A GAN-based Approach to Hydraulic Erosion

David Sommerfield
Winona State University
davidsommerfield.zg@gmail.com

Nicholas Moen
Winona State University
nickmoenpm@gmail.com

Abstract

We implemented a convolutional neural network that takes a heightmap as an input and outputs the result of a hydraulic erosion algorithm on that heightmap. The process of hydraulic erosion involves simulating the effects of water withdrawing sediment from the ground and flowing downward until the water either evaporates or reaches its sediment capacity. There are numerous implementations and methodologies to achieve this, but generally, when this happens, the water redeposits the earthen material it had picked up in its current location. The result is natural-looking terrain. The erosion algorithm that we based our model on is Gaea's hydraulic erosion algorithm. This algorithm takes about 0.6 seconds to run on one chunk of terrain. We tested our model by generating a batch of 32 on two different graphics card setups. On three Quadro RTX(6000/8000) GPUs, we can generate 32 images in 0.1 seconds. On one GeForce GTX 1060 we can generate 32 images in 0.3 seconds. The architecture we used is pix2pix. To capture the long-distance spatial patterns in the images like the long-distance water streaks that expand the length of the DEM, we needed to use dilated kernels and add more deep layers. Our model is intended to be used to enhance terrain in games that use procedural generation. The reason Gaea's algorithm and other erosion algorithms can't be used for procedural generation is that they are not fast enough to keep up with the rate at which new chunks will load as the player navigates the infinite world. It also has the potential to be used as a real-time terrain authoring tool.

1. Introduction

The problem at hand is converting a heightmap into a heightmap that has been through an erosion algorithm whose edges will match the edges of the surrounding chunks in procedural generation. To solve this problem we need to train a machine-learning model to generate hydraulic erosion based on information from the current chunk. The machine learning algorithm will not need to learn to connect adjacent chunks. To merge chunks that are next to each other, we will overlap chunks by an arbitrary amount and then perform a mathematical function called linear interpolation. This will give the edges of adjacent chunks generated by our generator a smooth look. The format we are using to learn terrain features is called DEMs. The input to our model is generated Perlin noise with different numbers of octaves and zoom levels. A DEM that is zoomed way in will appear flat and a DEM that is zoomed out will have more hills and bumps. The output was the corresponding DEM with erosion applied. There are many different types of terrain that we used to train our model to increase our model's ability to generalize.

2. Related Work

Generating Terrain Data for Geomorphological Analysis by Integrating Topographical Features and Conditional Generative Adversarial Networks

This paper proposes a terrain synthesizer based on the loess landform. The loess landforms are composed of silt and sand which are easily eroded by water. The researchers used a CGAN based on the Pix2Pix architecture. The input to their model is a sketch of terrain features. This sketch is composed of ridge lines and valley lines allowing the user to tailor their loess landform terrain output to their liking. Their output is a DEM that represents a loess landform. Their model has an advanced encoder-decoder architecture which allows the user to insert external conditions. In this

case, the external conditions are the lines that represent ridges and valleys. They have a U-Net encoder-decoder, which concatenates the feature maps from the encoding blocks to feature maps of the decoding blocks before applying transposed convolution. The discriminator is patch-based using a patch size of 30.

StyleDEM: a Versatile Model for Authoring Terrains

This paper introduces a novel generative adversarial network architecture called StyleDEM for terrain synthesis and authoring. What sets it apart from other implementations is it allows style considerations to be incorporated, which is important for designers in the entertainment industry. StyleDEM utilizes the StyleGAN architecture applied to Digital Elevation Models (DEMs), hence the name. The method allows for interactive authoring through user inputs by directly editing or manipulating the encoded latent space of the DEM. The mode can be broken into a few separate parts. The generator was trained on a dataset of DEMs, the encoder converts terrains into latent space representations. The generator and encoder are trained together. The training process occurs as it does in any other GAN architecture, with an interplay between the Discriminator and Generator. StyleGAN also uses a progressive growth mechanism where the network generates terrains and higher and higher resolutions.

Other applications are considered in this paper that do not directly translate to our problem but are nevertheless useful for understanding how the latent space can be manipulated in a StyleGAN. Style mixing is introduced, which lets a user mix features of different terrain types and scales. Experimental results demonstrate the effectiveness of StyleDEM in various editing tasks and sketch-based user authoring, style transform, terrain interpolation, and super-resolution. Finally, all of this is implemented into an add-on for the popular open-source 3D modeling and animation software Blender.

Interactive Example-Based Terrain Authoring with Conditional Generative Adversarial Networks

This paper proposes a multi-pass method where a real-world terrain database and a superimposed handmade sketch of its features are trained using a GAN to synthesize terrain given the sketch. From there an author can manually draw the features and the gan produces a terrain output, after which it is refined through erosion, erasing, and amplification. The most relevant of these details is the synthesized erosion, which uses a cGAN, making the computational cost of erosion simulation practically 0. They showed that for a given piece of data, the traditional algorithm takes around 40 seconds, whereas the synthesized erosion takes 0.025 seconds.

There are a few limitations to this implementation. Namely, if an image is too sparse, hallucinated artifacts may appear, which is undesirable, there is low modifiability of the erosion simulation that takes place, and the input for this algorithm is not a heightmap – instead the terrain is defined by crest line, rivers, and altitude lines which provide a lower degree of freedom when compared to a heightmap.

Image Super-Resolution Using Deep Convolutional Networks

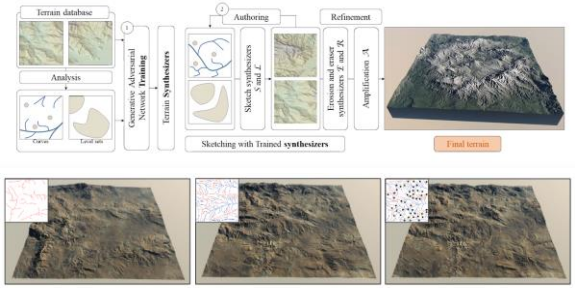
This paper proposes a deep learning approach for single-image super-resolution (SR). The paper introduces a novel convolutional neural network architecture specifically designed to directly learn an end-to-end mapping between low-resolution and high-resolution images. The SRCNN model comprises three convolutional layers and utilizes rectified linear units (ReLU) activations to enhance its capability in learning complex mappings. This research is relevant to our paper, because the architecture uses convolutions and tries to create an image output closely linked to the input image, but must infer details and minimize differences. This is a very similar problem statement to ours. Furthermore, the usage of multiple convolutional layers to extract different levels of detail is something we may have to use in our architecture due to the long-distance relationships present and this gives us insight into how such an architecture can be constructed.

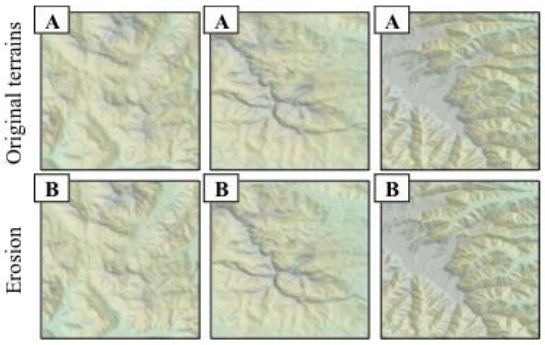
Interactive Terrain Modeling Using Hydraulic Erosion

The paper introduces an interactive physics-based modeling approach for terrains, enabling users to edit terrains with erosion and deposition algorithms. It couples hydraulic erosion algorithms, simulates slippage effects and provides various editing operations. The erosion simulation runs at least at 20

fps on standard computers for scenes with specific grid resolutions and material layers. A divide-and-conquer approach handles large terrain erosion by tiling and independent GPU calculations for each tile. The paper showcases erosion-based modeling features such as forming rivers, rain effects, and interactive manipulation with water bodies. It integrates three erosion algorithms, extends them to layered terrain representation, implements them on the GPU, and enables selective erosion of sub-tiles. The rendering is done both in real-time on the GPU and offline using Mental Ray for high-quality results. The system's GPU implementation allows for efficient simulation entirely on the GPU, without data transfer to the CPU during simulation, utilizing C++, OpenGL, and NVIDIA CG shading language. The approach provides immediate feedback and demonstrates feasible interactive physics-based terrain modeling.

There are three main components to this paper: terrain synthesis, user authoring, and terrain refinement. The terrain synthesizer is trained on Digital Elevation Models (DEMs) extracted from USGS Earth Explorer and NASA SRTM, these are implemented as 16-bit grayscale images, and superimposed handmade sketches corresponding to features on the terrain, namely: rivers, ridges, and altitude cues. From here the terrain synthesizer can take user-authored inputs, such as the previous cues, as well as cues used in the refinement process; level sets, erasure, and erosion. The terrain refinement step provides tools for refining terrains to the user through erosion and amplification. Erosion mimics natural erosion processes, while amplification adds small-scale details to the terrain, improving realism.





In the paper, the method is compared to other terrain sketching algorithms and physically-based simulations, highlighting the advantages of the proposed approach in terms of simplicity, interactivity, user-friendliness, and detail generation. There are limitations to this method, namely the need for a dedicated synthesizer for each task, potential failure to produce realistic results with sparse input sketches or in planar areas, and the user interface design restricting the simultaneous use of level set and curve sketches.

CAT to CT

This paper explores the transformative impact of deep learning on medical image analysis, particularly focusing on computed tomography (CT) and magnetic resonance imaging (MRI). It highlights how deep learning methods, particularly convolutional neural networks (CNNs), have revolutionized tasks such as lesion detection and classification, anatomical structure segmentation, image registration, and image enhancement and synthesis. The relevance of this research to our paper is that it takes black and white images and transforms it into an alternative that is informed by the original input, but also very different from the original input. Our project takes a heightmap, traditionally represented as a black and white image, and transforms it into a similar, but also different image.

3. Proposed Method

We used a Pix2pix architecture. This architecture uses an encoder-decoder generator and patch-based discriminator with a patch size of 70. The loss function of the GAN is MAE. The learning rate is 0.002. The optimizer used was Adam. The encoder has downsampling layers followed by dilated convolution layers with 5x5 kernels. The decoder has transposed convolutions followed by dilated convolution layers

with 5x5 kernels. The batch size is 4. The encoder brings the feature maps down to size 1x1x512. In our data, the image format used is 16-bit unsigned integers. In our network, the pixel values are normalized between -1 and 1.

Our GAN takes a procedurally generated input image. The size of images in our dataset is 1024x1024. We downsample these 1024x1024 DEM's into 256x256 DEM's using Lanczos interpolation. This is the best downsampling algorithm for reducing data loss.

We use data augmentation techniques to improve our model's ability to generalize. The images are rotated 90 degrees a random number of times from 0 to 3 before being input to the network. The loss function we use is MAE (mean absolute error) which we can use to evaluate the performance of our architecture. We also use the LPIPS metric to evaluate our model's performance. We use visual evaluation to compare the input and output of our network to the actual output generated by the erosion simulation after each epoch.

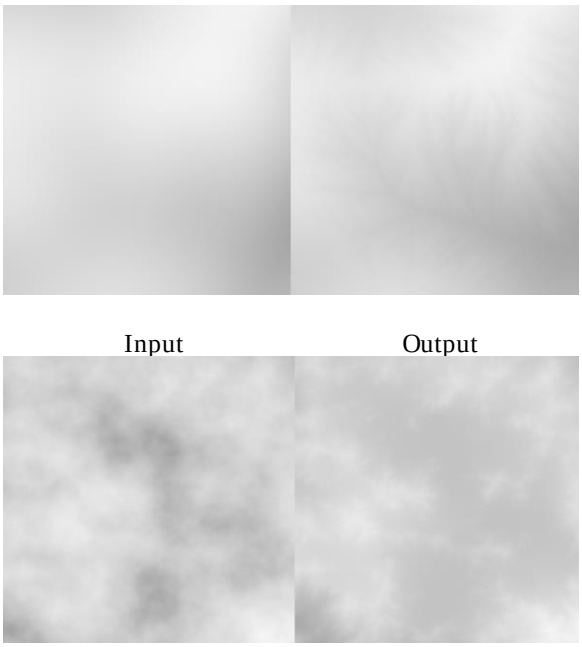
4. **Experiments**

4.1 Datasets

For our project, we used a custom data acquisition pipeline to generate the necessary terrain data.The primary tool utilized was QuadSpinner Gaea, a software application known for its capability to produce detailed height maps and simulate realistic erosion processes. The objective was to create a substantial dataset consisting of terrain heightmaps paired with corresponding erosion algorithm outputs.

The data generation process involved both the creation of input and output images encoding the results of an erosion simulation. The input images were generated using Perlin noise maps, incorporating various parameters such as scale, noise level, amplitudes, and variances to achieve desired terrain characteristics.

The DEM's in our dataset are 1024x1024 16-bit images. The file format we are storing the data in is the TIFF file format. We only use one color channel, as the DEMs we use are black and white.



Input Output

4.2 Evaluation metrics

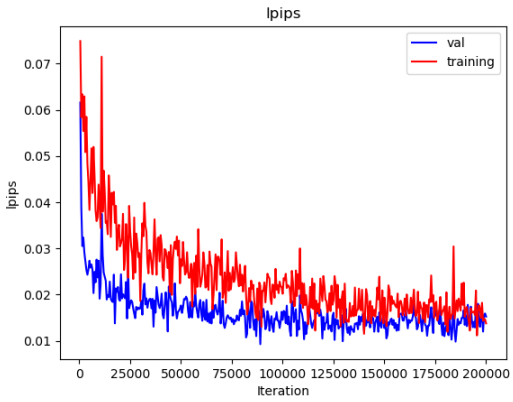
We used MAE and LPIPS to evaluate our results. We also used visual evaluation.

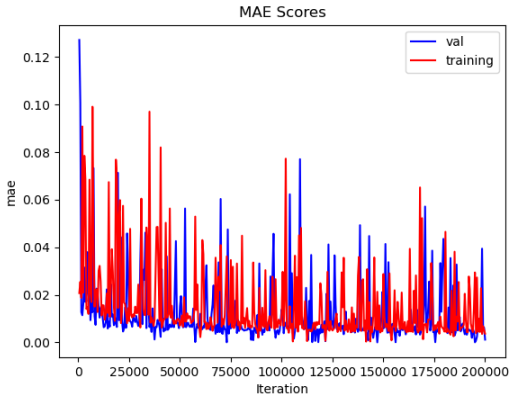
4.3 Implementation details

We use Python, Keras GPU, and Tensorflow. This may change in the future. We trained on 3 Quadro RTX (6000/8000) GPUs. Training took about 21 hours. We trained for 200 epochs.

4.4 Results

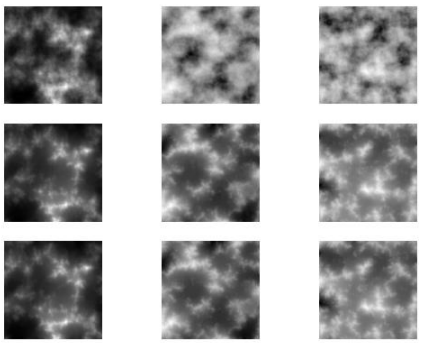
Our model was able to converge. Our LPIPs scores were around 0.011 at the end of training. The MAE score was as low as 0.0035.



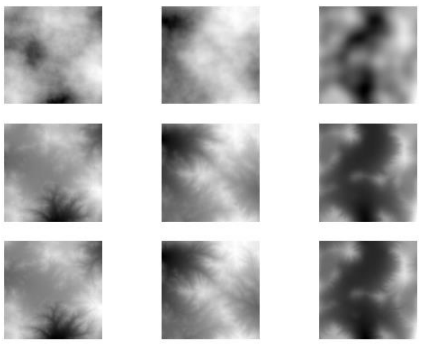


The next two images are created during training. The top row is input, the middle row is generated images, and the bottom row is the real output of the erosion algorithm.

196 epochs:

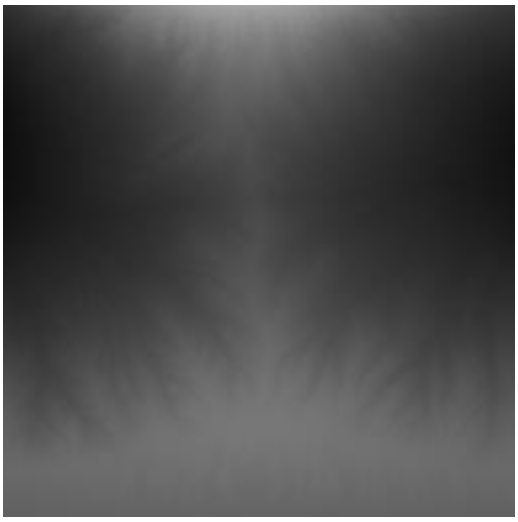


198 epochs:

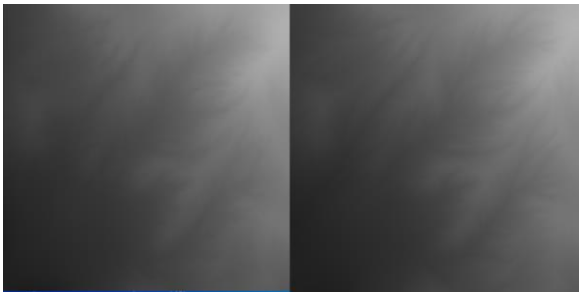


We can see it is doing a good job of capturing some long distance water streak lines and some more granular details in terrain that has higher octaves in the perlin noise. Next I will show a few examples of long distance water streak lines generated by our model.

Generated

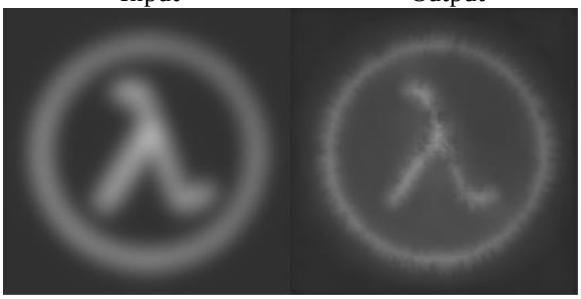


Generated Real



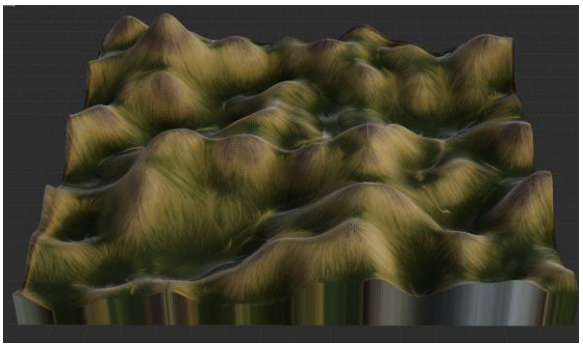
Our model has also been able to generalize well enough to perform erosion on input data that is not explicitly generated with perlin noise.

Input Output

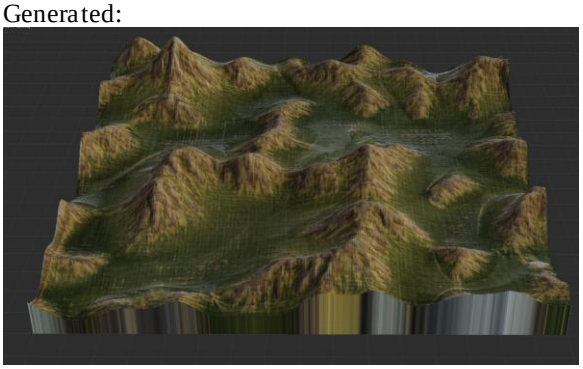


Here is an example of what our generated DEMs look like when rendered next to the input and real output DEM.

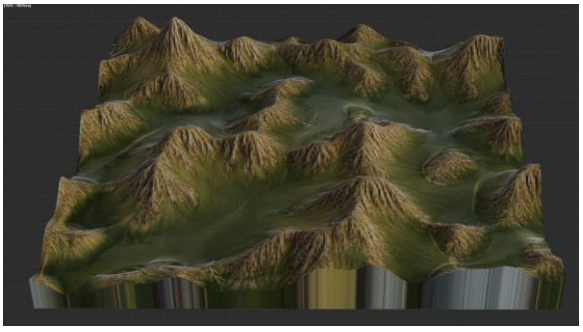
Input:



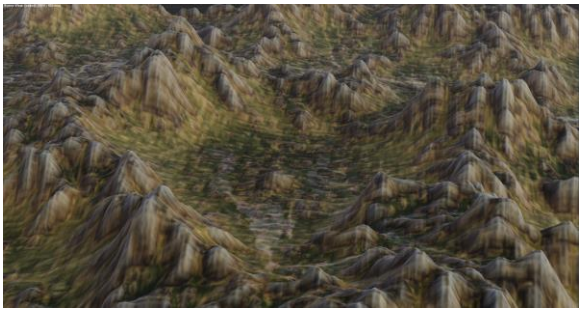
Generated:



Real:



Generated terrain render:



When you look closely you can see that the generated terrain has grid artifacts. It is extremely noticeable in regions on the DEM where the gradient is not very large. Using some post-processing we can clean up many of the noticeable grid artifacts. We can apply a blur to regions of the output DEM where the gradient is small. You find the slope for each part of the DEM then apply a blur effect to just the regions where the slope is below a certain threshold. Here is an example of what it looks like when we apply this blur to the generated erosion DEM.

5. **Rendering Our DEMs**

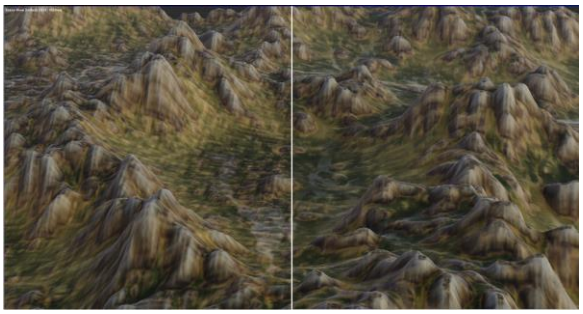
The terrain renders do have some noticeable differences between the real data and the generated data. There are some grid artifacts. All of the other pix2pix architectures that model erosion have resulted in similar artifacts. Some post-processing can be used to clean up a lot of the artifacts. Here are some examples of the artifacts that we have in our rendered terrain.

Real terrain render:

Generated terrain render with blur applied:

Generated versus real without blurring

Generated Real



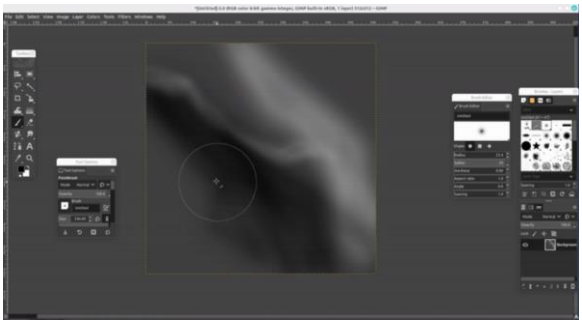
Generated versus real with blurring

Real Generated

6. **Real-time terrain authoring tool**

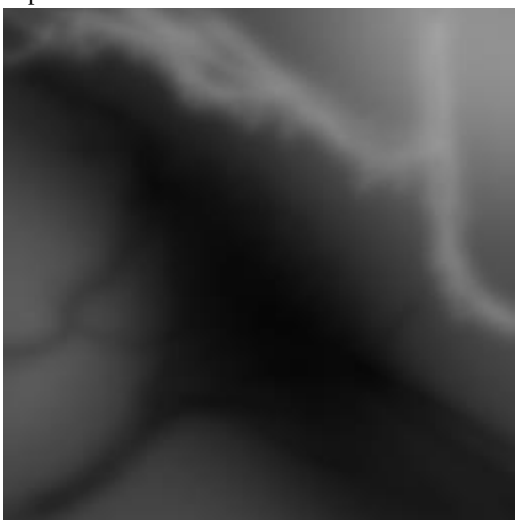
Our model can be used as the backend for a real-time terrain authoring tool. We would use a brush that can draw black and white lines/blur patterns. This brush would be called the terrain brush. There would be a drawing window and a rendering window where the output of the network is rendered as a 3d terrain model.

Here is an example of an input DEM being drawn in GIMP:

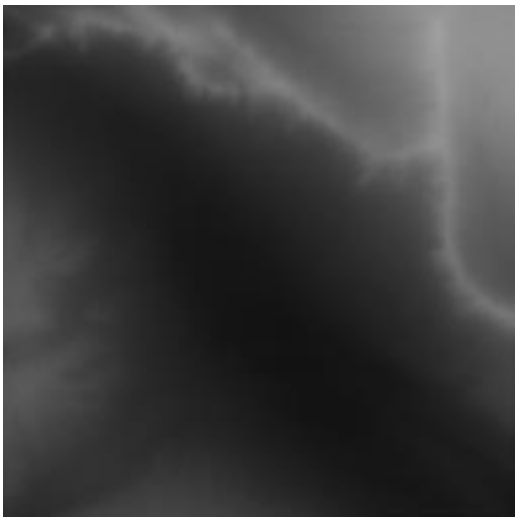


When we input this drawing to our network yields this result:

Input:



Output:



7. Conclusion

The goal of our architecture is to apply a hydraulic erosion algorithm to an input DEM. Our architecture is based on pix2pix. Our results were good at capturing spatial features like long-distance water streak lines. The level of detail produced by our generator was weaker than that of the original erosion algorithm. As with other pix2pix erosion simulation algorithms, we have grid artifacts. The small details that make the rock formations look more natural are also not showing in our generated outputs. There are some details, but not as many as the real erosion algorithm. Because we use DEM's as our input, our algorithm is flexible in its application. It can be used for procedural generation and it can be used for real-time terrain authoring. The real-time terrain authoring would give the artist more control over the output with our implementation than erosion CGANs that use ridge lines as input. Overall, our results show potential for a useful application. There are some improvements that would need to be made in output quality before our program will be useful for artists. However, for procedural generation our program would be able to generate terrains that are higher quality than current procedurally generated terrains. The only issue with this is that the model uses a large portion of GPU resources and may struggle to run in parallel with a videogame on smaller GPUs.

8. Resources

[1] [1]Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution

[2] using deep convolutional networks. IEEE transactions on pattern analysis and machine

[3] intelligence, 38(2):295–307, 2015.

[4] [2]́Eric Gúerin, Julie Digne, Eric Galin, Adrien Peytavie, Christian Wolf, Bedrich Benes, and

[5] Benoˆıt Martinez. Interactive example-based terrain authoring with conditional generative

[6] adversarial networks. ACM Trans. Graph., 36(6):228–1, 2017.

[7] [3] Hwang Woochan Jung Kyu-Hwan, Park Hyunho. Deep learning for medical image analysis:

[8] Applications to computed tomography and magnetic resonance imaging. hmr, 37(2):61–

[9] 70, 2017. doi: 10.7599/hmr.2017.37.2.61. URL http://www.e-sciencecentral.org/

[10] articles/?scid=1044313.

[11] [4] Ondˇrej ˇSt'ava, Bedˇrich Beneˇs, Matthew Brisbin, and Jaroslav Kˇriv́anek. Interactive terrain

[12] modeling using hydraulic erosion. In Proceedings of the 2008 acm siggraph/eurographics

[13] symposium on computer animation, pages 201–210, 2008

[14] Li, S., Li, K., Xiong, L., & Tang, G. (2022). Generating terrain data for geomorphological analysis by integrating topographical features and conditional generative adversarial networks. Remote Sensing, 14(5), 1166.

[15] Perche, S., Peytavie, A., Benes, B., Galin, E., & Guérin, E. (2023). StyleDEM: a Versatile Model for Authoring Terrains. arXiv preprint arXiv:2304.09626.