veneer based on NAS and multi-channel mask R-CNN". Sensors, Vol. 20, issue 16, PP. 4398, August 2020. **https://doi:10.3390/s20164398**

[25] Y. Guo, Ü. Budak, L.J. Vespa, E. Khorasani, A. Şengür. "A retinal vessel detection approach using convolution neural network with reinforcement sample learning strategy". Measurement, Vol. 125, PP. 586–591, September 2018. **Https: //doi.org/10.1016/j. measurement.2018.05.003**

[26] C.C. Park, Y. Kim, G. Kim. ‘‘Retrieval of Sentence Sequences for an Image Stream via Coherence Recurrent Convolutional Networks," in IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 40, issue 4, pp. 945–957, April 2018. **https:// doi.org/10.1109/TPAMI.2017.2700381**

[27] E. Johansson, T. Pahlberg and O. Hagman. "Fast visual recognition of Scots pine boards using template matching". In: Computers and Electronics in Agriculture, Vol. 118, pp. 85–91, October 2015. **Doi: 10.1016/j.compag.2015.08.026**

[28] Y. Zhang, S. Liu, J. Cao, C. Li, H. Yu."Wood board image processing based on dual-tree complex wavelet feature selection and compressed sensing", In: Wood Science and Technology, Vol. 50, pp. 297–311, 2016. **https://doi.org/10.1007/s00226-015-0776-y**

[29] N. Chen, X. Men, X. Han, X. Wang, J. Sun., H. Chen. ‘‘Edge detection based on machine vision applying to laminated wood edge cutting process". In: 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA), Wuhan, China, June 2018, IEEE, pp. 449–454. **Doi: 10.1109/ICIEA.2018.8397759**

[30] W. Funck, Y. Zhong, D.A. Butler, C.C. Brunner." Image segmentation algorithms applied to wood defect detection", Computers and Electronics in Agriculture, Vol. 41, issues: 1-3, PP. 157–179. **https:// doi.org/10.1016/S0168-1699 (03)00049-8**

[31] L. Lin, S. He, F. Fu, X. Wang. ‘‘Detection of wood failure by image processing method: influence of algorithm". In: adhesive and wood species, European Journal of Wood and Wood Products, Vol. 73, PP. 485–491, May 2015. **https://doi.org/10.1007/s00107-015-0907-z**

[32] Y. Zhang, C. Xu, C. Li, H. Yu, J. Cao. ‘‘Wood defect detection method with PCA feature fusion and compressed sensing." In: Journal of Forestry research, Vol.26, pp.745-751, April 2015. **Doi: 10.1007/s11676-015-0066-4**

[33] G.A. Ruz, P.A. Estevez, P.A. Ramirez, "Automated visual inspection system for wood defect classification using computational intelligence techniques. "In: International Journal of Systems Science, Vol. 40, issue 2, PP. 163-172. **DOI: 10.1080/00207720802630685**

[34] DT. Pham, RJ. Alcock, et al. "Automated visual inspection of wood boards selection of features for defect classification by a neural network". In: Proceedings of the Institution of Mechanical Engineers, Part E: Journal of Process Mechanical Engineering, Vol. 213, issue 4, PP. 231-245. **https://doi.org/10.1243/0954408991529852**

[35] M. Castellani, H. Rowlands." Evolutionary artificial neural network design and training for wood veneer classification". In: Engineering Applications of artificial Intelligence, Vol.22, issues: 4-5, PP.732–741, June 2009. **https://doi.org/10.1016/j.engappai.2009.01.013**

[36] M. M. Hittawe, S. M. Muddamsetty, and D. Sidibe, F. Mériaudeau. ‘‘Multiple features extraction for timber defects detection and classification using SVM.” In: 2015 IEEE International Conference on Image Processing(ICIP), Quebec City, Canada, IEEE, pp. 427–431, September 2015 . doi: 10.1109/ICIP.2015.7350834

[37] D. Qi, P. Zhang, X. Zhang, et al.”Appling a novel cost function to Hopfield neural network for defects for boundaries detection of wood image”. In: EURASIP Journal on Advances for Signal Processing, Vol. 6, PP. 1–8, June 2010. https://doi.org/10.1155/2010/427878

[38] Coolidge, Julian. “The origin of polar coordinates”. In: American Mathematical Monthly. Mathematical Association of America, Vol. 52, issue 2, PP. 78-85, 1952. doi:10.2307/2307104. JSTOR 2307104.

[39] Y. LeCun, G. Hinton, Y. Bengio. ”Deep Learning”, Nature 521, P.P.436-444, May 2015. https://doi.org/10.1038/nature14539

[40] Z. Zhang, P. Cui, W. Zhu. ”Deep Learning on Graphs: a Survey”. In: Transactions on Knowledge and Data Engineering, March 2020. https://doi.org/10.1109/TKDE.2020.2981333

[41] A. Shrestha, A. Mahmood.” A Review of deep learning algorithms and architectures”, IEEE Access, Vol. 7, P.P. 53040-53065, April 2019.

[42] J. Tang, S. Li, P. Liu.” A review of lane detection methods based on deep learning”. In: Pattern Recognition journal, Vol. 111, P.P.107-123, March 2021. DOI: 10.1016/j.patcog.2020.107623

[43] Z. Q. Zhao, P. Zheng, S.T. Xu, X. Wu.” Object detection with deep learning: a review”. In: IEEE Transactions on Neural Network and learning systems, P.P. 3212-3213, Vol. 30, issue 11, January 2019. DOI: 10.1109/TNNLS.2018.2876865

[44] W. Yang, X. Zhang, Y. Tian, W. Wang, J. H. Xue, Q. Liao. “Deep learning for single image super-resolution: a brief review”. In: IEEE Transactions on Multimedia, Vol. 21, issue 12, P.P.3106-3121, December 2019. DOI: 10.1109/TMM.2019.2919431

[45] A. Krizhevsky, I. Sutskever, G. E. Hinton.” ImageNet Classification with deep convolutional neural networks”. IN: Communications of the ACM, Vol.60, issue 6, P.P. 84-90, May 2017. https://doi.org/10.1145/3065386

[46] K. He, X. Zhang, S. Ren, J. Sun. “Deep residual learning for image recognition”. In: Proceedings of the IEEE conference on computer vision and pattern recognition, P.P 770-778, June 2016. DOI: 10.1109/CVPR.2016.90

[47] L. Alzubaidi, J. Zhang, A. J. Humaidi et al.”Review of deep learning: concepts, CNN architectures, applications, future directions”. In: Journal of big data, Vol. 8, issue 53, March 2021. https://doi.org/10.1186/s40537-021-00444-8

[48] M. Hossin, M. Sulaiman. ”A review on evaluation metrics for data classification evaluations “.In: International Journal of Data Mining & Knowledge Management Process, P.P. 1-11, Vol.5, issue 2, March 2015. DOI:10.5121/ijdkp.2015.5201

[49] D.J. Hand, R.J. Till. " a simple generalisation of the area under the ROC curve for multiple class classification problems". In: Machine learning and computer Science, Vol 45, issue 2, P.P. 171-178, October 2001. **DOI: 10.1023/A: 1010920819831**

[50] C. Shorten, T. M. Khoshgoftaar. A survey on image data augmentation for deep learning". In: Journal of Big Data, Vol.6, issue 1, 2019. **https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0197-0**

[51] S. Lofe, C. Szegedy." Batch normalization: accelerating deep network training by reducing internal covariate shift". March 2015. **https://doi.org/10.48550/arXiv.1502.03167**

[52] F. J. Moreno-Barea, F. Strazzera, J. M. Jerez, D. Urda, L.Franco."Forward noise adjustment scheme for data augmentation". In: 2018 IEEE symposium series on computational intelligence (SSCI), IEEE, P.P.728-734, November 2018. **DOI: 10.1109/SSCI.2018.8628917**

[53] D. Dua, E. K. Taniskidou. UCI machine learning repository. Irvine: University of California. School of Information and Computer Science; 2017. **http://archive.ics.uci.edu/ml**

[54] C. Pelletier, G. Webb, F. Petitjean." Temporal Convolutional Neural Network for the Classification of Satellite Image Time Series". In: Remote Sensing, Vol. 11, P.P. 523, March 2019**.** **DOI:10.3390/rs11050523**

[55] A. Khan, A. Sohail, U. Zahoora, A. S. Qureshi." A survey of the recent architectures of deep convolutional neural networks". In: Artificial intelligence review, Vol. 53, issue 8, P.P. 5455-5516, January 2019. **https://link.springer.com/article/10.1007/s10462-020-09825-6**

[56] A. Shrestha, A. Mahmood." Review of deep learning algorithms and architectures". In: IEEE Access, April 2019. **https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8694781**

[57] G. E. Dahl, T. N. Sainath, G. E. Hinton. "Improving deep neural networks for LVCSR using rectified linear units and dropout". In: 2013 IEEE international conference on acoustics, speech and signal processing. IEEE, p. 8609–8613. May 2013. **DOI: 10.1109/ICASSP.2013.6639346**

[58] URL: **https://www.baeldung.com/cs/learning-rate-batch-size**

[59] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting". In: Journal of Machine Learning Research. Vol.15, issue 1, P.P. 1929–1958, 2014. **DOI: 10.5555/2627435.2670313**

[60] Xu B, N. Wang, T. Chen, M. Li. "Empirical evaluation of rectified activations in convolutional network", May 2015. **DOI: arXiv: 1505.00853**

[61] S. Hochreiter. "The vanishing gradient problem during learning recurrent neural nets and problem solutions". In: international journal of uncertainty, Fuzziness and Knowledge Based Systems, Vol.6, issue 2, PP.107–116, 1998. **https://doi.org/10.1142/S0218488598000094**

[62] K. Simonyan, A. Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: arXiv preprint, April 2015. DOI: arXiv: 1409.1556

[63] M. D. Zeiler, R. Fergus. “Visualizing and understanding convolutional networks”. In: European conference on computer vision. Springer, p. 818–833, November 2013. https://doi.org/10.48550/arXiv.1311.2901

[64] M. Ranzato, F. J. Huang, Y. L. Boureau, Y. LeCun. “Unsupervised learning of invariant feature hierarchies with applications to object recognition”. In: 2007 IEEE conference on computer vision and pattern recognition, IEEE, PP. 1–8, July 2007. DOI: 10.1109/CVPR.2007.383157

[65] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, et al. ” Deep high-resolution representation learning for visual recognition”. In: IEEE Trans Pattern Anal Mach Intell, Vol. 43, issue 10, September 2021. https://doi.org/10.1109/TPAMI.2020.2983686

[66] B. Cheng, B. Xiao, J. Wang, H. Shi, T. S. Huang, L. Zhang. “Higherhrnet: scale-aware representation learning for bottom-up human pose estimation”. In: CVPR2020, March 2020. https://www.microsoft.com/en-us/research/publication/higherhrnet-scale-aware-representation-learning-for-bottom-up-human-pose-estimation/.

[67] URL: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

[68] URL: https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/AdamW

[69] URL: https://www.cse.msu.edu/~cse802/S17/slides/Lec_20_21_22_Clustering.pdf

[70] S. Grundberg, A. Grönlund. “Feature extraction with the aid of an x-ray log scanner”. In: Proceedings from the 3rd International Seminar/Workshop on Scanning Technology and Image Processing on Wood: Skelleftea, Sweden, August 17 - 19, 1998.

[71] URL: https://keras.io/api/layers/preprocessing_layers/image_augmentation/random_rotation/

[72] J. Lampinen and S. Samolander. “Self-organizing feature extraction in recognition of wood surface defects and colour”. In: International Journal of Pattern Recognition and Artificial Intelligence, Vol. 10, No. 2, PP. 97-113, 1996.