TopoGAN

Topology Optimization with Generative Adversarial Networks

Mathias Bernhard[1,*], Reza Kakooee[1,2], Patrick Bedarf[1], Benjamin Dillenburger[1]

[1] Digital Building Technologies DBT, Institute of Technology in Architecture ITA, ETH Zurich

[*] Corresponding author e-mail: mathbern@design.upenn.edu

[2] School of Information Technology, Lucerne University of Applied Sciences and Arts HSLU

Abstract

Topology optimization (TO) is a numerical simulation to identify an optimal distribution of solid and void. A more efficient distribution of material means a reduction of natural resources consumption. TO results in branching structures, difficult to manufacture with conventional methods. Advances in additive manufacturing allow the production of components at an unforeseen level of complexity. The computational cost and the need for expert knowledge in setup prevent TO from being part of architects' set of instruments. Data-driven artificial intelligence (AI) improved not only classification tasks but also spawned various synthetic models. We trained a generative adversarial network (GAN) with the boundary conditions as input and the result of a conventional TO as output. We chose a wall with randomly placed openings as a case study and produced three different training sets. The GAN was able to generate an output in a fraction of a second. The network learned to output structures close to the ground truth and generalized even across data sets. We measured the accuracy of the generated results with different metrics. The accuracy of the results was very encouraging within a few percents of the target value's deviation. The significant speed improvement is a first and promising indicator of how machine learning could provide real-time feedback to the designer. Integrated into a CAD environment, dynamic updates, even for complex tasks, are invaluable in the conceptual design phase. Such an instrument can help the designer save material and most efficiently layout the building structure.

Keywords: topology optimization, artificial intelligence, machine learning, generative adversarial network, structural design, computational design

1 Introduction

The global construction industry is responsible for a large share of natural resources consumption and a significant contributor to non-recyclable waste production and emission of pollutants (United Nations Environment Programme 2019). Even a relatively small reduction of used material in the constructive system has a big impact on absolute numbers. This offers a path to more sustainable building practice. Topology optimization (TO) is an analytical way to redistribute a specified fraction of material to satisfy or maximize a target value (e.g., thermal conductivity in heat sinks or maximum stiffness and minimum compliance in structural design) under a set of boundary conditions. This is done by running multiple iterations of finite element analysis and reassigning a new density value to each node at each step. TO was first introduced over 30 years ago (Bendsøe and Kikuchi 1988), and a vast collection of literature grew ever since. The geometries resulting from TO often feature intricate tubular structures and complex porosity. For conventional manufacturing methods, these designs would have to undergo a second round of shape optimization, e.g., to guarantee necessary draft angles for demoulding or accessibility for a CNC mill. Advances in digital fabrication and additive manufacturing brought these geometries within the realm of what can be produced directly.

1.1 Topology Optimization and Architecture

Whereas TO influenced design practices in industrial, aerospace, and automotive design significantly, only a few examples can be found for large-scale architectural components. In this field, pioneers were Japanese architects Arata Isozaki and Mutsuro Sasaki with their design proposals, Illa de Blanes and Santa Maria Novella, developed in the years 1998-2003 (Januszkiewicz and Banachowicz 2017). The designs for the iconic seaside resort in Blanes and the extension of the train station in Florence featured large cantilevering roofs and organic columns, generated with an evolutionary structural optimization algorithm (Ohmori 2011). Although unbuilt, these preliminary explorations influenced the two first built examples that used TO as the primary design method: the Akutagawa River Side project in Takatsuki from 2004 and the Qatar National Convention Centre in Doha from 2008 (Białkowski 2016).

Today the approach of using TO as a form-finding tool in the design of buildings is practiced mainly by structural engineers during the concept phase. This was demonstrated for the generation of graded exoskeleton patterns in high-rise structures (Stromberg et al. 2011) and the layout of structural frames for bridges and other cantilevering structures (Beghini et al. 2014). However, advancing fabrication

technologies such as 3D printing (3DP) and construction robotics extend the domain of what can be feasibly manufactured and allow for more complex geometries, which are typically featured in TO results.

Engineering consultancy Arup gave a first impression of the new design opportunities with their advanced 3D printed steel joint, which used TO to reduce its weight by 75 percent (Galjaard et al. 2015). Shortly after that, researchers used binder jet 3DP to create stay-in-place formwork for topologically optimized concrete slabs with significantly reduced material and weight (Jipa et al. 2016). In another example, 3D concrete printing was used to manufacture a post-tensioned girder designed by TO (Vantyghem et al. 2020).

Robotic fabrication enabled the assembly of advanced timber space-frame structures that used truss optimization algorithms with combined topology and sizing optimization (Søndergaard et al. 2016). Construction robotics was also instrumental for the fabrication of complex form-work systems for the casting of topology optimized concrete structures (Søndergaard et al. 2018).

However, TO in the design process remains in the hands of a few experts in research or consultancies. One of the reasons for this is the high amount of required expert knowledge needed for setting all parameters of TO simulations (Bialkowski 2018). Another reason is the high computational costs that result in time-intensive calculations. In order to become an integrated design instrument, TO needs to become more intuitive and real-time responsive.

1.2 Machine Learning

Enhancing the design process with forms of AI in general or machine learning more specifically turns the computer into an active assistant that learned from vast collections of precedents. Therefore, it can make informed predictions and propositions based on extracted patterns - without the need for calculating a solution from scratch and without prior knowledge. Since a massive increase of available computing power, the availability of ever bigger training data sets and the landslide victory of AlexNet at the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 (Krizhevsky et al. 2012), neural networks are used for many tasks like image classification, natural language translation and drug discovery.

Google Research's publication of DeepDream (Mordvintsev et al. 2015) awakened the interest in creative professionals. The idea behind it is to reverse a neural network, instead of asking what it sees in an image (classification task), it adjusts the image to maximize seeing something specific. This has spawned the development of many different generative AI models. A very prominent and promising type are

generative adversarial networks or GAN for short (Goodfellow et al. 2014). GAN models consist of two neural networks, called *generator* and *discriminator*. The goal of the generator is to fool the discriminator by generating fake data, which are as similar to the real data as possible. The role of the discriminator is to call the generator's bluff by perfectly distinguishing fake data from the real ones. An exciting expansion of GANs is Image-to-Image translation (Isola et al. 2016). Instead of sampling from a random noise distribution, the generator receives an input image and learns a mapping function to translate it into an output close to the target ground truth. Pix2pix has successfully been applied to image colourisation, season or daytime transfer, satellite photo to map image conversion (and vice versa), or to convert sketches of cats, shoes, and handbags into photos.

1.3 Topology Optimization and Machine Learning

These advances in AI awakened the interest of many disciplines for their promising potential, and they also entered the field of engineering. Early attempts used unsupervised learning to cluster the cells in the design domain for multi-material printing (Liu et al. 2015). Looking for a more compact representation and a speeding up of the process, Ulu et al. (2016) employ dimensionality reduction with principal component analysis (PCA) to predict the final result of the TO from a small set of eigenvectors. Convolutional Neural Networks (CNN), the powerful building blocks of modern computer vision, were used to predict the final stage of TO from the blurry intermediate stage after only 5 iterations (Sosnovik and Oseledets 2019), with a generated data set containing randomly distributed supports (priority given to border cells) and randomly distributed vertical loads. This approach with CNNs was even implemented in low-resolution three-dimensional space (Banga et al. 2018). Lei et al. (2019) train a model to rearrange discrete linear elements into an optimal topology using the moving morphable components (MMC) method. Variational auto-encoders were used to predict an optimized topology from only ten latent variables (Yu et al. 2019). Generative adversarial networks (GAN, see also **sec. 1.2**) were trained with data varying in volume fraction, penalty, and filter radius (Rawat and Shen 2018, 2019) and a variation thereof, conditional GANs on variations of the classic console example (Shen and Chen 2019). We build on these exciting prior works and extend them in three major new directions: the application at an architectural scale, mandatory voids as a varying input parameter for the training data, and a conversion of the data to continuous gradients through distance transformations to facilitate more robust learning.

2 Methods

We adapt the Image-to-Image Translation model presented by Isola et al. (2016) to our specific task. The task is to lay out a support structure around a variable set of openings, transferring the loads as efficiently as possible according to given boundary conditions. We train a GAN model to learn a generalized mapping function between an input: the arrangement of openings - and an output: the density distribution of an optimized topology.

2.1 Training Data Generation

Regardless of the application, type of data, or the algorithm at work, all machine learning methods rely on extensive collections of training data with hundreds, thousands, or up to millions of samples. For applications like object recognition or image classification, these images need to be collected "in the wild" and eventually labelled manually. The advantage of using a computational simulation is that the training samples can be generated in arbitrary large quantities "in silicio", and their labels are known.

The "Hello, World!" of TO is the MBB beam, a simple rectangular domain with rolling supports in its two bottom corners and a single point load at the centre of the top edge. More complex forms are often created for small monolithic parts (e.g., brake pedals, brackets, or consoles) by more complicated or even dynamically varying load and support cases. In these solutions, the regions around the mounting fixtures are set to be mandatory solid to permit strong connections, and thereby contributing to the global volume fraction. Despite being available in most TO algorithms and packages, the feature to set some mandatory void regions is less commonly in use.

2.2 Case Study

We apply TO at an architectural scale of an entire wall segment of 300x300 cm. The three examples in **fig. 1** show the dependence of the resulting structure on the proportions of the sample wall. The boundary conditions as shown in **fig. 2** (left) are constant across all examples and all data sets: a uniform linear load in negative Z direction (pink) on all the nodes in the top row and a uniform linear support blocking displacement in Z direction (blue) for all the nodes in the bottom row. These constraints are specified by raster images as well and could hence also vary and used for training the GAN.

Prior work on GANs for TO exists that varies the support and load cases as well as parameters such as volume fraction (see **sec. 1.3**). Therefore, we target mandatory

voids specifically. The variable element different in every sample is the distribution of openings. The openings vary in position, size, shape, and rotation. Three different data sets were generated, and the following **tab. 1** summarizes the parameters used for the generation of each data set.

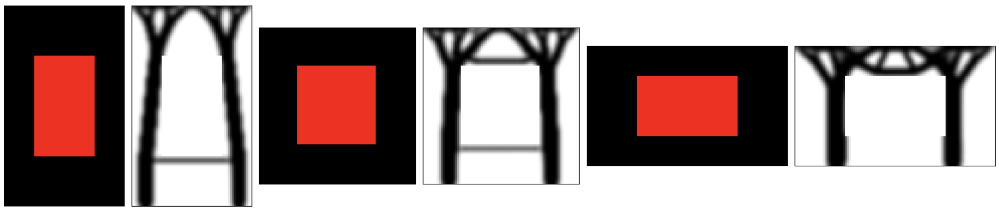

Figure 1: Proportion dependence of optimal topology, three tests with the same number of elements (FEA cells), each with a central void half the width and half the height, same linear load and support case.

Dataset	Shapes	Dimensions	Openings	Samples
1	Rectangles	position: random width: 12-150 cm height: 20-150 cm rotation: 0°	1-3	1,360
2	Ellipses	position: random width: 12-150 cm height: 20-150 cm rotation: 0°	1-3	453
3	Lines	position: random width: 23.4 cm length: 20-150 cm rotation: random 0-360°	3-5	571
1 df	Rectangles	same as data set 1 but converted to distance field (see **sec. 2.3**)	1-3	1360

Table 1: Overview of data set specifications.

We generated all sample pictures with a Processing script at a resolution of 128x128 pixels. For the size of the wall, this corresponds to a node element size of 2.3 cm. The TO script checks for the intensity of the red channel above a certain threshold to set the corresponding cells to void. Hence, black pixels are the remaining cells to work with, in which the TO algorithm optimally distributes the required amount of material.

We use the Python implementation of Solid Isotropic Material with Penalisation (SIMP) TO by Aage and Johansen (2013) with a custom adaptation allowing us to read loads, supports, solid and void regions from raster images. The image specifying the loads uses the red channel for vertical and the green channel for horizontal loads. The blue channel is discarded at the moment and set to 128 for all cells. The 8-bit integer value V is transformed with $(V-128)/128$ to adjust its range to $[-1,1]$. An orange pixel with the values (255,128,0) hence means a straight vertical load, yellow (255,255,0) would mean a 45° load from top left. The corresponding load colour map is shown in **fig. 2**, right.

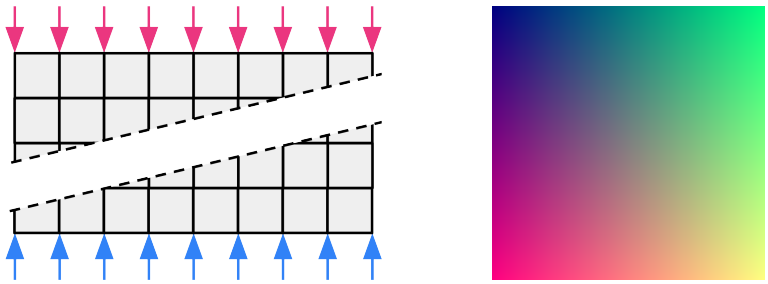

Figure 2: Left: Boundary conditions for all samples in all data sets; Right: colour map for load directions.

We batch process an entire folder of sample images and save the topology optimization result as an image file. We then concatenate these sample pictures with the TO simulation's result into 256x128 pixel image pairs. These pairs serve as training data for the GAN training. While the generator network only gets the right half (B) as an input to generate output from, the discriminator network gets both images to learn distinguishing between real (A, ground truth) and fake results.

2.3 Distance Fields

Neural Networks can be seen as general function approximation devices for any continuous function (Csáji 2001). In the case of TO, however, due to the penalization and sensitivity filtering, ultimately, the training data is almost binary or categorical. To overcome this and make the data more suitable to learn from on the one hand, and to have more gradual accuracy measurements for the output on the other hand, we post-process one of the data sets before training, converting the binary image to continuous gradients of grey levels. To this end, we first apply two Euclidean distance transforms (van der Walt et al. 2014) to both the solid and the void part, remap these distance maps to the range $[0,0.5]$, invert the one of the black part, combine them with mutually exclusive masks and add 0.5. The combined matrix is in the range $[0,1]$, with an iso-level of 0.5 along the boundary between solid and void of the original image, as shown in **fig. 3**.

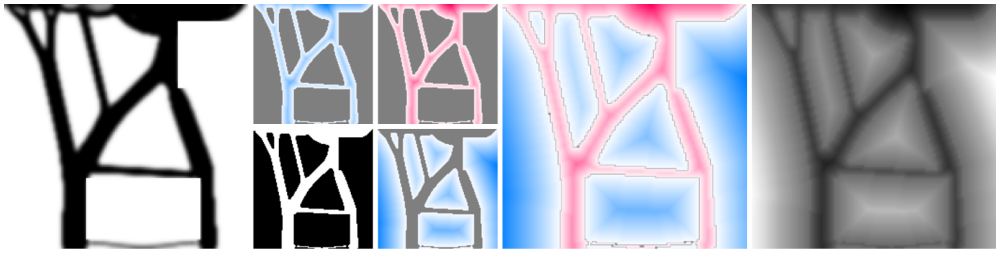

Figure 3: creation steps of the distance field training data set; from left to right: **1:** original training sample of data set 1, **2 top row:** distance map for solid part and its inversion, **bottom row:** inversion of the original and distance map for the void part, **3:** combination of the two maps, **4:** remapped range to $[0, 1]$, final post-processed training sample.

With gradual transitions in brightness, the performance measurement described in **sec. 2.5** below will return small fractions of error if the result is off to one side by a few pixels, as opposed to a full error if it is off by one pixel in (close to) binary image comparisons. In the example shown in **fig. 5**, these fractions add up to 2. In a binary version with a threshold of <0.5, the total error would also be 2. However, as shown in **tab. 3**, we achieve approximately 46% less error in the pixel difference category with the distance field data set.

Another advantage of the gradient transitions in the output is that by adjusting the threshold value to e.g. 0.6 or 0.4, the volume fraction can easily be adjusted up or down, respectively, by consistently offsetting the resulting solid part outside or inside. Instead of calculating the linear Euclidean distances in a post-processing step, the gradient values could stem directly from the TO algorithm, if the penalization and sensitivity filtering per iteration would be omitted.

2.4 Machine Learning Model

The training data consists of image pairs made of the input image (arrangement of openings) on the right and the ground truth (the result of the topology optimization) on the left, as shown in **fig. 4**. We run the training on a Windows 10 (64 bit) machine with 64 GB RAM, an AMD Ryzen™ Threadripper™ 2920X Processor and a NVIDIA GeForce RTX 2080 Ti graphics processing unit. An overview of time measurements for training and prediction is shown in **tab. 2**.

We train the generator and discriminator networks on all four data sets (see **tab. 1**) separately and on a collection of sets 1-3 combined. We run the training for 50 and 100 epochs. To verify how well the generator network learned to perform the mapping task, we retain 10% of the data for testing purposes. The discriminator network can be discarded after the successful training, and only the generator network is used to synthesize the output. The generator has not seen the test

Dataset	Training (100 epochs)	Prediction (ø/image)	Test Images
1	200 min	0.11 sec	136
2	70 min	0.14 sec	46
3	85 min	0.13 sec	57
1 df	200 min	0.15 sec	136

Table 2: Timings for training and predictions.

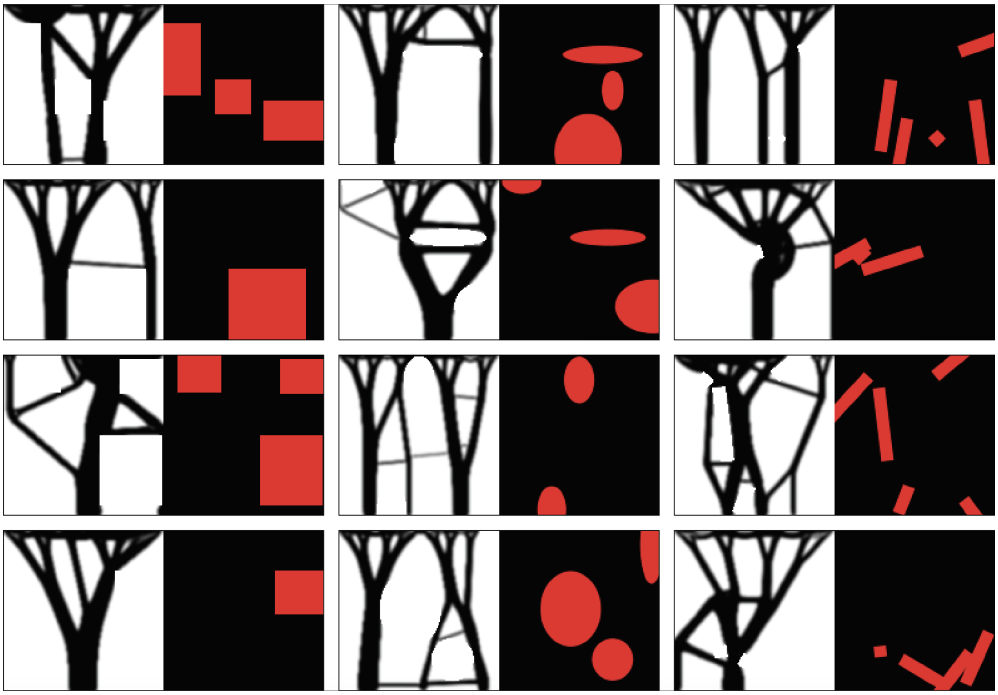

Figure 4: twelve samples of the training data, four of each data set; left: result of the topology optimization (ground truth), right: arrangement of openings, input for the generator.

inputs before during training. We run these tests for the four data sets separately, so the generator trained on the training samples of data set 1 is given the remaining test images of data set 1. Once the more time-consuming task of training the network is accomplished, we can easily test how well the model learned to generalize by running tests across different data sets. We do so by presenting the test images of one data set to a generator trained on another data set, as shown in **tab. 3**.

The bigger the collection of training data, the better most machine learning models learn their task, learn to generalize without overfitting. A common strategy to artificially expand the available samples is called *data augmentation*. While frequently applied transformations such as scaling and rotation are not applicable in our case due to the boundary constraints, we benefit from the fact that the TO

outputs are invariant to horizontal reflection. By randomly applying mirroring along the central vertical axis to the images prior to feeding them to the networks during training, we artificially double the number of samples (see **tab. 1**).

2.5 Performance Measurement

The only inputs for the two networks in the GAN are the 16'384 pixels' colour values of the training images. No semantic meaning whatsoever is taken into account during training. Examples for semantic meaning are that black means solid and white means void in the ground truth, that red pixels in the opening map must remain white, that 30% of the pixels should be darker than medium grey, that neighbouring pixels simulate physical connections to transfer loads or even the existence of a load and support condition at all. Likewise, the output of the generator network is also just an image with 16'384 colour values.

Visual Inspection

The most obvious and intuitive way to verify the result may be by visual inspection. The overview in **fig. 8** (see Appendix) shows how well the generator performs for most "regular" cases of branching trees. The most common error occurs with thin horizontal connections as they are occasionally misplaced, discontinuous, or missing altogether. More extreme cases like the one in the second column, third to last row, also seem to pose a problem, most likely because this type of structure is underrepresented in the training data. However, to gauge the capacity of the generator, precise numerical analyses are needed. Therefore, we implement four different measurement strategies to quantify the performance of the generated solution in comparison with the ground truth. They all measure a different type of similarity in different feature spaces. Most of them are based on numerical calculation and reveal differences not perceptible at first sight.

Pixel Difference

The brightness of the pixels in the TO algorithm's output represents material density values. Black means high density (solid), and white means low density (void). In the SIMP method used for this study, these values are not strictly binary (like in the BESO method) but instead are enforced through the penalization to tend towards one of the extreme ends of the spectrum, maintaining the target volume fraction constant. For every pixel, or cell in FEA terms, where the density (or brightness) of the generated result differs from the ground truth, it can be regarded as an error and the absolute difference its magnitude.

The computation of the pixel difference as an accuracy measurement involves two steps:

1. calculate a new matrix where each cell is the result of $A - B$, where A is the ground truth image, and B is the output of the generator network
2. sum the absolute values of all the cells in the new matrix

The first step is illustrated in **fig. 5**. A negative value means the generator did not put enough material in a cell, while a positive value means it added too much. The second step would sum all the absolute values of the third row; hence this toy example would have a PD value of 2.

A	1	0.75	0.5	0.25	0	0	0	0.25	0.5	0.75	1	1
B	1	1	0.75	0.5	0.25	0	0	0	0.25	0.5	0.75	1
A - B	0	-0.25	-0.25	-0.25	-0.25	0	0	0.25	0.25	0.25	0.25	0

Figure 5: calculation method for pixel differences.

The matrices generated in step 1 can also be rendered as an image for visual inspection of where any differences exist. The overview in **fig. 6** shows 28 examples of these matrices, with a colour map ranging from dark red (-1) over light grey (0) to dark blue (+1). Many red parts show that the generator sometimes fails to add thin horizontal or diagonal connections between principle vertical structures, or sometimes adds them at a different height shown in blue. Many blue parts are alongside the primary vertical structures. If these are blue on one and red on the other side, the generator slightly shifted the structure to either side. Blue on both sides means it generated the structure too thick overall.

Constraint Violation

One constraint enforced by the TO algorithm is that red cells in the input are mandatory to remain void (or density 0) in the FEA model. This is hard-coded and reset after every iteration of the FEA analysis. Like the volume fraction described above, this information is not explicitly available to the GAN machine learning model. Whether or not it succeeds in recognizing this pattern – an extracted rule to be applied by the generator – is a useful performance measurement for successful training.

The last column in **tab. 3** shows that this is a less significant performance measurement as only very few generated samples violate the void constraint at all or by very few pixels only. The tests performed across data sets - even though half of them got a maximum of 1 pixel wrong - show some outliers with higher numbers (trained on data set 2 and 3 and tested with data set 1). The 610 pixels correspond to a 3.7% error, which is still reasonably small.

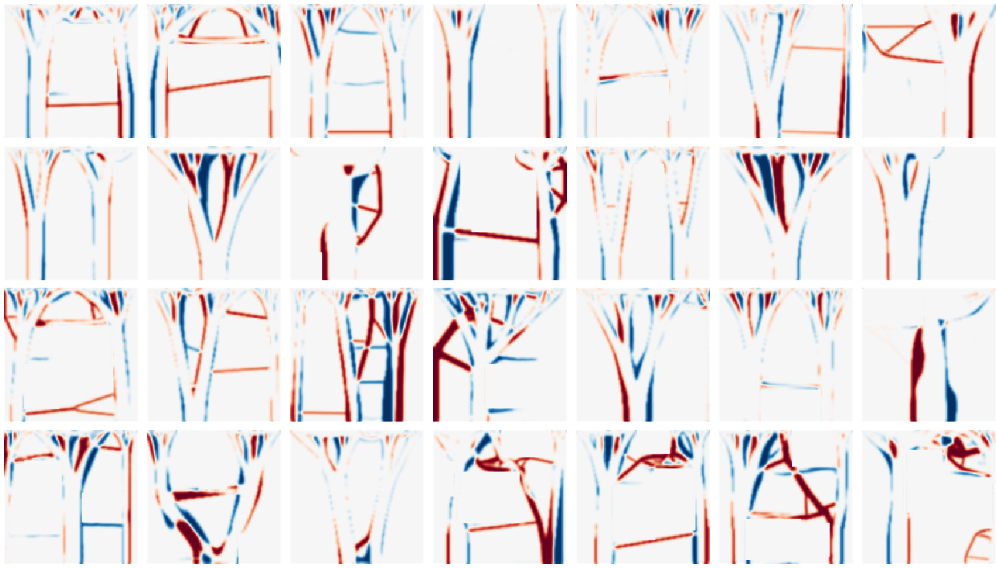

Figure 6: Pixel differences for 28 examples.

Volume Fraction

For the generation of the TO ground truth training data, we specified as a parameter a volume fraction of 30%. This means that all cells in the FEA model are set to an initial density value of 0.3. During the following optimization iterations, this virtual material is continuously redistributed to maximize stiffness or minimize compliance. In the end, most cells have a density of either 1 (solid, black) or 0 (void, white), but the total density sum remains constant. While the volume fraction constraint is numerically enforced after every iteration of the TO algorithm, this number is not available to neither the generator nor the discriminator network in the machine learning model. However, for the generator network to usefully and accurately complete the task, its output should also provide a solution using the same amount of material.

We measure the volume fraction in both the ground truth and the generated results by counting the number of pixels below 0.5 brightness level threshold and dividing it by the total number of pixels (128x128). As shown in **tab. 3**, columns four and five, the mean volume fraction in almost all the cases is within $\pm 1\%$ of the target value, the predictions of the generator just have a slightly higher standard deviation. Without being explicitly told, and without counting the black and white pixels in any layer of the machine learning model, it successfully learned this general rule and accurately applied it to generate the outputs.

Cosine Similarity of Network Weights

We feed the trained discriminator with a target image and the corresponding generated image to compute the cosine similarity values. We get the outputs of the last convolution layer of the discriminator. The shape of this output is [1,32,32,512]. We convert this tensor to a vector, so each result vector represents a point in 524,288 (32x32x512) dimensional space. We then compute the cosine similarity between the vector of the target image and the one of the corresponding generated image.

Compliance

The training data produced by the TO algorithm is optimized for minimal compliance and maximum stiffness. The 30% volume is distributed in a way to best respond to the specified load and support boundary conditions (see **fig. 2**). The quality of this response can be expressed as a number as well. As described in the previous sections, only differences based on the images have been measured for comparison so far. A relevant indicator for quantifying the success of the generator will be the structural performance of the generated structure compared to the ground truth. Hence, the next step will be to convert both images back to a FEA model with the density of the cells depending on the brightness of the pixels and calculate the compliance.

3 Results

We achieved promising results in both speed and accuracy according to the performance measurements presented in the previous **sec. 2.5**. They make TopoGAN an attractive candidate for future investigations.

3.1 Speed

For the production of the training data, we ran 50 epochs of TO per sample input, which took an average of $\pm$60 s per sample. The training of the GAN machine learning model is relatively time-consuming, depending on the number of samples (see **tab. 2**). However, once the generator network is trained, it can predict an output image in only tenths of a second. This corresponds to a factor 500 speed improvement. A single solution obviously does not justify the 200 min of GAN model training time. However, a frame rate of 8 fps converts an AI-assisted TO into a real-time computational design instrument. The designer can get live feedback upon every update of the design, where minor translations of openings can cause fundamental changes in the topology of the supporting structure. The 200 min of training time would be justified after only 25 s of interactive work.

3.2 Accuracy

The following **tab. 3** summarizes the results of the different performance measurements described in **sec. 2.5**. There are three major findings worth mentioning. The first is that the volume fraction for the predicted output is very close to the targeted 30%, with only a slightly higher standard deviation than the ground truth. The second is that constraint violations are 0 or close to 0 in all data sets for the corresponding test samples or when trained on data set 1. Because data set 1 contains the highest number of samples, it suggests that the other generator networks would also learn to generalize better with more training data. Finally, the third finding is the excellent performance of the distance field data set in the pixel difference category. The difference is 42% better than the average of data sets 1, 2 and 3, 36% better than data set 1 with the same number of samples. This post-processing step of converting the training data to continuous gradients described in **sec. 2.3** is beneficial for the GAN model to learn the mapping robustly.

Train	Test	PD	VF (GT)	VF (PR)	CV
1	1	1517 / 651	29.95 / 0.60	29.91 / 1.82	0.07 / 7
2	2	2747 / 1012	30.27 / 0.31	30.46 / 1.63	0.00 / 0
3	3	2703 / 780	30.26 / 0.23	30.67 / 1.59	0.00 / 0
1 df	1 df	972 / 393	29.98 / 0.60	30.42 / 1.86	8.96 / 144
1+2+3	1	1428 / 704	29.98 / 0.60	29.41 / 1.62	0.03 / 4
1	2	2546 / 940	...	30.22 / 1.34	0.02 / 1
1	3	3176 / 965	...	30.18 / 1.37	0.05 / 1
2	1	2364 / 917	...	30.34 / 2.34	4.18 / 287
2	3	3084 / 729	...	29.34 / 1.76	0.00 / 0
3	1	2639 / 946	...	29.86 / 2.60	24.03 / 610
3	2	3110 / 830	...	31.16 / 2.26	1.15 / 49

Table 3: Performed train and test combinations with the corresponding results; PD: pixel difference (μ/σ), VF: volume fraction (μ/σ), GT: ground truth, PR: prediction, CV: constraint violation($\mu/\max$), μ: mean value, σ: standard deviation, $\max$: maximum value.

4 Conclusion

We have shown how a machine learning model (GAN) can be trained on an extensive collection of precedent solutions to topology optimization problems. The generator network learns well to satisfy constraints such as the mandatory void regions and to satisfy a volume fraction within a small range of error without being explicitly told. The model learns the task well enough to generalize across data sets. The accuracy described in the results **sec. 3** made us very confident to investigate

the presented methods further. The massive speed improvement outperforms the slightly lower accuracy trade-off when compared to conventional TO algorithms. In early design phases of conceptual decisions, the more qualitative assessment of our instrument could be preferred over time-consuming simulations, for its agility and immediate response. By no means, we suggest replacing a quantitatively correct final structural analysis.

The 200 minutes of training do not pay compared to 1 minute of TO (50 epochs) for a single task. However, integrating an AI-assisted TO with a frame rate of 8 fps into a computational design instrument is a categorical game-changer. TO is no longer a computationally expensive method dictating a solution for given constraints that one has to obey. Instead, the architect can challenge the spatial layout of the structure and indirectly design it in an interactive process. The GAN model learns a type of intuition from a sufficient number of precedent cases, not 100% correct all the time but the gist of it, good enough to not always start from scratch as if no solution was ever computed before.

In its current state, our project presented in this paper has a clear focus. It targets the particular case of 2D wall elements with openings varying in shape and position, while dimension, resolution, and boundary constraints remain constant. We consciously set these constraints to investigate the viability of GANs for TO tasks. For TopoGAN to become a meaningful and more general-purpose design instrument, we need to increase two of the four V's of big data: volume and variety. As our training data is synthesized (see **sec. 2.1**), the volume can easily be increased. As the boundary constraints are specified through raster images as well, a GAN model could potentially also learn to interpret and correctly map these pixels' colours. An expansion of our method to the third dimension is possible with minor adaptations, but out of scope at the moment for reasons of resolution or computation time in data generation, respectively.

Instead of using machine learning to predict the final state from only the *first 5-10 iterations* of TO, as described in some prior works (see **sec. 1.3**), the output of the generator could be used to initialize the cells' density values in the FEA model. The TO algorithm would then only be used to calculate the *last 5-10 iterations*.

References

Aage, N. and V. E. Johansen (2013). Topology optimization codes written in python. `http://www.topopt.mek.dtu.dk/Apps-and-software/Topology-optimization-codes-written-in-Python`.

Banga, S., H. Gehani, S. Bhilare, S. Patel, and L. Kara (2018). 3D Topology Optimization using Convolutional Neural Networks.

Beghini, L. L., A. Beghini, N. Katz, W. F. Baker, and G. H. Paulino (2014). Connecting architecture and engineering through structural topology optimization. *Engineering Structures 59*, 716–726.

Bendsøe, M. P. and N. Kikuchi (1988). Generating Optimal Topologies in Structural Design Using a Homogenisation Method. *Computer Methods in Applied Mechanics and Engineering 71*(2), 197–224.

Białkowski, S. (2016). Structural Optimisation Methods as a New Toolset for Architects. In *Proceedings of the 34th Education and research in Computer Aided Architectural Design in Europe Conference*, Volume 2, pp. 255–264.

Bialkowski, S. (2018). Topology Optimisation Influence on Architectural Design Process-Enhancing Form Finding Routine by tOpos Toolset utilisation. *Computing for a better tomorrow - Proceedings of the 36th eCAADe Conference 1*, 139–148.

Csáji, B. C. (2001). Approximation with artificial neural networks. Faculty of Sciences, Eötvös Loránd University, Hungary.

Galjaard, S., S. Hofman, N. Perry, and S. Ren (2015). Optimizing Structural Building Elements in Metal by using Additive Manufacturing. In *International Association for Shell and Spatial Structures*, Number August.

Goodfellow, I. J., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio (2014). Generative Adversarial Networks.

Isola, P., J.-Y. Zhu, T. Zhou, A. A. Efros, B. Ai, and U. C. Berkeley (2016, nov). Image-to-Image Translation with Conditional Adversarial Networks. *arxiv*, 802–806.

Januszkiewicz, K. and M. Banachowicz (2017). Nonlinear Shaping Architecture Designed with Using Evolutionary Structural Optimization Tools. *IOP Conference Series: Materials Science and Engineering 245*(8).

Jipa, A., M. Bernhard, M. Meibodi, and B. Dillenburger (2016). 3D-Printed Stay-in-Place Formwork for Topologically Optimized Concrete Slabs. *2016 TxA Emerging Design + Technology* (Figure 2), 96–107.

Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems*

25, pp. 1097–1105. Curran Associates, Inc.

Lei, X., C. Liu, Z. Du, W. Zhang, and X. Guo (2019). Machine learning-driven real-time topology optimization under moving morphable component-based framework. *Journal of Applied Mechanics, Transactions ASME 86*(1).

Liu, K., A. Tovar, E. Nutwell, and D. Detwiler (2015, 08). Towards Nonlinear Multimaterial Topology Optimization Using Unsupervised Machine Learning and Metamodel-Based Optimization. *Volume 2B: 41st Design Automation Conference*. V02BT03A004.

Mordvintsev, A., C. Olah, and M. Tyka (2015). Deepdream - a code example for visualizing neural networks.

Ohmori, H. (2011). Computational morphogenesis: Its current state and possibility for the future. *International Journal of Space Structures 26*(3), 269–276.

Rawat, S. and H. Shen (2018). A Novel Topology Design Approach Using an Integrated Deep Learning Network Architecture. pp. 1–15.

Rawat, S. and M. H. H. Shen (2019). A Novel Topology Optimization Approach using Conditional Deep Learning.

Shen, M. H. H. and L. Chen (2019). A New CGAN Technique for Constrained Topology Design Optimization. pp. 1–14.

Søndergaard, A., O. Amir, P. Eversmann, L. Piskorec, F. Stan, F. Gramazio, and M. Kohler (2016). Topology Optimization and Robotic Fabrication of Advanced Timber Space-Frame Structures. *Robotic Fabrication in Architecture, Art and Design 2016*, 191–203.

Søndergaard, A., J. Feringa, F. Stan, and D. Maier (2018). Robotic abrasive wire cutting of polymerized styrene formwork systems for cost-effective realization of topology-optimized concrete structures. *Construction Robotics 2*(1-4), 81–92.

Sosnovik, I. and I. Oseledets (2019). Neural networks for topology optimization. *Russian Journal of Numerical Analysis and Mathematical Modelling 34*(4), 215–223.

Stromberg, L. L., A. Beghini, W. F. Baker, and G. H. Paulino (2011). Application of layout and topology optimization using pattern gradation for the conceptual design of buildings. *Structural and Multidisciplinary Optimization 43*(2), 165–180.

Ulu, E., R. Zhang, and L. B. Kara (2016). A data-driven investigation and estimation of optimal topologies under variable loading configurations. *Computer Methods*

in Biomechanics and Biomedical Engineering: Imaging and Visualization 4(2), 61–72.

United Nations Environment Programme (2019). Sustainable buildings. `https://www.unenvironment.org/explore-topics/resource-efficiency/what-we-do/cities/sustainable-buildings`.

van der Walt, S., J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors (2014, 6). scikit-image: image processing in Python. *PeerJ 2*, e453.

Vantyghem, G., W. De Corte, E. Shakour, and O. Amir (2020). 3D printing of a post-tensioned concrete girder designed by topology optimization. *Automation in Construction 112*(January), 103084.

Yu, Y., T. Hur, J. Jung, and I. G. Jang (2019). Deep learning for determining a near-optimal topological design without any iteration. *Structural and Multidisciplinary Optimization 59*(3), 787–799.

Appendix

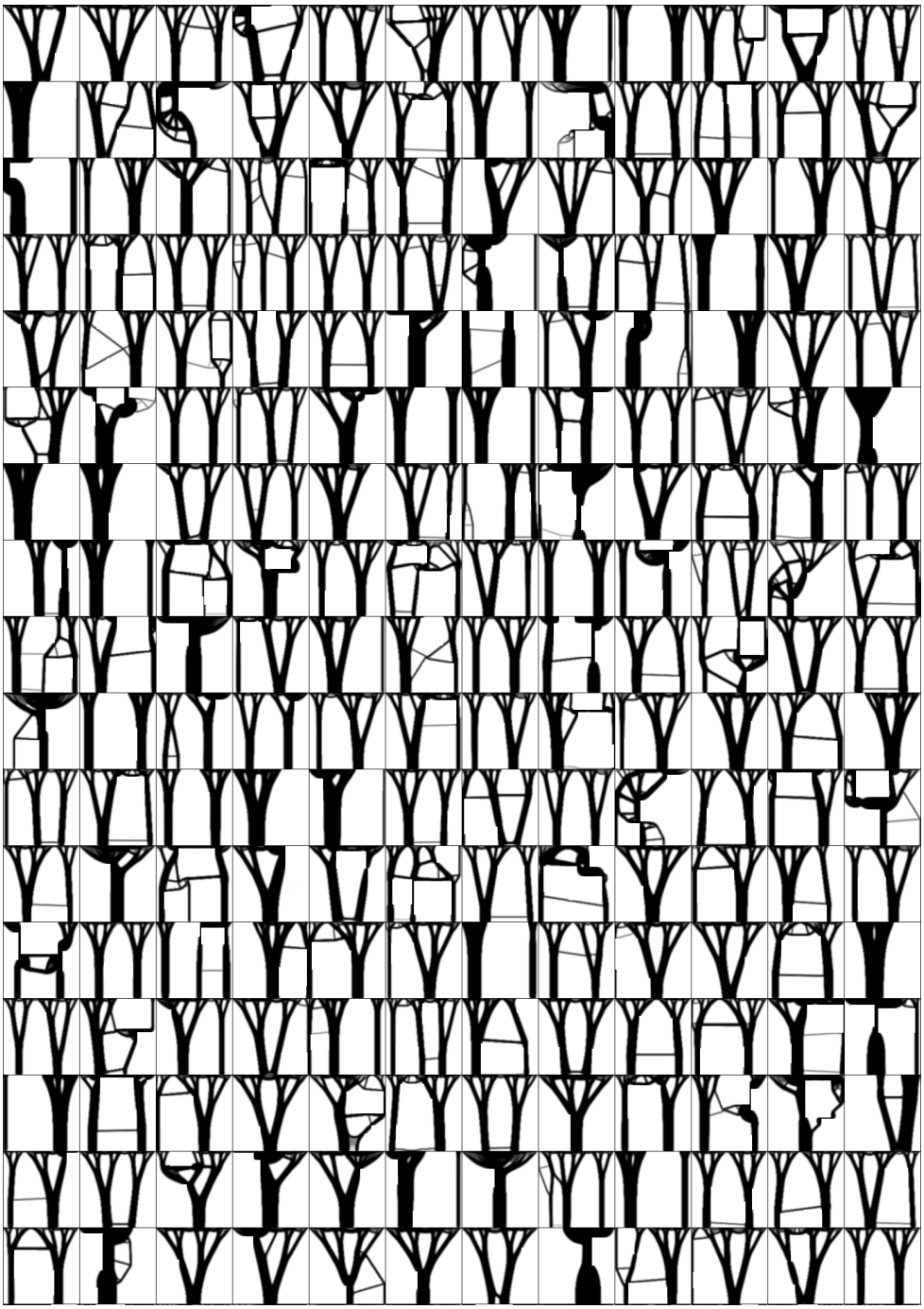

Figure 7: Collection of 204 training samples (ground truth).

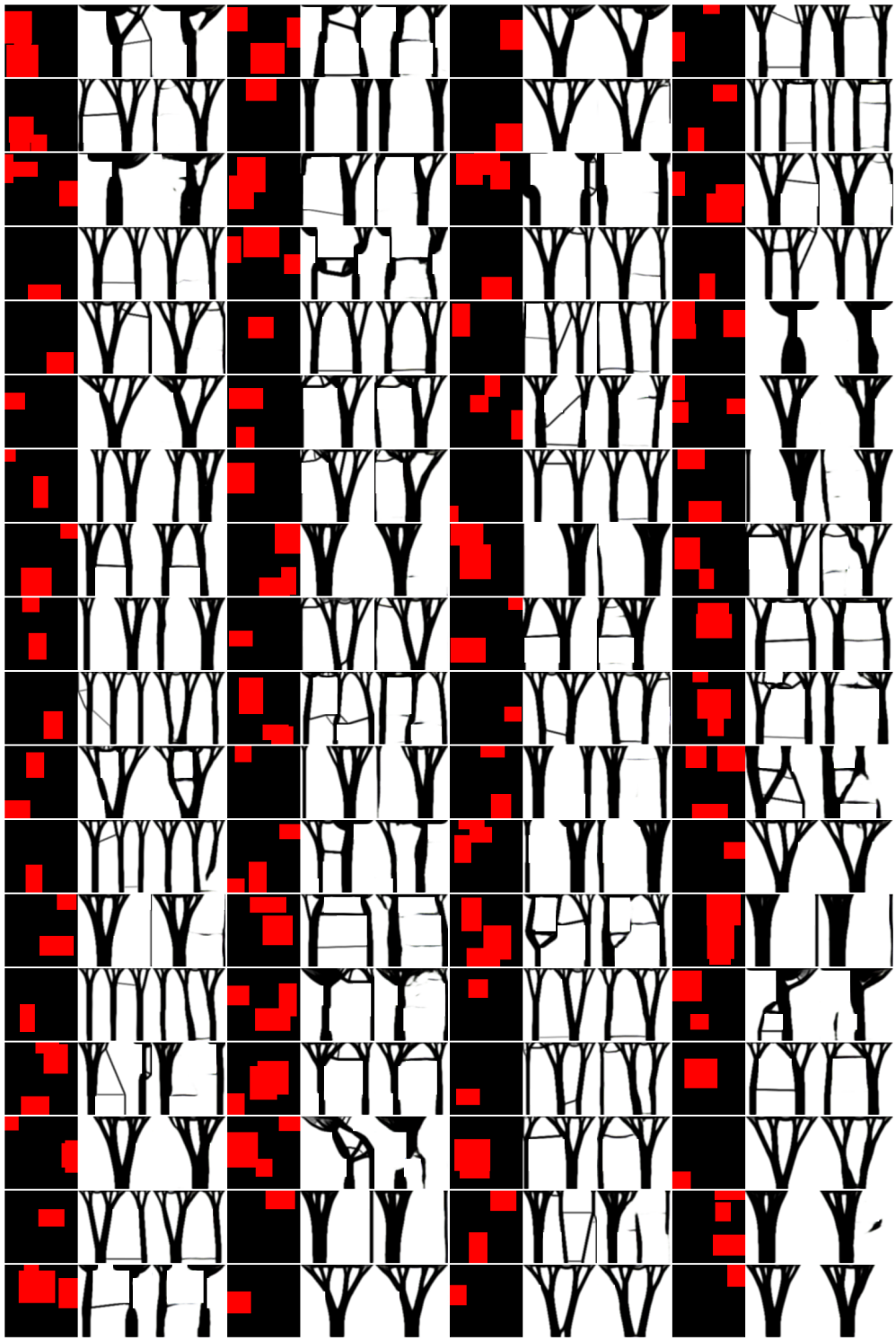

Figure 8: Collection of 72 results of the GAN, data set 1; image triples: input, ground truth, output.

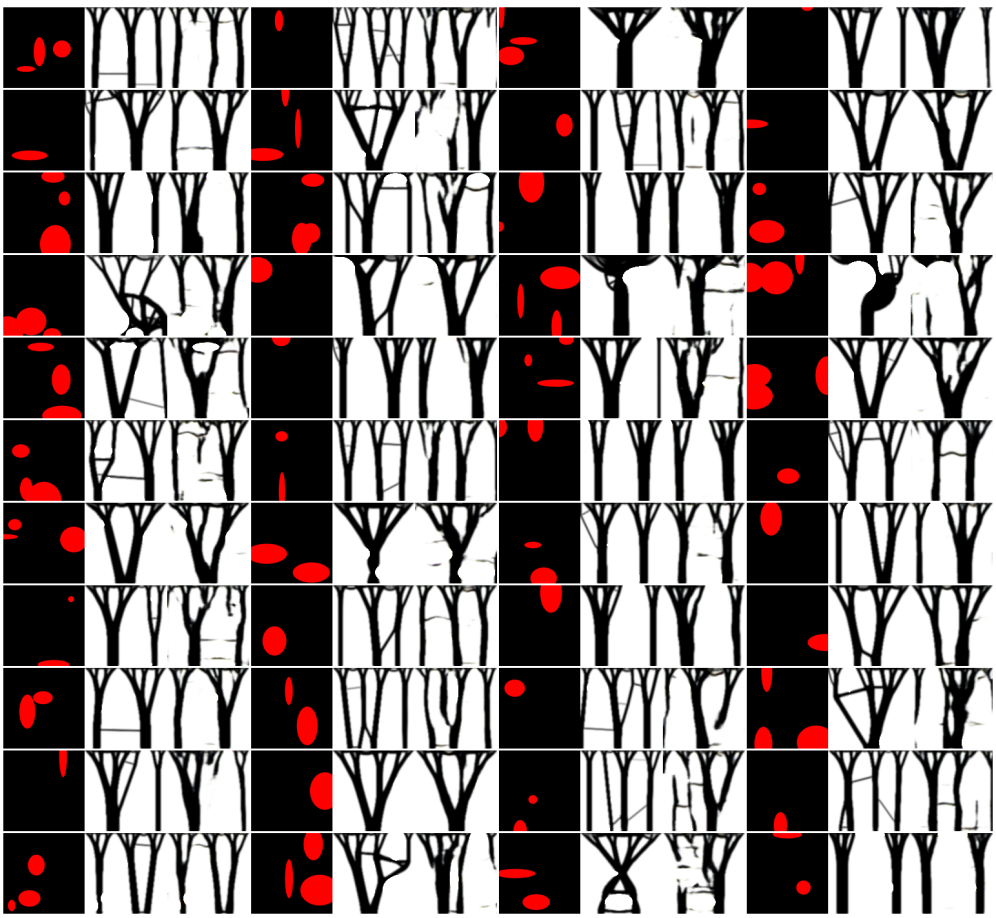

Figure 9: Collection of 44 results of the GAN, data set 2; image triples: input, ground truth, output.

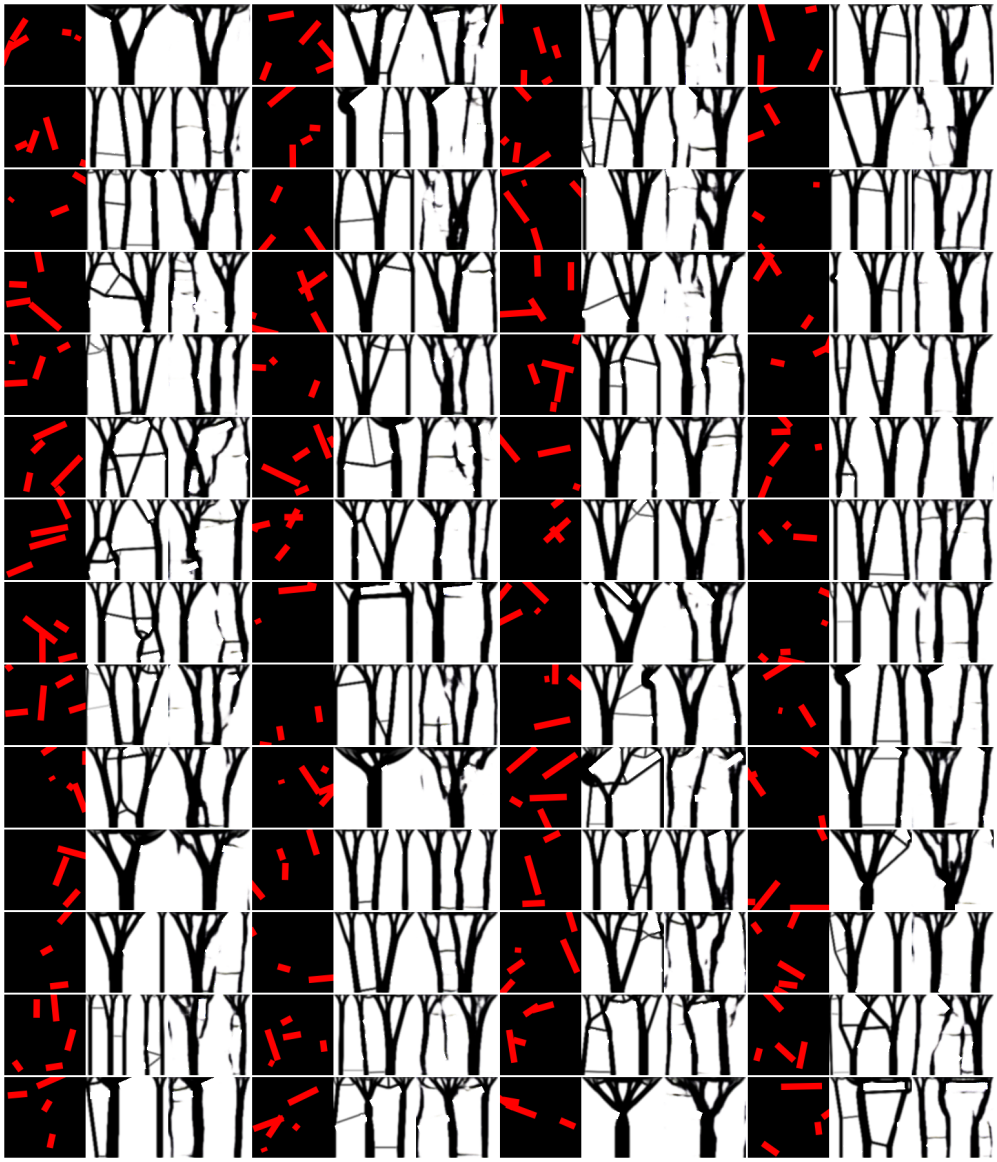

Figure 10: Collection of 56 results of the GAN, data set 3; image triples: input, ground truth, output.