Geomorph Genarator

Application of DC-GAN and W-GAN architectures to the issue of procedural generation of dungeon maps

Problem declaration

- Procedural generation of content is very important in the game industry because it's very cost effective. This is also true in the tabletop landscape[1] (example: Dungeons & Dragons battle maps)

- Approach: instead of generating full maps, use tessellated images that can be linked together in any orientation ➢ Geomorphs

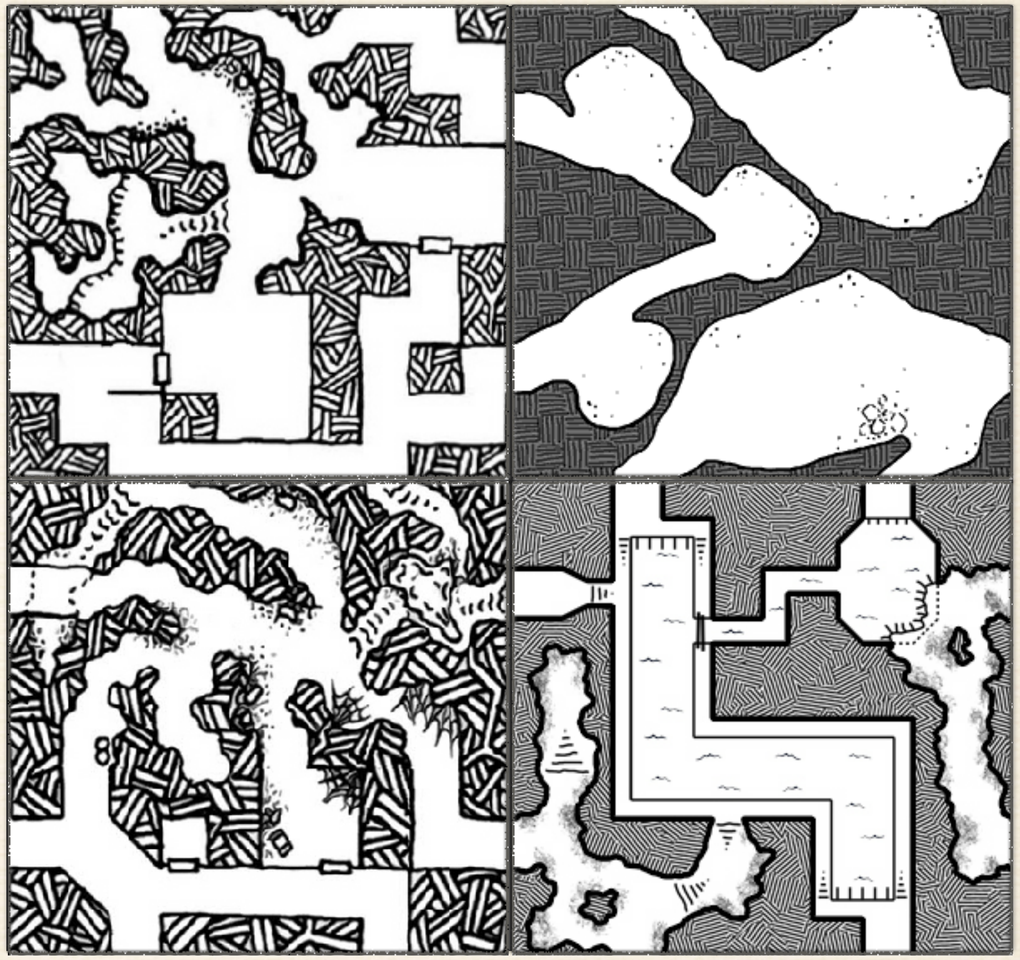
- By training our network with Geomorph tiles we are able to generate maps of adjustable sizes, which can be further refined with post-processing techniques and manual graphical adjustments

[1] D. Fernandes e Silva, R. Torchelsen, and M. Aguiar. "Dungeon level generation using generative adversarial network: an experimental study for top-down view games", in Anais do L Seminário Integrado de Software e Hardware, João Pessoa/PB, 2023, pp. 95-106, doi: https://doi.org/10.5753/semish.2023.229905

Dataset

- There is no pre-made Geomorph dataset available, so we made our own. All assets come from public github repositories[1] or artists' blogs[2]

- We augmented the dataset using combinations of rotation, flipping, dilation and erosion. After outlier removal, this resulted in a total number of images of 18'912

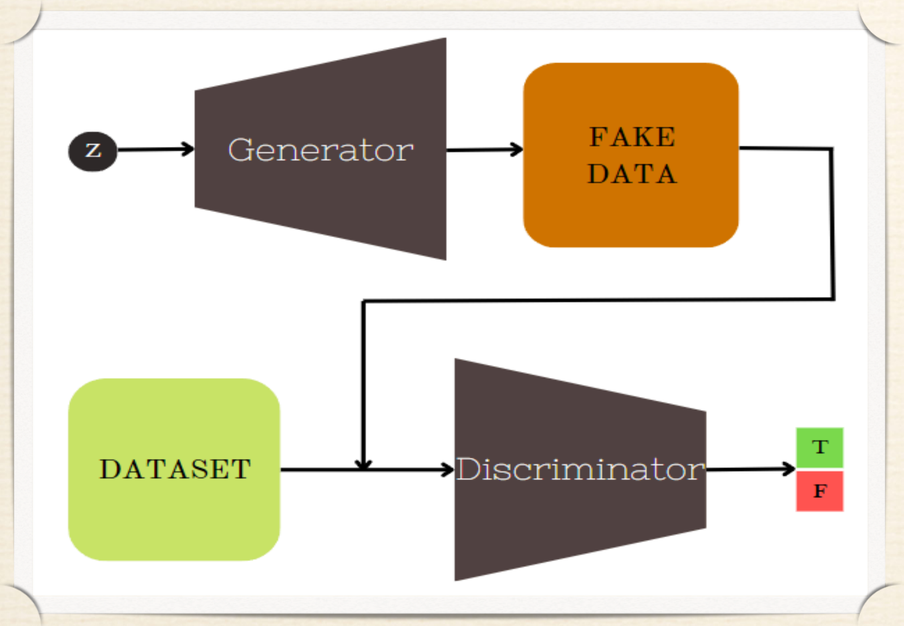


[1] Dave's Mapper GitHub repository: https://github.com/davmillar/DavesMapper
[2] Aeons & Augauries Blog: https://aeonsnaugauries.blogspot.com/

Neural architectures: DC-GAN (1)

Generative Adversarial Network: a game played by two competing players: a Generator that aims to minimize the following objective function, whilst the discriminator tries to maximize it

$$L(x, y) = E_x\left[log\left(D(x)\right)\right] + E_{\tilde{x}}\left[log\left(1 - D(\tilde{x})\right)\right]$$



DC-GAN architecture

DC-GAN: instead of using fully connected layers, opting for a convolutional neural network allows for less learnable parameters per layer, and in turn a higher number of layers

Neural architectures: DC-GAN (2)

Layer	Discriminator	Generator
1	Conv2D-LeakyReLU	ConvTranspose2D-BatchNorm2D-ReLU
2	Conv2D-BatchNorm2D-LeakyReLU	ConvTranspose2D-BatchNorm2D-ReLU
3	Conv2D-BatchNorm2D-LeakyReLU	ConvTranspose2D-BatchNorm2D-ReLU
4	Conv2D-BatchNorm2D-LeakyReLU	ConvTranspose2D-BatchNorm2D-ReLU
5	Conv2D-Sigmoid	ConvTranspose2D-Tanh

To understand how the resolution changes between layers, we refer to these formulas:

Con Layer	$out = (in - 1) \cdot stride - 2 \cdot padding + k\_size$
ConvT Layer	$out = \left\lfloor \frac{in + 2 \cdot padding - k\_size}{stride} \right\rfloor + 1$

Our experimentation landed us on two sets of hyperparameters:

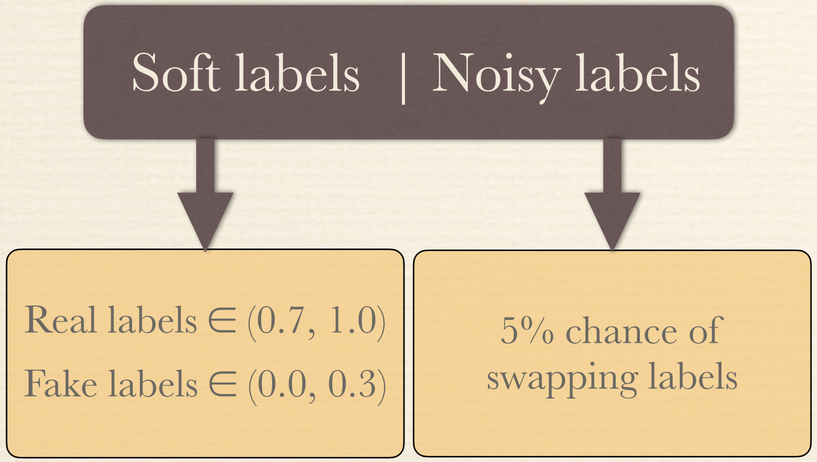
- k_size = 4; stride = 3; padding = 1 ➢ final output resolution: 284x284
- k_size = 4; stride = 2; padding = 1 ➢ final output resolution: 64x64

Note that the input layer of the Generator and the output layer of the Discriminator have different parametrization in stride and padding

DC-GAN training techniques

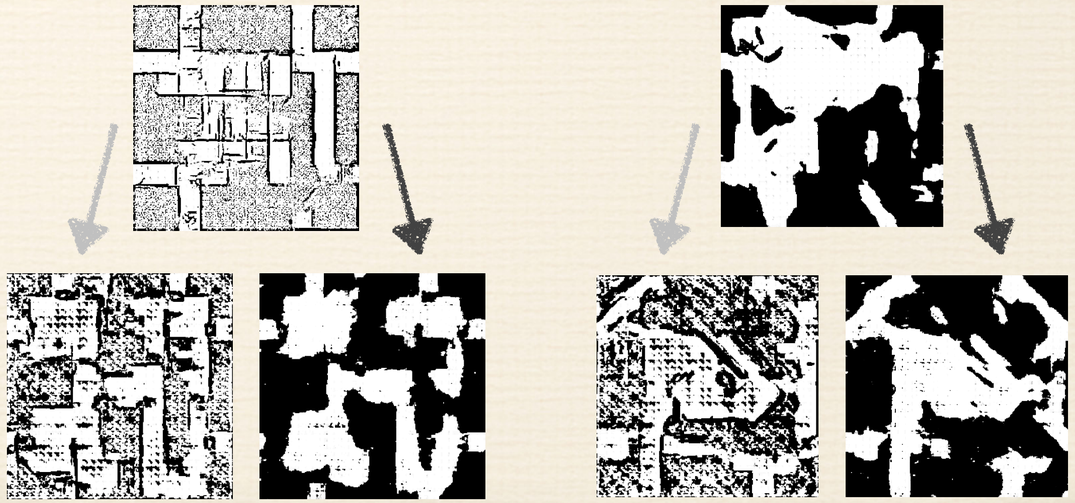
- We soon noticed that the Generator had trouble achieving good results: at best the training process broke after 320 epochs, at worst it outputted random noise

- This happens because the Discriminator trains much faster than the Generator, and is able to classify fake and real pictures extremely reliably, leaving no room for the Generator to learn



- To further fine-tune the results, it's possible to retrain the final Generator with a refined hand-picked dataset and an untrained Discriminator

- We found that this method improved some categories of geomorphs (conversely, caves and pictures with black backgrounds), whilst having adverse effects on others (mainly pictures with white backgrounds and labyrinth-type dungeons)



Neural architectures: W-GAN-GP

- This architecture substitutes the Discriminator with a Critic: instead of classifying pictures as fake ($\tilde{\mathbf{x}}$) and real ($\mathbf{x}$), it outputs the distance between the two data distributions in Wasserstain space

- This way the Generator always gets good feedback on its progress (provided the Critic is well trained), and doesn't get "stuck" as with DC-GAN

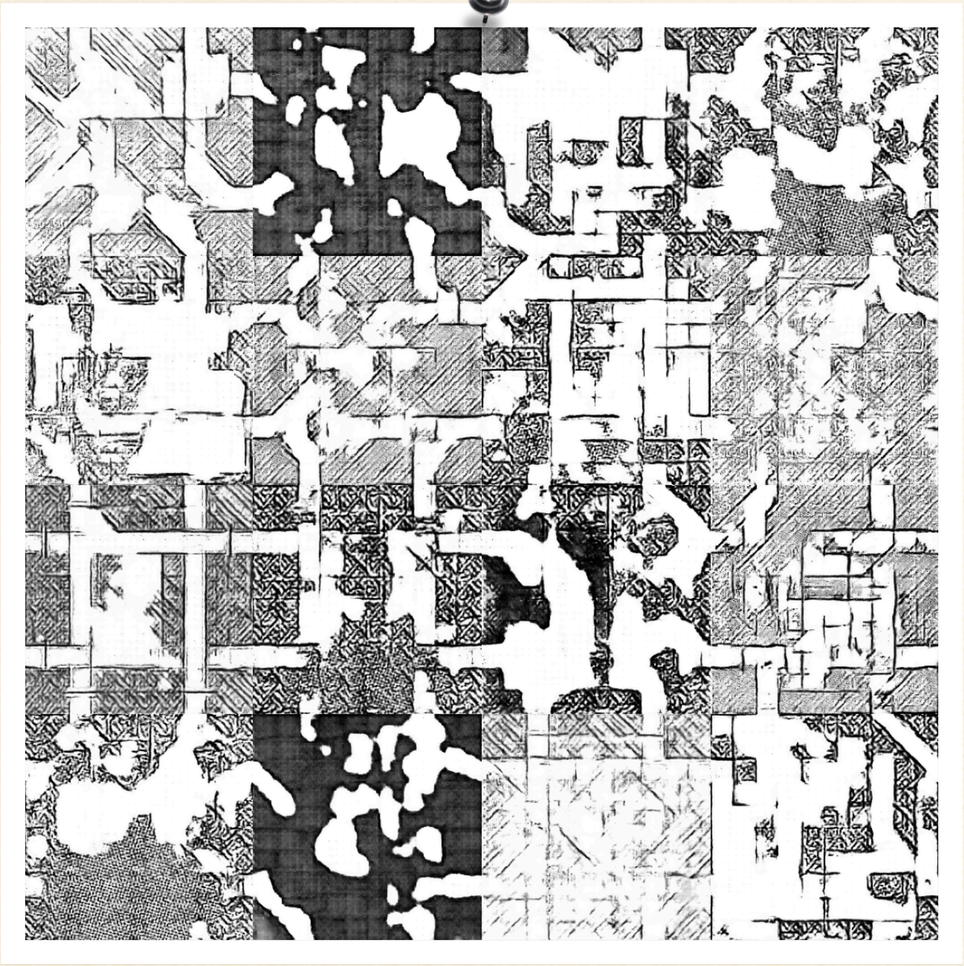
- The Wasserstain loss is not natively Lipschitzian; there exist many approaches to achieve this, and we selected Gradient Penalty[1], resulting in:

$$\begin{cases} L\left(x, y\right) = E_{\tilde{x}}\left[C\left(\tilde{x}\right)\right] - E_{x}\left[C\left(x\right)\right] + \lambda E_{\hat{x}}\left[\left(\| \nabla_{\hat{x}} C\left(\hat{x}\right) \|_2 - 1\right)^2\right] \\ \hat{x} = \alpha x + (1-\alpha)\,\tilde{x} \qquad \textit{where } \alpha \in [0,1) \textit{ and } \lambda = 10 \end{cases}$$

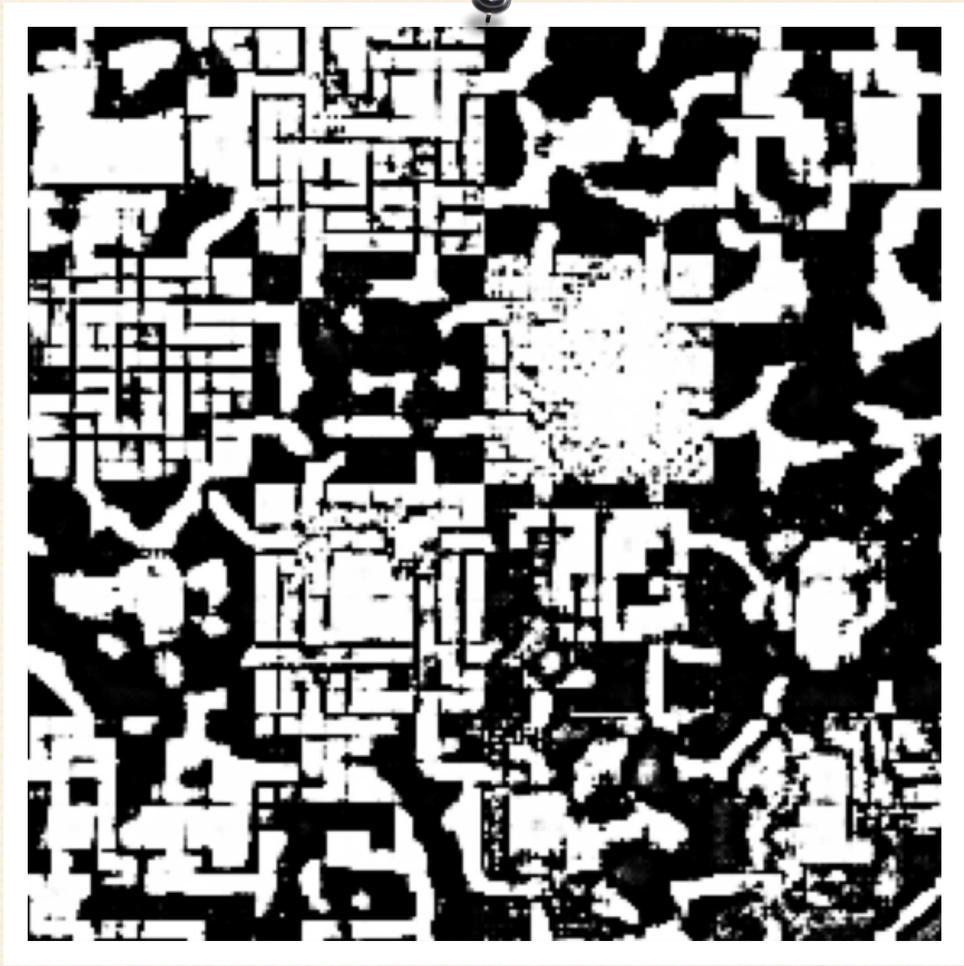
Layer	Critic	Generator
1	Conv2D-LeakyReLU	ConvTranspose2D-LayerNorm-ReLU
2	Conv2D-LayerNorm-LeakyReLU	ConvTranspose2D-LayerNorm-ReLU
3	Conv2D-LayerNorm-LeakyReLU	ConvTranspose2D-LayerNorm-ReLU
4	Conv2D-LayerNorm-LeakyReLU	ConvTranspose2D-LayerNorm-ReLU
5	Conv2D	ConvTranspose2D-Tanh

[1] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A. Courville ""Improved Training of Wasserstein GANs", doi: https://doi.org/10.48550/arXiv.1704.00028

Early results



DC-GAN
after 320 epochs of training, with a working resolution of 284x284, before applying soft&noisy labeling

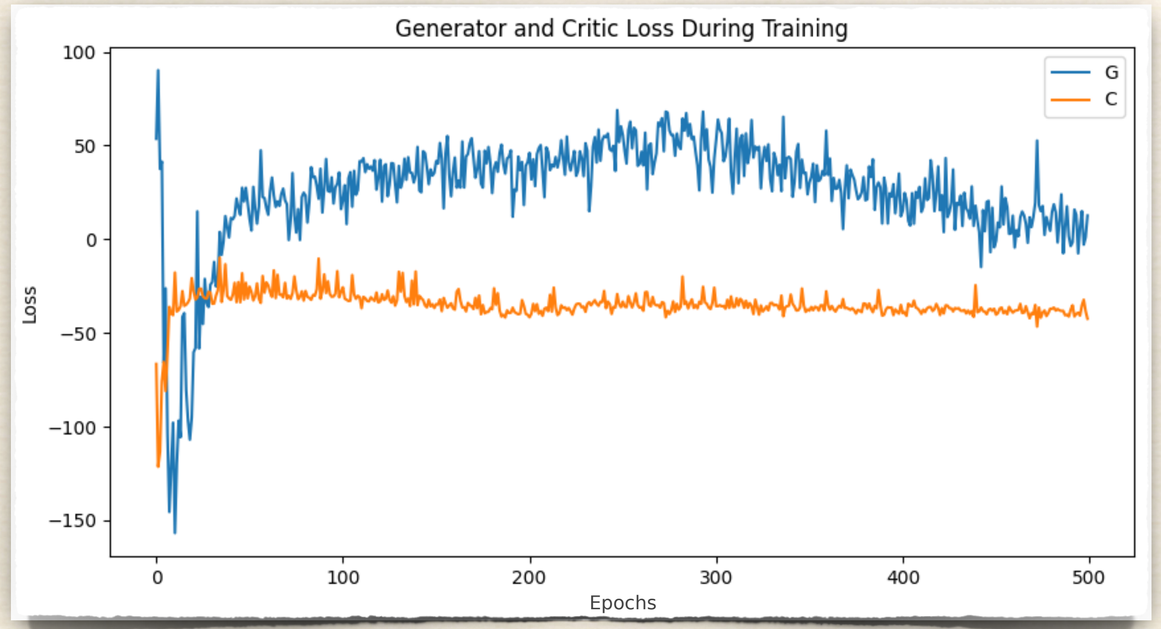


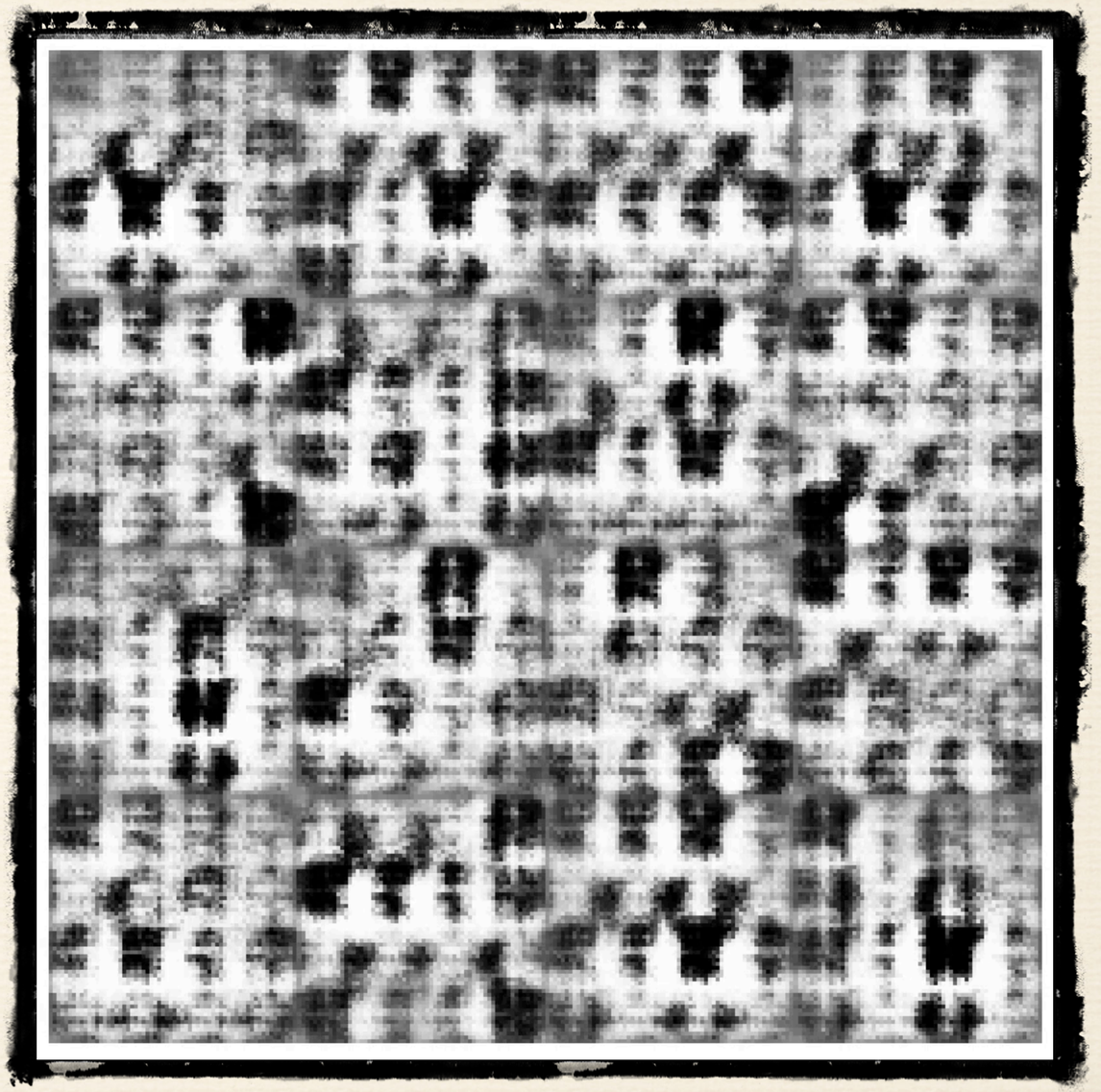
W-GAN
after 100 epochs of training, with a working resolution of 64x64

W-GAN-GP final results

At first we wanted to achieve 284x284 resolution, but found the training of such architecture impossible because of memory limitations within the GPU

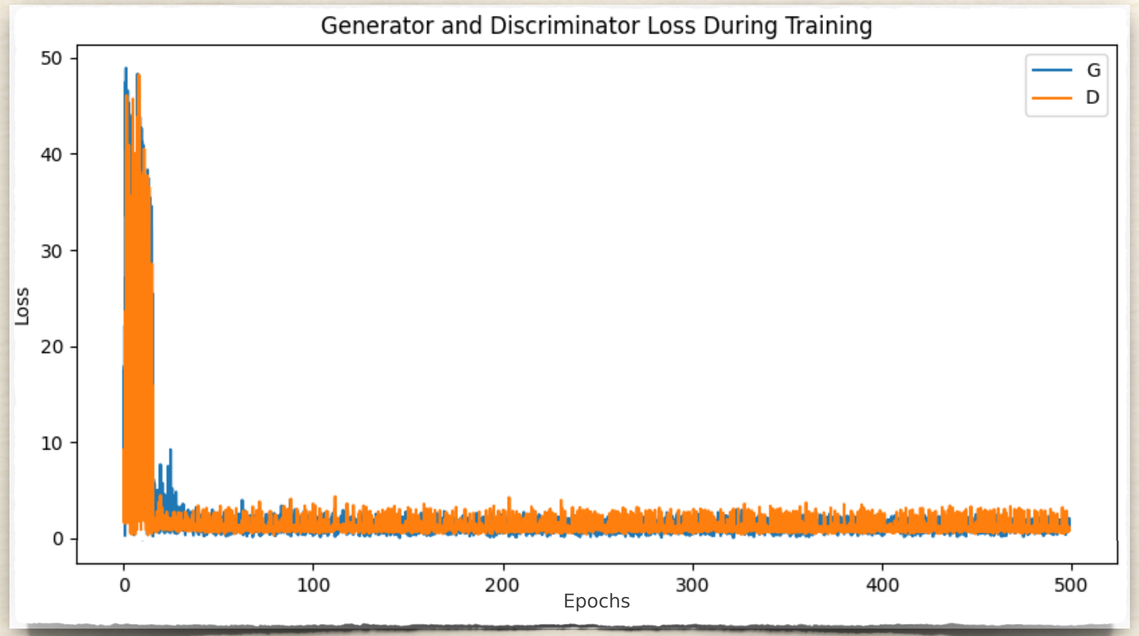
So we adjusted the resolution to 64x64, but doubled the number of kernels per layer

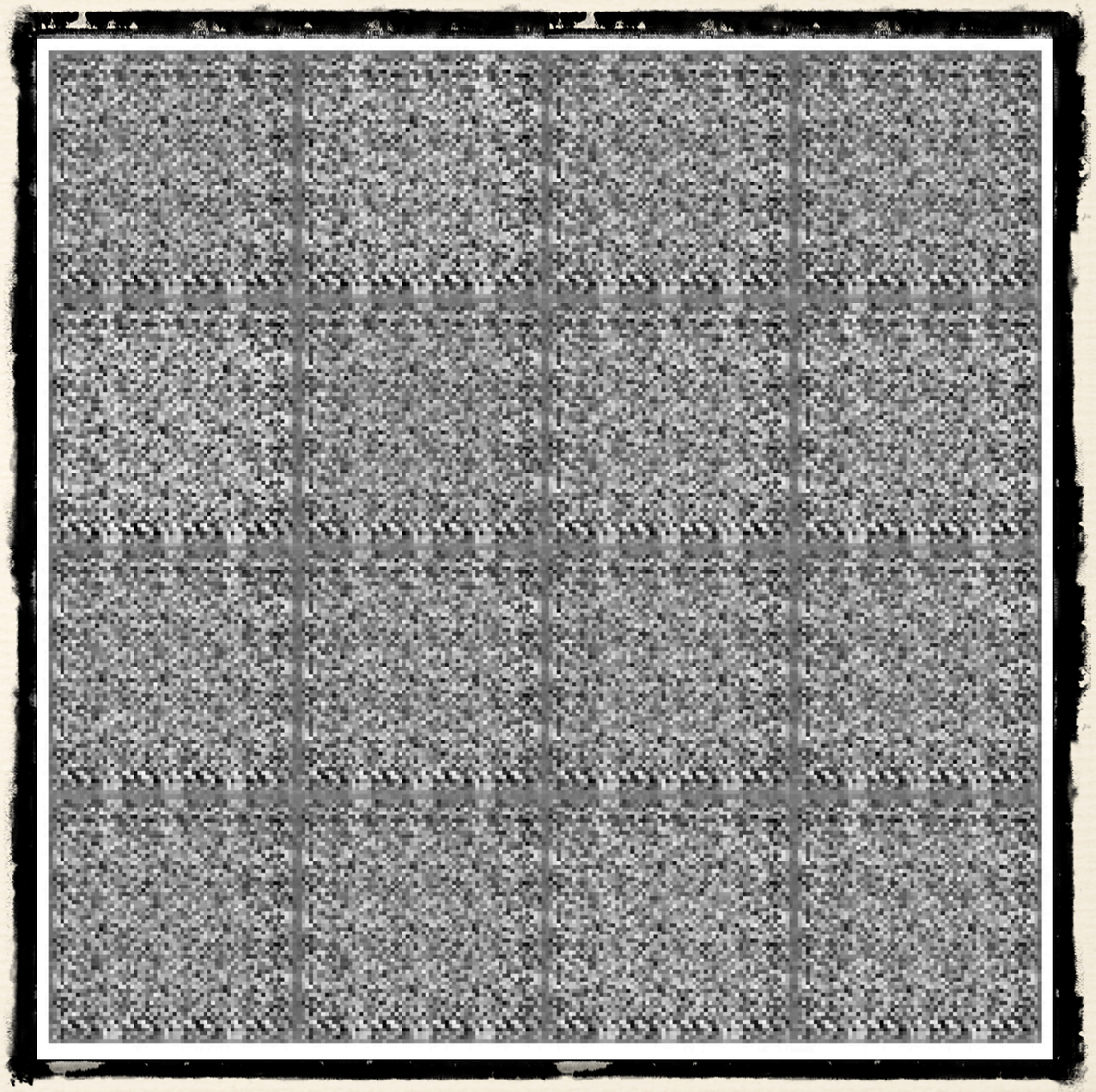




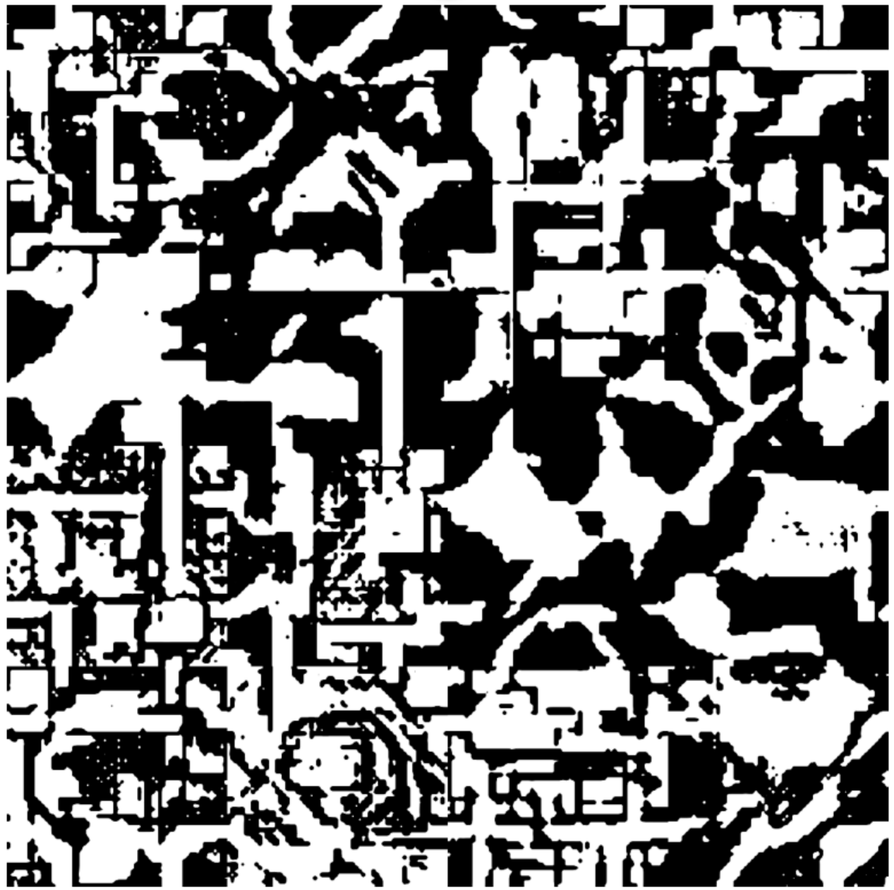
DC-GAN final results

To have a fair comparison between the architectures, we employed the same hyper-parameters used in the previous training

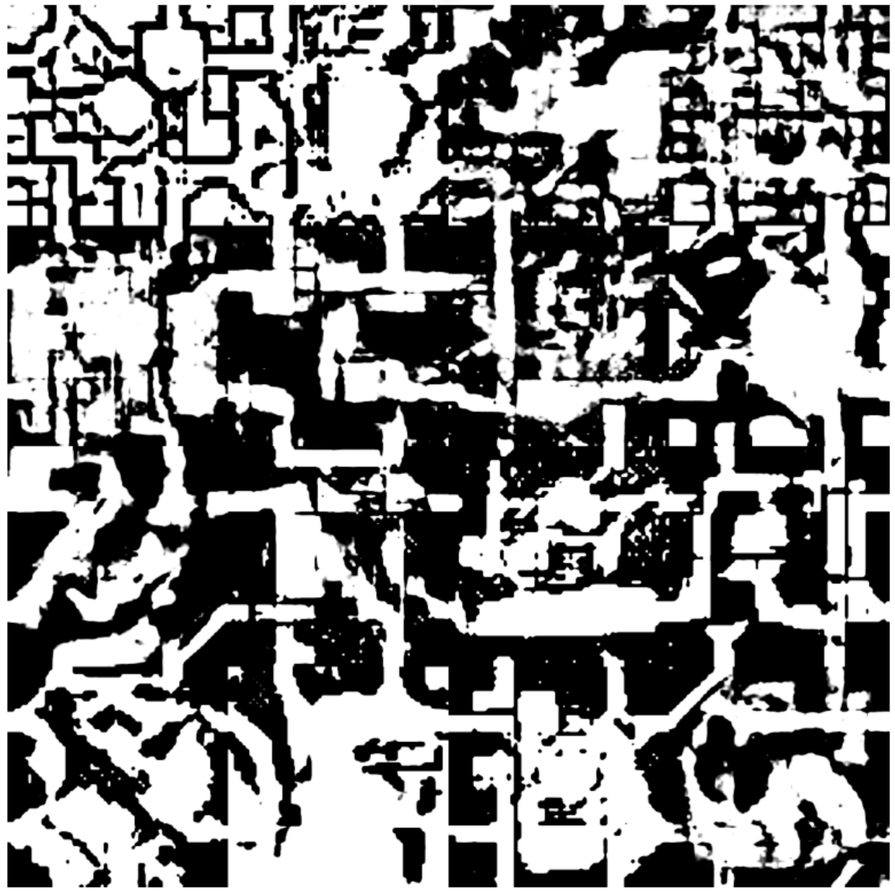




Comparison



DC-GAN with S&N labels
64x64 resolution, resized at 300x300, with contrast which helps removing white dots in the walls



Wasserstein-GAN-GP
64x64 resolution, resized at 300x300, with contrast to remove fog effect