Lappeenranta-Lahti University of Technology LUT
School of Engineering Science
Computational Engineering and Technical Physics
Computer Vision and Pattern Recognition

Daniel Batrakhanov

VIRTUAL SAWING USING GENERATIVE ADVERSARIAL NETWORKS

Master's Thesis

Examiners: Associate Professor Tuomas Eerola
Professor Sanjar Abdullaev

Supervisors: M.Sc. Fedor Zolotarev
Associate Professor Tuomas Eerola
Professor Lasse Lensu
Professor Heikki Kälviäinen

ABSTRACT

Lappeenranta-Lahti University of Technology LUT
School of Engineering Science
Computational Engineering and Technical Physics
Computer Vision and Pattern Recognition

Daniel Batrakhanov

Virtual sawing using Generative Adversarial Networks

Master's Thesis

2021

78 pages, 61 figures, 5 tables.

Examiners: Associate Professor Tuomas Eerola
Professor Sanjar Abdullaev

Keywords: computer vision, virtual sawing, image-to-image translation, generative adversarial networks, convolutional neural networks

The global trend towards digitalization along with the 4th Industrial Revolution allows building new innovative solutions to optimize manufacturing. In particular, the highly competitive sawmill industry is not an exception. The industry always depends on efficient raw material utilization, and thus, exploration of the internal log structure is an important feature in the timber conversion process. One common approach for a comprehensive internal wood structure examination is virtual sawing that is predicting the outcome of sawing process based on log measurements. The main objective of this thesis was to study the suitability of state-of-the-art generative adversarial networks for virtual sawing. The specific aims were to choose an appropriate generative adversarial network architecture to build a trainable model as an extension to an existing virtual sawing system. The proposed method for image-to-image translation is capable to synthesize the photorealistic images of the boards based on log measurements. The defects (knots) on virtual boards were detectable and their locations correspond to those in real boards.

PREFACE

I would like to thank Heikki Kälviäinen, Lasse Lensu, Tuomas Eerola, and Fedor Zolotarev for sharing their immense knowledge and precious time with me.
I am also grateful to my family, relatives, and friends for tremendous support during the studies.

Per aspera ad astra.

Lappeenranta, June 1, 2021

Daniel Batrakhanov

CONTENTS

LIST OF ABBREVIATIONS

2-D	Two Dimensional
3-D	Three Dimensional
Adam	Adaptive Moment Estimation
AP	Average Precision
AR	Average Recall
CEGAN	Consistent Embedded GAN
cLR-GAN	Conditional Latent Regressor GAN
CNN	Convolutional Neural Networks
COCO	Common Objects in Context
cVAE-GAN	Conditional Variational Autoencoder GAN
FC	Fully Connected Layer
FP	False Positive
FPN	Feature Pyramid Network
FN	False Negative
GAN	Generative Adversarial Network
IoU	Intersection-over-Union
LoG	Laplacian of Gaussian
MUNIT	Multimodal Unsupervised Image-to-Image Translation
QGAN	Quality-Aware GAN
PCC	Pearson Correlation Coefficient
R-CNN	Region-Based Convolutional Neural Network
ReLU	Rectified Linear Unit
RGB	Red, Green, Blue
SPA-GAN	Spatial Attention GAN
TanH	Hyperbolic Tangent
TP	True Positive
UNIT	Unsupervised Image-to-Image Translation
VAE	Variational Auto-Encoders
VTT	Technical Research Centre of Finland

1 INTRODUCTION

1.1 Background

The Finnish forest industry accounted for 20% of the total export value with 9 thousand million EUR generated GDP and employment of 62000 people [1]. The largest part of the sawmill business depends on log processing. The raw material expenses are normally about 60-70% of total costs [2]. Therefore, the sawmill profitability highly depends on efficient raw material utilization. Moreover, the potential gain from a more effective board production reduces the wood wastes (residues) and leads to a decline in the volume of excess logging and the development of sustainable forestry.

Thus, optimization of the sawing parameters is essential in terms of rational forest resource management and revenues of the mill companies. Currently, a sawing pattern and an angle are not based on the internal features of a log, since the first is decided beforehand, and the second is typically chosen randomly. The location and sizes of the internal wood defects strongly impact the end-product quality as they decrease the local strength of the sawn products. The most crucial internal defect is a knot. A knot is a place where the tree branches join the stem and they become an integral part of it (see Figure 1) [3]. Hence, a system that would provide a transparent feedback loop for the mill operator on the internal wood structure is capable to increase yield and highly valuable for the sawmill industry.

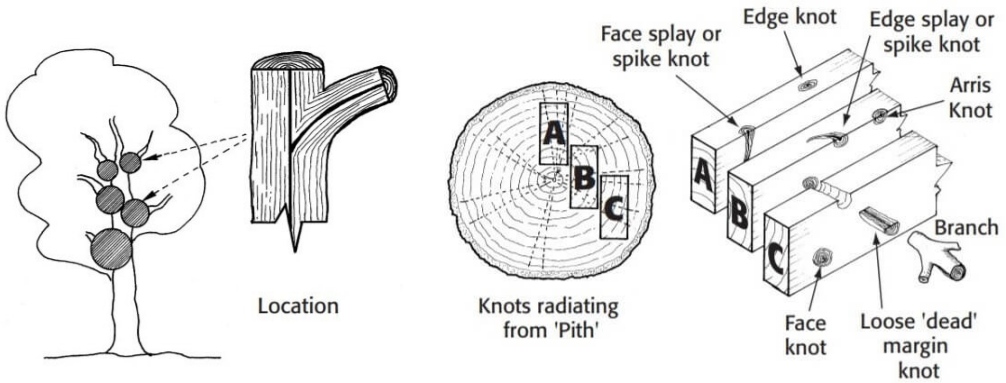


Figure 1. Knot is a natural structural wood defect [3].

The master's thesis focuses on virtual sawing, which simulates the sawing process with a digital reconstruction of a wooden log in order to predict the outcome of the real sawing process. The virtual sawing system can help an operator to make the correct decision

and to increase the production rates. More specifically, the thesis considers generative adversarial networks (GAN) to construct a trainable model for virtual sawing. GANs are powerful deep machine learning models that consist of two neural networks: generator and discriminator. GANs are commonly used for generating realistic images or translating them from one domain to another (image-to-image translation). The work builds around earlier research on virtual sawing where laser scans of a log surface were utilized to estimate the knot locations inside the log (see Figure 2) [4]. The method outputs grayscale images (knot maps) where the pixel values correspond to the probability of the knot appearing in that location on the sawn board. The grayscale knot maps are used as an input for the GAN, which translates them to photorealistic images. Moreover, the study explores the capability of an image-to-image GAN based technique to generate photorealistic images from raw surface heightmaps projected onto the boards surface.

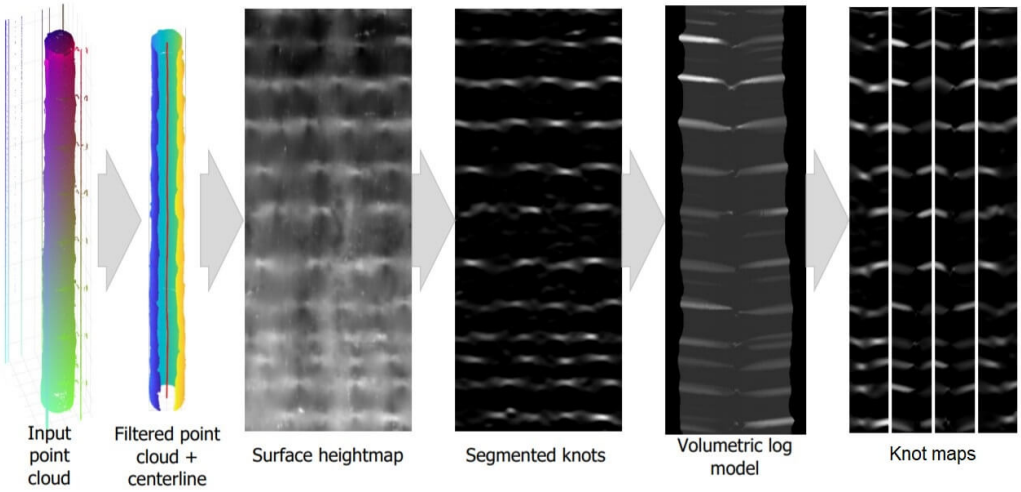


Figure 2. Step by step algorithm of obtaining of the knot maps [4].

This master's thesis continues the work carried out in the DigiSaw research project [5]. The DigiSaw project is oriented to develop and to modify sawmill manufacturing processes via the implementation of novel digital solutions.

1.2 Objectives and delimitations

The main goal of the master's thesis is to implement a trainable virtual sawing model based on the selected GAN architecture. The model should be capable to produce photorealistic images from the knot maps and the raw projected heightmaps obtained in the mentioned research (see Figure 3) [4]. The main objectives of this thesis are the following:

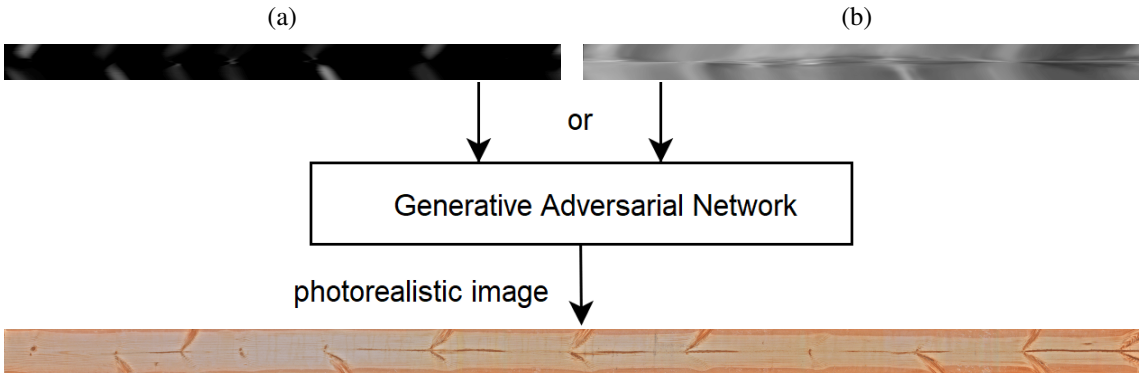


Figure 3. An input conversion to a photorealistic image: (a) Using as the input a knot map; (b) Using as the input a raw projected heightmap

1. Study the state-of-the-art models for generative adversarial networks.
2. Select a suitable model for the virtual sawing system.
3. Prepare a dataset of RGB images and corresponding knot maps and raw projected log surface heightmaps.
4. Implement, train and evaluate the applicability of the proposed GAN architecture.
5. Study the possibilities to modify the virtual sawing method proposed by Zolotarev et al. to improve the GAN performance [4].

As this work is based on the previous research, the data delimitation is inherited, only Scots pine (Pinus sylvestris) is considered [4]. However, the GAN method should be also applicable to wood species if proper training data is provided.

1.3 Structure of the thesis

This thesis organized as follows: Chapter 2 gives a short introduction to the sawmill process and existing machine learning frameworks for its optimization, including virtual sawing. Chapter 3 provides an overview of the generative models and state-of-the-art approaches for generative adversarial networks in terms of an image-to-image translation task. Chapter 4 discusses all the details of the measurements-to-image translation including the implemented GAN architecture. The results are presented in Chapter 5. In Chapter 6 the discussion of the achieved results and possible future work is given. Finally, the conclusions are drawn in Chapter 7.

2 SAWMILLING AND VIRTUAL SAWING

2.1 Sawmill process

The sawmilling process refers to a certain number of steps in order to transform raw wood material into the final product, e.g., planks or boards [6]. It includes a wide range of activities from harvesting and transportation of roundwood to timber drying, sorting, and grading (see Figure 4) [6]. Approximately, 50% of the raw material volume (roundwood) converts to viable boards and planks [7]. In the next subsections, the base outline is presented where the conversion chain depends on the sawmill equipment, environment, a species of wood material, and the desired product as well [3].

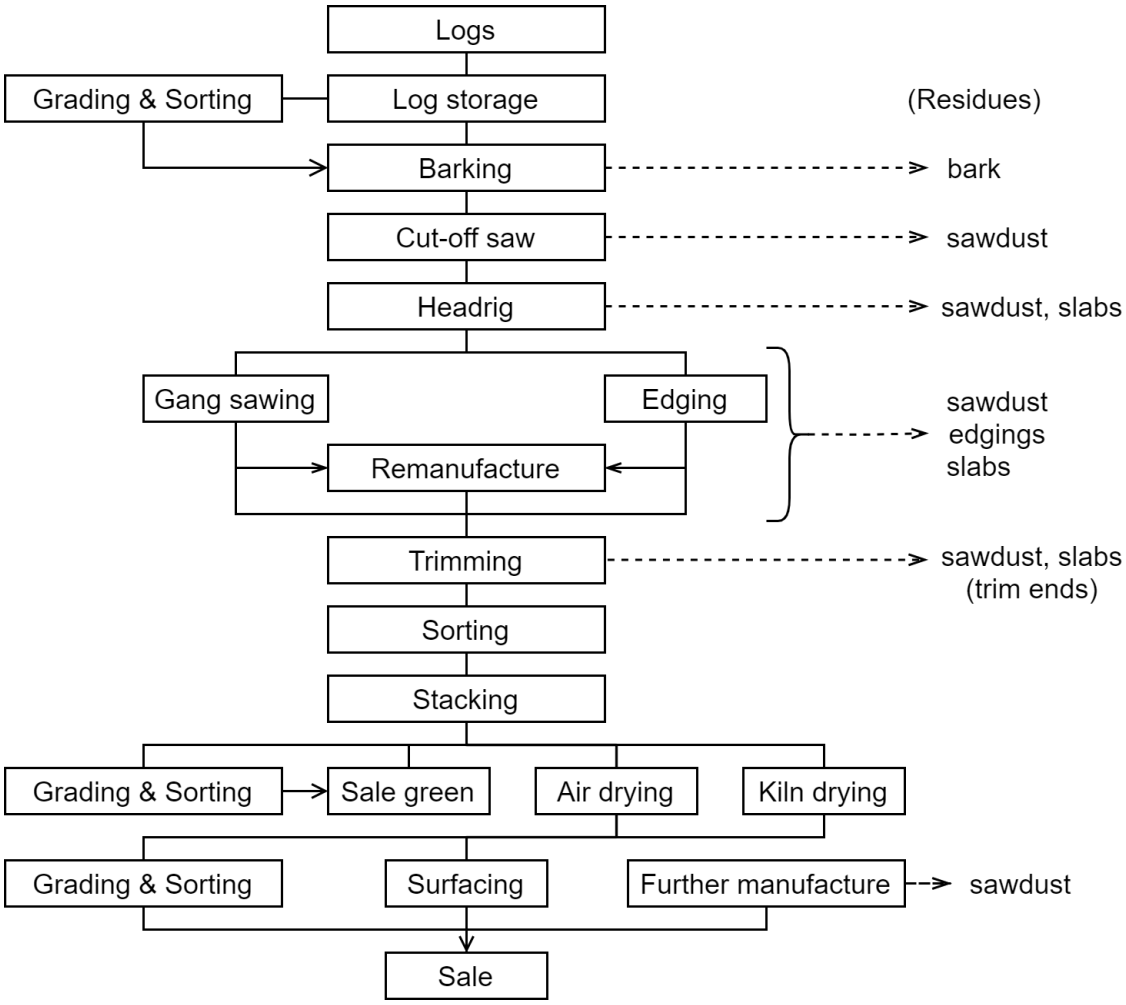


Figure 4. The sawmill process from forest to the sawn wood product [6].

2.1.1 Log sorting and barking

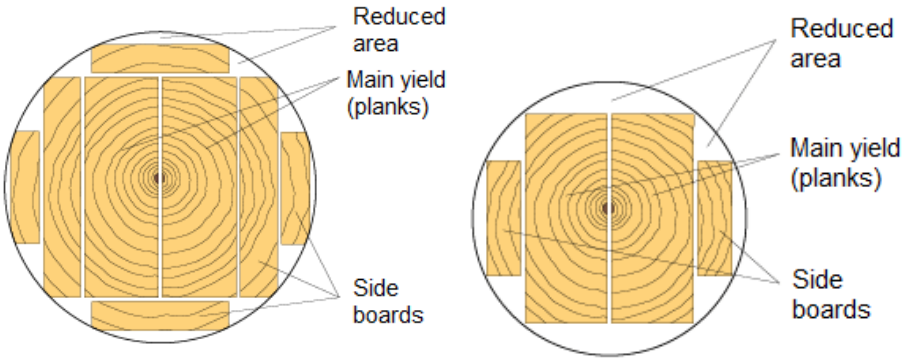
After harvesting trees from a forest, roundwood is delivered to a sawmill factory [7]. On arrival at the sawmill, roundwood is sorted by species, dimensions (length and diameter), end-use, and other relevant features. The logs are accumulated in sufficient amount to form a buffer which provides continuous manufacturing in the case of the material supply disruptions [6]. Further processing requires debarking the logs. Debarking is a protection measure from excessive wear and damage to the sawmill equipment, but it also contributes a sawyer to estimate the quality of an arrived timber [6].

2.1.2 Log sawing

Next the logs are cut to the maximum possible length with a cut-off saw and put onto a headsaw carriage [6]. A headrig operator chooses a sawing pattern to obtain the most efficient utilization with the minimum volume of waste and conveys accordingly the log through a headsaw. The sawing pattern is defined by various parameters such as a type of the sawing machines, a log size, a type of wood, condition of the wood, a proportion of heartwood to sapwood, a future use, structural or decorative, and also customer requirements for the thicknesses and widths (see Figure 5) [3] [8]. More specifically, wood can be divided into two basic subgroups, that is hardwood and softwood. Hardwood usually comes from deciduous trees which lose their leaves annually. Softwood generally comes from conifer trees, which usually remain evergreen. The trees from which hardwood is obtained tend to be slower growing, meaning the wood is usually denser. As a consequence, hardwood and softwood require distinct sawing approaches [3]. Finally, the received wood material is sawn into planks and boards. On the last step of this stage, the edges are aligned and the defects are pruned by edging and trimming in order to upgrade the lumber quality and to standardize the proportions [6].

2.1.3 Sorting and grading

At this stage, the sawn timber is sorted according to the dimensions and species, and then is graded [6]. Grading refers to the categorization of the lumber which comes from an overall quality and a condition [3]. Quality is a quantity, a size, and a type of the following defects (see Figure 6):



(a) Big log (b) Small log

Figure 5. Commonly used sawing patterns: (a) For a big log; (b) For a small log [8].

- Natural defects (e.g., knots).
- Defects as a result of drying (checking, splitting, distortion).
- Machining defects (as a result from saw/roller marks).

Condition is generally connected with surface staining and a moisture content [3].

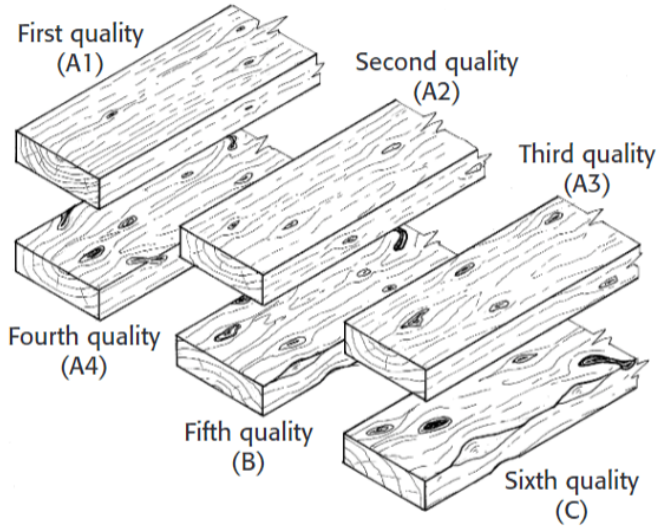


Figure 6. A possible appearance of the different grades of European softwood [3].

2.1.4 Drying

The timber that is not sold green (freshly felled, containing "free water") is enhanced further by a drying procedure [6]. The main goal of lowering the moisture content, i.e. the

drying, is to improve strength and color properties, to provide stability in size and shape, to reduce weight, and many other enhancements [3]. The drying process is generally performed by one of the following methods [3]:

- Air drying (natural drying).
- Kiln-drying (artificial drying).
- Air drying followed by kiln-drying.

After the drying phase, the timber is inspected then if defects are revealed, and it may be cut by trimming and regraded accordingly [6]. The final product is stacked and stored at a warehouse.

2.2 Machine learning in sawmill process optimization

Over the last two decades, machine learning has developed dramatically from laboratory research to practical technology with an extensive commercial use [9]. It has emerged as the method of choice for developing computer vision, speech recognition, natural language processing, robot control, and many other systems [9]. Machine learning systems can be described as algorithms that are capable to perform tasks by generalizing from examples [10]. This approach is often cost effective compared to traditional programming where codes and rules are formulated manually by a person [10]. The machine learning approaches are derived from data and how the data should be utilized by a learning system [11]. The fundamental goal of machine learning is to train an algorithm on a limited amount of data (training data) in such a way that chosen model performs well on the new data (test data, previously unseen).

Digitalization of the manufacturing processes leads to automation in production techniques, lowered response times, and increased operation accuracy, more efficient storing and distribution, and other industry optimizations [12]. In particular, the digitalization of the sawmill industry is proceeding via numerous researches in the application of machine learning and computer vision models. However, there are still many opportunities to be found and researched since the 4th Industrial Revolution has just begun.

2.2.1 Defect detection in sawn timber

Automated quality control is an important feature for the sawmill industry, as produced timber need to be efficiently sorted into different grades. Thus, it is vital to sort obtained planks and boards as accurately and fast as possible [13]. There exist numerous computer vision and machine learning based approaches for a surface inspection and an identification of different defects on sawn timber [14]. However, in terms of computer vision and machine learning systems three general steps can be highlighted as follows:

1. Localization of defects.
2. Feature extraction for defect characterization.
3. Classification of defects.

One of the frameworks for defect detection was built in the research [13] (see Figure 7). The convolutional neural network (CNN) model was applied for analyzing visual imagery reaching more than 92% classification accuracy in detection of the mechanical damages on boards [13].

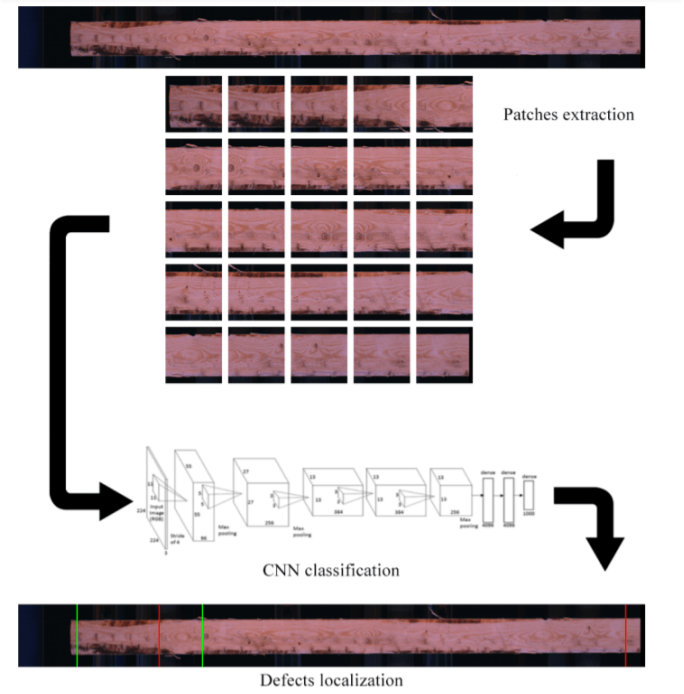


Figure 7. A diagram of the proposed method for the mechanical damages detection from sawn timber images [13].

2.2.2 Wood species identification

Another substantial task in the industry is the identification of species as their mixing can lead to operational costs due to the differences in wood processing between species. Manual verification of boards and planks is impossible for a modern sawmill environment with high production rates. This calls for an automated visual inspection system. The proposed method [15] for the wood species identification is based on CNN achieving almost a perfect accuracy of 99.4% (see Figure 8).

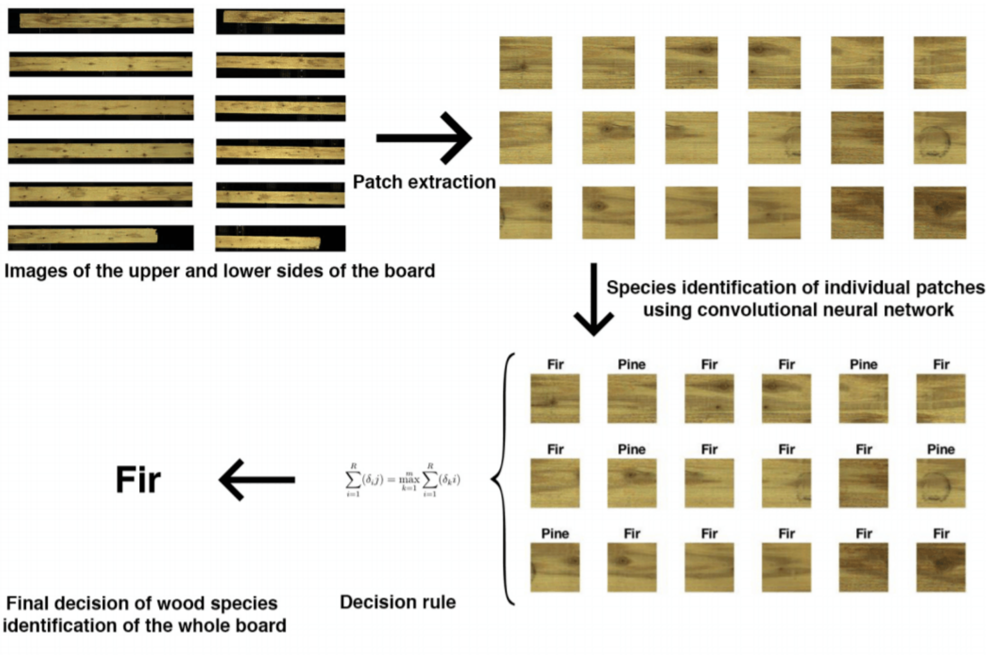


Figure 8. Proposed approach for wood species identification [15].

2.2.3 Timber tracing

Usually, in sawmills, a conversion pipeline is not straightforward and differs from log to log. Therefore, a timber tracing system from the raw material to the end product can be employed for efficient process control, the optimization of sawing, and the prediction of final product quality [16]. The non-invasive board-to-log tracking technique was designed for this purpose in the DigiSaw project. The outline of the solution is based on a convolutional encoder-decoder network and includes the following stages (see Figure 9):

1. Laser point cloud transformation to a heightmap.

2. Application of an encode-decode network to generate "barcode" from an image and an heightmap.

3. Conversion of a "barcode" into a 1-D signal and executing a cross-correlation algorithm to find the matching log.

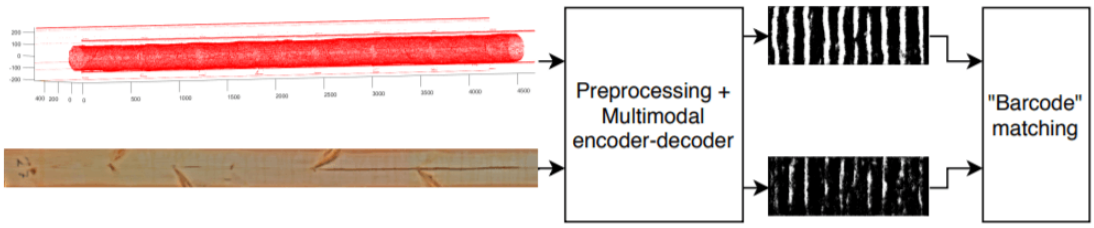


Figure 9. The board-to-log matching process [16].

2.3 Virtual sawing

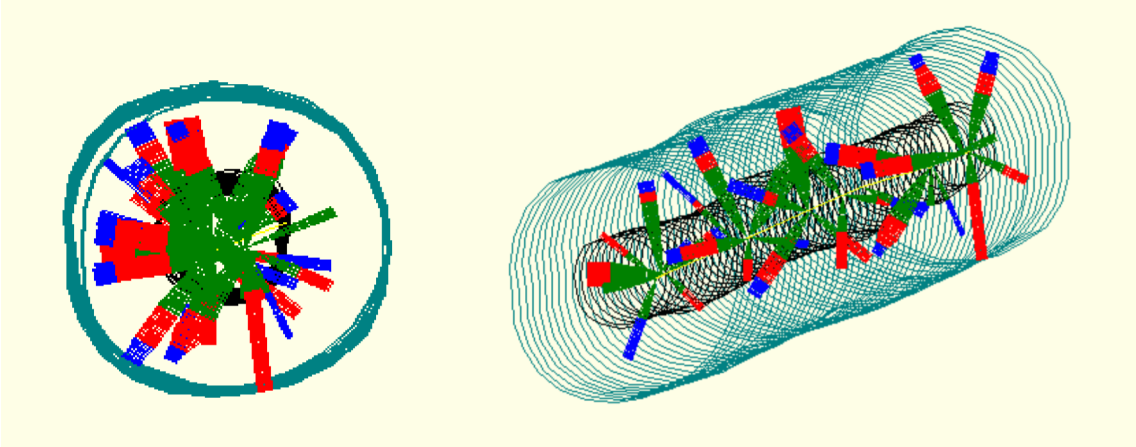
Virtual sawing or sawing simulation is one of the commonly used frameworks for increasing the sawmill yield. Multiple log breakdown simulators have been developed. There are products with a long history of researches and commercial attempts, such as SIMSAW [17] which the first version was released in 1975 at the Council for Scientific and Industrial Research [18]. WoodCIM [19] has been under development at the Technical Research Centre of Finland (VTT) since early 1970. The more recent software InnoSIM [20] was produced by the VTT in 2007, presumably as a replacement and a modification for WoodCIM. SEESAW [21] was initially designed by the New Zealand Forest Research Institute as a research prototype in 1988, and in 1990 system was enhanced into a much more powerful solution called AUTOSAW [22]. RAYSAW [23] was published in 2013 and is based on the surface laser-scanner data and applying a ray-tracing algorithm for virtual sawing.

Commonly, data from an X-ray or a laser surface scanner is used as an input for a sawing simulator. Then obtained information is mathematically reconstructed as a three dimensional (3-D) approximation model with a cylindrical shape, i.e., virtual log. Finally, the model is processed virtually considering the log internal structure. More advanced solutions consider the available sawmill equipment and even desired outputs. Those sophisticated techniques are often represented as a fully autonomous system that is capable to run various scenarios for optimization of sawmill patterns and sawing parameters. However,

even if a system is not fully autonomous, the results can be used as a recommendation by the sawmill employees in order to define the most beneficial sawing pattern. The systems for optimization of sawing patterns are widely employed in the mill environment and proven to have considerable evidence in profitability gain [24, 25].

2.3.1 WoodCIM

The WoodCIM is an integrated optimizing software system developed by VTT [19]. The WoodCIM (as well as InnoSIM) simulates the operations of the whole production chain from stem bucking to end products: timber, wood components, chips, and sawdust [26]. The sawing simulator consists of several modules and calculates the yield by comparing different sawing patterns and set-ups. As an input the programme requires an accurate geometry of logs, quality requirements such as length and grades, sawing dimensions [19]. After sawing a log the software allows reconstructing a log model scanning sawn flitches by the WoodCIM camera system (see Figure 10).



(a) (b)

Figure 10. Mathematical model of log of maritime pine demonstrating the internal structure: (a) 2-D reconstruction; (b) 3-D reconstruction [19].

According to [19], the framework demonstrated potential to solve an optimization problem for a mill as predicted outputs were close to the true outputs when simulations were compared. One substantial drawback of the WoodCIM that is considering the locations of knot inside a log only after an actual process of sawing.

2.3.2 RAYSAW

The RAYSAW [23] is a research tool for virtual log sawing of hardwood based on processing a high-resolution 3-D reconstructed laser-scan data (see Figure 11). The approach is using internal defect prediction models to evaluate locations of the defects on sawn board faces. The key feature of the system is to apply a ray-tracing algorithm for obtaining virtual board images.

Figure 11. A sample point cloud image of a 3-D laser-scanned red oak log [23].

The simulator saws to a predefined pattern. Boards produced by the software can be observed visually with a schematic view, and the results are possible to grade according to the National Hardwood Lumber Association rules (see Figure 12) [23].

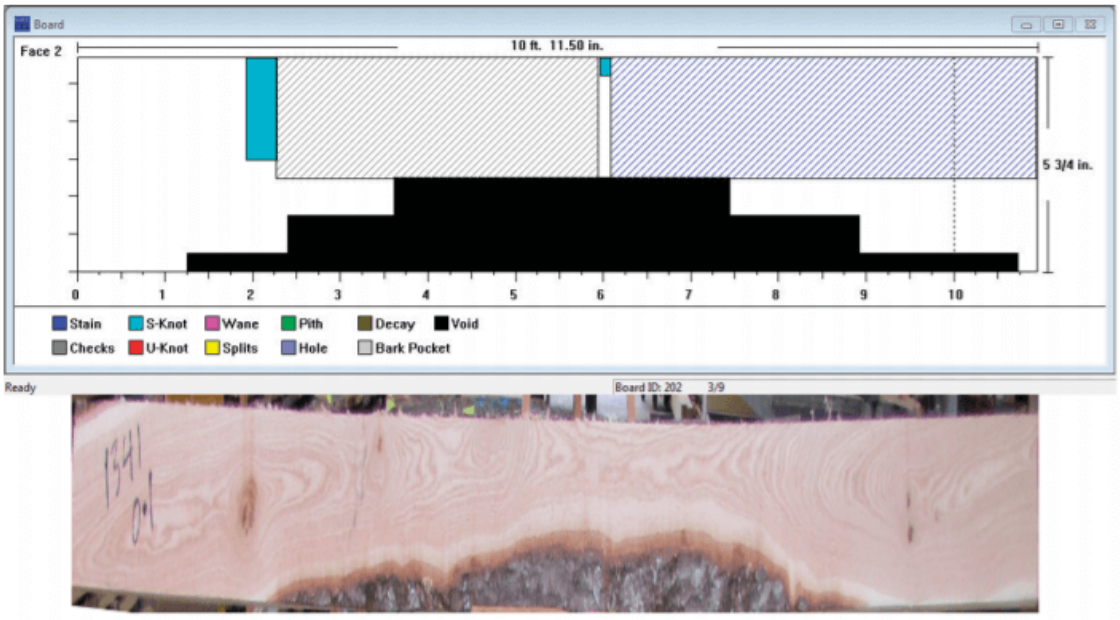


Figure 12. A computer generated and a true board from RAYSAW [23].

2.3.3 DigiSaw

An alternative method proposed in the DigiSaw project [4] estimates the internal structure using only external information, i.e., by a laser surface scanner. The approach allows the employment of more practical and affordable laser range scanners in comparison to expensive or rather low speed computer tomography or magnetic resonance imaging based systems. The framework comprises five major steps as follows (see Figure 2):

1. Point cloud filtering and centerline estimation.
2. Log surface heightmap extraction.
3. Knot segmentation.
4. Volumetric log reconstruction with knots.
5. Virtual sawing.

The final output represents the grayscale image of board faces with the depicted correlation between the probabilities of knot appearing and the pixel intensities in the corresponding locations (see Figure 13). This thesis focuses on the further development of the proposed approach by implementing a more transparent feedback loop that transforms a grayscale knot map into a photorealistic board image with an image-to-image translation GAN based technique (see Figure 3).

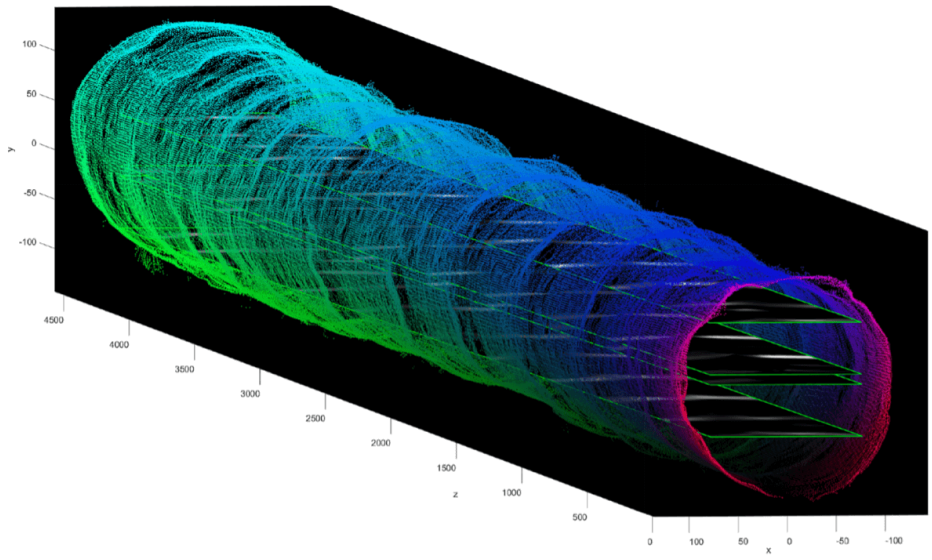


Figure 13. A schematic representation of a virtual log with the generated knot maps inside. [4].

3 GENERATIVE MODELS

The chapter gives an introduction to generative adversarial networks starting with generative and discriminative approaches in machine learning. After this, an outline for convolutional neural networks is given, since it is often used as the main building block for GANs. Next existing deep generative models are shortly discussed. Finally, GANs are examined including structure, application areas, and current state-of-the-art approaches. Several models are discussed in terms of the image-to-image translation task.

3.1 Generative and discriminative model

In contrast to a discriminative approach where the goal is to learn from the underlying distribution and to produce an accurate predictor, a generative approach models that underlying distribution over the data in order to estimate the parameters of the model [27]. More formally, generative classifiers try to learn a model of the joint probability $p(x, y)$ where x are the inputs and labels are described by y, or simply $p(x)$ if there are no labels. Then new samples x can be classified by the Bayes rule and estimated probability distributions $p(y|x)$. Discriminative classifiers directly utilize the posterior probability $p(y|x)$, or learn a direct map from inputs x to the labels y. However, they never solve a more general problem as an intermediate step such as modeling $p(x, y)$ [28].

Consider an art dataset, a certain part painted by Vincent van Gogh and the rest by Salvador Dalí [29] [30]. If a discriminative model is trained, the model would learn that some colors, shapes, textures are more probably to describe the Dutch master and another the Spanish painter. Then in the data space, the discriminative model draws a splitting line by studying the difference between Dali's and van Gogh's paintings. As a result, from a new observation *x* discriminative modeling estimates the probability of belonging to the categories and then outputs the most likely label as shown in Figure 14. A generative model after training is modeling the distribution throughout the data space. The generative model would be capable of solving a more difficult machine learning problem than analogous discriminative models, such as the image generation task (see Figure 15) [29] [30]. To summarize, the discriminative model is able to build a perfect predictor to identify van Gogh or Dali paintings. However, the discriminative model is still incapable to create a new painting that looks like van Gogh, Dali, or hybrid [30], whereas, the generative model can learn and then generate a distribution with the sets of pixels that belongs with a high probability to the training set.

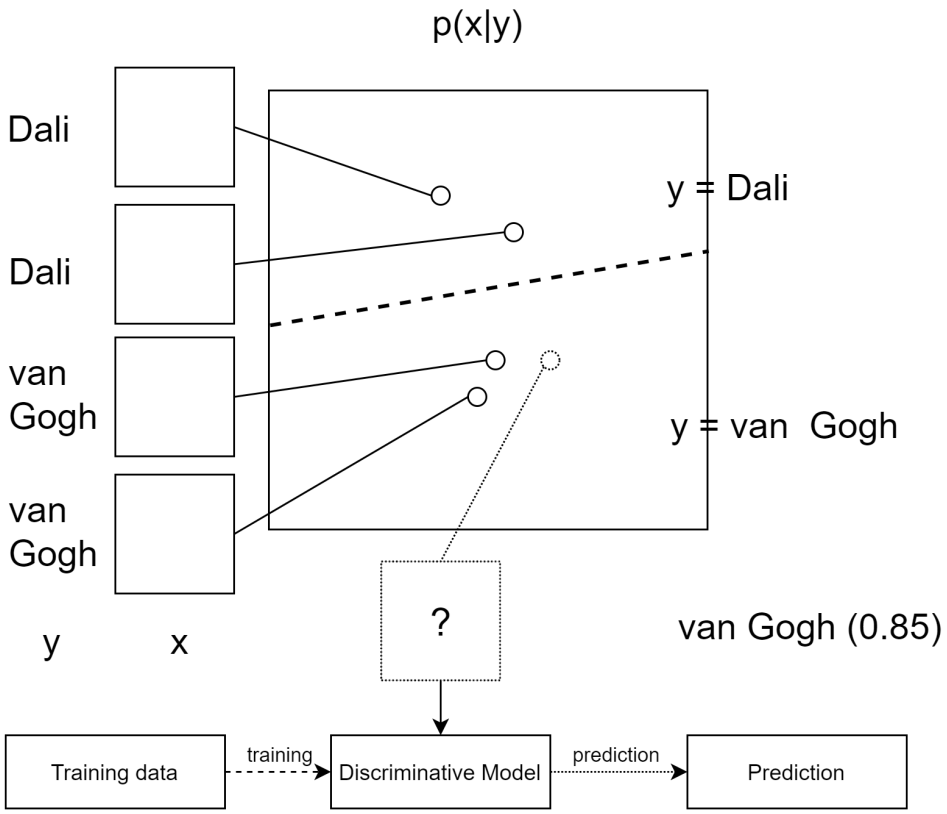


Figure 14. A discriminative modeling process modified from [29] [30].

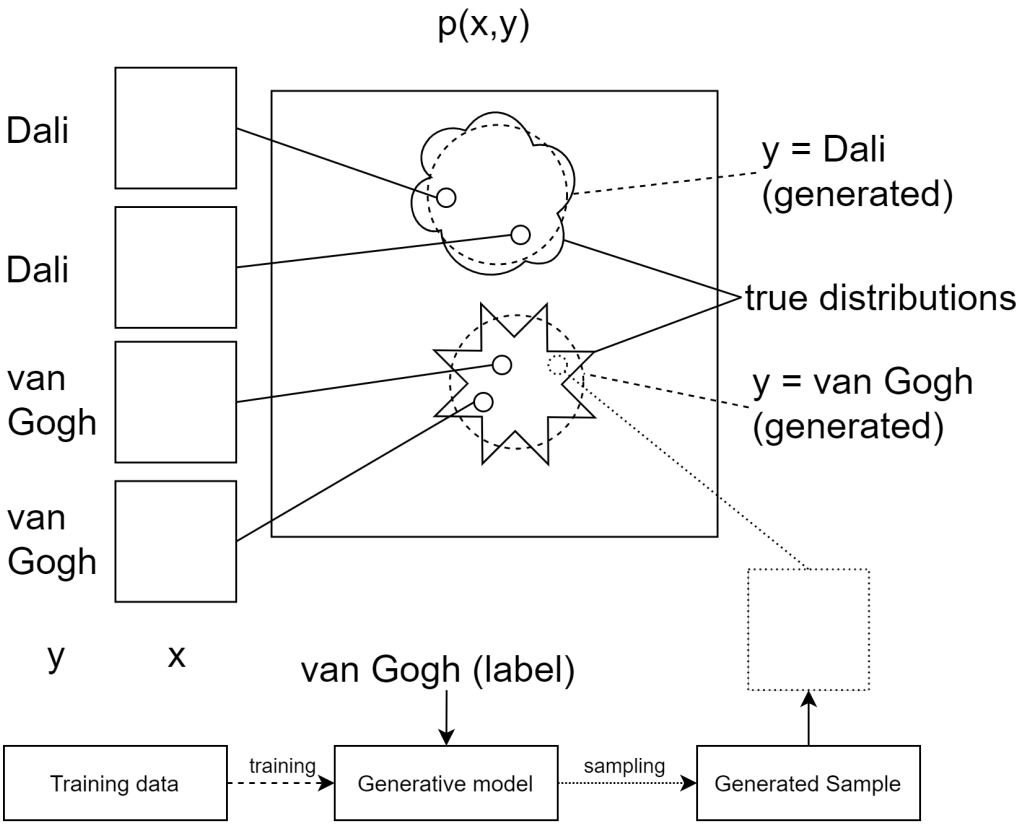


Figure 15. A generative modeling process modified from [29] [30].

3.2 Convolutional neural networks

CNNs are a specialized type of artificial neural network that is especially effective for processing grid-like topology data, e.g., images that are organized in 2-D arrays of pixels [11]. One of the first CNN with the backpropagation algorithm for an image recognition task was implemented by a research group at Bell Labs in 1990 [31]. LeCun et al. proposed the LeNet architecture for the recognition of handwritten postal codes which achieved an error rate of only 1% and a rejection rate of about 9% [31]. Later, CNNs have found tremendous success in practical applications, most commonly in analyzing visual imagery [32].

The "convolutional" in CNNs indicates that the network utilizes an operation called convolution over a general matrix multiplication in at least one of their layers [11]. Behind CNNs are the four important features: local connections, shared weights, pooling, and applying many layers [33]. A classic CNN is a composition of multiple building blocks as follows: (see Figure 16):

- Convolutional layer.
- Pooling layer.
- Fully connected layer.

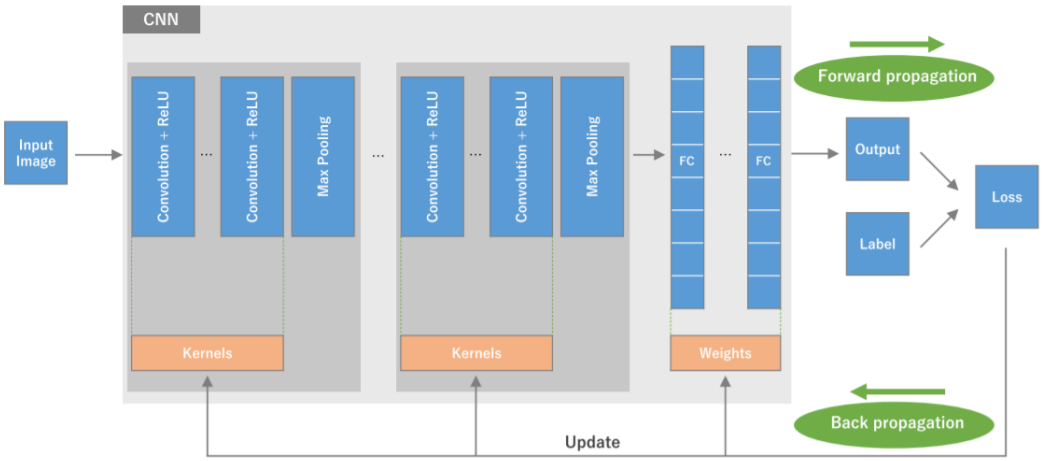


Figure 16. A typical structure of a CNN is formed by stacking the building blocks: the convolutional layer, the pooling layer, and the fully connected layer (FC). A training process of the CNN consists of a forward and a backpropagation algorithms [33].

When a neural network receives an input x and generates an output $\hat{y}$, a signal flows forward over the network [11]. The input x produces the primary signal which then propagates up to the hidden units at each layer and in the end, forms $\hat{y}$. During the training process the forwardpropagation can continue until producing a scalar cost $J(\theta)$, and this denotes forwardpropagation. Whilst, the backpropagation algorithm is used to compute a gradient by the signal from the cost which flows backward through the net [11].

The described CNN structure along with the backpropagation and the forwardpropagation algorithms allows to automatically and adaptively learn spatial features. As a consequence, manual feature extraction and a building stage are skipped. However, due to the previously mentioned factor, CNN is much more computationally expensive and requires an immense amount of training data to adjust typically millions of learnable parameters [34].

Convolutional layer

A convolutional layer is a general feature extraction component of the CNN that includes commonly linear (convolutional) and non-linear operations (activation functions) [11]. The convolutional operation is an element-wise product between a kernel and an input tensor which is calculated for each kernel location in the input tensor. An output tensor is called a feature map. The convolution layer parameters can be defined as follows: kernel size, number of kernels (channels), stride, padding, activation function (see Figure 17) [35]. Then the output tensor (result) is passed over a nonlinear activation function, for example, sigmoid, hyperbolic tangent (TanH), or rectified linear unit (ReLU). The choice of an activation function depends on an application area but presently ReLU is commonly used [34].

Pooling layer

A pooling layer performs a downsampling operation by reducing in-plane dimensionality of the output tensor [34]. With the reduced size of the input images, a number of the following learnable parameters are lowered, computational costs are decreased, and the identified features become more robust [35]. The types of pooling layers are as follows: max/min pooling, average pooling as shown in Figure 18 [34]. The hyperparameters for pooling layer are similar to the convolutional layer and include the pooling method, filter size, stride, padding.

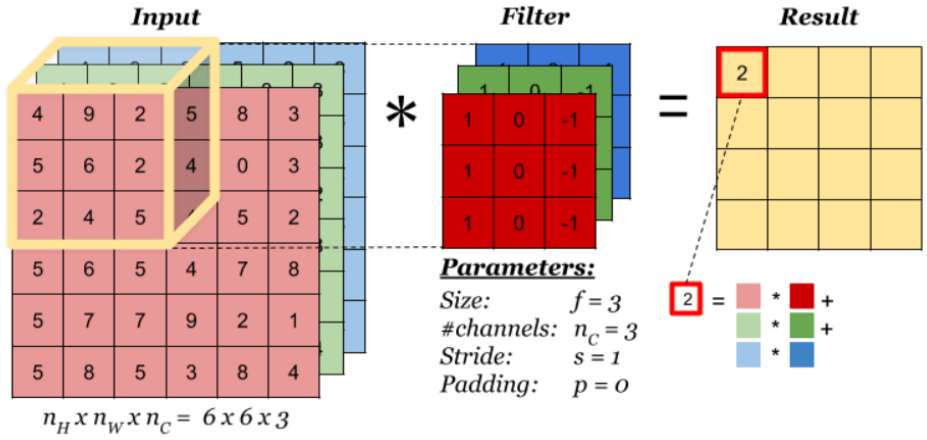
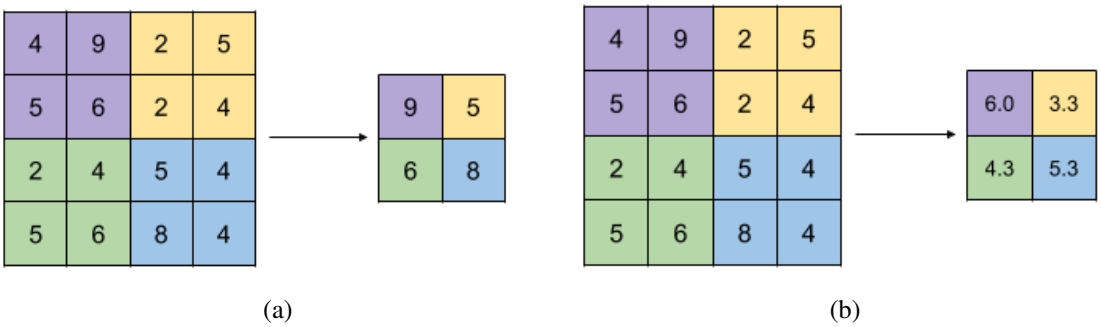


Figure 17. An example of convolution operation with 3 kernels, 3×3 sized, no padding, and a stride of 1 [35].



(a) (b)

Figure 18. Pooling layers: (a) Max pooling filter; (b) Average pooling filter with size 2×2, and stride 2 [34].

Fully connected layer

The last layer is typically transformed into a 1-D array of vectors each connected to a fully connected layer or a dense layer [34]. In the dense layer, each input is connected with each output by a learnable weight. The final layer usually consists of a number of outputs equals to the number of classes in the case of an image classification task. Moreover, an activation function in the fully connected layer is usually different from the others layers and needs to be chosen accordingly to a classification problem. For example, in the case of a binary classification problem sigmoid function is a good choice, while for a multiclass case softmax function is prevalent [34].

3.3 Deep generative models

Generative models can be separated into two groups of machine learning algorithms. The traditional generative models operate with diverse forms of probability density functions approximating the distribution. Infinite Gaussian mixture models [36], hidden Markov models [37], and hidden naive Bayes models [38] are not capable to learn complicated distributions [39]. In contrast, the deep generative models use such methods as stochastic backpropagation, deep neural networks, and approximate of Bayesian inference for producing new samples from variational distributions in large-scale datasets. Such models are deep Boltzmann machine [40], deep belief networks [41], variational autoencoders (VAE) [42], and GANs [43]. The last two techniques become prevalent in recent years as they demonstrate the most efficient results. However, likelihood-free approaches such as GANs outperform VAE models offering clear evidence of prevailing over autoencoders in terms of an ability to infer and to model successfully complicated data distributions for generating quality, realistic, and sharp images [39] [44] [45] [46].

3.4 Generative adversarial network

GANs are a kind of artificial intelligence algorithm introduced by Ian Goodfellow et al. in 2014, designed to solve the generative modeling problem [43] [47]. They have demonstrated a great practical value in realistic data generation tasks, most notably images (see Figure 19).

Figure 19. Evolution of GANs in terms of high quality and realistic image generation [43].

The fundamental goal of a generative model is to learn a probability distribution of a training dataset as stated in Section 3.1. The GAN training process can be described as a game between two machine learning models. The aim of the models is to find a local Nash equi-

librium on each step: a point that is a local minimum of each player's cost with respect to that player's hyperparameters [43]. Generally, deep neural networks, more precisely CNNs, are utilized as a baseline for a typical GAN architecture (see Figure 20) [29]. The first network is called a generator which is able to create samples from the distribution p_{model}. The generator samples come from a prior distribution $p(z)$ with a vector z that is used as input to the generator function $G(z; \theta^{(G)})$ where $\theta^{(G)}$ determines learnable parameters for the generator strategy in the game [43]. The vector z can be thought of as a seed in a pseudorandom number generator and is denoted as noise. The goal of the generator is to learn the $G(z)$ function with parameters $\theta^{(G)}$ that converts the noise z into realistic image instances.

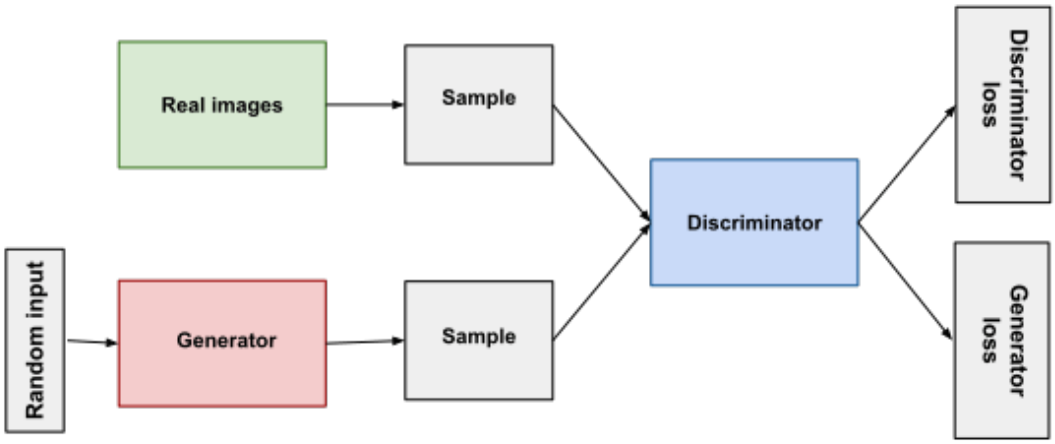


Figure 20. A generative adversarial network structure [29] [43].

The second network is defined as a discriminator which considers produced instances x and returns an estimate $D(x; \theta^{(D)})$ of whether x is a forgery or real [47]. Each network tries to minimize its own cost for a generator $J^{(G)}(\theta^{(G)}; \theta^{(D)})$ and for the discriminator $J^{(D)}(\theta^{(G)}; \theta^{(D)})$. However, the mentioned definition is not simply applicable in terms of model optimization, since each player cost is a function of the other player parameters [43]. The training process also endures several challenges, such as failure to converge, a mode collapse, and vanishing gradient [47]. Nevertheless, GAN is indeed a powerful deep generative model for diverse computer vision related problems: image-to-image translation, image synthesis, text-to-photo translation, image inpainting, and cartoon generation [39] [45] [48].

3.5 Challenges in training process of generative adversarial networks

The fundamental problem during the training process of generative adversarial networks is the issue of a non-convergence [49]. The model might oscillate, destabilize and never converge due to the fleeting nature of a convergence point.

For the most part, an optimization of deep models is searching for the lowest possible value of a cost function. Although optimization algorithms usually provide a reliable movement towards an optimum point, many problems can prevent an optimization. However, in the case of GANs, it is required to find the Nash equilibrium for two networks as stated in Section 3.4. Therefore, even if both networks appropriately move in the direction of the Nash equilibrium during the simultaneous gradient descent on the player's update. The update might move the other network in the opposite course. Thus, the networks are highly sensitive to the hyperparameters and may reach the equilibrium in one scenario, but in other cases. they can over and over recall each other's progress without achieving any successful result [49].

A special case of non-convergence is the mode collapse, also known as the Helvetica scenario [47]. The problem denotes an incident when a generator network trains to convert several different inputs z into the same output point. In Figure 21 the mode collapse issue is illustrated. The top row presents the target distribution expected to be learnt by the model. The lower row describes a series of distributions learnt with a particular time range of training. Instead of converging to the target distribution keeping all of the modes in the training data. The generator network only produces a single mode at a time cycling among different modes, since the discriminator network trains to reject the others [49].

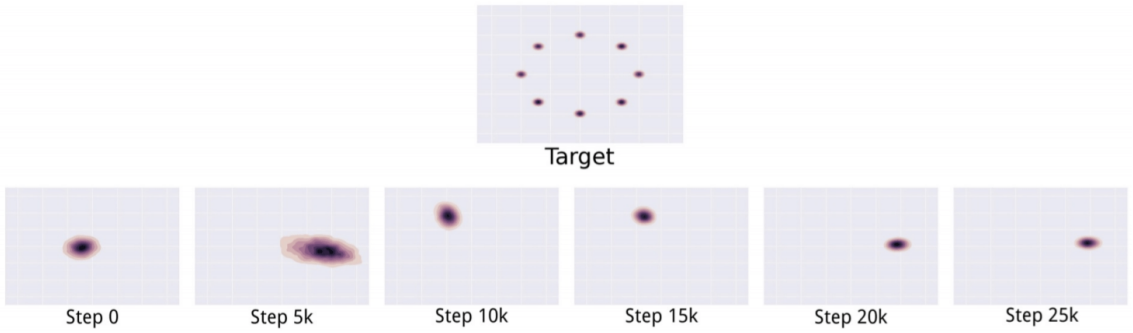


Figure 21. An example of the mode collapse problem [50].

3.6 Image-to-image translation

Various image processing, computer vision, and computer graphics tasks can be considered as the image-to-image translation problem. The idea of image-to-image translation was raised by Hertzmann [51], who described a nonparametric framework for the one-to-one image translation [52]. The image-to-image translation task refers to mapping a source image domain to a target image domain by changing the domain-dependent properties of the source image, for example, colors, but preserving the domain-independent features like content, e.g., season transfer and painting style transfer (see Figure 22) [39].

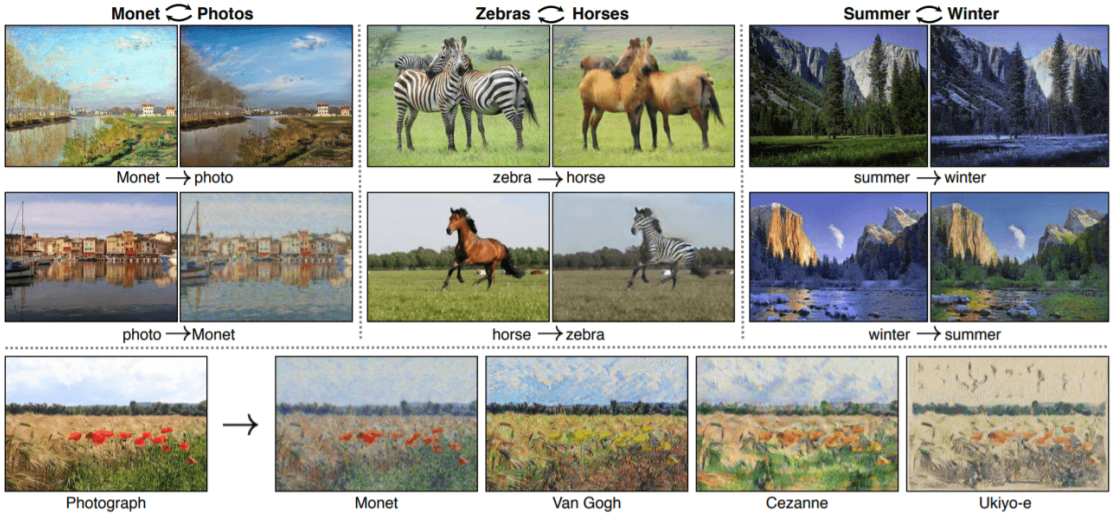


Figure 22. The image-to-image translation using Cycle-Consistent Adversarial Networks [52].

Variants of GANs have grown exponentially since the framework has been proposed in 2014. Researchers are constantly attracted to study GANs and have produced a large number of models due to the special structure and a great generation capability of the approach. However, in terms of the image-to-image translation task might be possible to categorize them as shown in Figure 23.

All the machine learning models depend on the type of training data. The type of training data usually defines an applicable approach: supervised or unsupervised. For the supervised image-to-image translation there is a labeled set of pairs of images (x, y) in domains (X, Y) where each image $x \in X$ is matched with the corresponding image from $y \in Y$. GAN learns a joint probability distribution $P_{\mathrm{X,Y}}(x, y)$ to produce a new sample by mapping from the first domain to the second $(X \rightarrow Y)$ [53]. The supervised approaches can be further divided by an objective function into directional and bidirectional.

The unsupervised image-to-image translation focuses on learning the conversion without aligned pairs in a set of images. With the given marginal distributions $P_{\mathrm{X}}(x)$ and $P_{\mathrm{Y}}(y)$. It is impossible to infer anything for the network since an infinite set of possible joint distributions can be obtained [53]. Different approaches and assumptions are applied which have resulted in the emergence of various unsupervised techniques for the image-to-image translation. The unsupervised approaches can be classified as follows: cyclic consistency-based, autoencoder-based, and methods using a disentangled representation [39]. The unsupervised techniques are beneficial for the ability to map effectively between the domains and to alleviate the challenge of limited training data.

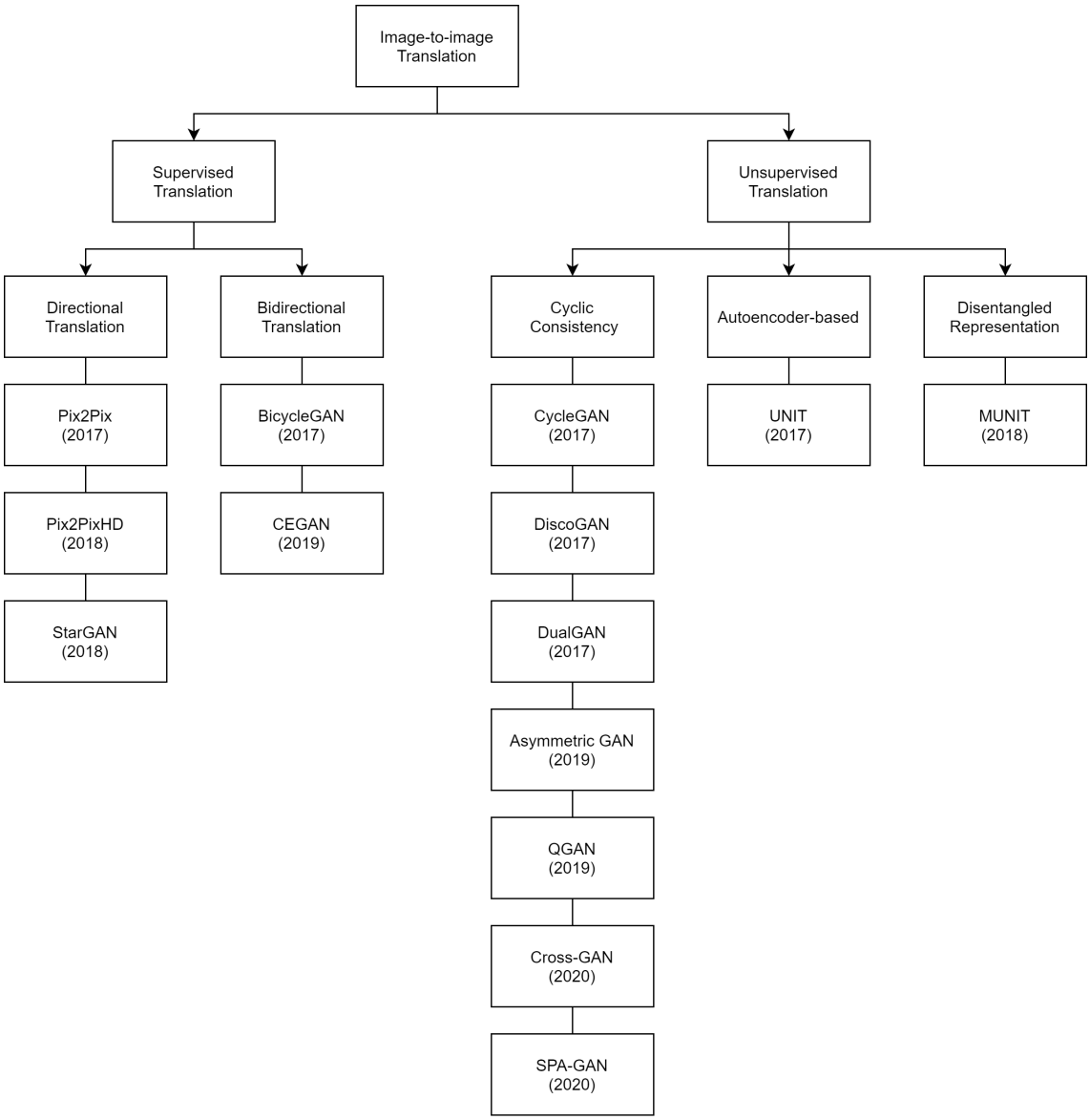


Figure 23. Taxonomy of the GANs modified from [39].

3.6.1 Directional supervised translation

The Pix2Pix [54] is a directional supervised translation method based on a conditional generative adversarial network. A pair of images is utilized for the one-to-one mapping. The training process is depicted in Figure 24.

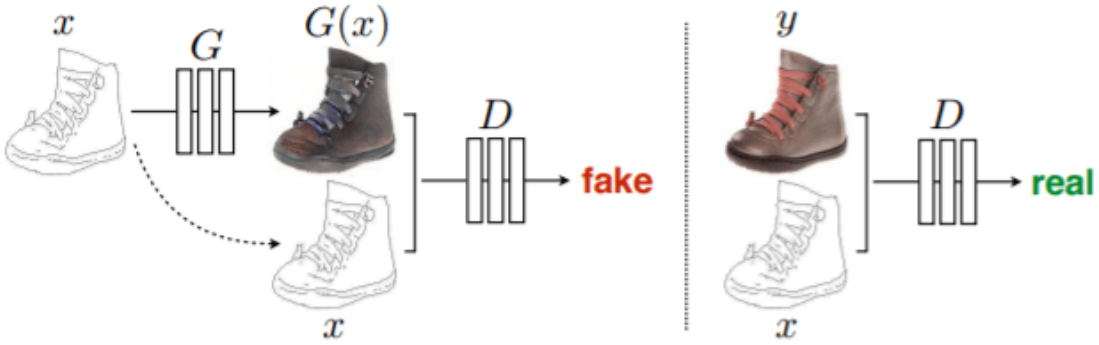


Figure 24. Training a conditional GAN to transform edges→photo. In contrast to an unconditional GAN, both a generator and a discriminator observe an input edge image [54].

The conditional GAN objective is expressed as

$$\mathcal{L}_{\text{cGAN}}(G, D) = \mathbb{E}_{x,y}[\mathbf{log} D(x, y)] + \mathbb{E}_{x,z}[\mathbf{log}(1 - D(x, G(x, z)))], \tag{1}$$

where G is a generator, D is a discriminator, x, y, and z are the input image, output image, and noise accordingly. G attempts to minimize this objective in opposition to adversarial D that tries to maximize it. In the Pix2Pix a traditional loss is also added to the adversarial loss. The $L1$ loss is described as follows

$$\mathcal{L}_{\text{L1}}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1]. \tag{2}$$

The final objective is a combination of the two losses, the adversarial and the traditional

$$G^* = \arg \min_G \max_D \mathcal{L}_{\text{cGAN}}(G, D) + \lambda \mathcal{L}_{\text{L1}}(G), \tag{3}$$

where λ is a hyperparameter defining an amount of the traditional noise. The images generated with the Pix2Pix [54] network are quite limited due to low resolution and blurriness. Later, the Pix2PixHD [55] architecture was proposed in order to increase the resolution of the output samples to 2048x1024 and make them more realistic by utilizing a multiscale generator and a discriminator. The Pix2Pix and the Pix2PixHD are limited to operate only in two domains, or in the case of a multidomain image-to-image transla-

tion, their architectures require several separate instances for each additional domain. The StarGAN [56] was proposed to address the limitation with a unified model architecture for a multidomain image-to-image translation (see Figure 25).

(a) (b)

Figure 25. Comparison between a common cross-domain architecture and the StarGAN architecture: (a) A classic GAN architecture requires additional instances for every pair of image domains; (b) The StarGAN architecture is capable to learn multidomain transformations using a single generator [56].

3.6.2 Bidirectional supervised translation

The BicycleGAN [57] is a multimodal and crossdomain translation method proposed as an attempt to mitigate a mode collapse, occurring in the Pix2Pix models by encouraging a bijective mapping between a latent encoding space and output modes. It is a hybrid approach with a baseline from the Pix2Pix model. There are two networks: the conditional Variational Autoencoder GAN (cVAE-GAN) and the conditional Latent Regressor GAN(cLR-GAN). The compound technique generates realistic and diverse images in comparison to the previously discussed Pix2Pix (see Figure 26) [57]. The objective function combines two objective functions:

1. cVAE-GAN objective function

$$G^*, E^* = \arg\min_{G,E}\max_{D} \mathcal{L}_{\text{GAN}}^{\text{VAE}}(G, D, E) + \lambda\mathcal{L}_{\text{L1}}^{\text{VAE}}(G, E) + \lambda_{\text{KL}}\mathcal{L}_{\text{KL}}(E). \quad (4)$$

2. cLR-GAN objective function

$$G^*, E^* = \arg\min_{G,E}\max_{D} \mathcal{L}_{\text{GAN}}(G, D) + \lambda_{\text{latent}}\mathrm{L}_{\text{L1}}^{\text{latent}}(G, E), \quad (5)$$

where λ, λ_{latent}, and λ_{KL} are the hyperparameters to control the relative importance of each term.

The final objective function is described as follows

$$G^*, E^* = \arg\min_{G,E}\max_{D} \mathcal{L}_{\text{GAN}}^{\text{VAE}}(G, D, E) + \lambda\mathcal{L}_{\text{L1}}^{\text{VAE}}(G, E) + \mathcal{L}_{\text{GAN}}(G, D) + \lambda_{\text{latent}}\mathcal{L}_{\text{L1}}^{\text{latent}}(G, E) + \lambda_{\text{KL}}\mathcal{L}_{\text{KL}}(E). \tag{6}$$

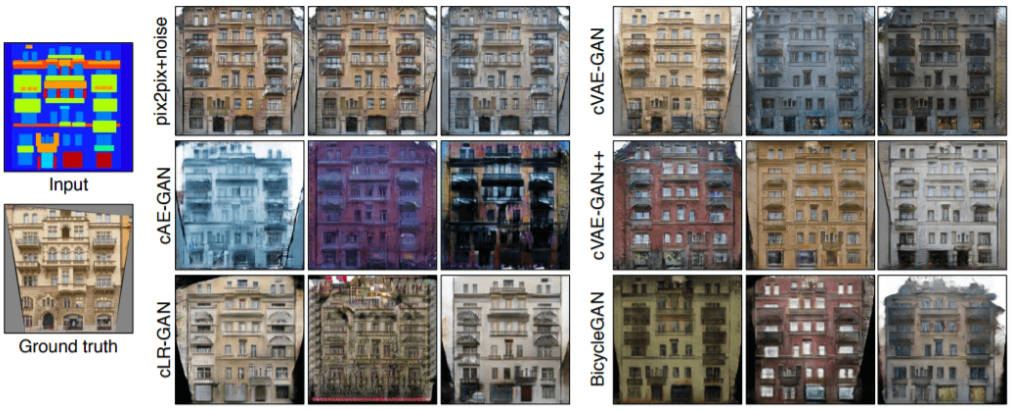


Figure 26. The BicycleGAN method produces results which are both realistic and diverse [57].

The method of Consistent Embedded GAN (CEGAN) [58] is another attempt based on a conditional generation model to tackle the problem of perceptually realistic and diverse images. The CEGAN learns the distribution by enforcing tight connections between the latent space and the real image space. The model consists of a generator, a discriminator, and an encoder. With the encoder and the discriminator, this model distinguishes the real and the fake samples in the latent space instead of the real image space in order to moderate the impact of redundancy and noise for generating realistic results [58].

3.6.3 Unsupervised translation with cycle consistency

The objective of the unsupervised image-to-image translation can be defined as follows: to learn a mapping $G : X \to Y$ such that the output $\hat{y} = G(x), x \in X$ is not distinguishable from the $y \in Y$ with an adversarial loss. As that transformation is highly constrained and leads to the mode collapse, in the CycleGAN [52] it was proposed to couple it with an additional translator $F : Y \to X$ that is an inverse mapping of the $G : X \to Y$ (see Figure 27).

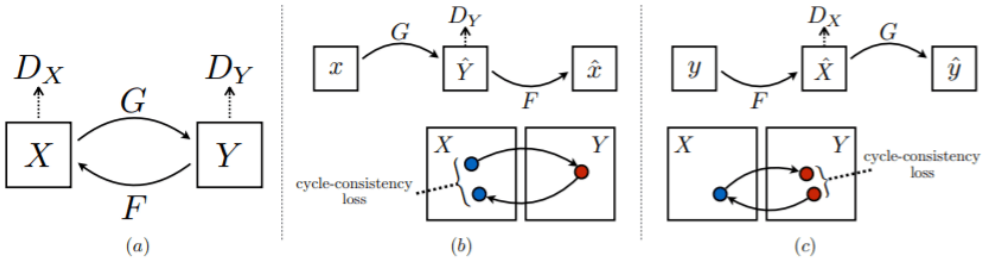


Figure 27. The CycleGAN model: (a) CycleGAN architecture; (b) Forward mapping; (c) Backward mapping [52].

In addition, two associated adversarial discriminators D_{Y} and D_{X} were introduced. D_{Y} supports G to translate X into results indistinguishable from domain Y, and backward for D_{X} helps F to map from the Y domain into the X domain. The authors introduced two cycle consistency losses:

1. The forward cycle consistency loss

$$x \to G(x) \to F(G(x)) = \hat{x}. \tag{7}$$

2. The backward loss cycle consistency loss

$$y \to F(y) \to G(F(y)) = \hat{y}. \tag{8}$$

The cycle consistency loss is expressed as follows

$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{x \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1]. \tag{9}$$

Then the final loss includes the adversarial losses for both mapping functions as the cycle losses

$$\begin{aligned}\mathcal{L}(G, F, D_{\mathrm{X}}, D_{\mathrm{Y}}) = \mathcal{L}_{\text{GAN}}(G, D_{\mathrm{Y}}, X, Y) + \mathcal{L}_{\text{GAN}}(G, D_{\mathrm{X}}, Y, X) \\ +\lambda \mathcal{L}_{\text{cyc}}(G, F).\end{aligned} \tag{10}$$

Finally, the main objective function to solve is

$$G^*, F^* = \arg \min_{G,F} \max_{D_X, D_Y} \mathcal{L}(G, F, D_X, D_Y). \tag{11}$$

Based on the CycleGAN various unsupervised image-to-image translation models emerged, e.g., the DiscoGAN [59], the DualGAN [60]. The method achieves good results on many

translation tasks including object transfiguration, season, and style transfer. However, the CycleGAN can only perform a one-to-one domain transition. Also, it fails when geometric conversions are essential [45].

Recently, for unpaired training data, Asymmetric GAN [61] was proposed. This asymmetric framework presents an auxiliary variable to resolve the issue of imbalanced information in case of transformation from the information-poor domain to the information-rich domain and vice versa. As a consequence, the solution outperforms the CycleGAN in image quality and diversity. In the unified quality-aware GAN (QGAN) [62] approach a quality loss function was developed for the unpaired image-to-image translation. The QGAN architecture compounds two methods of the quality-aware framework: a classical quality assessment loss, and a high level adaptive visual content structure loss as an addition to the adversarial loss [62]. The Cross-GAN architecture [63] consists of two adversarial autoencoders. It employs semantic consistency loss to capture the common feature representation of the source and target domains, and to learn the joint feature-space information rather than insufficient pixel-level data. The spatial Attention GAN (SPA-GAN) [64] introduces a spatial attention mechanism for the unpaired image-to-image translation. The SPA-GAN computes the spatial attention map directly in the discriminator and returns it back to the generator. Such an algorithm is used to focus the generator on the most discriminative areas between two domains. Also, an additional feature map loss is implemented to sustain domain specific features during the image conversion [64].

3.6.4 Unsupervised translation with autoencoder-based models

An autoencoder contains both an encoder that transforms the source images into a latent space and a decoder for inverse operation. The compressed vector can be used as an input to the generator in order to produce high-quality images [39].

The UNIT approach [53] combines generative adversarial networks (the CoGAN [65]) and VAE. The UNIT framework includes two encoders, two generators, and two discriminators (see Figure 28). The weight-share constraint is used in adversarial training to enforce the shared latent space and to generate corresponding images for source and target domains. However, the UNIT requires that the two images obtain analogous patterns for a good performance [45]. In addition, the Gaussian latent assumption is able to produce a unimodal output, on the one hand, while, on the other hand, training may be unstable due to the saddle point (floating state of convergence) [45].

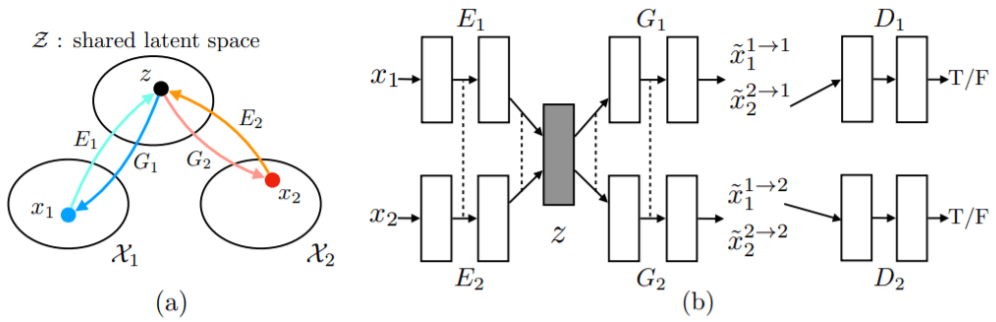


Figure 28. The UNIT framework: (a) The shared latent space assumption: corresponding images (x_1, x_2) in two different domains X_1, X_2 can be mapped to a same latent code z in a shared-latent space Z. E_1, E_2 are two encoding functions, mapping images to latent codes. G_1 and G_2 are two generation functions, mapping latent codes to images; (b) The network structure [53].

3.6.5 Unsupervised translation with the disentangled representation

The ability to understand high-dimensional unsupervised data and to transform them into useful representations remains an essential problem in deep learning. One approach to solve these challenges is through disentangled representations. The models with disentangled representations capture independent features of a given scene in such a way that if one feature changes, the others remain unaffected [66]. For example, each feature can be disentangled into narrowly defined vectors and encode as separate dimensions [67]. The MUNIT framework [68] assumes that an image latent space is possible to decompose into a content code that is domain-invariant and a style code that refers to domain-specific parameters. It is also assumed that images in different domains share a common content space but not the style space (see Figure 29) [68]. Then the MUNIT applies two autoencoders, and the latent code of each autoencoder is factorized into the content code C_i and the style code S_i.

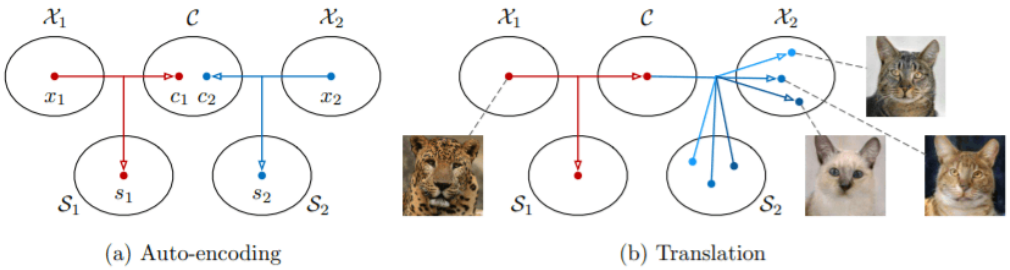


Figure 29. The MUNIT approach: (a) Images of each domain X_i are encoded to a shared content space C and a domain specific style space S_i. Each encoder obtain an inverse decoder (not depicted); (b) To translate an image from X_1 (a leopard) to X_2 (domestic cats), we recombine the content code of the input with a style code in the target style space. Different style codes lead to different outputs [68].

3.7 Summary

A quick overview on state-of-the-art techniques for image-to-image translation tasks would not give complete and comprehensive information to select the model for virtual sawing. However, with the described characteristics of GAN methods (see Table 1) and a limited amount of the available training data [4], some architectures are more suitable architecture than others.

First of all, the data is an aligned (paired) set of images. In addition, no multi-domain and unified structure are required as there are only two domains, as well as no multi-modal property is needed, the output should be deterministic. A bidirectional translation is also a redundant attribute since the study focuses only on the mapping to photorealistic images. Following the Occam's razor rule the Pix2Pix and the BicycleGAN are the most preferable approaches. Nevertheless, it can be beneficial to adapt properties from the more advanced architectures of GANs such as shared feature space and feature disentanglement to avoid a possible mode collapse. Finally, it is not possible to make the right decision from the beginning and only the results from experiments would be a determining factor in a practical sense of the research.

Table 1. Comparison of GANs methods [39].

Method / Property	Pix2Pix	BicycleGAN	StarGAN	CycleGAN	UNIT	MUNIT
Unpaired images	−	−	+	+	+	+
Multi-domain	−	−	+	−	−	+
Multi-modal	−	+	−	−	−	+
Unified structure	−	−	+	−	−	−
Bidirectional translation	−	+	+	+	+	+
Shared feature space	−	+	−	−	+	+
Feature disentanglement	−	−	−	−	−	+

4 IMAGE-TO-IMAGE TRANSLATION TO VIRTUAL SAWING

4.1 Pipeline

The main objective of the study is to create a supplementary GAN based method for the existing virtual sawing framework [4]. The existing system allows optimizing the density and distribution of knots on a board prior to sawing parameters without an actual process of log sawing which aids in increasing the grade of the final product, and thus, the overall profitability of a sawmill. The approach proposed by Zolotarev et al. translates point cloud data from laser range scanners to knot maps [4]. Additionally, the raw projected heightmap images are considered to explore the capability of an image-to-image translation method. The new supplementary module should be able to generate photorealistic board images from both grayscale knot map images and raw projected heightmap images introducing a more transparent feedback loop for a sawmill operator as shown in Figure 30.

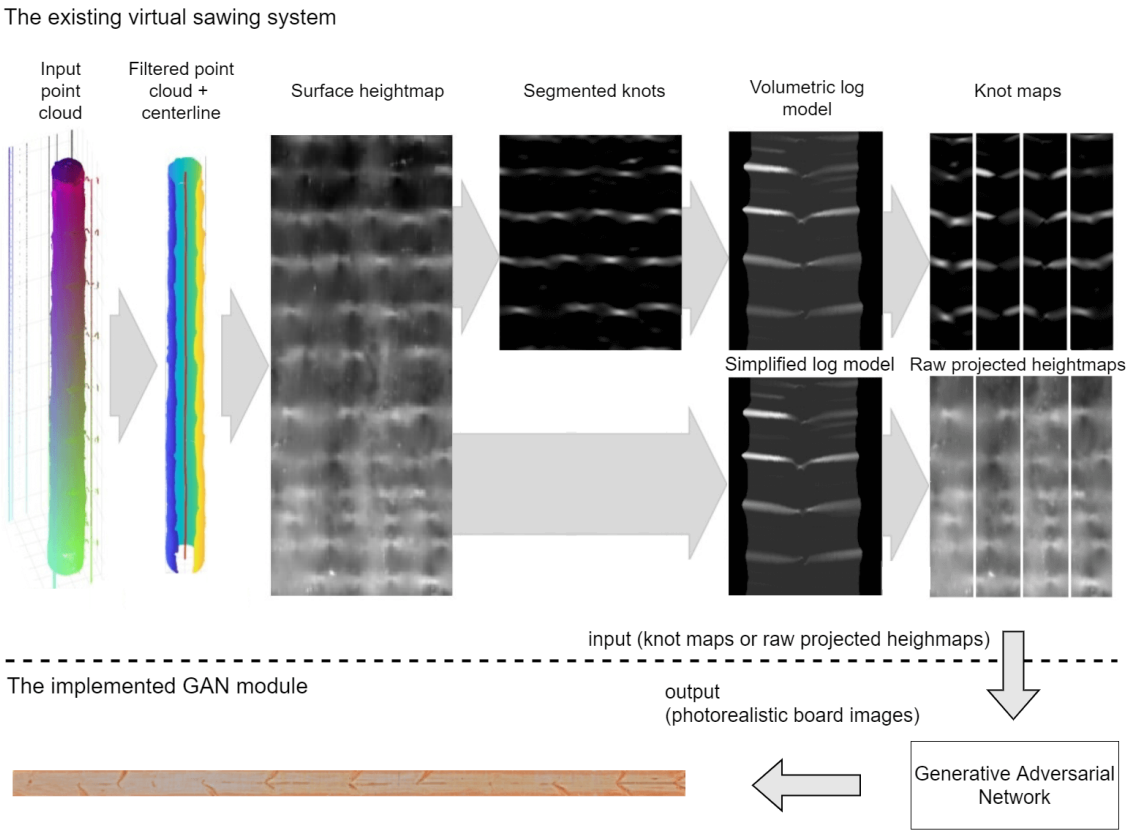


Figure 30. A complete pipeline for generation photorealistic board images.

4.2 Log measures-to-image translation

4.2.1 Knot map generation

The existing virtual sawing method is divided into the following five steps as illustrated in Figure 30:

1. Point cloud filtering and centreline estimation.
2. Log surface heightmap generation.
3. Knot segmentation.
4. Volumetric reconstruction of knots.
5. Virtual sawing.

Point cloud filtering and centerline estimation

In the first step of the measures-to-image translation, the point clouds extracted from laser range scanners are filtered to reduce the noise and potential artifacts such as metal rails holding the log. The filtering process is done layerwise where each cross section layer can be described as a point clouds on the same plane of z coordinate. The z axis is pointing up on the first and the second steps of the method depicted in Figure 30. Then a circle is fitted in order to get an estimation of a centerline. The clusters in each layer are filtered using the fitted circle and least square algorithm as shown in Figure 31.

Log surface heightmap generation

Afterward, the filtered point cloud data is transformed into a log centric coordinate system with a discretization step defined as $0.1N$, where N is the number of cross sections in a log. The obtained heightmap data and the relevant coordinate transformation are shown in Figure 32. The centerline (blue dashed line) is discretized into segments (red solid line). The new coordinates are estimated separately for each cross section along z axis where θ is the angle around the log, l is the position along the length of a log, and ρ is the distance to the centerline.

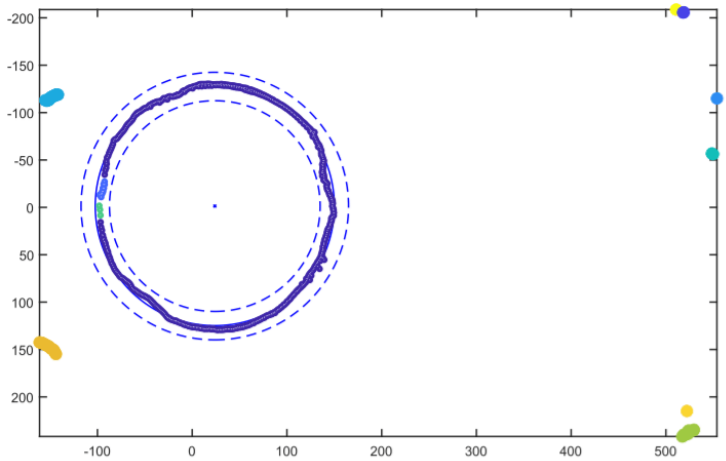
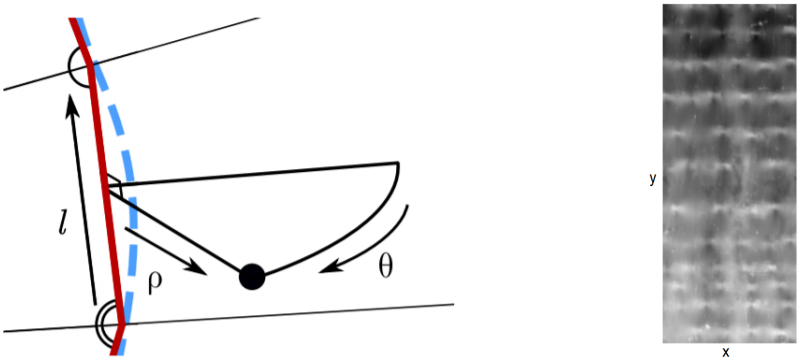


Figure 31. Filtering of a cross section layer and centerline estimation with circle fitting method. The fitted circle is plotted as a solid line, the least square thresholds are plotted with dashed line. The points outside of the two dashed lines are considered as a noise [4].

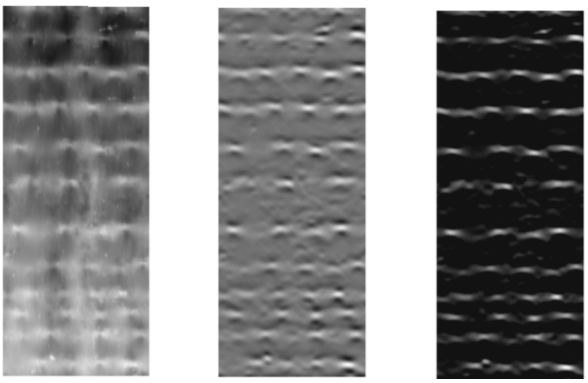


(a) (b)

Figure 32. The new coordinate system and the corresponding dimensions of a raw heightmap map image: (a) The transformation into the log centric coordinate system; (b) A raw heightmap image where x axis on the heightmap denotes the θ (an angle around the log), y axis defines a length of a log, and pixel intensities spacing between surface and a centerline point or ρ [4].

Knot segmentation

The third step, knot segmentation, applies Laplacian of Gaussian filtering (LoG) on the raw heightmap data, further all the negative pixel values from LoG filtered heightmap are set to zero (see Figure 33). As a result, the pixel intensities in the segmented knot locations correspond with a probability of a real knot being there.



(a) (b) (c)

Figure 33. The method of the conversion a raw heightmap image into a segmented heightmap image: (a) The raw heightmap; (b) The LoG filtered heightmap; (c) The segmented heightmap. [4].

Volumetric reconstruction of knots

The mapping of the segmented 2-D surface heightmaps into the inside of the log is possible using information about the biological properties of knot growth. For the extension, the simplified approximation of a full knot model is used. The main characteristics are the following:

1. Knot radius change: knots are rapidly growth during the first several stages of their life.
2. Knot vertical position: knots grow at a certain angle.

For modeling the knot vertical position [69]

$$z = (Z_H - C_{\rho_{max}}) \left(1 - e^{\frac{B}{\rho_{max} - \rho}}\right) + C_\rho \tag{12}$$

is applied where Z_H is the vertical distance between the knot location on the surface and its starting point, z is the vertical displacement at a distance ρ from the centerline, ρ_{max} is the maximum distance from the surface to the centerline, and B, C, are supplementary model parameters specific for the wood species [4]. For the knot radius

$$r = \frac{1}{2}\left((2r_{surface} - G_{\rho_{max}}) \left(1 - e^{\frac{F}{\rho_{max} - \rho}}\right) + G_\rho\right) \tag{13}$$

is applied where $r_{surface}$ is the radius of the knot at the surface, r is the radius of a knot at a distance l from the centerline, and F, G are model supplementary parameters specific for the wood species [4].

Rather than modeling each knot separately, model parameters remain constant for all knot intensity maps. As a result, instead of acting as functions $z(\rho)$ and $r(\rho)$ of the distance from the centre, 2-D maps in the stack are translated along l by $z(p)$ mm, simulating the vertical displacement of knot growth. The growth is simulated by morphological operations. The complete procedure of applying the growth model to a stack of heightmaps as a scheme is illustrated in Figures 34(a)– 34(c), then the stack of knot intensity maps transformed from the log centric coordinates to the Cartesian coordinates in order to receive the volumetric reconstruction of the knot growth intensity inside the log as depicted in Figures 34(d)– 34(e).

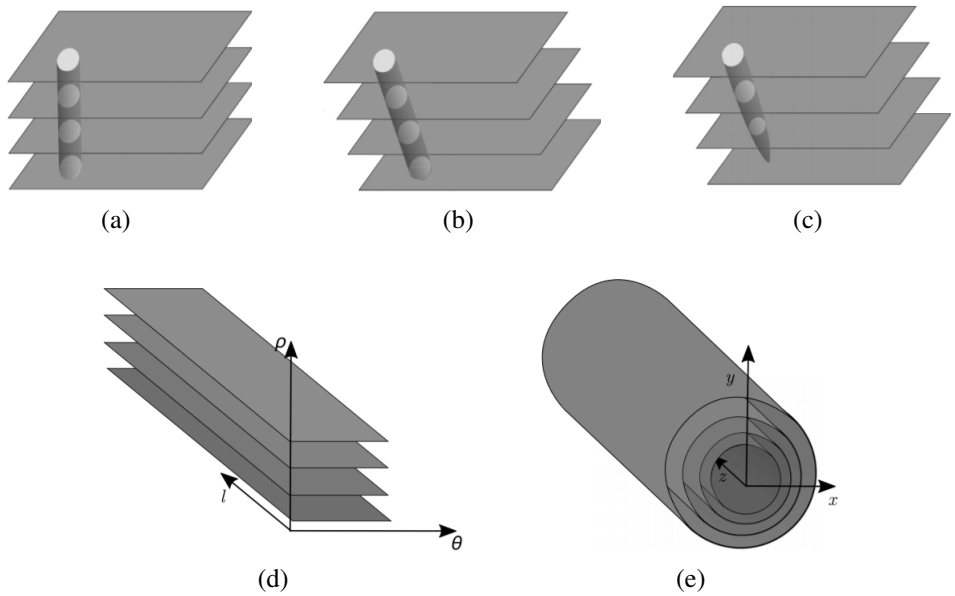


Figure 34. The full pipeline of the volumetric reconstruction of knots inside a log: (a) stacked initial heightmaps; (b) Shifted heightmaps; (c) Eroded and shifted heightmaps; (d) Stacked log centric heightmaps; (e) Stacked cartesian heightmaps [4].

Virtual sawing

Finally, virtual sawing can be performed. The process of virtual sawing is approximating the pixel intensity values accordingly to the placement of the boards inside the log (see Figure 35). The system possesses five degrees of freedom: α, is the sawing angle, rotation around the sawing axis, and x_1, y_1, x_2, y_2 define coordinates p_1 and p_2. The measurements of the sawn logs are estimated with predefined sawing patterns used during the actual process of sawing. The approximate measurements are calculated by fitting a sawing pattern in the center of a fitted circle produced at the stage of filtering and centerline estimation.

However, if a log is too curved, the following sawing coordinates can possibly lie outside of a log. In this case, the sawing coordinates are translated, such that the outliers are located inside the estimated log boundary. The sawing angles for each log were measured using videos of the log entering the sawing machine. Hence, the approximate error is significant, up to the 10°.

The result of virtual sawing is a set of images, each corresponding to a side of a board (see Figure 35). Pixels intensity values in images are correlated with the probability of a knot appearing in a specific location on the real board.

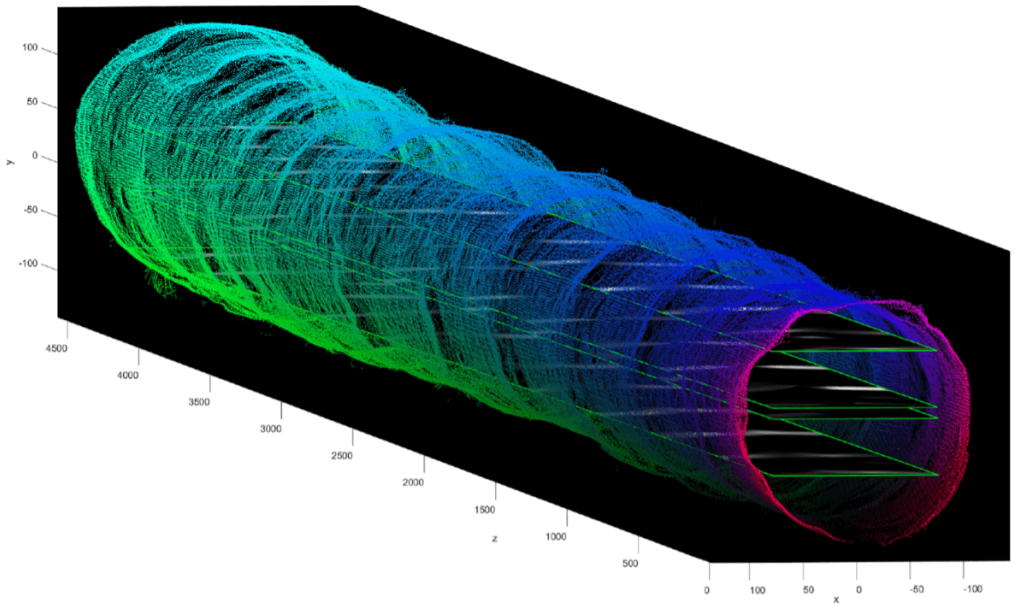


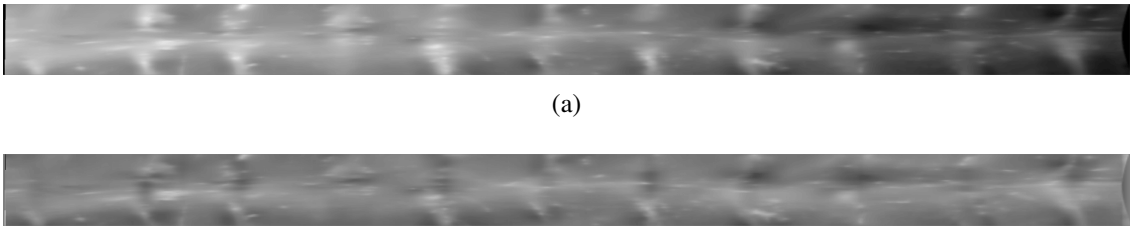
Figure 35. A schematic representation of a virtual log with the generated knot maps inside. [4].

4.2.2 Raw projected heightmap generation

An alternative input is considered to test the CNNs capability for extracting all the necessary information directly from raw heightmap data. If the GAN would able to generate photorealistic board images from such input, then the ad-hoc solution for knot segmentation and modeling biological properties can be avoided.

The raw projected heightmap images are produced using exactly the same method but the stage of knot segmentation is skipped and the simplified volumetric reconstruction is applied which does not simulate the knot growth. The pipeline for raw projected heightmap images is shown in Figure 30. Additionally, for the raw projected heightmap images,

a mean subtraction is performed. This helps to dissolve a homogeneous inclination of values among the board (see Figure 36). Nonuniform scaling of values among the board surface can force the network to learn only from the brighter part of a raw projected heightmap image.



(a)

(b)

Figure 36. The inclination of a raw projected heightmap image dissolves with mean subtraction: (a) A raw projected heightmap image with inclination; (b) A homogeneous raw projected heightmap image.

4.3 Board image generation using GAN based image-to-image translation

This section introduces the suggested extension for the translation of the produced knot maps or the projected heightmap images into realistic looking board images. The study proposes to apply an existing state-of-the-art architecture of GAN for image-to-image translation. From a large number of various architectures, the Pix2Pix model is selected as it is successfully employed for various image-to-image translation tasks and datasets. Two models for virtual sawing are proposed:

1. Pix2Pix model for patched board images.
2. Pix2Pix model for full board images.

The first model utilizes the vanilla version of the Pix2Pix model. For the second model, the vanilla Pix2Pix architectures are not applicable as it is, because the original structure of the GAN requires an input image to be cropped into patches. The initial model was built to operate with image data in which width and height dimensions are equal, i.e., representing a square and invariable within the dataset. Whereas, the size of the boards in the existing dataset is rectangular and changing from one instance to another. This calls to rebuild the vanilla architecture of the Pix2Pix GAN.

4.3.1 Specifications of the Pix2Pix model

In this section, detailed specifications on the defined Pix2Pix loss functions are revealed. The objective of the Pix2Pix conditional GAN is set with Eq. 1. It is possible to train the network without noise z, but the transformation of an input x to y in this case would be deterministic. Hence, the model would fail to match any distribution other than a delta function. The commonly applied Gaussian noise is also ineffective since a generator in the Pix2Pix architecture simply learns to ignore the noise. To address this problem the Pix2Pix framework uses the noise in the form of a dropout applied for the several first upsampling layers in the generator [54]. The final GAN objective is a mix of the GAN objective and a traditional loss function, i.e., $L1$ or $L2$ distance. Isola et al. explored both options and found out that using $L1$ over $L2$ distance promotes less blurring. Therefore, the final objective remains unchanged as it is designated in Eq. 3. The hyperparameter in the final objective function defines a multiplier for the $L1$ loss. Setting it to 0 gives a much sharper output image but injects visual artifacts. The inference from each loss component can be observed in Figure 37.

Figure 37. Different losses produce different results. Using only $L1$ generates blurry images, while solely utilization of a conditional GAN loss introduces artifacts to an images. $L2$ loss was not considered, since blurring of an output image is stronger than with $L1$ loss [54].

In conclusion, the objective function, the loss function, and the hyperparameter remain unchanged for the virtual sawing model as proposed in the study by Isola et al. [54]. The Pix2Pix vanilla network to update weights utilizes the Adaptive Moment Estimation (Adam) optimization algorithm which is a common and widely applied method in ma-

chine learning. The learning rate in Adam algorithm is adopted from the original paper and is set as 0.002. The other parameters are default for the Adam optimizer, the β_1 is 0.5, theβ_2 is 0.999. Finally, using conditional objective function previously defined in Eq. 3 the GAN training procedure can be defined as shown in Algorithm 1.

Algorithm 1 Training of the Pix2Pix GAN

INPUT: A set of input images (gray-level knot maps) X
a set of target images (RGB real boards) Y
a discriminator network D
a generator network G
a number of training epochs N
a number of batches in the splitted sets B
an objective function F
OUTPUT: a trained GAN, includes G and D

for number of training epochs N **do**
  for number of batches in datasets B **do**
    for every input and target image in a batch B_i **do**
      pick an image x_i from batch B_i of input set
      pick an image y_i from batch B_i of target set
      produce a generated image $\hat{x}$ with $G(x_i)$
      calculate a discriminator real output d_{real} with $D(x_i, y_i)$
      calculate a discriminator generated output d_{gen} with $D(x_i, \hat{x})$
      calculate a discriminator loss $F(d_{real}, d_{gen})$ (see Eq. 3)
      calculate a generator loss $F(d_{gen}, \hat{x}, y_i)$ (see Eq. 3)
      update the D weights using Adam optimization method
      update the G weights using Adam optimization method
    end for
  end for
end for

4.3.2 Pix2Pix model for patched virtual sawing

Generator architecture

In the Pix2Pix GAN, a modified encoder-decoder U-Net model is used as a generator network architecture. The vanilla structure comprises a contracting path (left side) and an expansive path (right side) as shown in Figure 38. The contracting path pursues a typical architecture of a convolutional neural network, i.e., repeating blocks of application

of convolutions, each followed by ReLU, and max pooling operation. While the left path consists of upsampling blocks of the activation map followed by a convolution that decreases the number of feature channels, a concatenation with the accordingly cropped feature map from the contracting path, and convolutions, each followed by ReLU. The cropping is necessary due to the loss of border pixels in every convolution. In the final layer, convolution is used to map each n-component feature vector to the desired number of classes [70].

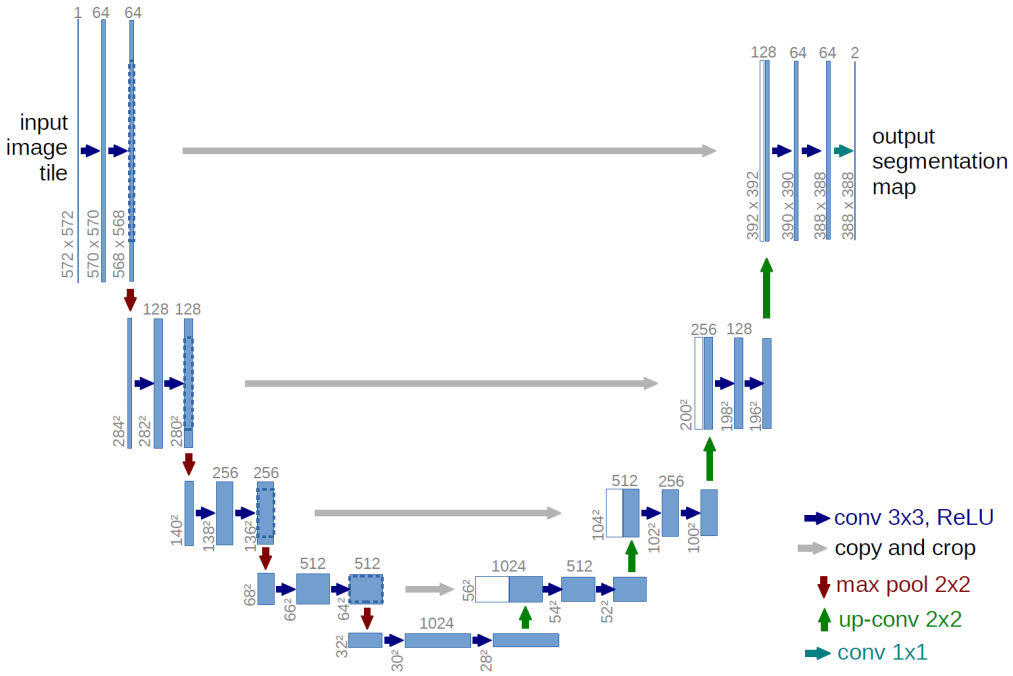


Figure 38. An example of the U-net architecture with 32×32 pixels resolution in the middle block. Blue boxes denotes a multi-channel feature map. On top of the boxes numbers indicate the number of channels in activation map. The x-y-dimensions are provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the corresponding operations [70].

The Pix2Pix adopts the idea of the U-Net architecture on a shared low level information between an input and ab output in the encoder-decoder network. The network translates a low level information through all the layers including the bottleneck, i.e., the middle layer. The skip connection feature in the architecture such information shuttles directly across the network and gives the generator a means to circumvent the middle layer (see Figure 39) [54]. Specifically, the skip connections are added between each layer i and layer $n-i$, where n is the total number of layers. Each skip connection simply concatenates all channels at layer i with those at layer $n-i$ [54]. Furthermore, several important modifications are employed to the generator. The encoder section of the generator network uses blocks of the following form: convolution layer followed by batch normalization layer

and ReLU. The batch normalization layer by reducing an internal covariate shift helps in accelerating the deep network training process [54] [71]. The implemented decoder network has the following structure: transpose convolutional layer followed by batch normalization layer and ReLU layer. The noise is applied in the form of dropout layers for the first 3 blocks in the decoder. The last block utilizes TanH activation function over ReLU converting a 128-component vector into RGB image scaled in a range $[-1, 1]$. Also in the last block batch normalization layer is not presented.

Discriminator architecture

As a discriminator in Pix2Pix network serves convolutional PatchGAN. The $L1$ loss function enforces to capture the low frequencies. This address to restrict the discriminator to map only high-frequency structure. Then would be sufficient to constrain the discriminator attention to the structure of local image patches [54]. The PatchGAN penalizes structure at the scale of those patches. The discriminator tries to classify if each $N \times N$ patch in an image is real or fake. The PatchGAN convolutionally iterates across the image and averages all the patches responses providing the final signal D. The ultimate advantage of such discriminator architecture that N can be much smaller than the initial image size, and the model would be still capable to generate high quality results. As a result, small PatchGAN contains a fewer number of parameters, calculates faster, and can be applied to arbitrarily large images [54]. The structure of the discriminator is depicted in Figure 40. Each block in the discriminator has the following structure: convolution layer followed by batch normalization layer, and ReLU layer as in the generator network. Except for the penultimate block of the discriminator which is slightly altered. It is surrounded by zero padding layers to get the desired shape of the output. Moreover, ReLU activation function is changed to the leaky ReLU. The last layer maps a 512-component activation map to a 2-D gradient.

4.3.3 Pix2Pix model for full board virtual sawing

The initial goal of the research is to generate full size board images. In order to obtain such output the following two approaches were taken into consideration:

1. Merge the patches into full board image.
2. Modify the architecture of the GAN.

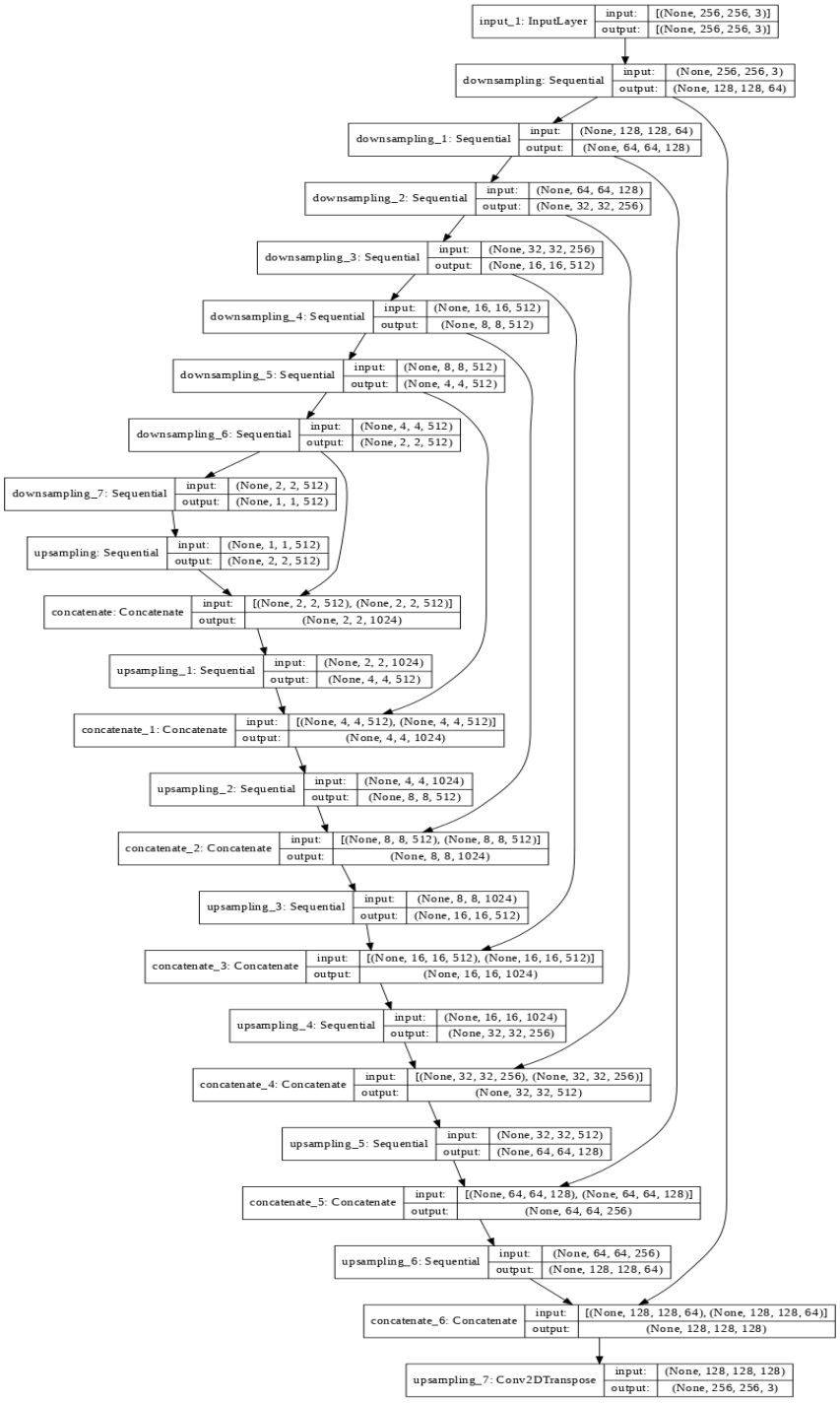


Figure 39. The original Pix2Pix generator network structure [72].

The first option implies to continue working with the patches obtained in the vanilla model. The second option calls for changes in the original framework architecture. This study does not focus on a texture synthesis algorithm that merges the patches into a full size board image. However, it can be an alternative method to be considered in future research. As a consequence, modification of the network structure can induce some unexpected visual artifacts and even problems with the training process such as mode collapse

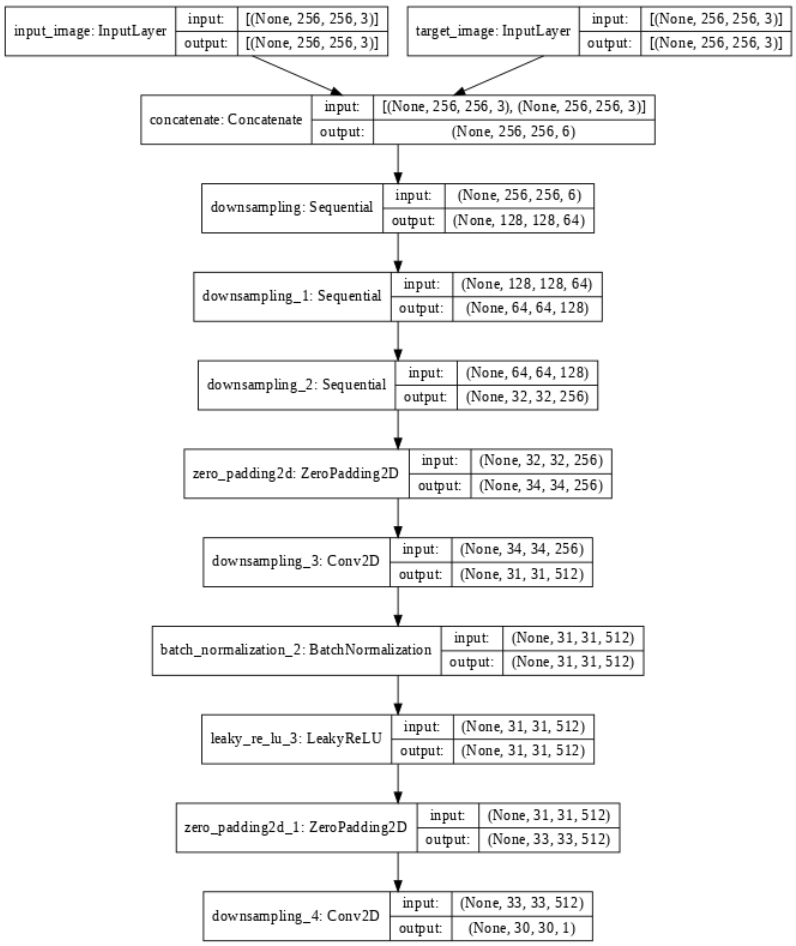


Figure 40. The original Pix2Pix discriminator network structure [72].

or failure in convergence. Although the second approach is more unpredictable and complex, it was chosen. The next section describes how the initial model is transformed, and the difficulties encountered are overcome.

Generator and discriminator architectures

The GAN generator architecture is based on the encoder-decoder U-Net network which baseline is a convolutional layer. In order to adjust the generator recognizing and producing the full image boards with high width to height ratio, the stride parameters in the upsampling and the downsampling block are altered. The changes allows to obtain the desired output from the input image of size 128×2048 pixels (see Figure 41). The discriminator uses the downsampling block from the generator, and thus, no complementary changes are required due to the advantageous architecture of the PatchGAN. The final architecture with the corresponding feature maps is illustrated in Figure 42.

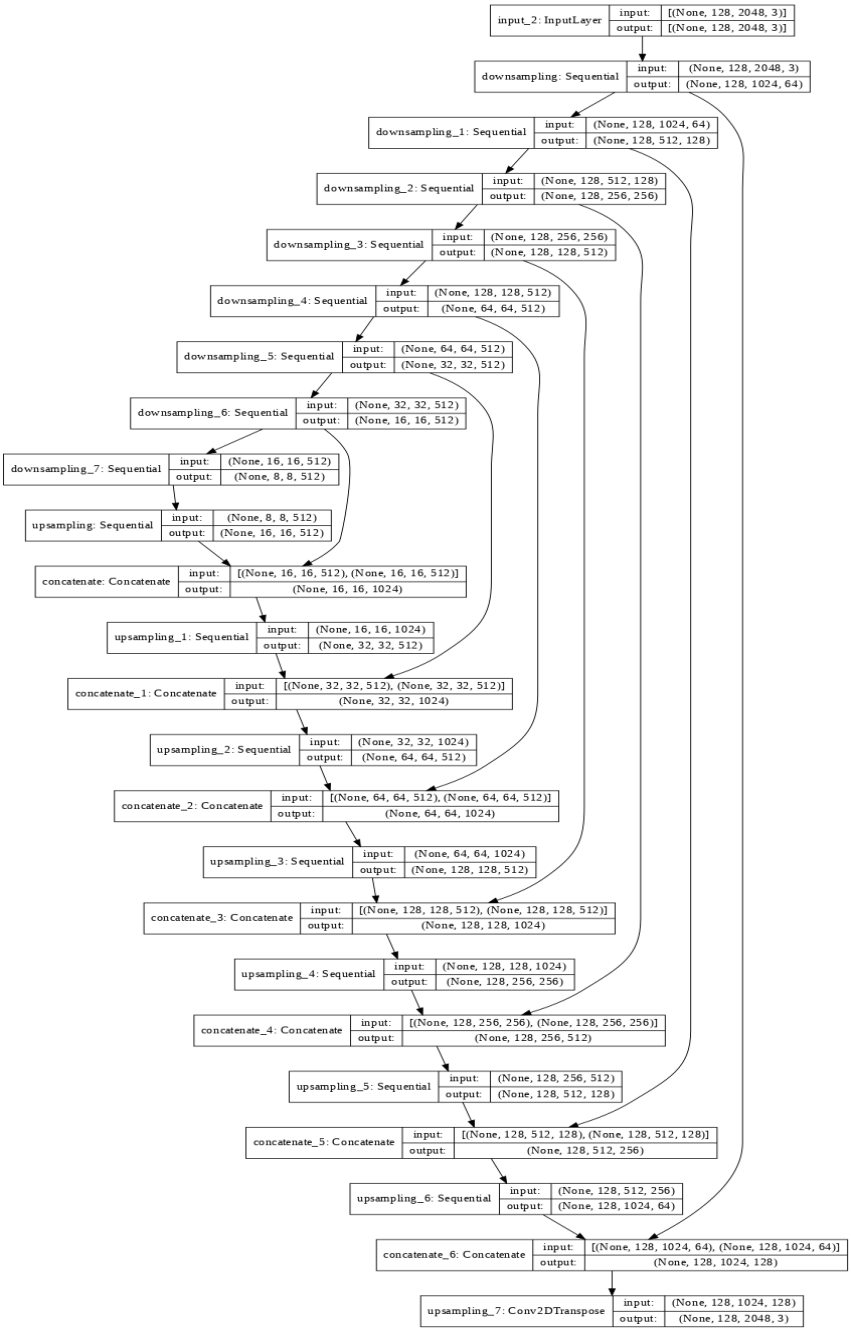


Figure 41. Modified Pix2Pix generator network structure with resize convolutional layers.

Checkerboard artifacts

The implemented GAN architecture after the encoding phase builds up the output image with the upsampling blocks in the decoder from a low resolution and a high feature map to a high resolution, and a low feature map (see Figures 41 and 42). The deconvolution or so called transpose convolution technique in the decoder network allows to describe the approximate image and then fill in the details. Unfortunately, transpose convolution easily

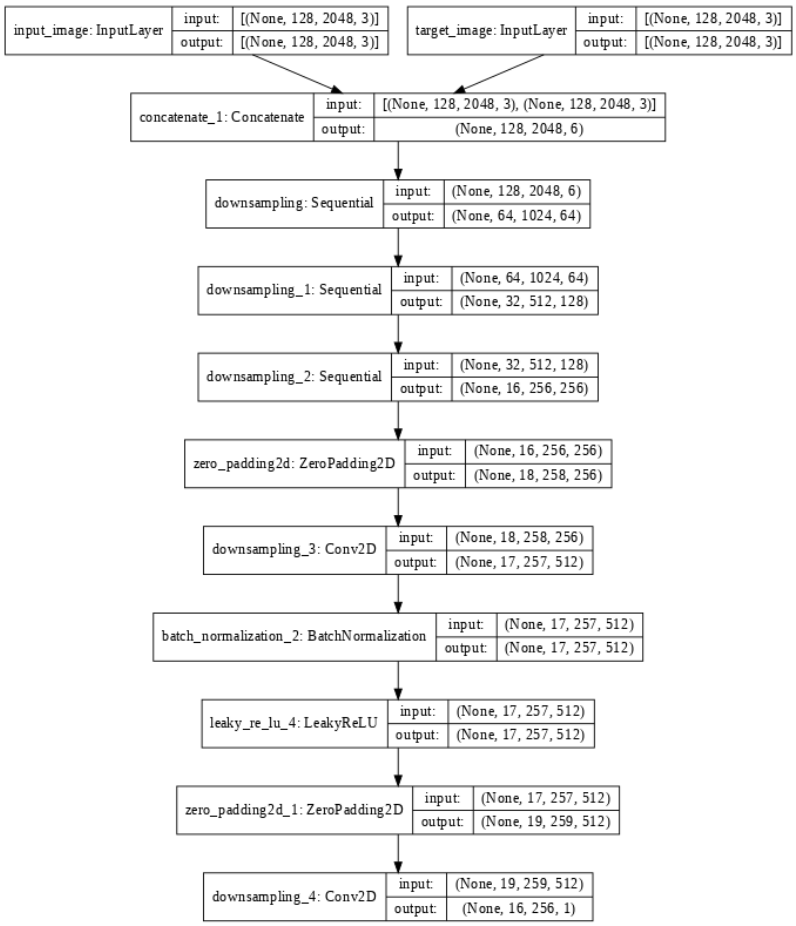


Figure 42. Modified Pix2Pix discriminator network structure.

tends to emerge artifacts into the output if the uneven overlap occurs (see Figure 43).

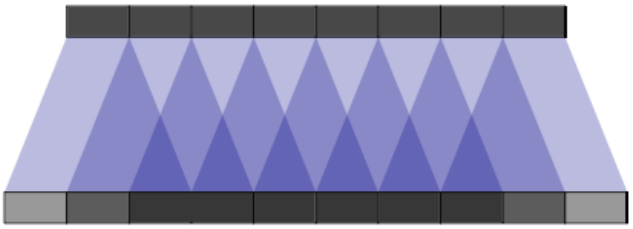


Figure 43. An uneven overlap from a deconvolution operation with kernel size 3 and stride 2. The upper line of rectangles is an input layer, the lower line is an output layer. The shades indicates an intensity of the overlap. The brightest regions on the edges of the output layer do not affected with an overlap while others suffer from the issue. [73].

In particular, an uneven overlap arises when a kernel size and an output window size are not divisible by a stride. The problem inclines to be more severe in two dimensions. Moreover, in the multilayer decoder architecture when output is constructed iteratively

from a stack of layers that turns out into extreme visual artifacts, i.e., checkerboard artifacts. Carefully chosen deconvolution parameters help to prevent the appearance of such output but bring significant restriction on filters, and in practice, the artifacts are still present in these models [73]. To address the problem the regular transposed convolutional layer is replaced with a resize convolutional layer as was proposed by Odena et al. (see Figure 44) [73]. The final upsampling block transforms from transpose convolutional layer followed by batch normalization layer, dropout layer (for the first 3 blocks), and ReLU layer to upsampling layer followed by reflection padding layer, and ordinary convolutional layer, the last three layers are the same. Although this method brings one substantial disadvantage, since the number of layers in blocks is increased from 4 to 6 which affects the computation time for the block, and thus, total training time.

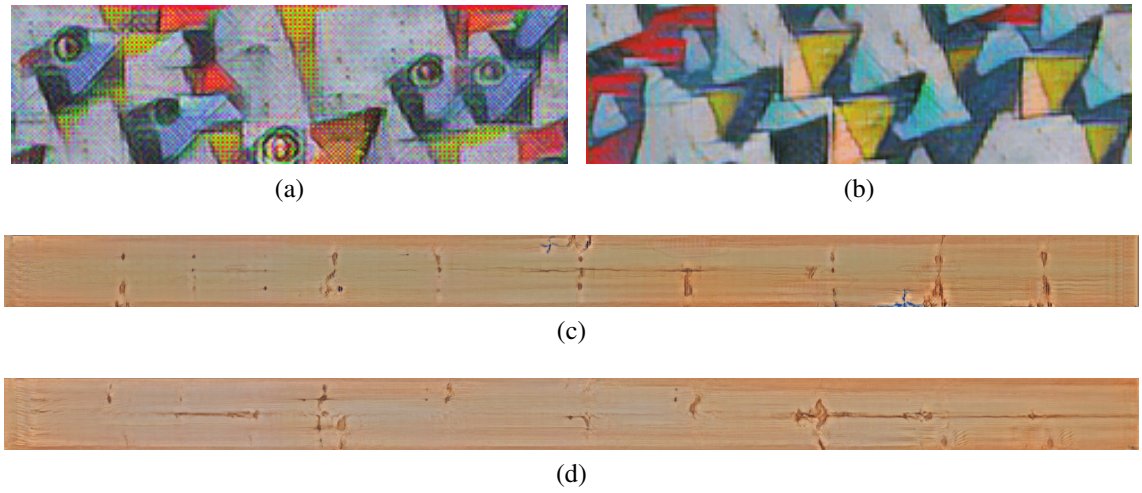


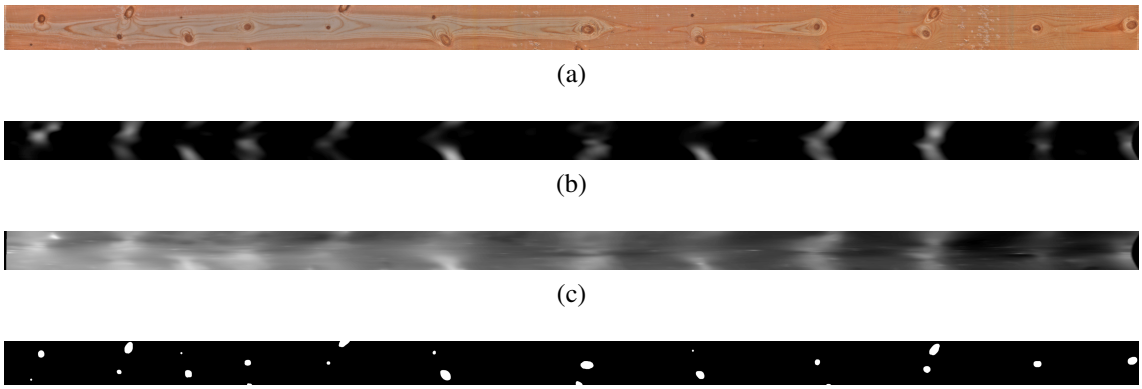
Figure 44. Switching deconvolution layers for resize convolution layers successfully dissolve the artifacts [73]: (a) An example of output from deconvolution layer; (b) An example of output from resize convolution layer. The checkerboard artifacts were successfully dissolved for the generated boards as well: (c) A board generated with deconvolution layer; (d) The same board generated with resize convolution layer.

5 EXPERIMENTS

5.1 Data

The data consist of 100 Scots pine *Pinus sylvestris*) logs. The logs were measured and sawn in the real sawmill environment. The produced boards were imaged with an RGB camera system. During the experiments, the following four modalities of the dataset are employed (see Figure 45):

- RGB images of real boards.
- Knot map images, obtained using the method from [4].
- Raw projected log surface heightmap images, computed as described in Section 4.2.2.
- Ground truth knots images, composed by manually annotation the knot locations.



(a)

(b)

(c)

(d)

Figure 45. The sample from four modalities of the dataset: (a) A real board image or a target image for the GAN; (b) A knot map image used as an input for the GAN; (c) A raw projected heightmap image used as an alternative input for the GAN; (d) An image with segmented ground truth knot locations.

5.2 Evaluation criteria

Despite significant progress in the field of various GAN applications, evaluation and comparison of GANs remain a challenging task. In fact, an efficient quantitative measure should meet a number of properties [74]:

- Support models generating high fidelity samples.
- Support models generating diverse samples.
- Support models with disentangled latent space.
- Possess a well-defined boundaries (lower, upper).
- Be perceptive for image distortions.
- Correspond to a human perceptual estimation.
- Possess a low sample and computational complexity.

In the scope of a virtual sawing task, an important characteristic of a generated board image is the accuracy of located knots, i.e., knots on a generated image correlate with knots on an actual image. Moreover, the generated knots should be similar to the knots of the real boards. As a consequence, an appropriate quantitative measure would be segmentation performance with a segmentation model trained on the real board images and using true knot locations as ground truth. More specifically, to compare the quality of the produced images, segmentation performance can be evaluated with the following measures (see Figure 46) [75]:

1. Precision.
   Precision is the ability of a model to recognize only relevant objects. In case of virtual sawing, it is the percentage of correct identified knot pixels to all knot pixels

$$Precision = \frac{TP}{TP + FP}, \tag{14}$$

   where True Positive (TP) is a number of correctly identified knot pixels and False Positive (FP) which is an amount of incorrectly identified nonknot pixels, or misplaced detections of knot pixels.

2. Recall.
   Recall is the capacity of a model to identify all relevant cases (all ground truth mask areas or bounding boxes). It is the percentage of correct knot pixel predictions among all given ground truths

$$Recall = \frac{TP}{TP + FN}, \tag{15}$$

   where TP is an amount of correctly identified knot pixels and False Negative (FN) is a number of undetected ground truth knot pixels.

3. Jaccard index (Intersection-over-Union).
   The Intersection-Over-Union (IoU), or simply the Jaccard Index, is one of the most typical similarity measures applied in instance segmentation

$$IoU = \frac{TP}{FP + TP + FN}. \tag{16}$$

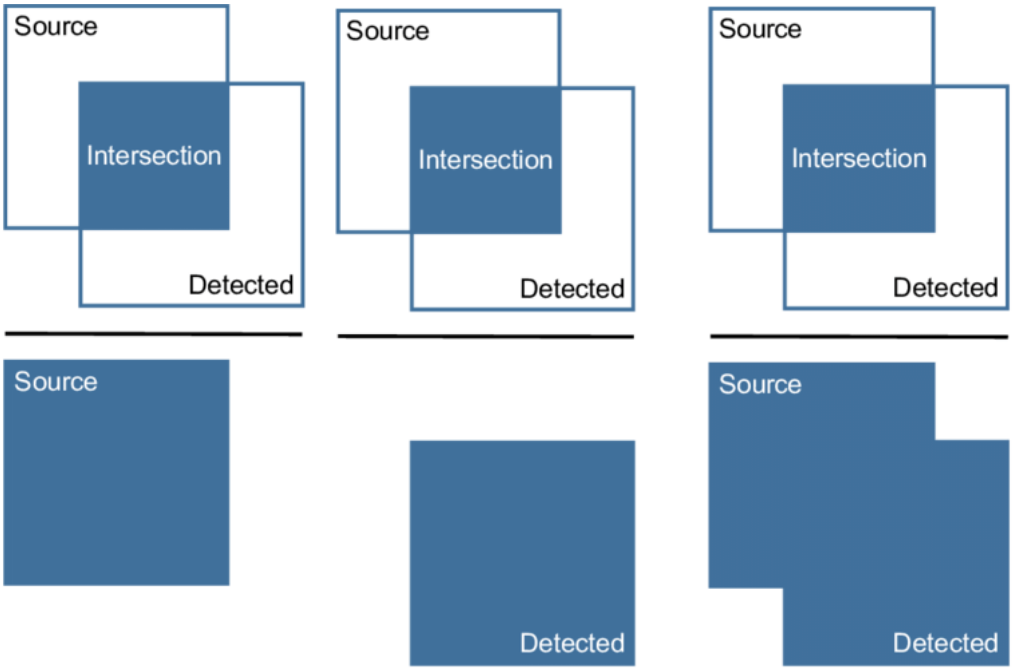


Figure 46. Demonstration of three measures for bounding boxes, the same are applied for mask areas: (a) Precision; (b) Recall; (c) IoU.

5.2.1 Detectron2 and COCO for knot segmentation performance

In order to find knot segmentation performance Detectron2 library is used [76]. However, a straightforward employment of Detectron2 is impossible with the existing format of the segmented ground truth knots images. One of the possible forms of annotation in Detectron2 is the Common Objects in Context (COCO) dataset format. As follows, the segmented ground truth knots dataset is transformed into the COCO format in which each knot representing a separate instance with the information about knot mask, bounding box, and also respective board (see Figure 47).

Eventually, the annotated dataset can be fed into a network that supports the COCO format. Next, from Detectron2 model zoo the pretrained Mask Region-Based Convolutional

(a)

(b)

Figure 47. An example of a board annotated in COCO format: (a) COCO annotated board; (b) Corresponding real board image.

Neural Network (R-CNN) [77] with the ResNeXt-101-Feature Pyramid Network (FPN) backbone is picked [78] which key properties are presented in Table 2. To estimate the quality of the generated images, the model trained on the real images then if an artificial image structure corresponds with a real image structure. The Detectron2 model would be able to successfully segment knot locations on a test subset of generated images.

Table 2. The main characteristics of the Mask R-CNN ResNeXt-101-FPN [78].

Characteristic / Model	Train time	Inference time	Train memory	Box average precision	Mask average precision
Mask R-CNN ResNeXt-101-FPN	0.690 seconds/iteration	0.103 seconds/image	7.2 GB	44.3 %	39.5 %

However, before utilizing the mentioned approach an optimal model among training epochs should be defined. In this research, only the loss function values are considered in searching for an optimal model, since the loss function values are not fully reliable measures as they do not possess the listed properties of efficient measures. Also due to the fact that the model tends to destabilize and oscillate due to the fleeting nature of a convergence point. The two different models with identical loss function values would certainly differ in a visual aspect. At the very least, this approach can cause selecting a nonoptimal model, but it might also cause various visual artifacts to the chosen model or a kind of a mode collapse.

5.2.2 Metrics for knot segmentation performance

Usually, an evaluation performance estimation of segmentation is done by a more sophisticated variant of precision and recall measures. An Average Precision (AP) and Average Recall (AR) metrics are commonly supplemented with a confidence level τ and IoU values [75]. Whereas, the Detectron2 library with COCO annotated dataset allows only to apply IoU values, and the following 11 metrics are used for characterizing the segmentation performance [79]:

- Average Precision across IoU.
  Includes two following metrics: $AP^{IoU=0.01:0.25:1}$, $AP^{IoU=0.01}$.

- AP across scales.
  Considers another three types of knots depending on the sizes (in pixels) as follows: AP^{small}, AP^{medium}, and AP^{large} for small, medium, and large knots, accordingly.

- Average Recall across detections.
  Recall calculated for a given number of detections: $AR^{max=1}$, $AP^{max=10}$, and $AP^{max=100}$ for the 1, 10, or 100 detections per image.

- AR across scales.
  The metric examine recall for various sizes of knots areas as follows: AR^{small}, AR^{medium}, and AR^{large} for small, medium, and large knots.

AR is the maximum recall given a fixed number of detections per image. All metrics are computed allowing for $IoU = 0.01$ at most 100 top-scoring detections per image unless otherwise mentioned. The evaluation metrics for detection with bounding boxes and segmentation masks are identical in all respects except for the IoU computation which is performed over boxes or masks, respectively. Additionally, the small, medium and large knots are defined for each segmentation evaluation individually as follows

$$\text{Knot type} = \begin{cases} \text{small}, & \text{if } x_i < \frac{\max\{X\}}{3} \\ \text{medium}, & \text{if } \frac{\max\{X\}}{3} < x_i < \frac{\max\{X\}}{2} \\ \text{large}, & \text{if } x_i > \frac{\max\{X\}}{2}, \end{cases} \tag{17}$$

where x_i is a knot area (in pixels) in a given set of knot areas X.

As a complementary performance metric, sample Pearson correlation coefficient (PCC) is applied to measure the linear correlation between a number of detected knots on generated board images and a true number of knots on corresponding RGB images of real boards. More specifically, PCC is the covariance of two variables in a sample set, divided by the product of their standard deviations

$$r_{xy}(PCC) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i}^{n}(y_i - \bar{y})^2}}, \tag{18}$$

where n is sample size, x_i, y_i are the sample points, $\bar{x}$ the sample mean, and analogously for $\bar{y}$.

5.3 Experiment 1: Patch architecture

The experiment with the unmodified Pix2Pix model comprises three main stages: the transformation of the data, the training process, and evaluation of the results.

5.3.1 Data preparation

The real board images, knot map images and ground truth images were transformed as follows:

1. Split dataset.
   The dataset was splitted to the train and test subsets. After partitioning each subset contained 80% (348 images) and 20% (87 images), respectively.

2. Image resizing.
   All the images were resized to the 179×5002 pixels which is median value of the images dimensions for the initial images.

3. Patch extraction.
   The images were cropped into square patches with the side of 179 pixels. Afterward, each board contained 28 patches, or 12180 patches in each modality.

4. Patch normalization.
   The extracted patches were normalized between $[-1, 1]$ in order to match an output of the TanH activation function in the last layer of the generator network.

5. Patch resizing.
   The patches were resized to the 256×256 pixels resolution matching the input and the output of the vanilla Pix2Pix architecture.

Additionally, the ground truth images are blurred by a Gaussian filter with a standard deviation of 10 to imitate the knot maps images where pixel intensities denote the probability of a knot being there. The value for standard deviation was chosen such that knots visually recall a knot map image.

5.3.2 Training process

To update network weights Adam optimization algorithm was used. The learning rate in the Adam algorithm was set as 0.002 adopting the one used in the original Pix2Pix architecture. The other parameters are tge default for Adam optimizer: the β_1 is 0.5 and the β_2 is 0.999. The hyperparameter for $L1$ loss equaled 100 which was also used by Isola et al [54]. The GAN was trained on 50 epochs for both the knot map images and ground truth images. During the training, the models oscillated and destabilized struggling to find a Nash equilibrium (see Figure 48).

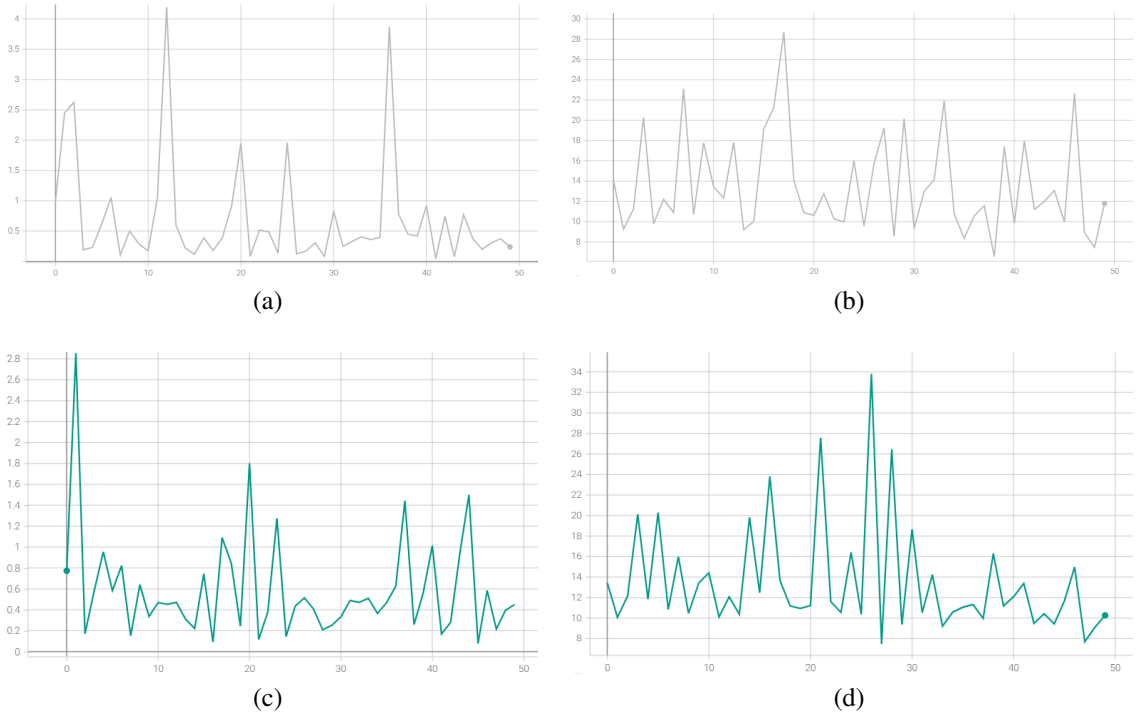


Figure 48. The graphics of the loss functions: (a) The discriminator loss for knot map images; (b) The generator loss for knot map images; (c) The discriminator loss for ground truth images; (d) The generator loss for ground truth images. A good reference point for discriminator loss is $log_e(2) = 0.69$. It indicates that a discriminator network is on average uncertain for both options if a reference image is real or generated.

5.3.3 Qualitative evaluation of results

The obtained patches from the knot map images are represented in Figure 49. The knot locations of generated patches for the most part correlate with the structure of a knot map image, but in some patches, divergence is presented. The GAN generated patches from the

ground truth dataset are represented in Figure 50. The obtained patches are realistically looking: the model is able to reproduce the various structures of knots, the low-level structure is also looking highly realistic. It is possible to conclude that the vanilla Pix2Pix model successfully managed to generate realistic patches for both datasets. Thus, an extension of a model for full board images is a reasonable decision.

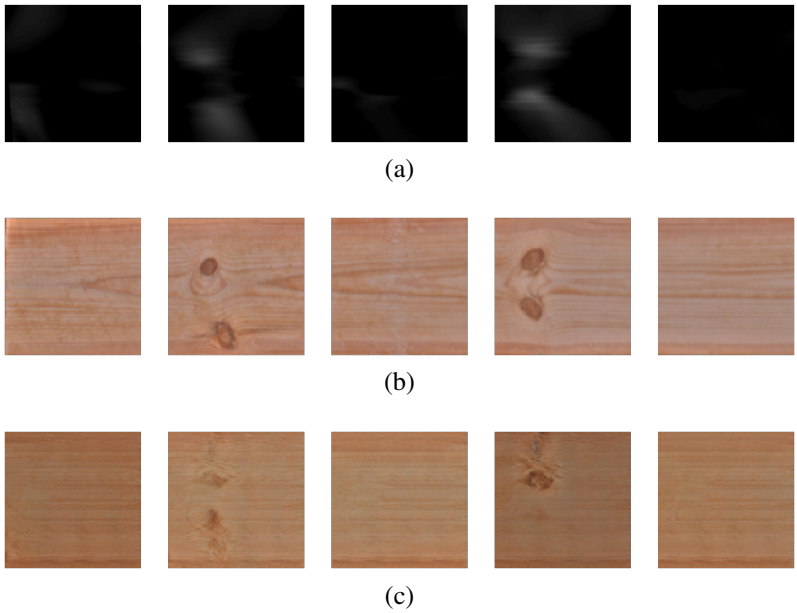


Figure 49. Example results from the vanilla model trained on the knot map images: (a) Patches of a knot map images; (b) Patches of a real board; (c) Patches of a generated board.

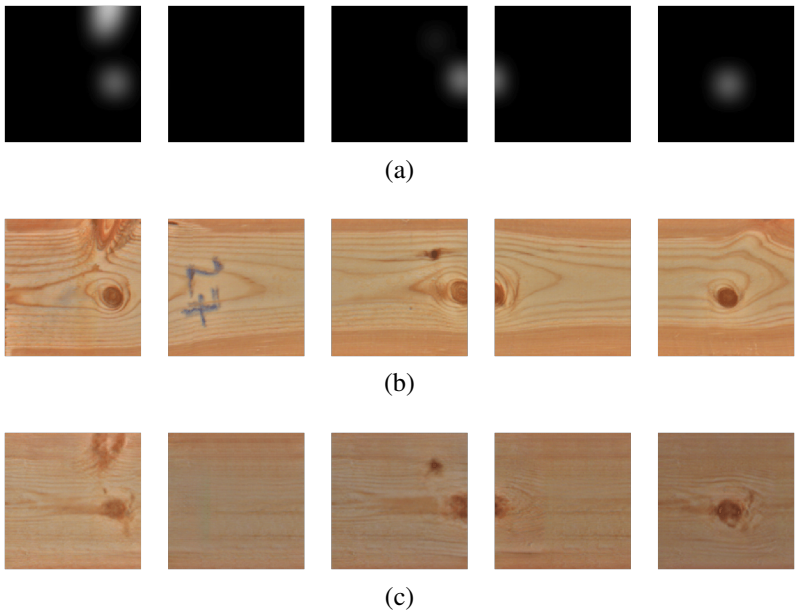


Figure 50. Example results from the vanilla model trained on the ground truth images: (a) Patches of a ground truth images; (b) Patches of a real board; (c) Patches of a generated board.

5.4 Experiment 2: Full board image generation with deconvolution block

In this experiment, the modified Pix2Pix architecture with deconvolution upsampling blocks was applied for the knot map images and raw projected heightmap images. The hyperparameter, as well as optimizer parameters, remained unchanged. The number of training epochs was increased to 200, as the number of training weights was increased significantly for full board image architecture. The input and target images were resized to 128×2048 pixels and then normalized.

5.4.1 Qualitative evaluation of results

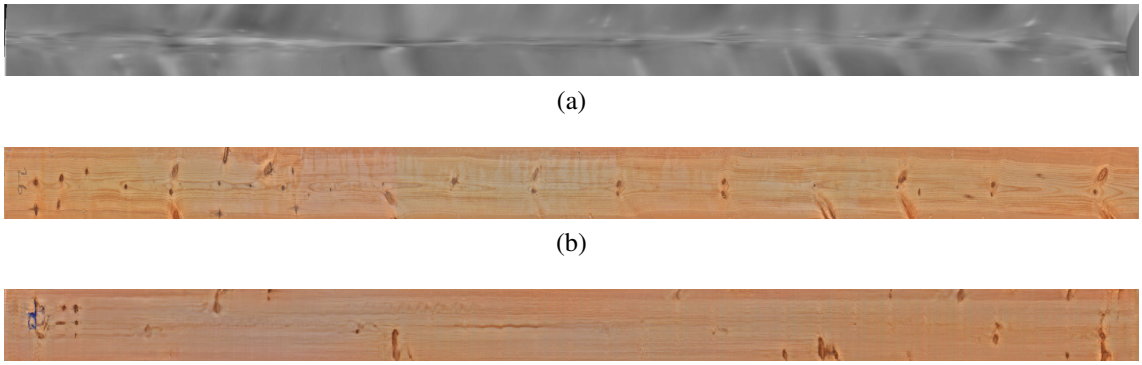
The model generates output with apparent visual artifacts such as checkerboard patterns on both dataset modalities the raw projected heightmap images and knot map images. Therefore, no segmentation performance metrics are applied for the defective model. The generated image from the model trained on knot map images is illustrated in Figure 51. Although the overall structure of the generated image correlates with the target image, since the image corrupted with the checkerboard artifacts, it can not be defined as realistic looking. The output of the GAN trained on the raw projected heightmap images is shown in Figure 52. Overall, the network struggles to reproduce the exact structure of the real board images from the raw projected heightmap images. The checkerboard artifacts are less presented in the generated image, but other visual artifacts are induced: a wavy low level structure, a repeating vertical line pattern.

(a)

(b)

(c)

Figure 51. Example results for the model trained on the knot map images with deconvolution upsampling block: (a) A full size knot map image; (b) A full size real board image; (c) A full size generated image.



(a)

(b)

(c)

Figure 52. Example results for the model trained on the raw projected heightmap images with deconvolution upsampling block: (a) A full size raw heightmap image; (b) A full size real board image; (c) A full size generated image.

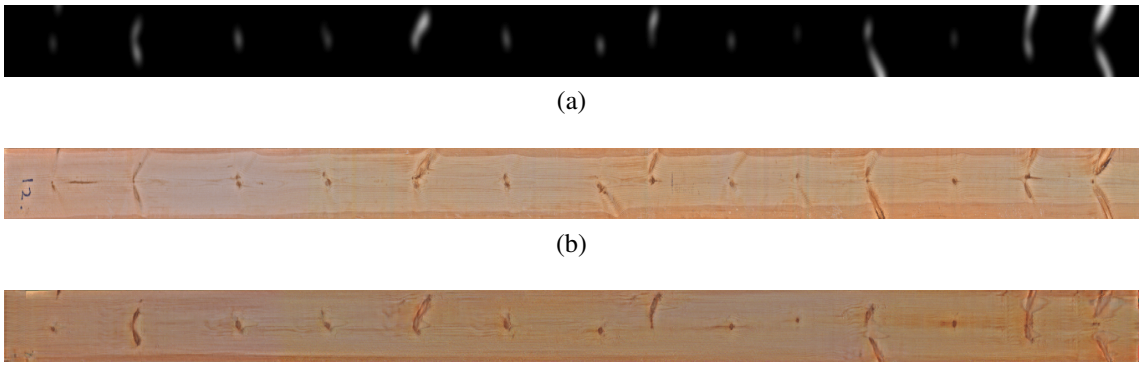
5.5 Experiment 3: Full board image generation with resize convolution block

In this experiment the deconvolution upsampling block is replaced with the resize convolution upsampling block. The dataset preparation algorithm and training procedures were adopted from the full board network approach with upsampling deconvolution block. The instance segmentation metrics were computed for all three inputs: knot map images, raw projected heightmap images, and ground truth images. The 358 real samples (80% of images) formed the training subset for the Detectron2 model. The rest of the generated boards that were 87 images (20% of images) composed the testing subset for segmentation metrics evaluation.

5.5.1 Quantitative evaluation of results for ground truth images

The Mask R-CNN ResNeXt-101-FPN segmentation model is trained on 5000 iterations with a base learning rate 0.025. The model training was carried out on the real board images, while metrics calculation was performed with generated images. Both for the real and generated datasets ground truth COCO annotation was used. The training model was able to extract knot locations from the generated boards. With ground truth dataset the implemented model is capable to generate realistic looking boards (see Figure 53). The structure of knots in the output image robustly coincides with a real board image. Although some visual artifacts are still presented. The calculated PCC and segmentation performance metrics are presented in Figure 54 and Table 3, respectively. The boundaries for the small, medium, and large knots areas are computed from the generated board im-

ages as defined in Eq. 17. The segmented binary masks and bounding boxes are illustrated in Figure 55.



(a)

(b)

(c)

Figure 53. Example results for the model trained on the ground truth images with resize convolution upsampling block: (a) A full size ground truth image; (b) A full size real board image; (c) A full size generated image.

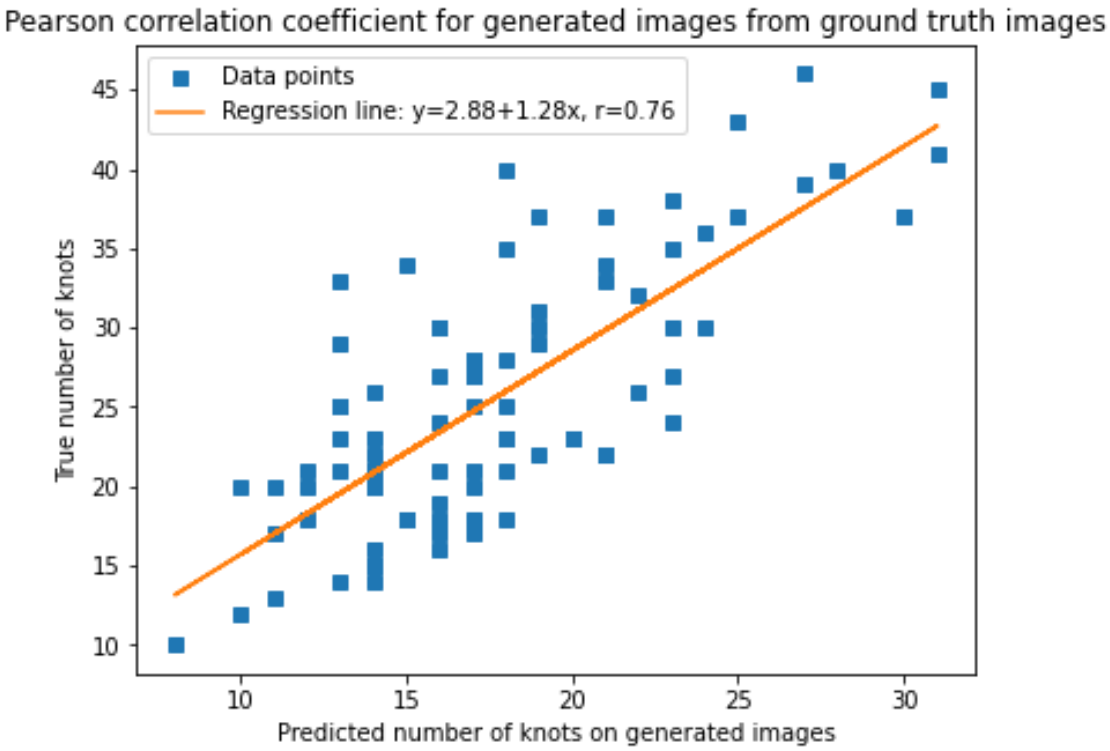


Figure 54. Scatter diagram for the sample set illustrates a correlation between the predicted number of knots and true number of knots for 87 images. The PCC represents a strong positive correlation with $r = 0.76$

Table 3. Segmentation metrics for the model trained on the ground truth images.

Metric \ Annotation type	Bounding boxes	Segmentation masks
$AP^{IoU=0.01:0.25:1}$	0.400	0.389
$AP^{IoU=0.01}$	0.683	0.682
$AP^{small}(20^2 > \text{pixels})$	0.683	0.681
$AP^{medium}(20^2 < \text{pixels} < 24^2)$	0.782	0.782
$AP^{large}(24^2 < \text{pixels})$	0.356	0.356
$AR^{max=1}$	0.039	0.039
$AR^{max=10}$	0.391	0.390
$AR^{max=100}$	0.686	0.684
$AR^{small}(20^2 > \text{pixels})$	0.688	0.686
$AR^{medium}(20^2 < \text{pixels} < 24^2)$	0.786	0.786
$AR^{large}(24^2 < \text{pixels})$	0.357	0.357

(a)

(b)

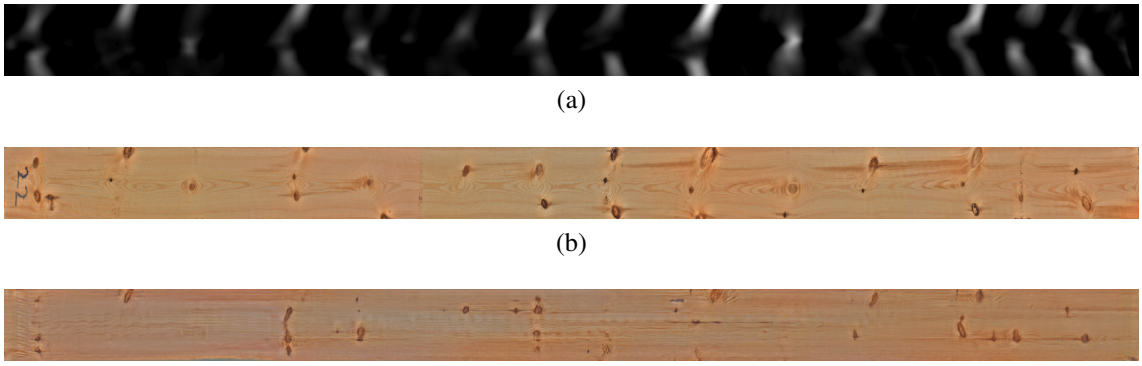
(c)

Figure 55. An example of successfully extracted knot locations with Detectron2 model: (a) A generated knot binary mask; (b) Bounding boxes and mask areas from a generated image; (c) Ground truth bounding boxes and mask areas.

5.5.2 Quantitative evaluation of results for knot map images

The checkerboard visual artifact completely dissolved with the resize convolutional block, but a wavy low level structure is still presented on the boards (see Figure 56). Overall, the positions of knots correlate with a pattern of the knot map image. However, in a scope of a segmentation performance, the locations of knot on the generated boards weakly correlate with the real boards. Since the problem is at least an order of magnitude more complex, than basic object detection. The segmentation metrics can not be expected as high results as in ordinary object detection. The calculated PCC and segmentation performance metrics for model trained on knot map images are presented in Figure 57 and Table 4. An example of the segmented bounding boxes and mask areas is represented

in Figure 58.



(a)

(b)

(c)

Figure 56. Example results for the model trained on the knot map images with resize convolution upsampling block: (a) A full size knot map image; (b) A full size real board image; (c) A full size generated image.

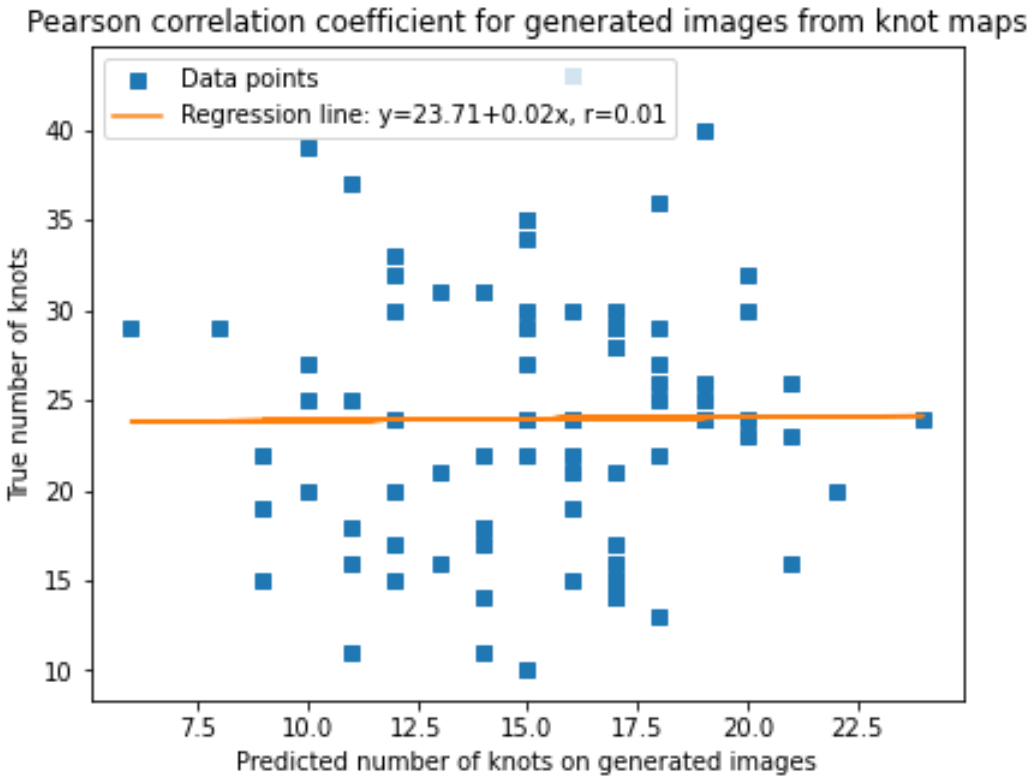


Figure 57. Scatter diagram for the sample set illustrates a correlation between 87 images. The PCC equals 0.01 which designates an absence of a correlation.

Table 4. Segmentation metrics for the model trained on the knot map images.

Metric \ Annotation type	Bounding boxes	Segmentation masks
$AP^{IoU=0.01:0.25:1}$	0.008	0.006
$AP^{IoU=0.01}$	0.025	0.015
$AP^{small}(22^2 > \text{pixels})$	0.020	0.014
$AP^{medium}(22^2 < \text{pixels} < 27^2)$	0.378	0.249
$AP^{large}(27^2 < \text{pixels})$	0.455	0.257
$AR^{max=1}$	0.009	0.007
$AR^{max=10}$	0.068	0.051
$AR^{max=100}$	0.108	0.078
$AR^{small}(22^2 > \text{pixels})$	0.097	0.070
$AR^{medium}(22^2 < \text{pixels} < 27^2)$	0.465	0.349
$AR^{large}(27^2 < \text{pixels})$	0.450	0.250

(a)

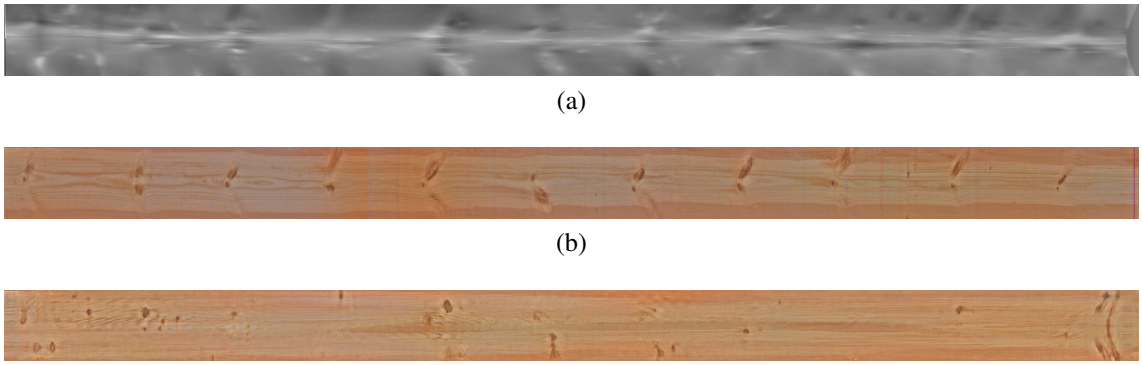
(b)

(c)

Figure 58. An example of successfully extracted knot locations with Detectron2 model: (a) A generated knot binary mask; (b) Bounding boxes and mask areas from a generated image; (c) Ground truth bounding boxes and mask areas.

5.5.3 Quantitative evaluation of results for raw projected heightmap images

The resize convolutional block confirms an efficiency in resolving the checkerboard pattern as presented in Figure 59. However, the knot locations from the generated image weakly correlate with the real one. This calls into question the application of the raw heightmap data as utilization of the raw projected heightmap data turned out into a big challenge for the GAN architecture. The calculated segmentation performance metrics decreased in comparison with the results from the model trained on the knot map images proving that the raw projected log surface heightmap data is a kind of challenge for the modified Pix2Pix architecture (see Tables 4 and 5). The PCC is shown in Figure 60. An example of the segmented bounding boxes and mask areas is represented in Figure 61



(a)

(b)

(c)

Figure 59. Example results for the model trained on the raw projected heightmap map images with resize convolution upsampling block: (a) A full size raw projected heightmap image; (b) A full size real board image; (c) A full size generated image.

Table 5. Segmentation metrics for the model trained on the raw projected heightmap images.

Metric \ Annotation type	Bounding boxes	Segmentation masks
$AP^{IoU=0.01:0.25:1.00}$	0.005	0.004
$AP^{IoU=0.01}$	0.019	0.013
$AP^{small}(17^2 > \text{pixels})$	0.011	0.008
$AP^{medium}(17^2 < \text{pixels} < 21^2)$	0.178	0.141
$AP^{large}(21^2 < \text{pixels})$	0.270	0.175
$AR^{max=1}$	0.010	0.009
$AR^{max=10}$	0.061	0.050
$AR^{max=100}$	0.100	0.080
$AR^{small}(17^2 > \text{pixels})$	0.075	0.060
$AR^{medium}(17^2 < \text{pixels} < 21^2)$	0.268	0.225
$AR^{large}(21^2 < \text{pixels})$	0.408	0.310

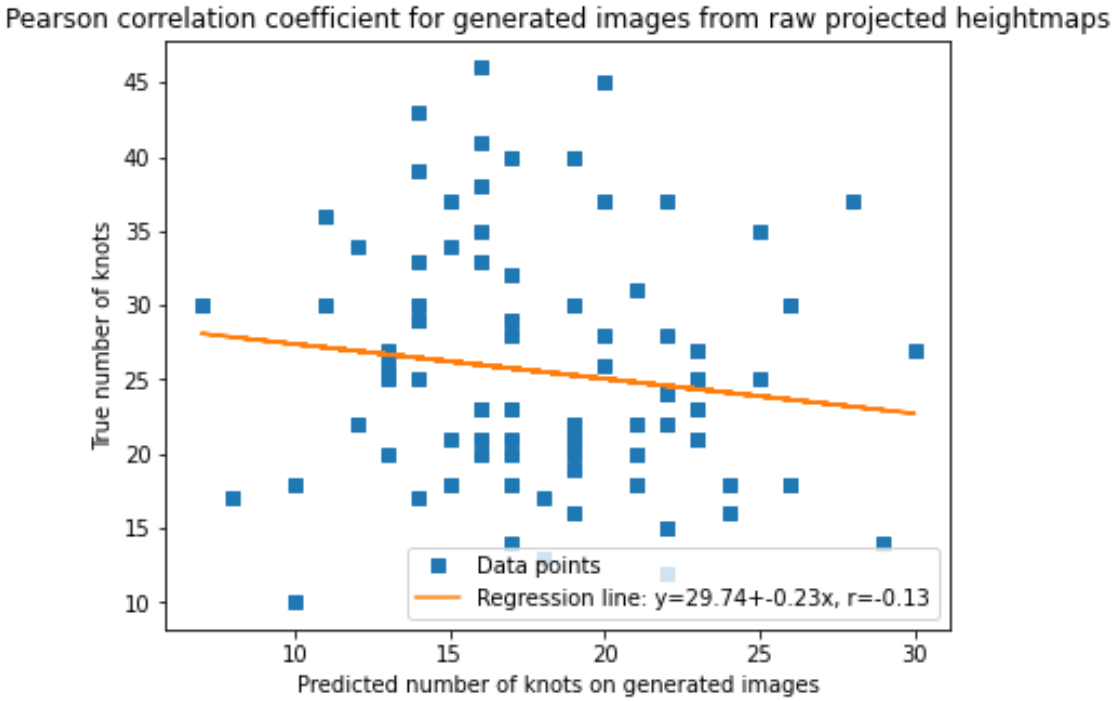
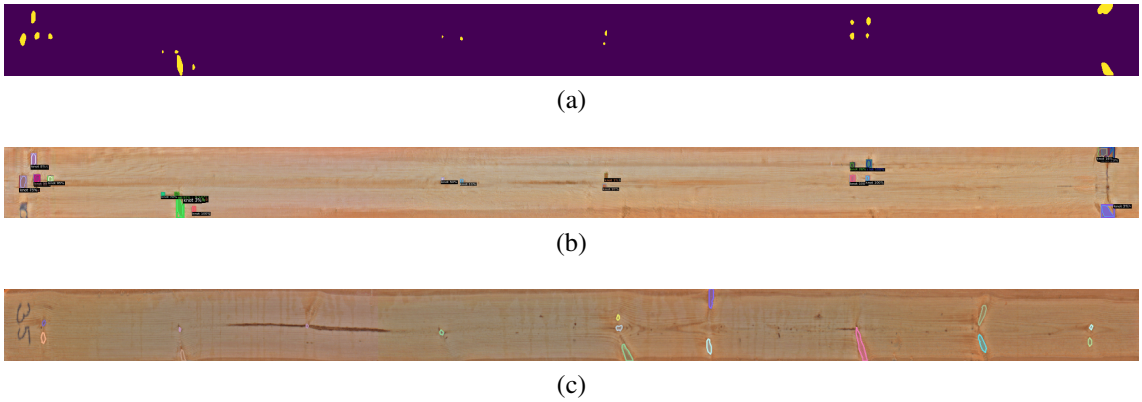


Figure 60. Scatter diagram for the sample set illustrates correlation between 87 images. The PCC equals -0.13 which can be considered as very weak negative correlation.



(a)

(b)

(c)

Figure 61. An example of successfully extracted knot locations with Detectron2 model: (a) A generated knot binary mask; (b) Bounding boxes and mask areas from a generated image; (c) Ground truth bounding boxes and mask areas. A generated image structurally do not correlate with a real image.

6 DISCUSSION

6.1 Current study

The size and the location of knots are the main factors affecting the board quality. Therefore, a system robustly predicting knot positions in the sawn board based on the external log measurements is highly valuable as it is capable to increase sawmill profitability.

The study proposed a supplementary module for an existing virtual sawing system introducing a more transparent feedback loop for a sawmill operator. The implemented image-to-image translation method is capable to convert the knot map data and the raw projected heightmap data into photorealistic board images. The method is based on the modified state-of-the-art Pix2Pix GAN architecture. The generated images are looking realistically from the visual perception. Moreover, the segmentation model trained on real board images is able to find knots on the generated images. However, the relatively low AP and AR segmentation measures and absence of the correlation between the numbers of knots on the generated and the real images build existing evidence of weak structural similarity with the real board images. More specifically, the model trained on the raw projected heightmap images producing an output where knot locations weakly correlate with the true positions of knots. The model trained with the knot map images achieved a more prominent morphological similarity. Nevertheless, an attribute of photorealism for generated images can not be directly interpreted through segmentation metrics. Thus, the qualitative and quantitative measures allow the generated images to be considered moderately realistic.

Despite that the training data was limited, only Scots pine is presented in the dataset. The proposed image-to-image translation method should be also applicable to other wood species if proper training data is provided.

6.2 Future work

As a continuation of the current study of the Pix2Pix architecture, many factors in data preparation, training process, and network architectural parameters can be considered or involved to generate more realistic-looking and diverse board images. As an example, a relatively small dataset requires to be extended with data augmentation techniques. During the training process, intermediate models should be examined with a quantitative

measure like Frechet Inception Distance in order to find the optimal one. In addition, an extension of a generator and a discriminator with supplementary downsampling and upsampling block, in theory, can force the GAN to produce high fidelity and diverse samples of board images. Moreover, different architectural parameters like a number of feature maps in downsampling and upsampling blocks should be tested thoroughly. As an alternative, a disentangled representation of a feature space can be employed as it can help the network to generate more diverse and realistic instances with a small amount of training data.

Besides the research of the Pix2Pix model, further studies can employ and evaluate other GAN architectures. Also, the abovementioned Frechet Inception Distance can be applied as a quantitative measure to evaluate the photorealism of produced images. Another important task to carry out is an improvement of an existing virtual sawing framework, which requires a more accurate and in depth evaluation of the obtained results and the implemented GAN network architecture.

Furthermore, using as input the knot maps has limited practical use since knot locations are already defined and can be used for optimization as such. Whereas, application of raw projected log surface heightmaps provides a more general end-to-end framework where the networks learn to extract the necessary information from the log measurements. This unbounded approach can be applied for various wood species skipping wood specific properties on the stage of modeling knot growth and segmentation. Also, an interesting option for future research would be to apply new input modalities such as x-ray log data for unsupervised machine learning image-to-image models.

7 CONCLUSION

The study investigated the applicability of GANs for the virtual sawing task. The existing state-of-the-art GAN architectures were reviewed. The principals of CNNs and generative models were surveyed. The dataset for training the selected GAN architecture was prepared. The trainable modification of the Pix2Pix architecture was implemented. The proposed method was evaluated for the capability to generate realistic board images from both inputs the raw projected log surface heightmap images and knot map images. According to the demonstrated ability of a segmentation model to detect knots on the generated images, the proposed image-to-image translation method can produce realistic looking board images. Finally, for the continuation of the research were suggested to extend the implemented model with Frechet Inception Distance or disentangled feature space. Also an application of more advanced network architecture or utilization of x-ray log data as an input is worth considering for future study of virtual sawing systems.

REFERENCES

[1] Markus Lier, Kari T Korhonen, Tuula Packalen, Tiina Sauvula-Seppälä, Tarja Tuomainen, Jari Viitanen, Antti Mutanen, Eeva Vaahtera, and Jouni Hyvärinen. Finland's forests 2019. 2019.

[2] Markku Husso and Erlend Nybakk. Importance of internal and external factors when adapting to environmental changes in sme sawmills in norway and finland: The manager's view. *Journal of Forest Products Business Research*, 7(1):14, 2010.

[3] Porter Brian and Tooke Chris. *Carpentry and Joinery 1*, volume 3rd ed. Routledge, 2001.

[4] Fedor Zolotarev, Tuomas Eerola, Lasse Lensu, Heikki Kälviäinen, Tapio Helin, Heikki Haario, Tomi Kauppi, and Jere Heikkinen. Modelling internal knot distribution using external log features. *Computers and Electronics in Agriculture*, 179:105795, 2020.

[5] Digisaw project, Finland. http://www2.it.lut.fi/project/digisaw/index.shtml, 2020. [Online; accessed January, 15, 2021].

[6] Pullar Gordon and T. Título. Energy conservation in the mechanical forest industries. *FAO Forestry Paper*, 93, 1990.

[7] Michael H. Ramage, Henry Burridge, Marta Busse-Wicher, George Fereday, Thomas Reynolds, Darshil U. Shah, Guanglu Wu, Li Yu, Patrick Fleming, Danielle Densley-Tingley, Julian Allwood, Paul Dupree, P.F. Linden, and Oren Scherman. The wood from the trees: The use of timber in construction. *Renewable and Sustainable Energy Reviews*, 68:333 – 359, 2017.

[8] The composition of pine and spruce and grading. https://www.swedishwood.com/India/pine_spruce_grading/, 2020. [Online; accessed February, 1, 2021].

[9] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[10] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.

[11] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[12] Päivi Parviainen, Maarit Tihinen, Jukka Kääriäinen, and Susanna Teppola. Tackling the digitalization challenge: how to benefit from digitalization in practice. *International Journal of Information Systems and Project Management*, 5(1):63–77, 2017.

[13] Nikolay Rudakov, Tuomas Eerola, Lasse Lensu, Heikki Kälviäinen, and Heikki Haario. Detection of mechanical damages in sawn timber using convolutional neural networks. In *German Conference on Pattern Recognition*, pages 115–126. Springer, 2018.

[14] Ummi Rabaah Hashim, Siti Zaiton Hashim, and Azah Kamilah Muda. Automated vision inspection of timber surface defect: a review. *Jurnal Teknologi*, 77(20), 2015.

[15] Dmitrii Shustrov, Tuomas Eerola, Lasse Lensu, Heikki Kälviäinen, and Heikki Haario. Fine-grained wood species identification using convolutional neural networks. In *Scandinavian Conference on Image Analysis*, pages 67–77. Springer, 2019.

[16] Fedor Zolotarev, Tuomas Eerola, Lasse Lensu, Heikki Kälviäinen, Heikki Haario, Jere Heikkinen, and Tomi Kauppi. Timber tracing with multimodal encoder-decoder networks. In *International Conference on Computer Analysis of Images and Patterns*, pages 342–353. Springer, 2019.

[17] M. Singmin and National Timber Research Institute (South Africa). *SIMSAW: A Simulation Program to Evaluate the Effect of Sawing Patterns on Log Recovery*. CSIR special report: HOUT. National Timber Research Institute, Council for Scientific and Industrial Research, 1978.

[18] Berndt G Lindner, PJ Vlok, and C Brand Wessels. Determining optimal primary sawing and ripping machine settings in the wood manufacturing chain. *Southern Forests: a Journal of Forest Science*, 77(3):191–201, 2015.

[19] Isabel Pinto, Helena Pereira, and Arto Usenius. Sawing simulation of pinus pinaster ait. *Coordinates*, 1(S2):S3, 2002.

[20] Tiecheng Song, Arto Usenius, and Innovood Seminar. Innosim-a simulation model of wood conversion chain. In *COST Action E44 Conference on Wood Processing Strategy Helsinki*, pages 95–108, 2007.

[21] CL Todoroki. Seesaw: A visual sawing simulator, as developed in version 3.0. *New Zealand Journal of Forestry Science*, 18(1):116–23, 1988.

[22] CL Todoroki. Autosaw system for sawing simulation. *New Zealand Journal of Forestry Science*, 20(3):332–348, 1990.

[23] R Edward Thomas. Raysaw: A log sawing simulator for 3d laser-scanned hardwood logs. In *The 18th Central Hardwood Forest Conference*, volume 117, pages 325–334, 2013.

[24] Andreas Rais, Enrico Ursella, Enrico Vicario, and Federico Giudiceandrea. The use of the first industrial x-ray ct scanner increases the lumber recovery value: case study on visually strength-graded douglas-fir timber. *Annals of Forest Science*, 74(2):28, 2017.

[25] Martin Pernkopf, Martin Riegler, and Manfred Gronalt. Profitability gain expectations for computed tomography of sawn logs. *European Journal of Wood and Wood Products*, 77(4):619–631, 2019.

[26] Anu Kantola. The structure of norway spruce (picea abies [l.] karst.) stems in relation to wood properties of sawn timber. *Dissertationes Forestales*, 2008, 08 2008.

[27] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[28] Andrew Ng and Michael Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14, pages 841–848. MIT Press, 2002.

[29] Generative Adversarial Networks. https://developers.google.com/machine-learning/gan, 2020. [Online; accessed January, 25, 2021].

[30] David Foster. *Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play*. O'Reilly Media, 2019.

[31] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems*, pages 396–404, 1990.

[32] M.V. Valueva, N.N. Nagornov, P.A. Lyakhov, G.V. Valuev, and N.I. Chervyakov. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Mathematics and Computers in Simulation*, 177:232 – 243, 2020.

[33] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[34] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, 9(4):611–629, 2018.

[35] S. Mustafa and M. Dauda. Evaluating convolution neural network optimization algorithms for classification of cervical cancer macro images. In *The 15th International Conference on Electronics, Computer and Computation*, pages 1–5, 2019.

[36] Carl Edward Rasmussen. The infinite gaussian mixture model. *Advances in Neural Information Processing Systems*, 12:554–560, 2000.

[37] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *The Institute of Electrical and Electronics Engineers*, 77(2):257–286, 1989.

[38] L. Jiang, H. Zhang, and Z. Cai. A novel bayes model: Hidden naive bayes. *IEEE Transactions on Knowledge and Data Engineering*, 21(10):1361–1371, 2009.

[39] Aziz Alotaibi. Deep generative adversarial networks for image-to-image translation: A review. *Symmetry*, 12(10):1705, 2020.

[40] Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In *Artificial Intelligence and Statistics*, pages 448–455. Machine Learning Research, 2009.

[41] Geoffrey E Hinton. Deep belief networks. *Scholarpedia*, 4(5):5947, 2009.

[42] Diederik P. Kingma and M. Welling. Auto-encoding variational bayes. In *The 2nd International Conference on Learning Representations*, 2014.

[43] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[44] M Ehsan Abbasnejad, Qinfeng Shi, Anton van den Hengel, and Lingqiao Liu. A generative adversarial density estimator. In *Conference on Computer Vision and Pattern Recognition*, pages 10782–10791, 2019.

[45] Lei Wang, Wei Chen, Wenjia Yang, Fangming Bi, and Fei Richard Yu. A state-of-the-art review on image synthesis with generative adversarial networks. *Institute of Electrical and Electronics Engineers Access*, 8:63514–63537, 2020.

[46] Michael U Gutmann, Ritabrata Dutta, Samuel Kaski, and Jukka Corander. Likelihood-free inference via classification. *Statistics and Computing*, 28(2):411–425, 2018.

[47] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27, pages 2672–2680. Curran Associates, Inc., 2014.

[48] Zhengwei Wang, Qi She, and Tomas E Ward. Generative adversarial networks in computer vision: A survey and taxonomy. *arXiv preprint arXiv:1906.01529*, 2019.

[49] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[50] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. In *The 5th International Conference on Learning Representations*, 2017.

[51] Aaron Hertzmann, Charles E Jacobs, Nuria Oliver, Brian Curless, and David H Salesin. Image analogies. In *The 28th Annual Conference on Computer Graphics and Interactive Techniques*, pages 327–340, 2001.

[52] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *International Conference on Computer Vision*, pages 2223–2232, 2017.

[53] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *The 31st International Conference on Neural Information Processing Systems*, pages 700–708, 2017.

[54] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Conference on Computer Vision and Pattern Recognition*, pages 1125–1134, 2017.

[55] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Conference on Computer Vision and Pattern Recognition*, pages 8798–8807, 2018.

[56] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Conference on Computer Vision and Pattern Recognition*, pages 8789–8797, 2018.

[57] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *The 31st International Conference on Neural Information Processing Systems*, pages 465–476, 2017.

[58] Feng Xiong, Qianqian Wang, and Quanxue Gao. Consistent embedded gan for image-to-image translation. *Institute of Electrical and Electronics Engineers Access*, 7:126651–126661, 2019.

[59] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. In *International Conference on Machine Learning*, pages 1857–1865. PMLR, 2017.

[60] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. Dualgan: Unsupervised dual learning for image-to-image translation. In *International Conference on Computer Vision*, pages 2849–2857, 2017.

[61] Yu Li, Sheng Tang, Rui Zhang, Yongdong Zhang, Jintao Li, and Shuicheng Yan. Asymmetric gan for unpaired image-to-image translation. *IEEE Transactions on Image Processing*, 28(12):5881–5896, 2019.

[62] Lei Chen, Le Wu, Zhenzhen Hu, and Meng Wang. Quality-aware unpaired image-to-image translation. *IEEE Transactions on Multimedia*, 21(10):2664–2674, 2019.

[63] Amélie Royer, Konstantinos Bousmalis, Stephan Gouws, Fred Bertsch, Inbar Mosseri, Forrester Cole, and Kevin Murphy. Xgan: Unsupervised image-to-image translation for many-to-many mappings. In *Domain Adaptation for Visual Understanding*, pages 33–49. Springer, 2020.

[64] Hajar Emami, Majid Moradi Aliabadi, Ming Dong, and Ratna Babu Chinnam. Spagan: Spatial attention gan for image-to-image translation. *IEEE Transactions on Multimedia*, 23:391–401, 2020.

[65] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *The 30th International Conference on Neural Information Processing Systems*, pages 469–477, 2016.

[66] Evaluating the unsupervised learning of disentangled representations. https://ai.googleblog.com/2019/04/evaluating-unsupervised-learning-of.html, 2019. [Online; accessed January, 28, 2021].

[67] Ruoqi Wei, Cesar Garcia, Ahmed El-Sayed, Viyaleta Peterson, and Ausif Mahmood. Variations in variational autoencoders-a comparative evaluation. *Institute of Electrical and Electronics Engineers Access*, 8:153651–153670, 2020.

[68] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *European Conference on Computer Vision*, pages 172–189, 2018.

[69] E Duchateau, Fleur Longuetaud, Frédéric Mothe, C Ung, D Auty, and A Achim. Modelling knot morphology as a function of external tree and branch attributes. *Canadian Journal of Forest Research*, 43(3):266–277, 2013.

[70] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[71] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456. PMLR, 2015.

[72] Pix2Pix Tensorflow. https://www.tensorflow.org/tutorials/generative/pix2pix, 2021. [Online; accessed May, 8, 2021].

[73] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.

[74] Ali Borji. Pros and cons of gan evaluation measures. *Computer Vision and Image Understanding*, 179:41–65, 2019.

[75] Rafael Padilla, Wesley L Passos, Thadeu LB Dias, Sergio L Netto, and Eduardo AB da Silva. A comparative analysis of object detection metrics with a companion open-source toolkit. *Electronics*, 10(3):279, 2021.

[76] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. Detectron. https://github.com/facebookresearch/detectron, 2018. [Online; accessed May, 12, 2021].

[77] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *International Conference on Computer Vision*, pages 2961–2969, 2017.

[78] Detectron2 Model Zoo and Baselines. https://github.com/facebookresearch/detectron2/blob/master/MODEL_ZOO.md, 2021. [Online; accessed May, 8, 2021].

[79] COCO Detection Evaluation. https://cocodataset.org/#detection-eval, 2021. [Online; accessed May, 12, 2021].