Procedural Tree Generation

Efficiently predict branching structures from foliage

Sam Taklimi

Supervisor(s): Prof. Dr. Elmar Eisemann, Dr. Petr Kellnhofer, Lukas Uzolas

EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
January 28, 2024

Name of the student: Sam Taklimi
Final project course: CSE3000 Research Project
Thesis committee: Prof. Dr. Elmar Eisemann, Dr. Petr Kellnhofer, Dr. Marcel Reinders, Lukas Uzolas

Abstract

The objective of this project is to train a model that transforms a tree with its foliage into only its branch structure. This is achieved by employing machine-learning techniques, specifically Generative Adverserial Networks (GANs). By utilizing the proposed method, a predictive model is built that automatically minimizes its own error function through a comparison of a set of input and ground-truth tree images, which are tree images with and without leaves, respectively. The adoption of GANs has shown promising results, both visually and metrically.

Figure 1: Transforming trees from leaves to branches.

1 Introduction

The intricate morphology of trees plays an important role in ecological studies, environmental monitoring, and landscape analysis [12], urban planning and forestry management [3], and biodiversity assessment. Traditional methods of branch identification and mapping often rely on labor-intensive field surveys, making them time-consuming and resource-intensive. Recent advancements in the domain of Convolutional Neural Networks (CNNs) have presented opportunities for automating the analysis of tree structures which is the basis of this paper, namely the algorithm laid out in the Pix2Pix paper [6] in which they investigate conditional adversarial networks as a general-purpose solution to image-to-image translation problems. These networks not only learn the mapping from input image to output image, but also learn a loss function to train this mapping.

The specific point of focus in our research is correctly identifying branches by separating them from the foliage. This is going be to done on a 2-D level.

This paper addresses the research question of efficiently predicting branching structures from foliage. It successfully outlines a custom specialized version of the Pix2Pix algorithm for accurately separating tree branches from foliage, representing a crucial step in further analyzing tree skeletons.

2 Related Work

In the exploration of image-to-image translation algorithms for the primary research objective, which involves the separation of leaves from foliage, several established methods employing machine learning techniques were considered.

The Pix2Pix paper [6] emerged as a prominent solution when paired training data is accessible. Employing a conditional Generative Adversarial Network (cGAN), Pix2Pix excels in crafting lifelike images for intricate tasks like isolating leaves from trees. The utilization of adversarial training further amplifies visual realism by instructing the generator to yield results virtually indistinguishable from authentic images in the target domain. This aligns seamlessly with our objective of transforming trees from one domain (with leaves) to another (without), making this algorithm a fitting choice for our case.

Focused on unsupervised image-to-image translation, Liu et al [8] emphasizes learning a joint distribution of images in different domains. It introduces a shared-latent space assumption and employs Coupled GANs for the translation framework. Comparisons with other methods showcase high-quality results in challenging tasks such as street scene, animal, and face image translation.

Zhu et al [14] addresses the scenarios where aligned image pairs are unavailable for training, this approach aims to learn a mapping from a source domain (X) to a target domain (Y) without paired examples. It introduces mappings G: X $\rightarrow$ Y and F: Y $\rightarrow$ X, coupled with a cycle consistency loss to ensure F(G(X)) is X and vice versa. The method demonstrates superior performance in tasks without paired training data, emphasizing its adaptability across various domains.

3 Preliminaries

Unlike traditional approaches requiring hand-engineered loss functions, cGANs automatically learn a loss function, making them adaptable to a broad spectrum of problems without the need for specialized formulations. The Pix2Pix research demonstrates the success of cGANs in synthesizing photos from label maps, reconstructing objects from edge maps, and colorizing images [13]. Furthermore, the Pix2Pix paper addresses the commonality in image processing tasks—predicting pixels from pixels—and seeks to establish a unified framework for various image translation challenges.

The generator of the Pix2Pix conditional Generative Adversarial Network (cGAN) takes the form of a modified U-Net, comprising an encoder (downsampler) and decoder (upsampler). The encoder consists of blocks involving Convolution, Batch normalization, and Leaky ReLU, while the decoder incorporates Transposed convolution, Batch normalization, Dropout (applied to the first three blocks), and ReLU. Skip connections between the encoder and decoder, akin to a U-Net architecture, are present.

The generator loss (Gen-Loss) is the sum of the adversarial loss (Gan-Loss) and the L1 loss (L1-Loss) scaled by a hyperparameter (LAMBDA). The L1 loss measures the absolute difference between the generated and target images, while the Gan-Loss represents the success of the adversarial training process. The LAMBDA value is a constant multiplier set by the Pix2Pix authors, typically at 100.

A discriminator is defined to receive two inputs: the input image and the target image. The target image is classified as real, while the input image, which is the generated image (output of the generator), is classified as fake.

4 Method

We build our approach on the Pix2Pix framework [1] to extract the branching and trunk structure from the foliage.

Given that the Pix2Pix algorithm is a versatile and general domain-independent method, we achieved reasonable results by applying it to our dataset with minimal adjustments. To tailor the Pix2Pix algorithm to our specific case, we decided to run experiments by replacing the original loss function as explained below.

The Loss Function

The Pix2Pix algorithm [6] originally uses the BinaryCrossEntropy method to calculate the loss between the input and the output. However, in order to optimize the results further, we considered experimenting with different loss functions namely: BinaryCrossEntropy, BinaryFocalCrossEntroy and MeanAbsoluteError, which were selected to see how the results would change.

Binary Cross Entropy (BCE) [9] is a common loss function used in binary classification tasks. It is often employed as the pixel-wise loss to measure the difference between the generated image and the ground truth image.

The Binary Cross Entropy loss for a single pixel is given by:

$$\text{BCE}(y,t) = -\frac{1}{N}\sum_{i=1}^{N}[t_i \log(y_i) + (1-t_i)\log(1-y_i)]$$

where

N is the number of elements in the input tensors, which is in our case 256 × 256, resulting in 65,536.

y is the predicted probability of the pixel belonging to the positive class (foreground).

t is the true label (ground truth) of the pixel (0 for the negative class/background, 1 for the positive class/foreground).

Binary Focal Cross Entropy (BFCE) [7] is an extension of Binary Cross Entropy designed to address the issue of class imbalance in binary classification tasks. BFCE can be employed as a pixel-wise loss to further improve the model's handling of difficult-to-classify examples, such as ours specifically since there can be a lot of noise and overlap concerning the edges of the branches and the leaves for each tree.

The Binary Focal Cross Entropy loss is given by:

$$\text{BFCE}(p,y) = -\alpha\cdot(1-p)^{\gamma}\cdot\log(p) - (1-\alpha)\cdot p^{\gamma}\cdot\log(1-p)$$

where

p is the predicted probability of the pixel belonging to the positive class (foreground).

y is the true label (ground truth) of the pixel (0 for the negative class/background, 1 for the positive class/foreground).

α is the balancing parameter, controlling the balance between positive and negative classes.

γ is the focusing parameter, determining the amount of focus on hard-to-classify examples.

Mean Absolute Error (MAE) is a regression loss commonly used to measure the average absolute difference between predicted and true values. MAE can be used to assess the pixel-wise difference between the generated image and the ground truth image.

The Mean Absolute Error loss for a single pixel is given by:

$$\text{MAE}(y,t) = \frac{1}{N}\sum_{i=1}^{N}|y_i - t_i|$$

where

y_i is the predicted value for the i-th pixel.

t_i is the true (target) value for the i-th pixel.

N is the total number of pixels, which is in our case 256 × 256, resulting in 65,536.

MAE is a suitable choice for regression tasks, where the goal is to generate images with pixel values that are close to the ground truth.

We developed a function for generating weighted maps, taking an image as input and converting it into a TensorFlow tensor. The function utilizes thresholding to create a binary mask, designating pixels above the threshold as 255 (white) and others as 0 (black). The weighted map is then formed by combining this binary mask with foreground and background weights. In our application, this weighted map is generated for each image produced by the generator during the training process. It is then utilized in the Adversarial Network, when BinaryFocalCrossEntropy (BFCE) is employed.

The model was trained for each one of the mentioned loss functions separately, with 20k steps for each model. The training model was performed on the same test set, consisting of 10 Maple trees, 5 Acacia trees and 5 Birch trees. The results of this experiment will follow in the next section.

Similarity Measurement

To specifically be able to see the similarity between predicted images and its ground truth, we adopted 3 different ways of finding the similarity between two images such as Hausdorff (HD) [5] , Structural Similarity Index (SSIM) [11] and Mean Squared Error (MSE).

Hausdorff (HD) [5] the Hausdorff distance is a measure used to assess the similarity or dissimilarity between two sets, often applied to binary images, each image is represented as a set of points, for each point in one set (image), it finds the closest point in the other set and measures the distance between these pairs of points. It identifies the maximum distance among all point pairs. The final Hausdorff distance is the larger of the two maximum distances. It quantifies the dissimilarity between the two images, considering both directions.

Similarity Index (SSIM) [11] the Structural Similarity Index (SSIM) is a metric used to quantify the similarity between two images. It takes into account three components: luminance, contrast, and structure. SSIM evaluates how well the luminance (brightness) of corresponding pixels in the two images align. The metric assesses the similarity of contrast, measuring how the local patterns of intensity variations in the images resemble each other. SSIM also looks at the structural information in the images, examining how edges and other high-frequency details are preserved. The SSIM index ranges from -1 to 1. A value of 1 indicates perfect similarity, while lower values indicate increasing dissimilarity.

Mean Squared Error (MSE) is a metric commonly used to measure the difference between two images. MSE calcu-

lates the squared difference between the pixel values of corresponding pixels in the two images. The squared differences are then summed up for all pixels. The total sum is divided by the total number of pixels to obtain the mean squared difference. A lower MSE value indicates a smaller average difference between the images, suggesting greater similarity. Conversely, a higher MSE implies a larger average difference and, therefore, greater dissimilarity.

5 Experimental Setup and Results

Datasets

We employed TreeIt [2] software to generate all the images of the trees.

We created a large dataset containing 400 images of trees, with and without leaves. This dataset includes 200 images of trees with leaves and 200 images of trees without leaves, all sized 256*256 pixels. The images with leaves serve as input, while those without leaves act as ground truth. Additionally, our dataset includes validation data with 50 mixed trees (25 with leaves and 25 without leaves) representing all three tree species. We have a separate test set containing 40 trees (20 with leaves and 20 without leaves), representing all three tree species, which will be used after training to evaluate the success of our trained model. It's important to note that these test sets are not used during training and exclusively serve to assess the model's performance.

The dataset includes three types of trees to demonstrate the flexibility of our approach across different species of trees. Those are: 200 Maple trees (100 with leaves and 100 without leaves), 100 Acacia trees (50 with leaves and 50 without leaves) and 100 Birch trees (50 with leaves and 50 without leaves). Each image with leaves corresponds to a specific tree without leaves, forming pairs where the input image (with leaves) matches the ground truth image (without leaves) for the same tree. These trees in the images varied from each other by the number of leaves, camera angle, and number of branches. Notably, all the data used for this study is derived from our internally generated dataset, and no external data sources were used.

Setup

We employed Google Colab to utilize the Pix2Pix GAN [6]. The CPU is Intel(R) Xeon(R) CPU @ 2.20GHz, with 2 cores. The System memory is 12.7 GB. The GPU is Tesla T4 with 16GB of GDDR6 memory and 2,560 CUDA cores. This allowed for successful training with Pix2Pix [1] after minor adjustments.

Here is the link to our GitHub repository [10] to access our generated data (trees with and without leaves).

Results

As we start experimenting, we found out that SSIM was not to be able to correctly assess the similarity of the predicted image and ground truth, while MSE and HD were found to be the best options. The reason for this conclusion was that SSIM always produced a high similarity index (more than 0.9, indicating high similarity) even if the images were not visually similar. However, MSE and HD were quite acceptable as they provided more accurate results. Another reason for this conclusion was that MSE and HD were the only metrics that noticed the difference when comparing the predicted image of an input with a completely different tree image. Therefore, we adopted MSE and HD to check the similarity between the images as evaluation metrics.

Table 1 and Figure 2 illustrate the results of the same predicted image with different ground truths, emphasizing the significant differences between HD and MSE. Meanwhile, SSIM consistently indicates high similarity, evident in both numerical values and visual representation in the figure.

Table 1: Comparing predicted image with different ground truth and it's own ground truth.

Predicted image with different ground truth			
MSE	SSIM	HD	Tree
5.825	*0.92*	**4143.363**	Acacia-Maple
Predicted image with its own ground truth			
MSE	SSIM	HD	Tree
1.304	*0.96*	**600.772**	Acacia-Acacia

Figure 2: Comparing similarities visually.

Table 2 displays how well the predicted images match the actual ones using different loss functions. The model underwent 20K steps of training for each loss, all using the same training datasets. Post-training, we evaluated the model on the same sample test for result comparison.

Table 3 illustrates the comparison between ground truth and predicted images using BFCE with a weighted map. The inclusion of a weighted map improves results by assigning a higher weight to the foreground. This experiment utilized the same training data as the previous one, and after training, the model was tested on the same sample set for meaningful comparison.

As evident from Table 2, it can be concluded that Binary and Focal Cross Entropy yield comparable results. However,

Table 2: Results from different loss on the same sample test consist of Maple, Birch, and Acacia.

BinaryCrossEntropy		
MSE	HD	Tree
2.188	882.393	Acacia
2.720	788.440	Birch
3.206	773.173311	Maple
BinaryFocalCrossEntropy		
1.405	**794.134**	Acacia
1.677	**769.324**	Birch
1.786	**774.437**	Maple
MeanAbsoluteError		
1.814	818.385	Acacia
2.225	782.424	Birch
2.478	742.211	Maple

Table 3: Results from adding weighted map to BFCE of Maple, Birch and Acacia.

BinaryFocalCrossEntropy		
0.844	**730.499**	Acacia
1.486	**731.255**	Birch
1.230	**746.859**	Maple

when the weighted map is applied to Binary Focal Cross Entropy, as seen in Table 3, even better results are achieved.

Figure 3 showcases three different tree species used in Table 2 and Table 3 for our experiments on the same test set. This enables us to evaluate the results across different loss functions and make comparisons. Specifically, Figure 3 illustrates the performance of the trained model using Binary Focal Cross Entropy (BFCE) with a weighted map.

Figure 4 displays the results obtained after 20,000 training loops/steps, showcasing the trained model's performance on the test set. The figure represents from left to right the number of loops, the input image, its ground truth, and the predicted image based on the trained model. As shown in the figure, the model improves its predictions as it undergoes more training. Initially, the predictions may appear noisy, but with increased training, the quality of predictions improves.

Figure 5 displays the results of 20,000 training loops/steps, using BinaryCrossEntropy loss on the test sets. The figure represents from left to right the number of loops, the input image, its ground truth, and the predicted image based on the trained model.

Figure 6 displays the results of 20,000 training loops/steps, using BinaryFocalCrossEntropy loss on the test sets. The figure represents from left to right the number of loops, the input image, its ground truth, and the predicted image based on the trained model.

Figure 3: Results of 20k steps of training BFCE with weighted map

6 Discussion

Limitations and Future Work

Following our initial analysis, the branches are better prepared for our colleagues to start their analysis, focusing on branches and using a grammar and geometric analysis. For example, 'Inverse Procedural Modeling of Branching Structures by Inferring L-Systems' [4] is a method that extracts the grammar of a tree from its branches, which are the most powerful and correct ways to create automatic digital content in computer graphics.

As part of future work, improvements can be made to extend the analysis to 3D, as our current focus is on 2D. Additionally, exploring other loss functions can help evaluate how we can enhance the performance of both the generator and discriminator for improved results.

Conclusion

Quite expectedly, the domain-independent Pix2Pix algorithm proved to be a promising method for predicting branching patterns from foliage. However, for increased versatility, it is essential to expand the dataset, including a broader range of tree species. Notably, our use of Binary Focal Cross Entropy demonstrated improved efficiency in predicting branches from foliage in this study.

Ethical Considerations

Since our data consists of generated tree images, there are no ethical concerns within our case study. However, it's important to note that the algorithm can generate fake images re-

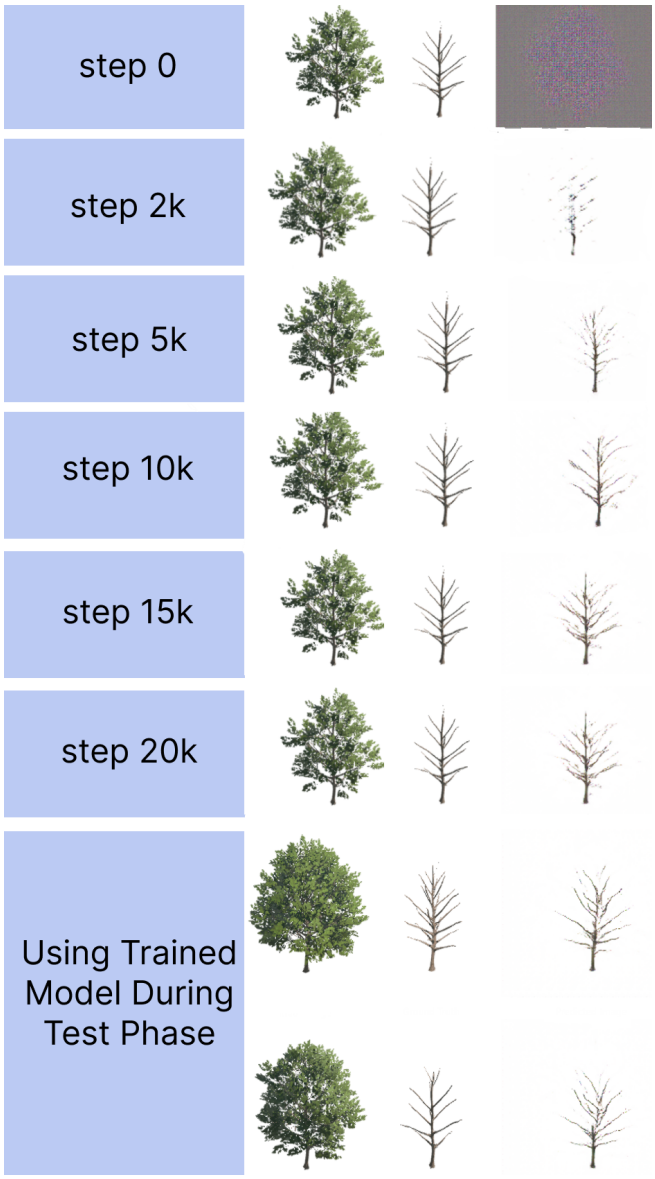

Figure 4: 20k steps of training with the original BCE loss function and its performance on test sets

Figure 5: Results of 20k steps of training with MAE as loss function

Figure 6: Results of 20k steps of training with BFCE as loss function

sembling real ones, raising ethical considerations depending on the user's intent and application.

A Appendix

References

[1] Google colaboratory. https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/generative/pix2pix.ipynb. Accessed: 2024-1-9.

[2] Tree it on steam. https://store.steampowered.com/app/2386460/Tree_It/. Accessed: 2024-1-27.

[3] Suraj Amatya, Manoj Karkee, Aleana Gongal, Qin Zhang, and Matthew D Whiting. Detection of cherry tree branches with full foliage in planar architecture for automated sweet-cherry harvesting. *Biosyst. Eng.*, 146:3–15, 2016.

[4] Jianwei Guo, Haiyong Jiang, Bedrich Benes, Oliver Deussen, Xiaopeng Zhang, Dani Lischinski, and Hui Huang. Inverse procedural modeling of branching structures by inferring l-systems. *ACM Trans. Graph.*, 39(5):1–13, 2020.

[5] Daniel P Huttenlocher, Gregory A. Klanderman, and William J Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on pattern analysis and machine intelligence*, 15(9):850–863, 1993.

[6] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. 2016.

[7] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. pages 2980–2988, 2017.

[8] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. 2017.

[9] Mahdi Shafiei Neyestanak, Hamid Jahani, Mohsen Khodarahmi, Javad Zahiri, Mir Saeed Yekaninejad, et al. A quantitative comparison between focal loss and binary cross-entropy loss in brain tumor autosegmentation using u-net.

[10] S. Taklimi. ProceduralTreeGeneration: From foliage to its branches. https://github.com/Samtaklimi87/ProceduralTreeGeneration.

[11] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[12] Jing Zhang, Long He, Manoj Karkee, Qin Zhang, Xin Zhang, and Zongmei Gao. Branch detection for apple trees trained in fruiting wall architecture using depth features and Regions-Convolutional neural network (R-CNN). *Comput. Electron. Agric.*, 155:386–393, 2018.

[13] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *Computer Vision – ECCV 2016*, pages 649–666. Springer International Publishing, Cham, 2016.

[14] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. 2017.