Signal, Image and Video Processing (2024) 18:6481–6489
https://doi.org/10.1007/s11760-024-03331-w

ORIGINAL PAPER

Image generation of log ends and patches of log ends with controlled properties using generative adversarial networks

Dag Björnberg[1,2] · Morgan Ericsson[1] · Johan Lindeberg[3] · Welf Löwe[1] · Jonas Nordqvist[4]

Received: 25 November 2022 / Revised: 13 January 2023 / Accepted: 29 May 2024 / Published online: 18 June 2024

Abstract
The appearance of the log cross-section provides important information when assessing the quality of the log, where properties to consider include pith location and density of annual rings. This makes tasks like estimation of pith location and annual ring detection of great interest. However, creating labeled training data for these tasks can be time-consuming and subject to misjudgments. For this reason, we aim to create generated training data with controlled properties of pith location and amount of annual rings. We propose a two-step generator based on generative adversarial networks in which we can completely avoid manual labeling, not only when generating training data but also during training of the generator itself. This opens up the possibility to train the generator on other types of log end data without the need to manually label new training data. The same method is used to create two generated training datasets; one of entire log ends and one of patches of log ends. To evaluate how the generated data compares to real data, we train two deep learning models to perform estimation of pith location and ring counting, respectively. The models are trained separately on real and generated data and evaluated on real data only. The results show that the performance of both estimation of pith location and ring counting can be improved by replacing real training data with larger sets of generated training data.

Keywords Generative adversarial network · Image generation of log ends · Training data generation · Conditional GAN · CycleGAN

1 Introduction

Multiple quality measurements of wood can be estimated from the log cross-section. One measure to take into consideration is pith location, which serves as a point of reference to determine other properties of the cross-section [1]. One such important property is annual ring width [2, 3], in which knowing the location of the pith is a prerequisite. For these reasons, tasks such as estimation of pith location and annual ring counting are interesting.

For these tasks, different computer vision technologies can be employed, e.g., classic image processing as used in [2] or supervised machine learning (ML) as used in [4]. Provided with labeled training data, i.e., images with known properties, supervised ML approaches show high accuracy. However, creating labeled training data can be cumbersome and error-prone. It can also be costly if additional resources are needed to perform such labeling tasks, especially if many datasets are to be labeled. This is not an unlikely scenario within forestry since we may consider images of log ends from various species taken in different environments.

✉ Dag Björnberg
dag.bjornberg@lnu.se

Morgan Ericsson
morgan.ericsson@lnu.se

Johan Lindeberg
johan.lindeberg@lnu.se

Welf Löwe
welf.lowe@lnu.se

Jonas Nordqvist
jonas.nordqvist@lnu.se

1 Department of Computer Science and Media Technology, Linnæus University, Växjö, Sweden

2 Softwerk AB, Växjö, Sweden

3 Department of Forestry and Wood Technology, Linnæus University, Växjö, Sweden

4 Department of Mathematics, Linnæus University, Växjö, Sweden

We examine the possibility of generating training data with controlled properties. The research question behind this is whether training data that is artificially generated can improve the accuracy of supervised ML/DL computer vision approaches, at least in the researched analysis tasks. We use deep learning (DL). More specifically, variants of generative adversarial networks [5] are the fundamental approach throughout the process.

We propose a two-step generator based on prior works on image-to-image translation [6, 7]. Its input images are drawings where we have control of the properties such as the number of annual rings and pith location. We transform these drawings into photo-realistic images maintaining the properties that serve as labeled training data. The proposed generator does not require any manual labeling during training, which makes it possible to retrain it for other types of log end data in different environments, with only minor parameter changes. The details of our method are described in Sect. 4.

The paper is organized as follows. Section 2 briefly introduces the exploited fundamental DL technology. Section 3 discusses related work. Section 4 details the suggested image generation method. Section 5 presents the results of our experimental validation of the method. Finally, Sect. 6 concludes the paper and shows directions for future work.

2 Theoretical background

2.1 Generative adversarial networks

Generative adversarial networks (GANs) have shown impressive results in image generation [5, 8]. The generated outputs are forced to be, in principle, indistinguishable from real images.

GANs are trained through an adversarial process, where the *generator* tries to mimic a real-world data distribution p_{real} by transforming a prior noise distribution p_z to outputs y, i.e., $G : z \mapsto y$. The generated outputs are then fed into a *discriminator* that tries to distinguish the generated outputs from p_{real}. The discriminator tries to minimize the error of this classification while the generator tries to maximize it. Through this adversarial training procedure, the generator will get better and better at producing realistic outputs, eventually making the discriminator unable to tell them apart from real data.

Conditional GANs (cGANs) are extensions of the standard GAN, where the generator and discriminator are provided with conditional information x [9]. This conditional information could be any kind of auxiliary information such as feature attributes [10] or real images [6]. Extending regular GANs, cGANs learn the mapping $G : \{x, z\} \mapsto y$, which makes it possible to guide the generated outputs to preserve the information provided by the condition x. The conditioning is performed by feeding x into both the generator and the discriminator as an additional input layer. Figure 2 displays the schematics of a cGAN that is trained to perform edge map → photo of entire log ends.

Image-to-image translation seeks to transfer images from a source domain to a target domain while preserving the content representations of the source domain [11]. In this paper we will perform image-to-image translation, where we adopt the methods outlined in [6] (for paired data) and [7] (for unpaired data). Unpaired image-to-image translation occurs as a natural problem when there is no paired data, i.e., no source to target image pairs, to train on.

CycleGAN consist of two generators and can address the unpaired image-to-image translation problem. The idea is that if we translate an image from source domain X to target domain Y and then translate it back from Y to X we should arrive at where we started [7]. To ensure that the generators learn the correct mappings between the domains X and Y, a *cycle consistency loss* is introduced. In other words, for the two generators, $G : X \rightarrow Y$ and $F : Y \rightarrow X$, the cycle consistency loss ensures that $F(G(x)) \approx x$ and $G(F(y)) \approx y$. Figure 3 displays the schematics of a CycleGAN that is trained to perform drawing → edge map for patches of log ends.

2.2 Objective functions

2.2.1 cGAN

Let G and D be the generator and discriminator of a cGAN respectively. Further, denote by x the conditional information, y a real image and z a random noise seed vector. The objective of a cGAN can then be expressed as:

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

The generator G aims to minimize the objective against the adversary D which tries to maximize it, i.e. $\min_G \max_D \mathcal{L}_{cGAN}(G, D)$.

It has proven useful to add regularization to the objective, in terms of the L1 or L2 distance [6, 12], enforcing the generator to not only generate realistic samples but also making the outputs closer, in L1 or L2 sense, to the corresponding real images. The discriminator's task, however, remains the same. We adopt the method in [6] and add an L1 loss to the objective function:

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

Through this adversarial procedure, we optimize the generator as:

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda\mathcal{L}_1(G). \quad (1)$$

2.2.2 CycleGAN

For the generator $G : X \to Y$ and its discriminator D_Y, the adversarial loss can be formulated as:

$$\mathcal{L}_{GAN}(G, D_Y) = \mathbb{E}_y[\log D_Y(y)] + \mathbb{E}_x[\log(1 - D_Y(G(x))].$$

The generator G tries to generate realistic samples $G(x)$ and the discriminator D_Y tries to distinguish generated samples $G(x)$ from real samples y, i.e. $\min_G \max_{D_Y} \mathcal{L}_{GAN}(G, D_Y)$.[1] Similarly, for the mapping $F : Y \to X$ and its discriminator D_X, F tries to generate realistic samples $F(y)$ and D_X tries to distinguish generated samples $F(y)$ from real samples x, i.e. $\min_F \max_{D_X} \mathcal{L}_{GAN}(F, D_X)$.

In addition to the adversarial loss, a cycle consistency loss is included in the objective function for CycleGAN. This means that for the generators G and F, we enforce that $F(G(x)) \approx x$, which we refer to as *forward cycle consistency* [7]. Similarly, G and F should satisfy *backwards cycle consistency*, i.e. $G(F(y)) \approx y$. Thus, we formulate the cycle consistency loss as:

$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_x[||F(G(x)) - x||_1] + \mathbb{E}_y[||G(F(y)) - y||_1].$$

Similar to the cGAN regularization with L1 or L2 norms, it is suggested to introduce a so-called *identity loss* in order to enforce that the generators are near identity mappings when provided with samples from target domain as input [13]:

$$\mathcal{L}_{id}(G, F) = \mathbb{E}_x[||F(x) - x||_1] + \mathbb{E}_y[||G(y) - y||_1].$$

To conclude, the full objective can be expressed as:

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) = &\mathcal{L}_{GAN}(G, D_Y)\\ &+\mathcal{L}_{GAN}(F, D_X)\\ &+\lambda_0\mathcal{L}_{cyc}(G, F)\\ &+\lambda_1\mathcal{L}_{id}(G, F),\end{aligned} \quad (2)$$

[1] In this paper we use the original implementation of CycleGAN [7], with the exception that we use the cGAN generator in [6]. This implies an implicit random noise prior z in the form of dropout in the networks of our generators. However, to follow the standard presentation of CycleGAN we do not include this in our presentation.

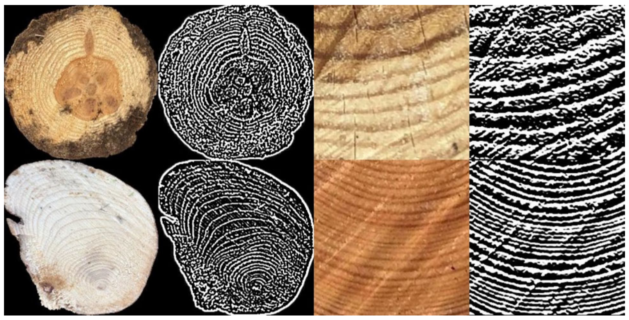


Fig. 1 Examples of our proposed edge maps. LHS: Laplacian edge maps of entire log ends. RHS: Sobel edge maps of patches. Instead of manually creating paired data we will use these pairs when training the cGAN

in which we aim to solve:

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

2.3 Edge detection

Edge detection is an image processing technique used to find boundaries of objects within images by detecting discontinuities in brightness [14]. By sending an image through an edge detector, an *edge map* is produced which contains important information about the structure of the image itself.

Since we are interested in generating images with controlled properties like number of rings and pith location from controlled schematic drawings, it is important that we find edge detectors that preserve these properties as well as possible. In this way we can condition on the important features we are interested in without manually creating training data.

For both entire log ends and patches, we convert the images to gray-scale and use Gaussian blur in order to smooth the images. We find that for entire log ends, the Laplacian edge detector [15] is suitable and at the patch level the Sobel edge detector [16] works well, see Fig. 1.

3 Related work

Similar approaches to ours have been used in order to make quality assessments of boards. In [17], conditional GANs are used to create binary representations (drawings) of Norway spruce timber boards for automatic detection of annual rings and pith location. In [4], a cGAN is trained to create virtual boards from binary representations, to be used as training data for estimation of pith location. Unlike our generative model, both of these approaches require manual effort to create paired training data for the cGAN.

As in our framework, previous work have been made with edge maps based on known edge detection algorithms as

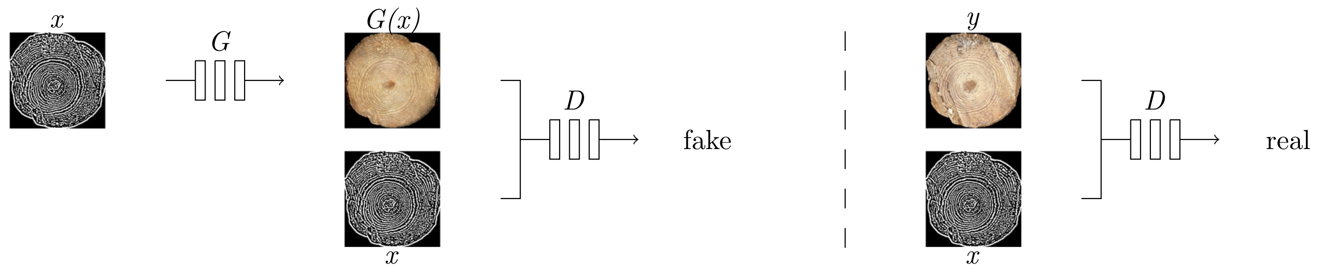


Fig. 2 Training of a cGAN to perform edge map → photo of entire log ends. The generator learns to fool the discriminator, and the discriminator learns to distinguish between real and fake samples. Note that both the generator and the discriminator are provided with the conditional information x

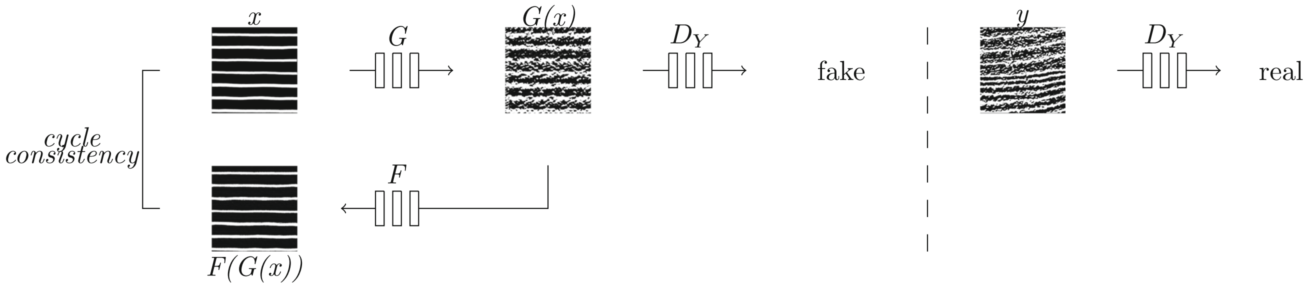


Fig. 3 Training of CycleGAN for unpaired image-to-image translation between drawings and edge maps for patches

training data for GANs. In [18] a very similar approach to ours is used to perform face age translation, utilizing both a CycleGAN and a cGAN. Young and old faces are converted to edge maps with the Canny edge detection algorithm [19], and a CycleGAN is used to perform image-to-image translation between the two domains. Conditional GANs are trained to perform the mappings young face edge map → young face and old face edge map → old face.

Image generation of log ends has been considered earlier. In [20] the author creates a stochastic model to generate images of Scotch pine log ends with certain controlled properties. These include pith location and annual ring patterns, but also presence of knots, heartwood, colour and sawing patterns. The difference compared to our approach is that all these features are modeled explicitly, using gathered statistics and visual assessment.

4 The image generation process

4.1 Our proposed method

We propose a two-step generator where we use both paired and unpaired data during training. First, we consider an unpaired image-to-image translation between drawings and edge maps using a CycleGAN [7]. Then, we consider using paired data of real images and their corresponding edge maps

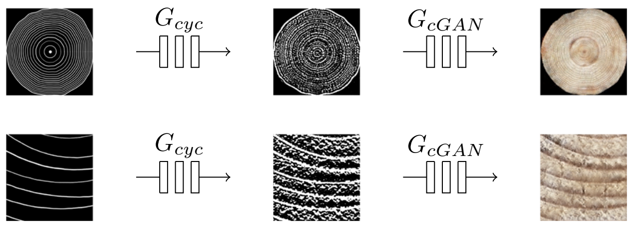


Fig. 4 Our two-step generator

using a cGAN [6]. Through this procedure, we create a training scheme in which we can completely avoid manual labeling. Thus, if we want to generate images from different log ends or patches from a different environment, we argue that we can use the same method only with minor changes in the drawings and edge map parameters.

The training phases are outlined in the Figs. 2 and 3. When training is completed, we chain the two generators together and generate images according to the principle schematic drawing → generated edge map → generated image, see Fig. 4.

Initial experiments of applying solely CycleGAN to perform schematic drawing → generated image was tested. However, visually unsatisfactory results, and on the other hand the promising results of using CycleGAN as an intermediate step prior to cGAN was the reason for using our suggested approach.

4.2 Network architectures

We will use the same network architectures during both training phases. Precisely as in [6], the generators are encoder-decoders with skip connections, following the shape of a "U-Net" [21]. For the discriminators, we use a 70 × 70 *PatchGAN* classifier, developed in [6], that classifies each 70 × 70 patch of an image as either real or fake.

4.3 Model for generating drawings of log ends and patches

The shape and ring width of a log end depend on several factors like climate, genetics of the tree and presence of other trees in close surroundings [4]. Thus in real trees the shapes are not cylindrical and the distances between annual rings are not constant. In general, annual rings are thicker closer to the pith and thinner at larger distances from pith. In order to capture this behaviour, we create a stochastic model in which we create circles around pith using a square-root fit with added noise, i.e. the radius r from ring number x to pith, is modelled as $r = a_0 + a_1\sqrt{x} + \epsilon$. The model is based on data from 401 trees harvested from a Norway spruce field trial, located at Tönnersjö in south-west Sweden (56.66°N, 13.09°E; altitude, 90 m above sea level). A more detailed explanation of the dataset is found in [22].

To capture variety of shape and pith location we add several "bumps" to the circles, meaning that we add some shape variety locally, in such a way that we reach realistic results in terms of shape of the entire log end and pith location. For a more extensive example of a stochastic model for generating drawings of log ends we refer to [4].

For the patches, we use a very simple approach, where the rings are generated by a second degree polynomial function with varying parameters in order to create realistic variations. For patches including pith we simply generate a half-circle with variations in radius and crop it randomly around the pith. To further increase the variety of the ring structure we apply elastic transformation [23] to our patches drawings.

4.4 Training details

For the two training phases, we use the same settings as in the original papers [6, 7], but replacing the generator in [7] with the generator in [6]. For both our experiments we use a batch size of 1.

For our cGAN experiment, we train for a total number of 200 epochs (for entire log ends) and 20 epochs (for patches). While training our CycleGAN, we faced issues with unstable training and mode collapse. We therefore decided to make an early stopping during training (after 20 and 2 epochs, respectively), knowing well that the generator may not be optimized, and leaving it as a future work on how we could improve the training of our CycleGAN.

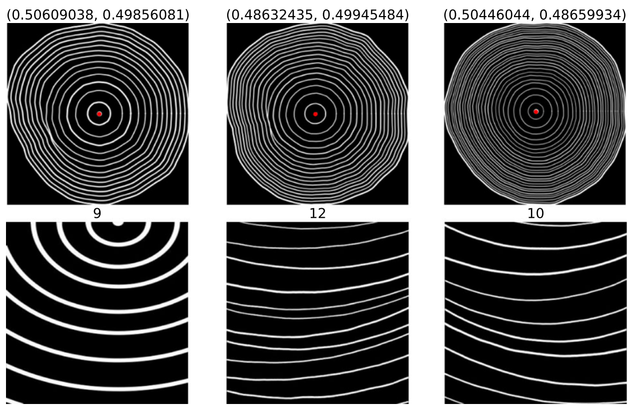


Fig. 5 Drawings of log ends and patches of log ends with controlled properties. For drawings of log ends the pith location (here marked with red) is expressed in normalized (x, y) coordinates. For drawings of patches, the labels are the number of rings

5 Experiments

Since we want to investigate the performance of deep learning approaches when trained on generated data, we consider two proofs of concept: pith estimation and ring counting, where we create baseline models using real data and then compare its performance to models being trained on generated data. We decide to measure the performance in terms of validation error on real data. To conclude, we use the following approach:

(i) generate 10.000 images.
(ii) create a baseline model using real training data, and store the validation error.
(iii) take a random sample of generated images, and create models by training with increasing subsets of this sample. Store the validation error for every model.
(iv) repeat steps (ii–iii) 10 times.

5.1 Baseline models and training details

We decide to create baseline models using transfer learning [24], based on the efficientnetb0 architecture [25], with 150 layers frozen for both cases. For pith estimation, we avoid overfitting by applying several augmentations to our training data, in our case we found that 70 augmentations to each image worked well. For ring counting at the patch level, we did not find this useful, and the ability to add augmentations like translation is limited to the fact that labels will change by that augmentation. Instead, we increase the dropout parameter from 0.2 to 0.95, and apply 20 augmenta-

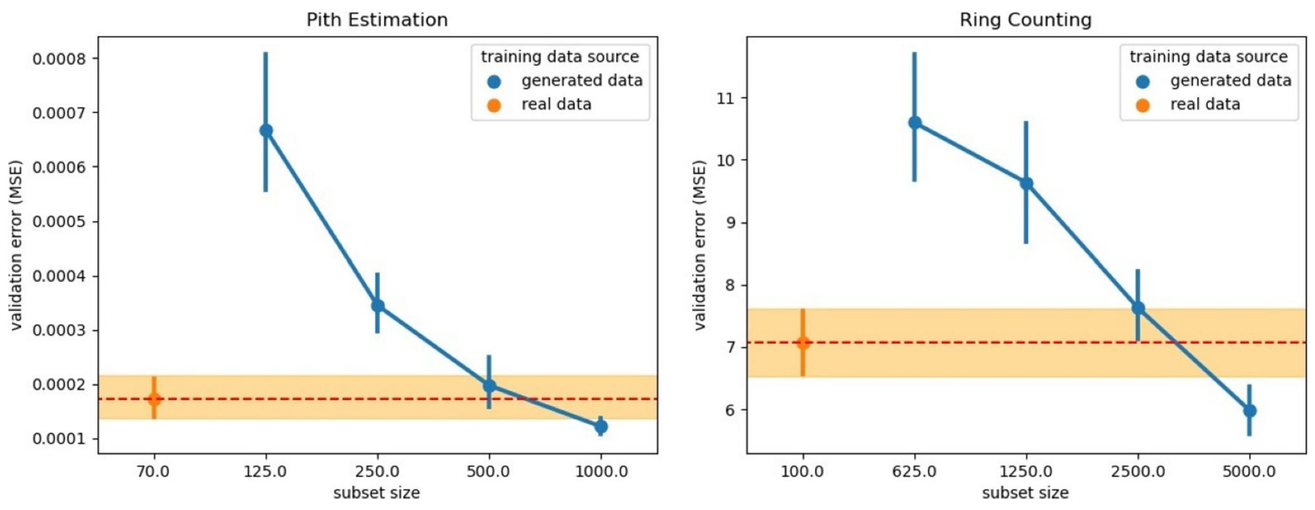


Fig. 6 Comparison of performance using real training data and generated training data for two different tasks; pith estimation (to the left) and ring counting (to the right). The performance is measured in terms of lowest validation error on real data. The bars represent the 95% confidence intervals

tions to each image in order to stabilize training. We use 50 epochs for every training iteration, with patience = 7 as an early stopping criterion. In both cases a fixed learning rate is used (1e−04 for pith estimation and 1e−05 for ring counting). After every training iteration the lowest validation error is stored, i.e. the model selected is the one which gives the lowest validation error.

The same settings are used when creating models using generated training data, except for the number of augmentations that we lower to 10 and 1, for pith estimation and ring counting, respectively.

5.2 Dataset selection

We consider two distinct datasets, both consisting of Norway spruce and Scotch pine.

For entire log ends, we have a total dataset size of around 900 images of Norway spruce and Scotch pine. We clean this dataset by removing log ends that are—by the authors' quick assessment—more than 30 years old, to ensure that the ring structure is visible in 256×256 resolution. We also remove log ends covered with too much dirt, spray color, snow etc., leaving us with 242 log end images. Out of these 242 images, we use 70 images to train our baseline model, 50 images for validation and the remaining 122 images are fed into the generator training phase.

At the patch level, we have patches from a total number of 31 different log ends (12 Norway spruce and 19 Scotch pine), roughly between the ages of 60 and 120, divided into patches. This results in around 5500 patches in total. We clean the dataset by removing patches which we cannot label by counting the amount of rings on them. We also remove patches containing bark. This leaves us with a total number of 1674 patches from 31 log ends. We decide to use 25 trees for training and generation and the remaining 6 trees (4 Scotch pine and 2 Norway spruce) for validation. We take 4 patches from each of the 25 trees for training and generation, leaving us with 100 patches that are used to train our baseline model. The remaining patches (1138) are used to train the generator. Out of the 6 trees used for validation, we select 8 or 9 patches from each tree to a total number of 50 patches to be used as validation data.

5.3 Results

Initial experiments indicated that we needed around 1000 generated images (for pith estimation) and 5000 generated images (for ring counting) to be able to improve the performance compared to our baseline models. Therefore, we decided to take random samples of 1000 and 5000 generated images in step (ii), respectively. In order to track the improvement of the performance for generated data we decided to use subset sizes of 125, 250, 500, 1000 and 625, 1250, 2500, 5000, respectively. The results shown in Fig. 6 strongly indicates that we were able to improve performance for both cases using generated training data compared to real training data when increasing the size of the generated training dataset.

6 Discussion and future work

We have created a two-step generator by adopting the methods in [6] and [7]. Training this generator consists of both training a cGAN (for paired training data) and training a CycleGAN for unpaired training data. Instead of manually creating the pairs used in the cGAN we propose the idea

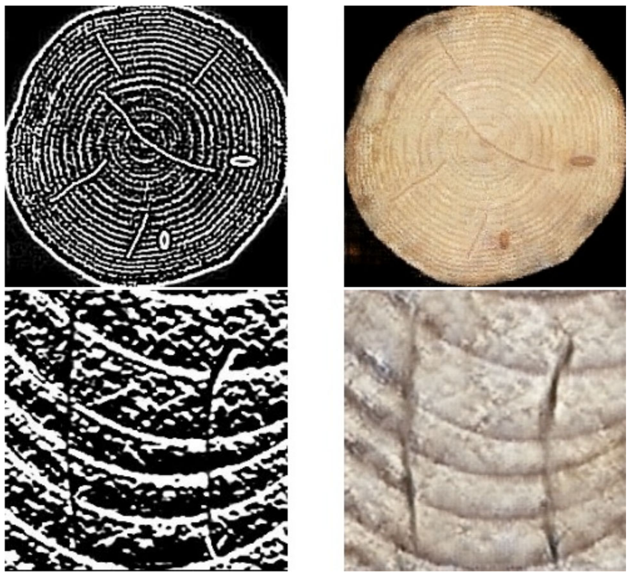


Fig. 7 Examples of added perturbations to the generated images. A perturbations is added to the generated edge map. We add black or white lines to the generated edge maps to represent cracks, and oval structures to represent knots. These representations translate into quite realistic representations of cracks and knots in the generated image

of using edge maps based on already known edge detection algorithms.

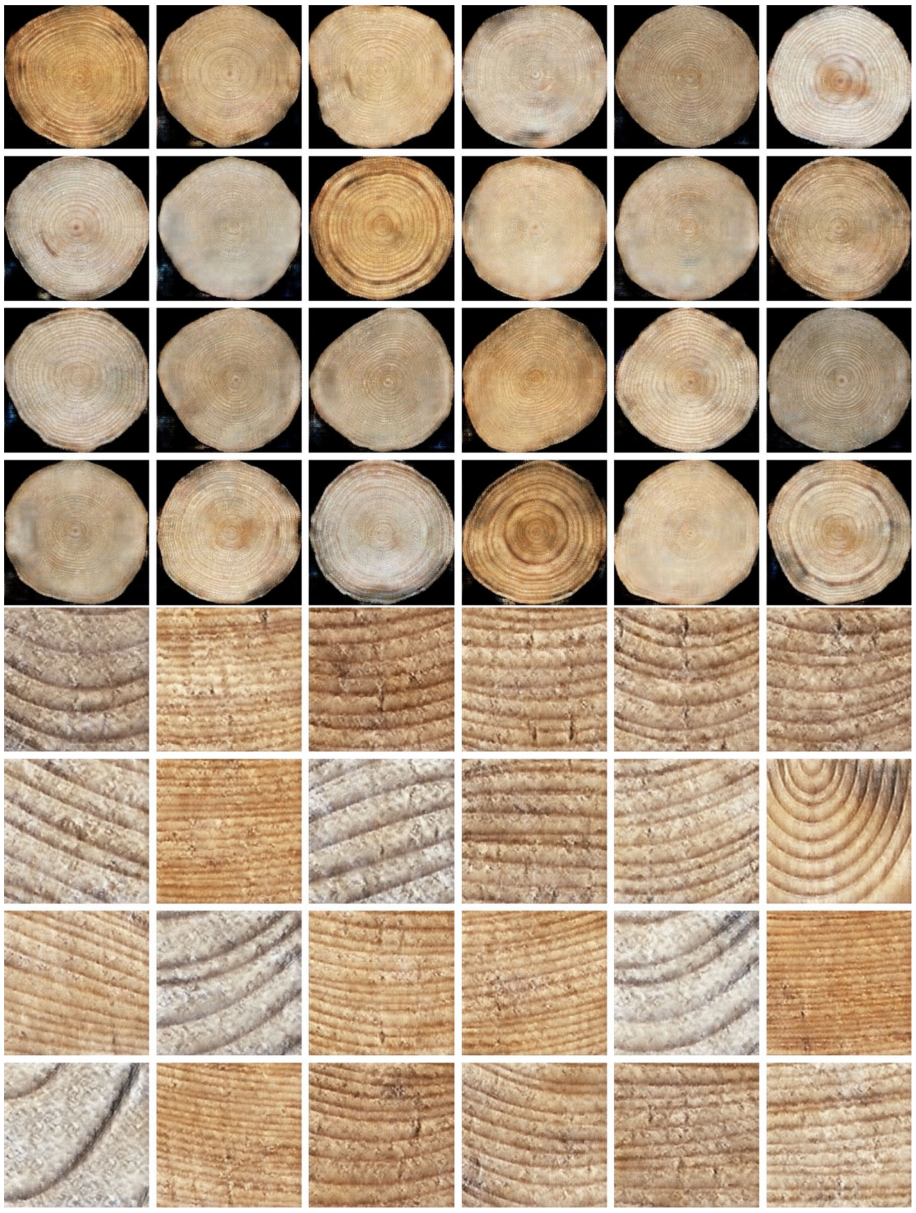
We explore the possibility of improving the performance of deep learning approaches with generated training data by considering two proofs of concept; pith estimation and ring counting.For both of these tasks, we were able to improve the performance using generated training data compared to real training data. However, this comes at the cost of more training data. On the other hand we can completely avoid manual labeling, which could be useful not only for the two tasks presented here but also for other potential tasks like annual ring detection, where labeling training data can be very time-consuming.

There are alternative methods to our suggested approach. We could for instance have considered a more traditional image processing technique, as in [20], using gathered statistics from the images to generate synthetic data. We could also have considered a semi-automatic approach as in [4] where we only label a smaller dataset for training of the generator. However, our method can directly be applied to other environmental settings and types of logs with only minor parameter changes.

Previous work [6] have shown that designing cGANs that can capture the full entropy of the conditional distributions they model is a challenging problem. This is in line with our visual assessment of the generated images where we think the generated samples are photo-realistic but do not have the same variety as the real samples. It is therefore plausible that this is why we needed more generated images compared to real images to be able to improve the performance of the two machine learning tasks used as proofs of concept.

Interesting future works include the above mentioned ring detection task, where we could consider to train a cGAN (as in [17]) in order to see if we could generate binary representations from real patches, but instead using generated data with corresponding drawings as training data. Another (more unrelated task) would be to examine the possibility of adding realistic perturbations to the images, which could be useful for image recognition of log ends, as outlined in [26], or to further diversify the generated datasets. Since interesting features like cracks and knots can be preserved by the edge maps, we might add perturbations to the generated edge map (or the drawing), thus creating perturbations to the final image, see Fig. 7.

Appendix A Examples of Generated Images



Acknowledgements We would like to thank Rafael Messias Martins, Anders Olsson and Osama Abdeljaber for their helpful comments on the paper.

Author Contributions DB conducted research and wrote the paper. JN participated in the writing. All authors participated in the research and reviewed the manuscript.

Funding Open access funding provided by Linnaeus University. The work has been funded by the Industry Graduate School on "Data Intensive Applications (DIA)" at Linnaeus University which is partially funded by the Knowledge Foundation (project id 20190336).

Data Availability Samples of 1000 generated images for both log ends and patches of log ends with corresponding labels are available at https://github.com/dagbjornberg/Image-Generation-Log_Ends.

Declarations

Conflict of interest All authors declare that they have no Conflict of interest.

Ethical approval Not applicable.

References

1. Norell, K., Borgefors, G.: Estimation of pith position in untreated log ends in sawmill environments. Comput. Electron. Agric. **63**(2), 155–167 (2008)
2. Norell, K.: Automatic counting of annual rings on *Pinus sylvestris* end faces in sawmill industry. Comput. Electron. Agric. **75**(2), 231–237 (2011)
3. Österberg, P., Ihalainen, H., Ritala, R.: Measurement of wood quality parameters from annual rings using color analysis with digital images (2011)
4. Habite, T., Abdeljaber, O., Olsson, A.: Determination of pith location along Norway spruce timber boards using one dimensional convolutional neural networks trained on virtual timber boards. Constr. Build. Mater. **329**, 127129 (2022)
5. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks. Adv. Neural Inf. Process. Syst. (2014). https://doi.org/10.1145/3422622
6. Isola, P., Zhu, J.-Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1125– 1134 (2017)
7. Zhu, J.-Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2223– 2232 (2017)
8. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv e-prints, 1511–06434 (2015). arXiv: 1511.06434 [cs.LG]
9. Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv e-prints, 1411–1784 (2014). arXiv: 1411.1784 [cs.LG]
10. Woldesellasse, H., Tesfamariam, S.: Prediction of lateral spreading displacement using conditional generative adversarial network (cGAN). Soil Dyn. Earthq. Eng. **156**, 107214 (2022)
11. Pang, Y., Lin, J., Qin, T., Chen, Z.: Image-to-image translation: methods and applications. IEEE Trans. Multimed. (2021)
12. Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., Efros, A.A.: Context encoders: Feature learning by inpainting. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2536– 2544 (2016)
13. Fang, Y., Deng, W., Du, J., Hu, J.: Identity-aware CycleGAN for face photo-sketch synthesis and recognition. Pattern Recognit. **102**, 107249 (2020)
14. Hu, X.-X., Kou, K.I.: Phase-based edge detection algorithms. Math. Methods Appl. Sci. **41**(11), 4148–4169 (2018)
15. Wang, X.: Laplacian operator-based edge detectors. IEEE Trans. Pattern Anal. Mach. Intell. **29**(5), 886–890 (2007)
16. Nausheen, N., Seal, A., Khanna, P., Halder, S.: A FPGA based implementation of Sobel edge detection. Microprocess. Microsyst. **56**, 84–91 (2018)
17. Habite, T., Abdeljaber, O., Olsson, A.: Automatic detection of annual rings and pith location along Norway spruce timber boards using conditional adversarial networks. Wood Sci. Technol. **55**(2), 461–488 (2021)
18. Wang, Z., Liu, Z., Huang, J., Lian, S., Lin, Y.: How Old Are You? Face Age Translation with Identity Preservation Using GANs. arXiv e-prints, 1909–04988 (2019). arXiv: 1909.04988 [cs.CV]
19. Song, R., Zhang, Z., Liu, H.: Edge connection based canny edge detection algorithm. Pattern Recognit. Image Anal. **27**(4), 740–747 (2017)
20. Norell, K.: Creating synthetic log end face images. In: 2009 Proceedings of 6th International Symposium on Image and Signal Processing and Analysis, pp. 353– 358. IEEE (2009)
21. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical Image Computing and Computer-assisted Intervention, pp. 234– 241. Springer (2015)
22. Hallingbäck, H.R., Högberg, K.-A., Säll, H., Lindeberg, J., Johansson, M., Jansson, G.: Optimal timing of early genetic selection for sawn timber traits in *Picea abies*. Eur. J. Forest Res. **137**(4), 553–564 (2018)
23. Simard, P.Y., Steinkraus, D., Platt, J.C., et al: Best practices for convolutional neural networks applied to visual document analysis. In: Icdar, vol. 3. Edinburgh (2003)
24. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Trans. Knowl. Data Eng. **22**(10), 1345–1359 (2009)
25. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning, pp. 6105– 6114. PMLR (2019)
26. Wimmer, G., Schraml, R., Hofbauer, H., Petutschnigg, A., Uhl, A.: Two-stage CNN-based wood log recognition. In: International Conference on Computational Science and Its Applications, pp. 115– 125. Springer (2021)