Multi-Theme Generative Adversarial Terrain Amplification

YIWEI ZHAO, EA Digital Platform – Data & AI, Electronic Arts
HAN LIU, EA Digital Platform – Data & AI, Electronic Arts
IGOR BOROVIKOV, EA Digital Platform – Data & AI, Electronic Arts
AHMAD BEIRAMI, EA Digital Platform – Data & AI, Electronic Arts
MAZIAR SANJABI, EA Digital Platform – Data & AI, Electronic Arts
KAZI ZAMAN, EA Digital Platform – Data & AI, Electronic Arts

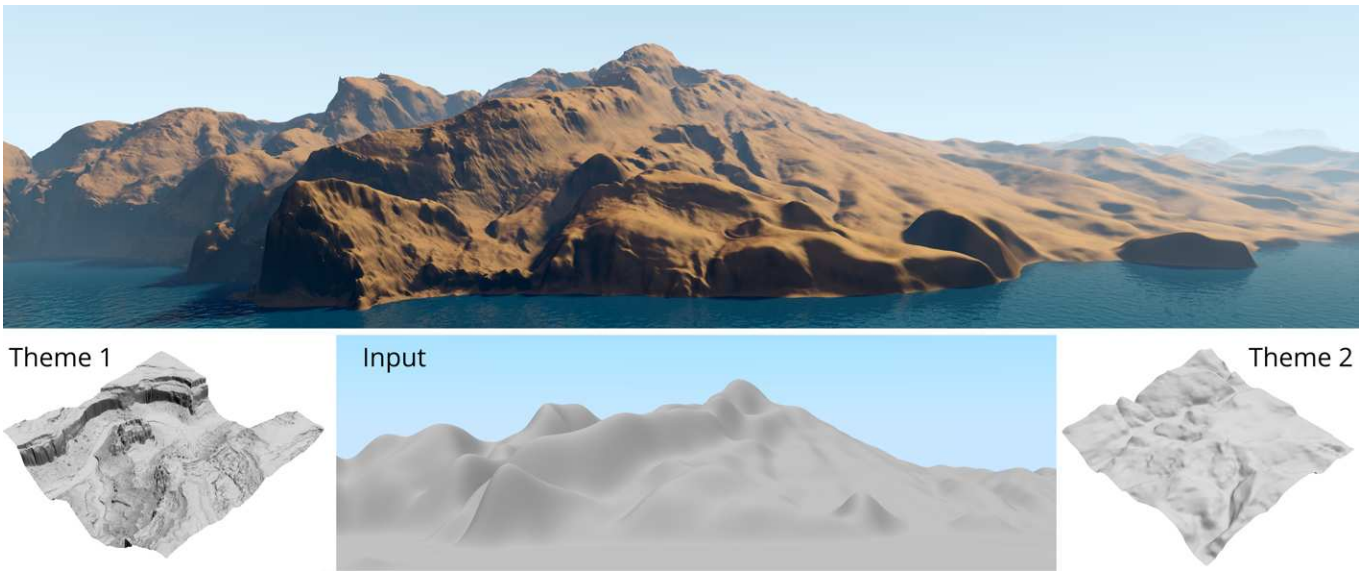

Fig. 1. The proposed Generative Adversarial Terrain Amplification (GATA) algorithm learns to amplify a low-resolution input terrain to a high-resolution one while transferring a desired output theme. By introducing the concept of theme embedding, GATA can gradually interpolate between two themes (Themes 1 and 2) from left to right while providing local and global coherence. This brings the quality of terrain detail amplification to a new level while adding a novel dimension to the control over the theme. The plot is the output of the generator and we did not perform any kind of postprocessing to the final rendering of the texture.

Achieving highly detailed terrain models spanning vast areas is crucial to modern computer graphics. The pipeline for obtaining such terrains is via amplification of a low-resolution terrain to refine the details given a desired theme, which is a time-consuming and labor-intensive process. Recently, data-driven methods, such as the sparse construction tree, have provided a promising direction to equip the artist with better control over the theme. These methods learn to amplify terrain details by using an exemplar of high-resolution detailed terrains to transfer the theme. In this paper, we propose Generative Adversarial Terrain Amplification (GATA) that achieves better local/global coherence compared to the existing data-driven methods while providing even more ways to control the theme. GATA is comprised of two key ingredients. The first one is a novel embedding of themes into vectors of real numbers to achieve a single tool for multi-theme amplification. The theme component can leverage existing LIDAR data to generate similar terrain features. It can also generate new fictional themes by tuning the embedding vector or even encoding a new example terrain into an embedding. The second one is an adversarially trained model that, conditioned on an embedding and a low-resolution terrain, generates a high-resolution terrain adhering to the desired theme. The proposed integral approach reduces the need for unnecessary manual adjustments, can speed up the development, and brings the model quality to a new level. Our implementation of the proposed method has proved successful in large-scale terrain authoring for an open-world game.

CCS Concepts: • **Computing methodologies** → **Neural networks**; **Learning latent representations**; **Shape modeling**;

Authors' addresses: Yiwei Zhao, EA Digital Platform – Data & AI, Electronic Arts, yiwzhao@ea.com; Han Liu, EA Digital Platform – Data & AI, Electronic Arts, haliu@ea.com; Igor Borovikov, EA Digital Platform – Data & AI, Electronic Arts, iborovikov@ea.com; Ahmad Beirami, EA Digital Platform – Data & AI, Electronic Arts, ahmad.beirami@gmail.com; Maziar Sanjabi, EA Digital Platform – Data & AI, Electronic Arts, maziar.sanjabi@gmail.com; Kazi Zaman, EA Digital Platform – Data & AI, Electronic Arts, kzaman@ea.com.

0730-0301/2019/11-ART200
https://doi.org/10.1145/3355089.3356553

Additional Key Words and Phrases: terrain amplification, generative models, generative adversarial networks (GANs), style transfer, embedding.

ACM Reference Format:
Yiwei Zhao, Han Liu, Igor Borovikov, Ahmad Beirami, Maziar Sanjabi, and Kazi Zaman. 2019. Multi-Theme Generative Adversarial Terrain Amplification. *ACM Trans. Graph.* 38, 6, Article 200 (November 2019), 14 pages. https://doi.org/10.1145/3355089.3356553

1 INTRODUCTION

1.1 Motivation

The graphical quality of games has advanced significantly over the past decade and is already approaching cinematic quality [NVIDIA 2019]. In addition, the game worlds are also growing in size to give the player the freedom to explore and an illusion of an open world. For example, Fortnite's map is 5.5km squared and Battlefield V's Halvoy map is 12km squared [Cotton 2019]. The first step towards creating such an open world is a controllable pipeline for creating large-scale high-resolution (hi-res) terrains, which adhere to the artists' intended terrain and game theme.

There are two common methodologies to generate such large-scale high-resolution terrains. The first one starts with finding a suitable real-world terrain, which mostly satisfies the needs of the design. The terrain is then manually edited to incorporate the requirements of the artists. Unfortunately, it is usually very difficult, if not impossible, to find real-world terrains that would satisfy all the needs of the artists, such as overall topographical structure and theme, given the ever growing size of the maps.

The second way of generating high-resolution terrains is a two-step process. In the first step, the artists resort to a plethora of existing resources and tools to generate their desired terrains but in low resolution. These tools include [BiteTheBytesUG 2019; Daz3D 2019; DigitalElement 2019; E-onSoftware 2019; PlanetsideSoftwareLLC 2019]. However, these low-resolution (lo-res) terrains would not satisfy the needs of cinematic quality graphics and would not carry the stylistic details desired by the artists. Thus, there is a need for a second step, called terrain amplification, where a lo-res terrain is refined to a hi-res one while adding consistent stylistic details. Throughout this paper, we call the stylistic details of a terrain, which is not captured by the lo-res terrain, the *theme* of the terrain. Our focus in this paper is on terrain amplification, while enabling to transfer stylistic details from different themes to the lo-res input. The existing terrain amplification methods could generally be categorized into three groups [Galin et al. 2019]: (1) physical simulation-based [Chiba et al. 1998; Musgrave et al. 1989], (2) procedural [Ebert and Musgrave 2003], and (3) example-based [Guérin et al. 2016]. It is worth noting that although some of these methods could also be used for terrain generation, we present them here in the context of amplification.

The physical simulation-based methods work via simulating a physical model of the environment, such as hydraulics or corrosion, for a long period until realistic details on the environment emerge. Hence, they are relatively slow. Moreover, it is usually difficult to work backwards from a desired terrain theme to tune their parameters. Hence, in spite of their ability to generate realistic terrains, artists are often reluctant to use these methods to build large-scale terrains.

Procedural methods, on the other hand, try to model the terrains without any use of physical models. They often resort to non-physical properties of the terrains, such as fractal characteristics, to generate outputs. Similar to the physical simulation-based methods, they can be difficult to control. While these methods tend to be relatively fast, the final results are usually not the most visually appealing.

Example-based methods learn how to generate relatively small but detailed and high-quality patches of terrain from a hi-res exemplar of a desired theme. These patches are then stitched together to obtain large-scale terrains. Example-based methods are relatively fast, but they are prone to generating output results that suffer from local and global incoherence, particularly at the stitching points between the patches [Argudo et al. 2017]. Moreover, the existing methods can only generate themes that are fed to them in a comprehensive exemplar and changing the theme requires replacing an entire exemplar.

Before discussing our solution for terrain amplification let us introduce a desired set of requirements that we would like an ideal terrain amplification method to satisfy. It is worth mentioning that this set is similar to the one proposed in [Galin et al. 2019, Section 7].

- **Fast Interactivity.** As we have emphasized previously, the method should be capable of amplifying terrains with minimal manual intervention from the artists. Moreover, if the artists need to interact with the tool, it should be fast enough so that they can quickly iterate on the design to obtain desired outcomes.
- **Local & Global Coherence.** Most terrain amplification tools start by amplifying a relatively small patch of terrain and then stitching different patches together in order to get large-scale terrains of the desired size. These patches usually have overlaps and therefore one would use blending techniques to put them together. When the output patches do not resemble each other at the stitching area, some local incoherence, with respect to the theme, might occur [Argudo et al. 2017; Galin et al. 2019]. Furthermore, it is difficult to maintain the global coherence of the generated terrain when using patch-based methods [Galin et al. 2019]. Thus, we wish for a patch-based method that locally adheres to the theme (local coherence), while maintaining better global coherence with the input, when compared with the state-of-the-art patch-based methods.
- **Control over Theme.** The artists usually desire seamless control over the theme of the output with the least amount of manual intervention. Hence, we want our method to give the artists such control over the stylistic details. Ideally, the method can also generate terrains with themes that are not directly represented in its exemplar.

1.2 Our Contribution

In this paper, we propose to use a new example-based approach, which we call Generative Adversarial Terrain Amplification (GATA),

to amplify lo-res terrains while handling multiple themes. Our approach is based on a machine learning technique, called Generative Adversarial Networks (GANs), introduced in [Goodfellow et al. 2014]. GANs have achieved great success in synthesizing images in many tasks. For more details on prior work on GANs, especially in contexts that are related to our work, see Section 2. GATA uses an embedding space to encapsulate the information of multiple themes. We learn these embeddings alongside our generative model, which inputs an embedding and a lo-res terrain to output a hi-res one. Note that we use the word model here to refer to the final generative tool that is modeling the process of amplifying a lo-res terrain into a hi-res one. As a consequence, the proposed approach supports the following properties:

- GATA is sufficiently fast to provide the artists with an interactive terrain amplification tool for editing and beautifying the desired parts of the terrain.
- GATA exhibits better coherence with the input, compared to the state of the art example-based methods; see Section 4 for more details. This leads to consistency at stitching points of the patches as well as better global consistency with respect to the input.
- Controlling the output's theme in GATA is as easy as changing the input embedding vector of the generator and rerunning the inference, which takes a fraction of a second.
- GATA allows for the creation of new fictional themes that do not exist in the input exemplar (and may not even exist in the real world) by tweaking the theme embedding vector. For example, we can interpolate between two themes and smoothly transition the stylistic details of the output from one to another.

2 RELATED WORK

In this section, we review the existing methods that are most relevant to ours. We start by reviewing physical simulation-based methods (Section 2.1), procedural methods (Section 2.2), and example-based methods (Section 2.3) for terrain amplification. See [Galin et al. 2019] for a recent comprehensive review of these methods. We will then review the relevant work on generative models in graphics and their relevance to the proposed method for terrain amplification (Section 2.4).

2.1 Physical Simulation-Based Methods

The physical simulation-based methods use physical processes such as water diffusion or erosion [Musgrave et al. 1989], thermal weathering [Musgrave et al. 1989] and chemical processes [Wojtan et al. 2007] to generate hi-res realistic terrains from lo-res inputs. In these methods, the terrain would be represented as a 2D heightfield map [Chiba et al. 1998; Cordonnier et al. 2016], layered data structures [Benes and Forsbach 2001], or 3D volumetric data [Beneš et al. 2006]. Then, detailed and fine simulations are run on those representations to obtain the final hi-res results. Running physical-based methods is time-consuming, especially if the end goal is to obtain a detailed hi-res terrain. To overcome this, [Guérin et al. 2017] proposed to approximate erosion simulations using machine learning techniques. Another main problem with physical simulations-based methods is the lack of flexibility in generating different stylistic details, i.e. themes. In most of these methods it is impossible to change the theme and in some others it requires a careful handcrafting of the simulation parameters and the method still might be unable to generate all the desired stylistic details.

2.2 Procedural Methods

In this category of methods, one utilizes non-physical fractal properties of the terrains to model them through stochastic processes such as fractional Brownian motion [Mandelbrot and Van Ness 1968]. They often use techniques such as subdivision [Mark and Aronson 1984], faulting [Ebert and Musgrave 2003; Mandelbrot 1982], or noise functions [Lagae et al. 2009; Perlin 1985; Worley 1996] to achieve this goal. Despite their speed, the final result of procedural methods often lacks realism as well as local and global coherence. Moreover, these methods are often very difficult to control.

2.3 Example-Based Methods

In contrast to procedural and simulation-based methods, example-based algorithms are data-driven. By extracting patches from a hi-res exemplar and stitching them to enhance the input, the idea of example-based methods is widely used in texture generation [Wei et al. 2009] and image super resolution [Freeman et al. 2002]. [Zhou et al. 2007] introduced this method to the terrain generation domain, which was followed by several improvements for better user control [Gain et al. 2009, 2015], virtual world's interactive editing [Emilien et al. 2015], and blending of different land-form features [Génevaux et al. 2015; Guérin et al. 2016]. By enhancing the input with moisture, soil type and the vegetation density information, [Argudo et al. 2017] achieve improved local and global coherence. However, the extra information may not always be available in the dataset.

To minimize the amount of manual labor in this process of patch replacement, [Guérin et al. 2016] proposed sparse modeling, which uses dictionary learning methods to choose the best matching patches for a target patch from the exemplar based on elevation similarity. Although this process can minimize the amount of manual labor, sparse modeling also suffers from a couple of shortcomings detailed below:

(1) The local elevation similarity does not guarantee a good fit from the source patch to the target patch. Moreover, all the patches are always chosen independently of each other. Hence, even if the local features of the terrain are preserved, sparse modeling still cannot guarantee consistency at the intersection of the patches. In fact, we will show some results on the local and global coherence of sparse modeling in Section 4.
(2) Sparse modeling requires a careful selection of the exemplar, where all the samples have a consistent theme that also fits the input. After choosing an exemplar, the method is locked into generating only one theme and changing the theme would require changing the entire exemplar. Even in that case, the method can only generate output themes from which sufficient examples are available to form a comprehensive exemplar.

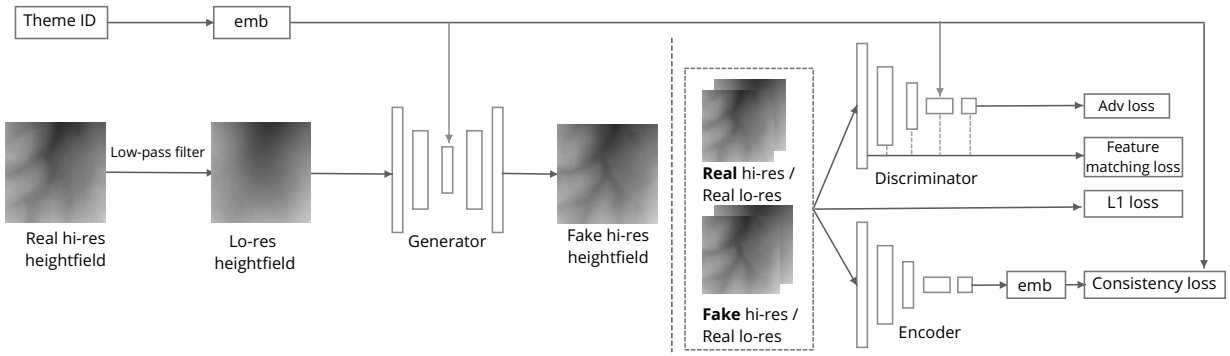

Fig. 2. Training architecture and pipeline overview for GATA. There are multiple components, including generator, embedding, discriminator, and encoder that are described in Section 3.1.

2.4 Generative Models

Generative models learn a target probability distribution from its samples. For example, one can use a generative model to learn the distribution of facial images [Radford et al. 2015], and then use it to draw new samples (i.e., generate fake facial images). In the past few years there have been several major breakthroughs in the area of generative modeling in the machine learning community, especially with the introduction of Variational Auto Encoders (VAEs) [Doersch 2016; Kingma and Welling 2013] and Generative Adversarial Networks (GANs) [Goodfellow et al. 2014]. These methods have been shown to be very powerful at learning distributions of complex data such as natural images that live in low-dimensional manifolds that are hard to describe manually. They have been successfully used in many image related tasks, such as image generation [Karras et al. 2018], image to image translation and image style transfer [Isola et al. 2017; Zhu et al. 2017a].

Our work builds on GANs [Goodfellow et al. 2014]. In particular, we use a variant of GANs called conditional GAN (cGAN) [Mirza and Osindero 2014], which learns a distribution conditioned on a low-dimensional control input containing further information about the target probability distribution. Due to their superb capability in modeling conditional distributions, cGANs have been successfully used in achieving super resolution [Ledig et al. 2017], and performing image to image translation [Isola et al. 2017] (style transfer). [Isola et al. 2017] proposes a Pix2Pix network architecture, which inputs a style and an image, and tries to transfer the style over to the input image. Although this method is effective in transferring one style to another, one of its limitations is that it would require to train one such model for any source and destination style tuple. In order to incorporate the multi-theme property into our formulation we use an architecture similar to [Isola et al. 2017], but we also include a novel theme embedding that is trained alongside the generative model; see the supplementary material for more details.

The use of latent embeddings in generative models, similar to our theme embedding (see Section 3), has been at the core of many recent works in machine learning community [Donahue et al. 2016; Perarnau et al. 2016; Ulyanov et al. 2018]. While BiGAN [Donahue et al. 2016] and AGE [Ulyanov et al. 2018] use encoder and a similar consistency loss (see Section 3.), they only consider the unconditional problem with no optimization in the latent space. On the other hand, IcGAN [Perarnau et al. 2016] which considers the conditional GAN problem only uses given fixed latent vector in cGAN alongside an encoder. Finally, GLO [Bojanowski et al. 2018], which is the closest to our approach, optimizes the latent representation, but only considers unconditional problem. In contrast to all these approaches, our work focuses on the conditional generation, where the latent representations, i.e. theme embeddings, are trainable; see Section 3. In such a case, decoupling the latent representations, i.e. theme embeddings, from the conditions, i.e. lo-res inputs, is of utmost importance. Such a decoupling is achieved through the careful design of the training pipeline; see Section 3.

Generative models have also been extensively used in the past few years in the area of computer graphics, e.g., 3D shape modeling and generation [Liu et al. 2017; Wang et al. 2018], rendering hair and 3D scenes [Feygina et al. 2018; Wei et al. 2018], fluid flow generation [Xie et al. 2018], food image generation [Fujieda et al. 2017], generating mass models [Kelly et al. 2018], animated facial image generation [Geng et al. 2018], transferring shape deformation [Gao et al. 2018], photo-to-caricature translation [Cao et al. 2018], face swapping [Natsume et al. 2018], and selfie-to-avatar translation [Nagano et al. 2018].

In the specific area of terrain generation, [Guérin et al. 2017] used a cGAN to generate heightfields from simple input controls, such as sketches or contours, that contain high-level information about the location of valleys and peaks. The generative model in [Guérin et al. 2017] falls short of generating hi-res terrains with a specific theme. [Guérin et al. 2017] proposed to amplify the lo-res output of the generative model using sparse modeling [Guérin et al. 2016] to generate hi-res terrains. Another related work in this domain is [Argudo et al. 2018], which uses super resolution techniques to amplify a tuple of an aerial image and its corresponding lo-res terrain to achieve a hi-res one. This method cannot generate multi-theme outputs and also needs an extra aerial image which might not be available in many cases.

It is worth noting that another line of work that uses Convolutional Neural Networks (CNNs), but not generative models, to perform style transfer on normal images is presented in [Gatys et al.

2015, 2016]. In this method, the input is two hi-res images, which would be combined in a way that the style of one input is overlaid on the details from the other image. Although, this method works fine with normal images, it is not applicable to our problem as one of our input images is of low resolution. Moreover, due to the fact that this method is designed for normal images, and not heightfields, we have found this method to perform poorly on our specific terrain amplification task; see Section 4.

3 PIPELINE OVERVIEW

In this section, we explain our overall approach for training and deploying the proposed Generative Adversarial Terrain Amplification (GATA) method. Similar to other example-based methods, such as sparse modeling [Guérin et al. 2016], we first focus on generating a fixed-size output patch from an input patch of the same size. If the chosen patch size is too small, many stylistic details in the patch will be lost. On the other hand, a large patch size will increase the computational cost of training. Note that training is only performed once, covering inference for all themes. As such, we choose the patch size for our method to be 256×256 pixels. See Section A.1 for more details on the choice of our dataset and patch size; and Section 4 for experiments on the impact of the patch size.

To generate terrains at arbitrarily large scales, we generate overlapping patches of size 256×256 pixels and blend them to obtain the overall terrain. We have fixed the amount of overlap between the adjacent patches to be 128 pixels. The use of overlapping patches and blending them with a mask is a standard approach in the literature, e.g., [Argudo et al. 2018; Guérin et al. 2016]. In order to blend the patches we use a weighted mask that was previously proposed by [Guérin et al. 2016].

We prepared our training data using LIDAR images from the State of Oregon Department of Geology and Mineral Industries (DOGAMI). The landscape in the State of Oregon is relatively diverse spanning mountains, hills, lakes, rivers, beaches, and deserts, which makes a rich dataset for creating different themes. We break the dataset into non-overlapping train and test patches used throughout the paper. See Section A.1 in the supplementary material for more information on the training and test datasets.

3.1 Training Architecture

The overall training architecture for GATA is summarized in Fig. 2. The main module is the generator, which is implemented using a CNN. The generator operates on lo-res inputs and produces hi-res ones. Remember that we wish the model to be multi-theme, i.e., it should be capable of amplifying an input terrain to an output with an arbitrary theme. Hence, we need to feed the theme information into the generator module as well. For more information on the generator architecture see Section A.3 in the supplementary material.

The theme information is embedded in 1024 dimensional vectors, called theme embeddings. Note that these embeddings are also *learnable* and are trained alongside the generator at training time. We refer to the collection of all themes used in training as the theme glossary; see Section A.1 for further information. See Section A.4 in the supplementary material for more details on the embedding vectors.

In addition to the generator and theme embeddings, the overall training architecture incorporates two more components, namely the discriminator and the theme encoder. Similar to the generator, the discriminator is also implemented using a CNN. It performs the classification task between fake and real terrains. In other words, it takes in a tuple of lo- and hi-res terrains and decides if the hi-res one is a real amplification of the lo-res terrain or a fake one generated by the generator. In order to help the discriminator, we also feed in the theme embedding for the corresponding input to the discriminator. See Section A.5 in the supplementary material for more details.

The encoder module is also a CNN (see Section A.6) that extracts the embedding vector from hi-res results. We use the output of this encoder to ensure that the learned theme embedding is consistent and can be derived from the hi-res terrains. Thus, we incorporate a cycle consistency loss [Zhu et al. 2017a] between the theme embedding and the encoder's output to our training. As it is common in all adversarial training methods [Goodfellow et al. 2014; Isola et al. 2017; Mirza and Osindero 2014], the goal of the discriminator is to increase its accuracy in differentiating between real and fake height fields, while the generator tries to reduce this accuracy by generating more realistic results. This constitutes the adversarial part of the training procedure. See Section 3.3 for the overall formulation of the problem incorporating all losses.

Although adversarial training is essential to generating believable fake results, such as fake facial images [Liu et al. 2015; Radford et al. 2015], it is known to be insufficient for complex tasks, such as ours. [Dosovitskiy and Brox 2016; Isola et al. 2017; Salimans et al. 2016] proposed to augment the adversarial training with other losses to ensure consistency and help the generator produce more realistic results; for more information on the overall loss and training see Section 3.2.

3.2 Training Losses

The overall loss used to train our generative model is comprised of several different components. Before we proceed with more details about these parts, let us define some notations. We refer to (T_{low}, T_{real}) as the tuple of the lo-res and the real hi-res terrains. We use T_{fake} to refer to the fake hi-res terrain, which is obtained as the output of the generator G applied to T_{low} and the theme embedding *emb*. We use *Emb* to denote the matrix of all theme embeddings for all of the themes used in training. We denote the output score of discriminator D on real and fake inputs by s_{real} and s_{fake}, respectively. We use $\mathbb{E}_{real}$ and $\mathbb{E}_{fake}$ to denote the empirical expectation over real and fake samples, respectively. Finally, we use E to refer to the encoder.

With these definitions in place, we are now ready to discuss different components of the loss function. The first component is an adversarial loss [Goodfellow et al. 2014] which relates to the adversarial nature of the generator and discriminator networks. The adversarial loss, L_{adv}, is given by:

$$L_{adv}(G, D, Emb) = \underbrace{\mathbb{E}_{real}[\log(s_{real})]}_{L^r_{adv}(D,Emb)} + \underbrace{\mathbb{E}_{fake}\left[\log(1 - s_{fake})\right]}_{L^f_{adv}(G,D,Emb)}, \quad (1)$$

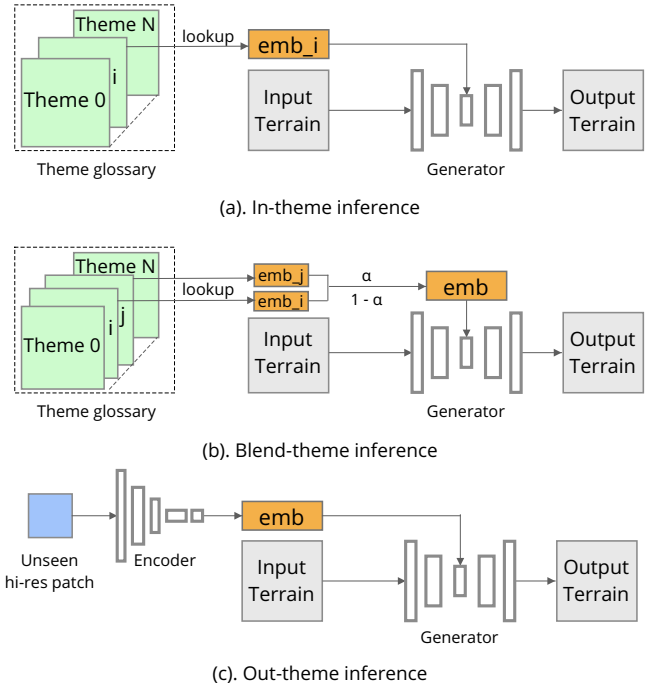

Fig. 3. Three different inference pipelines for GATA, namely in-theme inference, blend-theme inference, and out-theme inference. These pipelines are described in Section 3.4.

which is comprised of two parts, $L^r_{adv}(D, Emb)$ and $L^f_{adv}(G, D, Emb)$, that depend on real and fake samples, respectively. One can view L_{adv} as the negative of the cross entropy between the scores of discriminator D and the true real/fake labels. Thus, it is natural for the discriminator to adversarially increase L_{adv}. On the other hand, the generator wants to decrease L_{adv}. It is important to note both terms in L_{adv} depend on D and Emb, while L^f_{adv} depends on G as well.

To increase the fidelity of the hi-res reconstruction, we also use an ℓ_1 loss between the real and fake hi-res terrains in the overall loss. Such a loss could be defined as L_{ℓ_1}:

$$L_{\ell_1}(G, Emb) = \mathbb{E}_{real}\left[\left\|T_{fake} - T_{real}\right\|_1\right], \tag{2}$$

where T_{fake} is the output of generator that corresponds to the (T_{low}, T_{real}).

Perceptual feature matching losses, which are defined based on the activation vectors in the intermediate layers of a pretrained model, e.g., VGG, are reported to be useful in improving visual results [Dosovitskiy and Brox 2016] and avoid mode collapse [Salimans et al. 2016]. However, since the VGG or other pretrained networks are only trained on normal images, and not on terrains, their activation vectors are not very informative for our application. Thus, here we follow a different approach, which was proposed in [Xie et al. 2018], to define the perceptual feature matching loss based on the activation vectors of the discriminator layers:

$$L_{fm} = \mathbb{E}_{real}\left[\sum_{i=1}^{L}\left\|D^{(i)}(T_{real}) - D^{(i)}(T_{fake})\right\|_1\right], \tag{3}$$

where $D^{(i)}$ is the output of the activation function in the i-th layer of the discriminator and L is the total number of layers in the discriminator.

The final component in the loss that we use in our training enforces consistency between the encoder's output and the theme embedding using an ℓ_2 loss, denoted by L_{con}, and defined as:

$$\begin{aligned} L_{con}(G, E, Emb) = {} & \mathbb{E}_{real}\left[\|E(T_{real}) - emb\|\right] \\ & + \mathbb{E}_{fake}\left[\left\|E(T_{fake}) - emb\right\|\right]. \end{aligned} \tag{4}$$

In Section C, in the supplementary material, we report results of an ablation study on all of the loss terms and demonstrate that all terms are necessary for the final performance of the trained model.

3.3 Training Algorithm

To perform the training we use stochastic gradient descent (SGD). In particular, we use ADAM [Kingma and Ba 2014] which is a popular implementation of SGD with adaptive step-size tuning. As it is common in adversarial training, we partition the parameters into multiple blocks and at each step of the algorithm we only update one block, while keeping the other blocks fixed. We cycle through updating the blocks to finish one iteration. More specifically, the updates are partitioned in three categories as detailed below.

3.3.1 Generator and embedding updates. The generator and embedding updates include the parameters for generator, G, and the embedding, Emb. The update of the generator is simply a step in the negative gradient of the following loss function with respect to the parameters of G:

$$L_G = L_{adv} + \lambda_{\ell_1} L_{\ell_1} + \lambda_{fm} L_{fm} + \lambda_{con} L_{con}, \tag{5}$$

where λ_{adv}, λ_{fm}, λ_{ℓ_1} and λ_{con} are non-negative hyperparameters that balance the influence of the respective losses. For the update of Emb, we use the gradient of another loss, defined as

$$L_{emb} = \lambda_{\ell_1} L_{\ell_1} + \lambda_{fm} L_{fm} + \lambda_{con} L_{con}, \tag{6}$$

where we use the same hyperparameters (weights) to define L_{emb} that were used for the generator update.

We remark that the embedding vector impacts L_{con} through two separate paths, one directly through emb and the other indirectly through $\widetilde{emb}$. We use a stop gradient on the direct path when taking the gradient of L_{con}, otherwise the emb variable would only follow the output of the Encoder. The stop gradient could be viewed as making a copy of emb parameters and calling them $\widetilde{emb}$, which is used when computing the gradient of L_{con} with respect to emb. Note that this copy, $\widetilde{emb}$, would be updated to become equal to emb, as soon as the emb variable is updated.

3.3.2 Discriminator update. The update of the discriminator is straightforward and only includes a step towards negative gradient of $L_D = -L_{adv}$ with respect to the parameters of D.

3.3.3 Encoder update. Finally, the update of the encoder is also straightforward and only includes a step towards negative gradient of L_{con} with respect to the parameters of E.

The overall training for our Generative Adversarial Terrain Amplification (GATA) is summarized in Algorithm 1. We use ADAM with initial learning rate $\alpha = 2e-4$. The toal number of iterations was $T = 1,200,000$. Note that the gradients in all updates are stochastic with a batch size of 1. We also use $\lambda_{\ell_1} = 100$ and $\lambda_{fm} = \lambda_{con} = 1$ for the computation of the losses.

ALGORITHM 1: Train GATA

INPUT: # iterations: T, learning rate: α;
for $t \in \{0, \ldots, T-1\}$ **do**
(Update Generator Block)
$G = G - \alpha \nabla_G L_G$ and $Emb = Emb - \alpha \nabla_{Emb} L_{emb}$
(Update Discriminator)
$D = D - \alpha \nabla_D L_D$
(Update Encoder)
$E = E - \alpha \nabla_E L_{con}$
end for

3.4 Inference: Terrain Amplification

The use of theme embeddings and the learned encoder in GATA opens up the door for several interesting inference time usages. The three main usages that GATA supports are detailed in Fig. 3. The most straightforward way of using it is to choose a theme ID from the theme glossary, which contains all of the multiple themes used in the training phase. We call this *in-theme* inference. For these themes we can directly look up the theme embedding from our learned embeddings. As we will show in Section 4, GATA can generate locally and globally coherent results with these themes as the input.

The second possible use case is to blend different themes from the theme glossary. We call this *blend-theme* inference. In this method, one can choose two or more themes from the glossary, and interpolate between their embeddings and use the interpolated vector as an input for the embedding vector to the generator. As we shall show in Section 4, GATA is capable of generating high-quality results by smoothly interpolating between multiple themes, when changing the interpolation coefficients. This control knob could be useful, for example when the artists would like to move from one theme to another one smoothly across a larger landscape. Note that the interpolated themes have never been seen during training and this use case is only made possible by learning the embedding space. Thus, GATA is learning to generate results that do not necessarily belong to its exemplar. Interestingly, we could also use the same method to extrapolate some of the themes to exaggerate some features such as rockiness to the extreme such that the output can look like a fictional theme. Overall, the ability to tweak the vector of theme embeddings in a meaningful way to tune the output theme provides the opportunity to pass down this flexibility as an extra control knob to the artists to fine tune some of the terrain features.

Finally, there might be a case when an artist would want to amplify a terrain based on a new theme, which they have only one small sample. As we shall see, sparse modeling methods fail at this task because there is not enough data to extract a comprehensive exemplar from a small patch of the input theme. In this case, we can still use our method in the following way. We use the trained encoder on an input theme patch to obtain a theme embedding, using which we can then perform amplification. Our experiments show that this approach, which we call *out-theme* amplification, can generate high-quality results in this case as well; see Section 4.

4 EXPERIMENTAL RESULTS & EVALUATION

In this section we set out to perform a comprehensive evaluation of GATA against the wish-list that we proposed at the beginning of the paper.

4.1 Benchmark Methods

As discussed in the introduction, output coherence and control over stylistic details are of utmost importance to the artists. This rules out procedural and physical simulation-based methods as potential candidates. The example-based methods, on the other hand, provide such control to the artists. Hence, we compare with the state-of-the-art example-based method in this paper. We also compare with neural style transfer methods that have been successful in style transfer for real images but are not tailored to the application at hand.

4.1.1 Sparse Construction Tree (SCT) [Guérin et al. 2016]. SCT is the state of the art in example-based terrain amplification. It is fast as it uses fast dictionary learning sub-routines to fill in the patches of the input with a sparse summation of the patches (atoms) in the exemplar. In terms of flexibility with the output theme, it performs relatively well given enough samples from the target theme are available to create a comprehensive exemplar. SCT has three important hyper-parameters: (1) patch size, (2) offset, and (3) sparsity level. For the terrain amplification using SCT, we tried different hyperparameter settings (see Appendix D) and chose the setting that generates the most visually appealing results. In this setting the patch size is 192, the offset is 96, and the sparsity level is 1. Throughout this section, we only present results for SCT with this best hyperparamter setting. For more results on SCT with other hyperparameter settings see Appendix D.

Throughout our experiments we use SCT with a nominal amplification factor of 16 (ratio between the areas of the input and output in pixels), which is commonly used in practice [Guérin et al. 2016]. As discussed in Section A.2 in the supplementary material, the SCT amplification factor is smaller than the one we have chosen for GATA, i.e., ∼50. Note that we have re-implemented the open-source code of SCT method in Python for the ease of use, and the reported results for this method are based on our implementation. Finally, it is worth noting that [Argudo et al. 2017] improved over SCT by including extra information, such as vegetation and soil moisture. Since our dataset does not contain such information, we are unable to directly compare with this work.

4.1.2 Neural Style Transfer [Gatys et al. 2015, 2016]. Neural style transfer method is designed as a general tool for transferring style from one image to another. We emphasize that style transfer is not

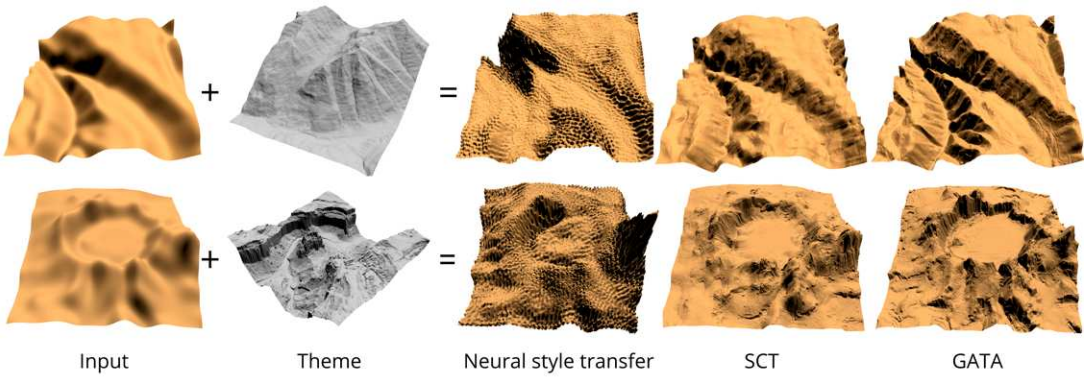

Fig. 4. Comparison between SCT [Guérin et al. 2016], neural style transfer [Gatys et al. 2015, 2016], and GATA [Ours]. Each method receives a lo-res input as well as a target theme exemplar and generates a hi-res output. As can be seen, GATA generates a globally coherent terrain, which locally adheres to the desired theme.

designed for our application but we use it for the sake of completeness.

4.1.3 Generative Adversarial Terrain Amplification (GATA) [Ours]. Throughout the results section, we always use GATA with patch size of 256×256 and the offset of size 128. We acknowledge that changing the patch size for GATA would require redesigning the neural architecture and retraining it, which would come at a huge cost. Hence, we do not treat these parameters as hyperparameters for obtaining visually appealing results given the input or the target theme. Training is only performed once for generating all of the results throughout the paper.

4.2 Local & Global Coherence

To assess the local and global coherence, we perform both visual and numerical comparisons. We also comment on the computation time for training and inference (amplification) in Section 4.2.3.

4.2.1 Visual Comparisons. In Figure 4 we show that our method can consistently transfer the details from the target theme to the lo-res input terrain to generate visually appealing hi-res results. For our method, we use the *in-theme* inference, defined in Section 3.4, which relies on one of the themes from the glossary. We also include the results of amplification for the baseline methods, i.e., SCT and style transfer. For a fair comparison, we provide SCT with an exemplar consisting of the entire 1024×1024 target theme terrain. We also choose the hyperparameter settings (patch size, offset and sparsity level) for SCT that produce the most appealing visual results. Note that numerical experiments quantifying the impact of these hyperparameters are presented in Appendix D. As can be seen in Figure 4, our method generates high quality results, that are locally coherent with the target theme. GATA also preserves the high- and mid-level topographical details of the original input. On the other hand, the results of SCT suffer from local inconsistencies with the target theme. We believe this is due to the inconsistencies between chosen adjacent patches (see Section 4.2.2 for a numerical study on this). Finally, neural style transfer completely fails at producing appealing results, which is not surprising as it is not designed for the terrain amplification task at hand. Thus, in the rest of our experiments we opt out of comparing against neural style transfer.

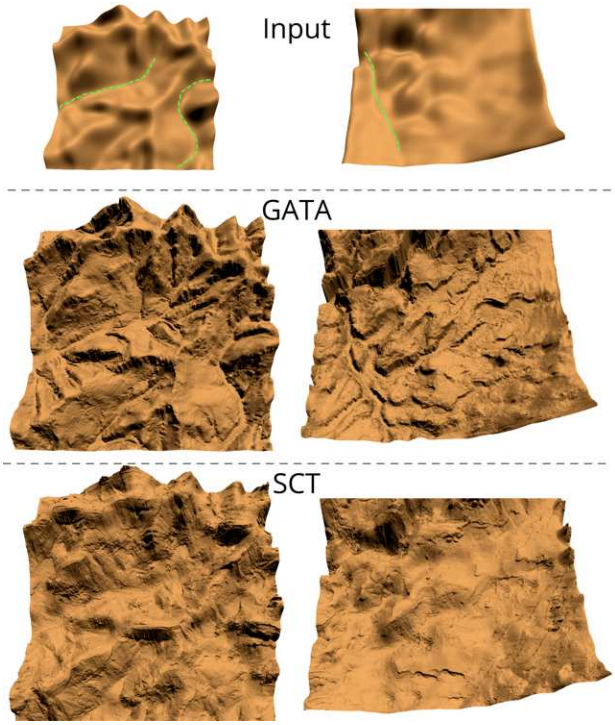

Fig. 5. This figure depicts the concept of global coherence. Since GATA better adheres to the local features of the input when local patches are stitched together, it outperforms SCT at preserving global details of the input. For example, considering the the waterways in the input, marked with green dashed line, GATA outperforms SCT in preserving these drainage patterns. Note that we used the Canyon theme from Figure 7 for this experiment.

It is very difficult for patch-based methods to preserve the global coherence. But, in Figure 5 we highlight that GATA improves over SCT at preserving the global patterns, such as waterways of the

Table 1. Comparison between the amplified terrain and the ground truth using different metrics. GATA outperforms SCT in terms of reconstructing the original hi-res terrain from the theme and the input.

Method	without amplification	SCT	GATA
Avg. ℓ_1 dist.	1012±692	1432±733	**749** ±611
SSIM	0.6890±0.2355	0.5743±0.2102	**0.6891**±0.2391
PSNR	34.84±4.86	31.47±3.70	**37.29**±4.83

input. It is worth noting that these global patterns are significantly larger than the patch size for GATA and preserving those, when transforming the input to output, is a result of better adhering to the local details of the input.

4.2.2 Numerical Comparisons. Numerically measuring concepts, such as coherence of the output is a difficult task. But if we have the ground truth for a hi-res terrain (which is the case in our test dataset as discussed in Section A.1), we can always compare the output of each method, when it is fed with the lo-res terrain and the ground truth theme, against the actual ground truth. In this section, we make these comparisons based on three standard metrics: (1) average ℓ_1 distance: average ℓ_1 distance between two images I_1 and I_2, of the same size S pixels, is defined as $\frac{1}{S}\|I_1 - I_2\|_1$. (2) structural similarity index (SSIM) [Wang et al. 2004], and (3) peak signal-to-noise ratio (PSNR) [Hore and Ziou 2010]. The results of this evaluation are presented in Table 1, where we report the average value for all these metrics over the patches in the test set. In addition to the results for SCT, we also report the numerical similarity of the lo-res input to the target hi-res terrain as a reference point. This would give us a perspective on which method can add details to the lo-res input and make it more consistent with the original hi-res terrain. Note that these results show that GATA can add details to the lo-res patches to obtain hi-res ones that are more consistent with the hi-res target.

The next set of numerical experiments are designed to compare the methods in terms of local coherence which is also a very difficult task. In order to capture this, we examine two overlapping input patches and compare the output of the methods for these two patches in the overlapping area. More specifically, we compute the average ℓ_1 distance between the pixel values in the overlap. In Fig. 6 we depict this average ℓ_1 distance versus the amount of offset between adjacent patches. Note that the amount of overlap between adjacent patches is equal to the size of the patch minus the offset. Note that a large difference between adjacent patches in the overlapping area would result in blurring of the details at the stitching points between the patches during the blending time [Guérin et al. 2016]. This could result in lack of local coherence, as also demonstrated visually in Fig. 4.

As is evident from Fig. 6, GATA achieves a low average ℓ_1 distance regardless of the size of the overlap. On the other hand, the results for SCT are not only worse, but are also more dependant on the amount of overlap, i.e., they deteriorate with larger offset. Although this average ℓ_1 distance can be used as a proxy for local coherence, it does not tell the whole story. While one might conclude from Fig. 6 that using a smaller offset would increase the local coherence and quality, this is indeed not the case. In practice, a smaller offset

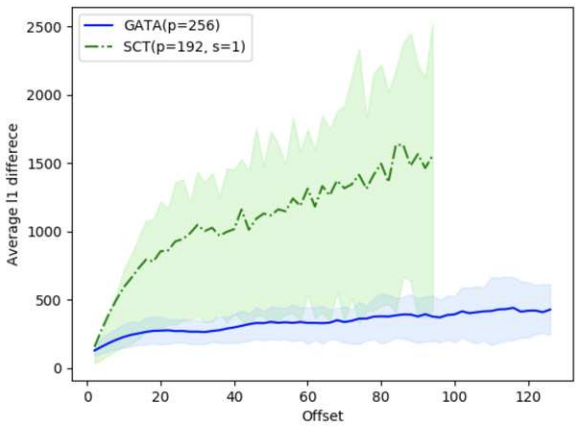

Fig. 6. We plot the average ℓ_1 difference on the overlapping area of the adjacent output patches for each method. For SCT we choose patch size $p = 192$ and sparsity $s = 1$. This is the setting that generates the most appealing results. For more hyperparamter settings see Fig. 18 in Appendix D. Note that the size of the overlapping area decreases linearly with the offset. We sweep all the offsets between 1 pixel and half of the patch size in each case. To give a better perspective in addition to the average, we also plot the 25% and 75% percentile values for each constellation of the methods. As it can be seen, our method is relatively robust and the distribution of the average ℓ_1 differences does not change much with the value of the offset, even though the chosen patch size is large. Note that although extremely small offsets seem to be better in terms of this coherence measure, they are not used in practice due to other considerations (see Section 4.2.2 for a detailed discussion).

would result in more output patches covering each pixel. As a result, more blending would be needed to obtain the height for each pixel. This would in turn lead to more blurring and loss of local coherence as well as the theme details. Moreover, a smaller offset would result in more computation during amplification. As a result of these facts, the offset is normally chosen to be half the patch size to obtain the best visual results carrying the theme details and satisfying local coherence.

4.2.3 Computation Time. As we emphasized in previous sections, GATA can generate high quality results and does not require any hyperparameter tuning. But as we mentioned in our wish-list we want our method to be fast, meaning that it should be capable of amplifying large-scale terrains relatively fast such that artists can use it iteratively to improve their designs. In this section we report the computation time for our method. As opposed to SCT, GATA has a significant computational cost up front at the training time. However, the model only needs to be trained once. We emphasize that all of the results in this paper are produced with a single training of GATA. The training of GATA takes $\sim$ 70 hours using TensorFlow [Abadi et al. 2016] on a computer equipped with an Nvidia Tesla V100 graphics card.

At the inference time, we used GATA to amplify terrains of size 1km^2 (remember that our resolution is 2 meters per pixel as described in Section A.1). As mentioned earlier, our patch size is fixed

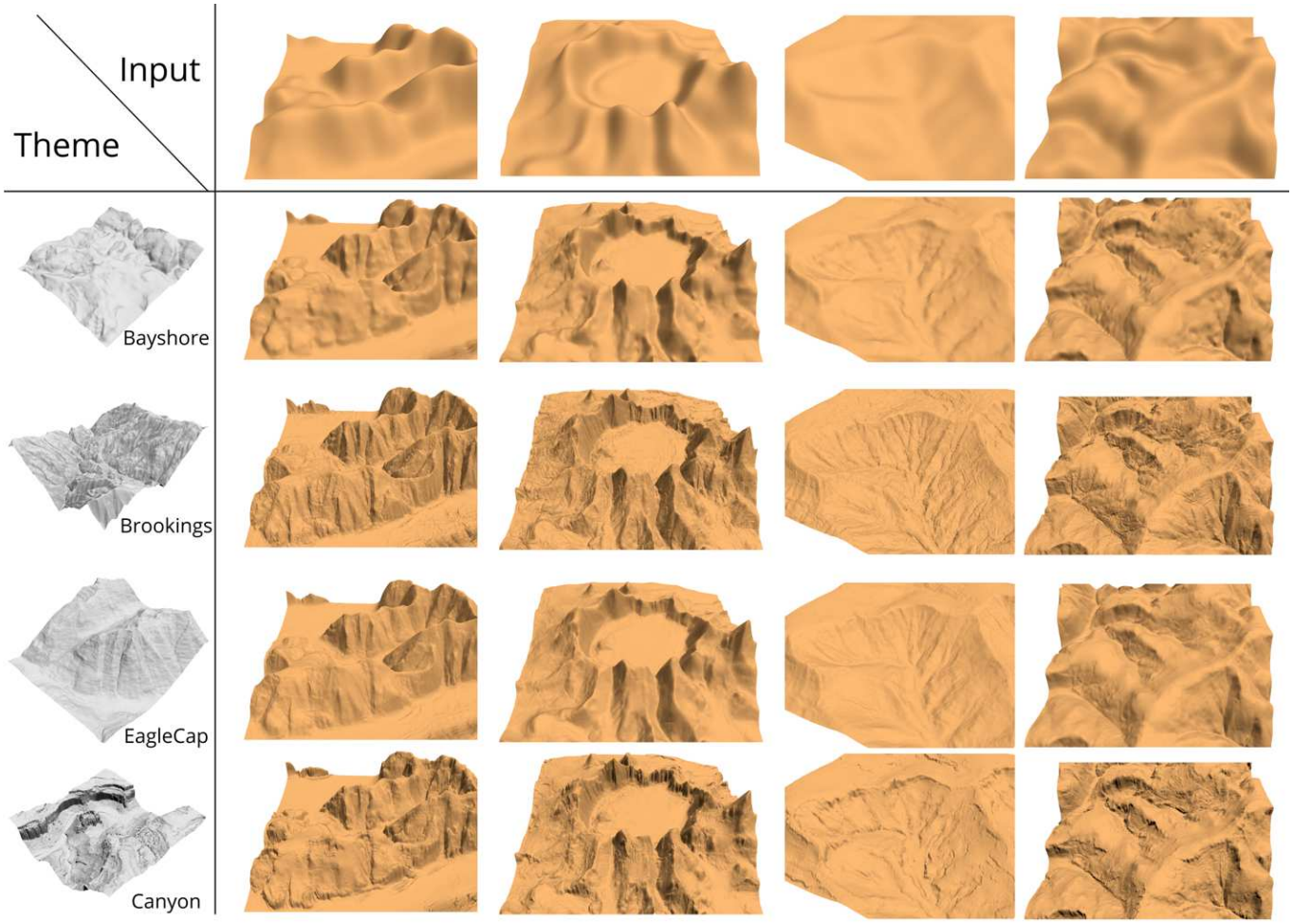

Fig. 7. Results of *in-theme* amplification. GATA receives the los-res terrain and the target theme from the glossary and generates hi-res terrain. We can easily get multiple themes for the output by changing the target theme.

at 256×256 and the offset for our method is 128. We ran our inference on a desktop equipped with Intel Core i7, clocked at 3.7 GHz with 64G RAM and Nvidia Tesla V100 graphics card. Running on the CPU only, GATA, on average takes around $\sim$ 10 sec. to amplify a $1km^2$ terrain. Due to the fact that we use deep networks, these numbers can be improved substantially on a GPU. GATA's average GPU performance is $\sim$ 0.7 sec. for amplifying $1km^2$ terrains. Note that almost all of our inference time is spent on amplifying patches (blending them does not take very long).

GATA's patch amplification time is constant and does not depend on the size of the training set, or the exemplar. Thus, the only factor that dominates the speed of our method is the surface area of the terrain. It is worth noting that the SCT method is much faster compared to GATA when amplifying the same $1km^2$ terrain ($\sim$ 0.03 sec.). This difference in the computation time is the price we have to pay in order to gain global/local coherence, high quality details, and complete control over the theme (for experiments on theme controllability see Section 4.3). Note that despite this difference, our method is still capable of providing an interactive editing tool for the artists to flexibly amplify many areas of their terrain.

4.3 Control over Stylistic Details (Theme)

In the previous section, we showcased the local/global coherence of the hi-res terrains amplified by GATA. In this section, we demonstrate the different ways in which GATA provides control over theme to the artists.

4.3.1 In-Theme Amplification. To further show the flexibility of GATA in generating multiple themes of output from a lo-res input, we apply it on the same input using multiple themes from our glossary (*in-theme*) and depict the results in Figures 7 and 8. As is evident, GATA is capable of coherently amplifying the same input to multiple themes from the glossary while satisfying local and global coherence.

4.3.2 Blend-Theme Amplification. As discussed in Section 3.4, the training of embeddings vectors allows us to manipulate them to

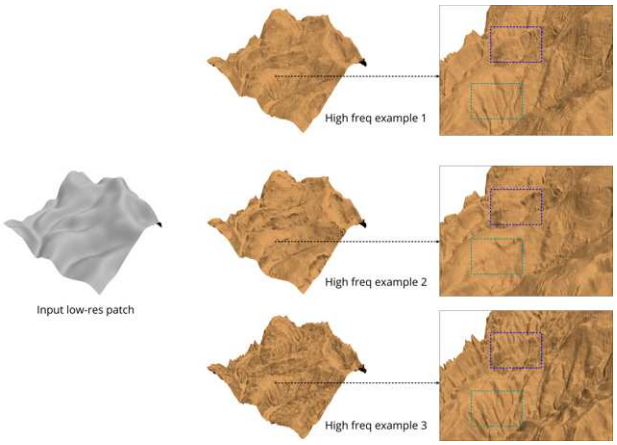

Fig. 8. In this figure we present more evidence that GATA is capable of generating different stylistic details (themes) with the same input. In this figure we pick three very different rocky styles (with high frequency details) from the glossary and apply them on the same input. As it is evident from the figure we get very different results; see the highlighted areas in the figure. For instance, example 2 introduces some cliffs in the hi-res output which are absent in the other two examples.

obtain new themes that are not part of the glossary and the training exemplar. One easy way to do that is to blend multiple themes. For example, in Fig. 9, we show the output results when we feed GATA with the vector that is obtained by interpolating between the theme embedding vectors for two initial themes. It can be seen that the output result smoothly varies from one theme to the other, without losing local/global coherence and the details for $0 \leq \alpha \leq 1$. This feature could potentially provide the artists with easy knobs to tune the final theme of the terrain by changing only one coefficient. We can even see that GATA can also extrapolate between the themes to exaggerate the stylistic details from one of the themes ($\alpha = -1$).

It is worth noting that these blend-themes have never been seen by GATA during the training phase. This is only made possible by the introduction of the theme embeddings that captures semantic stylistic information of the terrain (see Section 4.4). We are not aware of any other terrain amplification method that can provide such a flexibility in generating terrains with multiple themes.

4.3.3 Out-Theme Amplification. Finally, the last experiment is on generating a terrain from a small input hi-res patch (as the exemplar), which has never been observed by our method in the training phase. For this purpose, we use a small 256×256 patch of the heightmap from Death Valley (California) as our target theme. Recall that our entire dataset is from the State of Oregon. We run our encoder on the patch and obtain an embedding vector and then use it to amplify a lo-res terrain. As can be seen in Fig. 10, GATA can still generate visually appealing results that are coherent and transfer the desired target theme even though it has never seen it before.

To put the difficulty of this task into perspective, we also show the results of SCT method, when it is fed with an exemplar with the single 256×256 from Death Valley. As it is obvious from Fig. 10, the result for SCT algorithm is not comparable in terms of coherence with GATA, and the desired theme is completely lost. We emphasize that the *out-theme* amplification itself is a completely novel addition to the existing terrain amplification literature, which is made possible by the introduction of the theme embeddings. In the next section, we provide more details on the learned theme embedding space.

4.4 More on Theme Embedding

One promise of the embeddings is that they can encapsulate the information about abstract concepts, such as stylistic detail, i.e., themes, into a compact mathematical form such as a vector. As we have seen so far these vector representations can convey meaningful relationships. For example, in Fig. 9, we showed that interpolating between the theme embeddings can actually be translated by the model into interpolation between the stylistic features. This means that simple arithmetic calculations on these vectors can have abstract semantic meanings in terms of themes.

In this section, we explore the theme embeddings from a different perspective. We use a simple visualization/dimension reduction technique called Principal Component Analysis (PCA). The hypothesis is that if the theme embeddings live on a high-dimensional manifold that is simple, i.e., close to a hyper-plane, then linear techniques such as PCA should be capable of extracting meaningful information from these embeddings. In other words, we would be able to see meaningful semantic patterns along the main principal components of the theme embedding data. Moreover, terrains with similar theme should be close to each other in the embedding domain.

To test this hypothesis, we plot all the theme embeddings, projected on the first two principal components of the the embedding matrix, *Emb*, in Fig. 11. We visualize some of the themes that lie in different areas of the representation space. It can be seen that moving along the first principal component, i.e., x-axis, from left to right would represent going from having more land to having more water in the terrain. In other words, the extreme points on the right are comprised of terrain pieces that are mostly water, while the extreme ones on the left are mostly land and do not have any water. The ones in the middle have a mixture of water and land.

The second principal component, y-axis, seems to represent the rockiness. In one extreme, i.e., at the bottom, the terrains seem to be mostly smooth, while the ones at the top tend to be very rocky. Similar to the first principal component, the ones in the middle have a mixture of rockiness and smoothness. Although this is not a comprehensive study, but it re-affirms our hypothesis that our model learns embeddings that encapsulate useful semantic information from the theme and are simple to manipulate in meaningful ways using simple arithmetic operations. We believe that a more careful exploration of this phenomenon would be an interesting avenue for future research.

5 CONCLUDING REMARKS & FUTURE WORK

Data-driven techniques are gaining steam in multiple applications in computer graphics, including terrain generation and amplification. In this paper, we proposed a novel multi-theme generative adversarial terrain amplification method, called GATA. By leveraging the

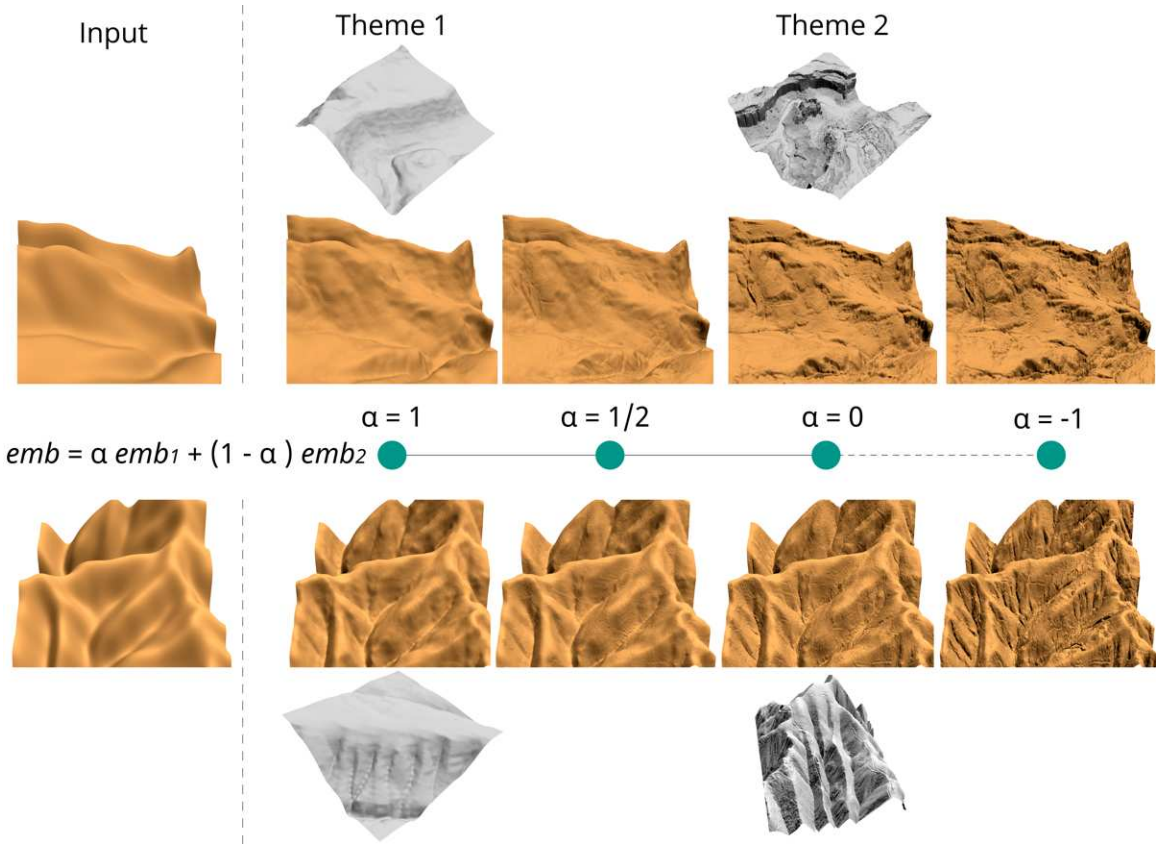

Fig. 9. *Blend-theme* inference: We blend the theme embeddings by linearly combining their corresponding vectors. We first do it by interpolating between the two vectors. By changing the linear interpolation coefficient our amplified terrain smoothly transitions from one theme to the other. We can even take this further and extrapolate between two themes. For example, when we extrapolate from *Theme 1* outward from *Theme 2*, we can see that the properties of *Theme 2* that are not present in *Theme 1*, e.g., rockiness, get even more exaggerated in the final output.

entire training theme glossary in offline training, GATA is capable of generating high-resolution terrains that take both quality and control over theme to a new level. GATA is sufficiently fast so that it could provide a flexible interactive tool for the artists to amplify parts of their terrain in applications such as designing open-world games.

GATA owes its quality and controllability to the novel use of theme embeddings. The theme embeddings enable GATA to generate terrains from a blending of multiple themes or themes that do not even exist in the glossary used in training. As we showed in Fig. 1, we can also generate large-scale terrains that smoothly transition from one theme to another. This adds a new dimension to the controllability in terrain amplification.

We believe that we have only scratched the surface in understanding the capabilities of theme embeddings. Future work would naturally consider a better apprehension of this tool. Moreover, while GATA is sufficiently fast for interactive terrain amplification, it remains an open problem to scale this method up for oneshot amplification of very large terrains. We believe that combining GATA with a multi-resolution approach would be a promising direction for achieving such scalability. Finally, the scope of this work is limited in the sense that we only considered heightmaps while general terrains cannot necessarily be represented solely by heightmaps. It would be interesting to see how different components of this work may generalize to other terrain representations, including general 3D models.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their helpful suggestions and constructive comments. We would also like to thank all the members of EADP Data & AI team at Electronic Arts for their continuous support, especially Navid Aghdaie, Tushar Bansal, Harold Chaput, Sundeep Narravula, Reza Pourabolghasem, and Mohsen Sardari. We would also like to acknowledge the valuable feedback on terrain quality and artifacts from Mathieu Guindon, and Cody Ritchie.

REFERENCES

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 265–283.

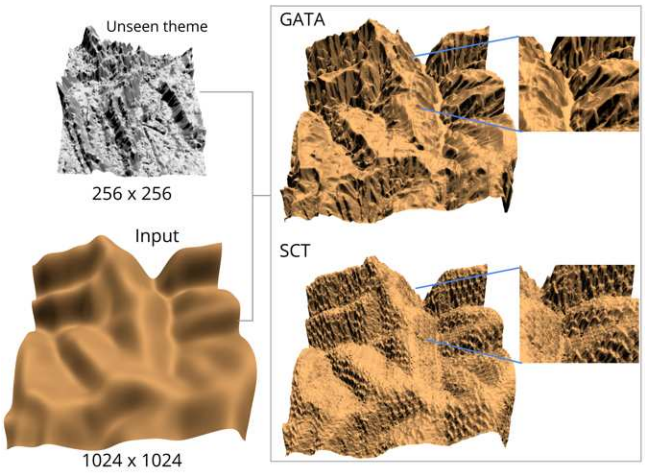

Fig. 10. *Out-Theme* amplification: The top-right terrain shows the result of applying GATA on an input lo-res terrain with the theme embedding that is generated by the encoder from a patch of an un-seen terrain. The lower-right terrain is the results of running SCT on the input terrain with the exemplar coming from the small patch of the hi-res target. Due to the small size of the exemplar we had to reduce the patch size and offset for the SCT to 64 and 32 respectively.

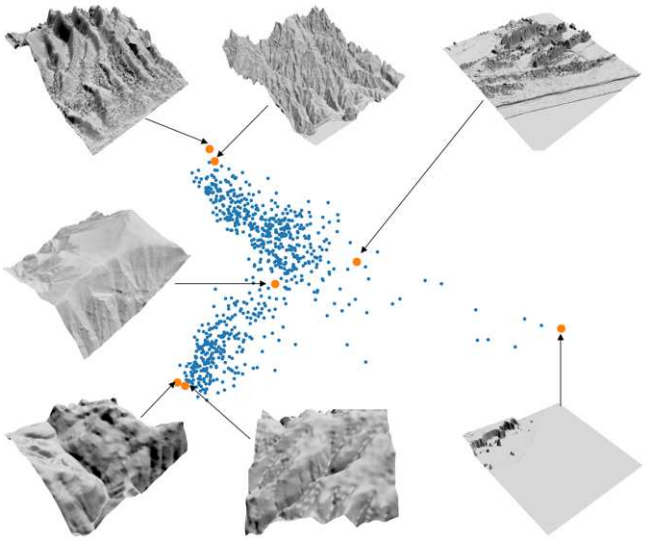

Fig. 11. Each data point in the scatter plot represents one theme in the glossary. The embeddings vectors are first centered and scaled before computing the PCA. Then they are projected onto the first two principal components of the resulting embedding matrix. The x and y axes represent the value of each embedding vector projected on first and second principal components respectively.

Oscar Argudo, Carlos Andujar, Antonio Chica, Eric Guérin, Julie Digne, Adrien Peytavie, and Eric Galin. 2017. Coherent multi-layer landscape synthesis. *The Visual Computer* 33, 6-8 (2017), 1005–1015.

Oscar Argudo, Antoni Chica, and Carlos Andujar. 2018. Terrain Super-resolution through Aerial Imagery and Fully Convolutional Networks. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 101–110.

Bedrich Benes and Rafael Forsbach. 2001. Layered data representation for visual simulation of terrain erosion. In *sccg*. IEEE, 0080.

Bedřich Beneš, Václav Těšínskỳ, Jan Hornyš, and Sanjiv K Bhatia. 2006. Hydraulic erosion. *Computer Animation and Virtual Worlds* 17, 2 (2006), 99–108.

BiteTheBytesUG. 2019. World Creator. https://www.world-creator.com/

Piotr Bojanowski, Armand Joulin, David Lopez-Pas, and Arthur Szlam. 2018. Optimizing the Latent Space of Generative Networks. In *International Conference on Machine Learning*. 599–608.

Kaidi Cao, Jing Liao, and Lu Yuan. 2018. CariGANs: unpaired photo-to-caricature translation. In *SIGGRAPH Asia 2018 Technical Papers*. ACM, 244.

Norishige Chiba, Kazunobu Muraoka, and Kunihiko Fujita. 1998. An erosion model based on velocity fields for the visual simulation of mountain scenery. *The Journal of Visualization and Computer Animation* 9, 4 (1998), 185–194.

Guillaume Cordonnier, Jean Braun, Marie-Paule Cani, Bedrich Benes, Eric Galin, Adrien Peytavie, and Eric Guérin. 2016. Large scale terrain generation from tectonic uplift and fluvial erosion. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 165–175.

Paul Cotton. 2019. How does Battlefield V's Firestorm battle royale map compare to Fortnite and Apex Legends? https://www.dexerto.com/battlefield/battlefield-firestorm-map-size-472053

Daz3D. 2019. Bryce 7 Pro. https://www.daz3d.com

DigitalElement. 2019. WorldBuilder Pro 4. https://www.digi-element.com/worldbuilder-pro/

Carl Doersch. 2016. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908* (2016).

Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. 2016. Adversarial feature learning. *arXiv preprint arXiv:1605.09782* (2016).

Alexey Dosovitskiy and Thomas Brox. 2016. Generating images with perceptual similarity metrics based on deep networks. In *Advances in neural information processing systems*. 658–666.

Vincent Dumoulin, Ethan Perez, Nathan Schucher, Florian Strub, Harm de Vries, Aaron Courville, and Yoshua Bengio. 2018. Feature-wise transformations. *Distill* 3, 7 (2018), e11.

E-onSoftware. 2019. Creator Solution. https://info.e-onsoftware.com/creator-solution

David S Ebert and F Kenton Musgrave. 2003. *Texturing & modeling: a procedural approach*. Morgan Kaufmann.

Arnaud Emilien, Ulysse Vimont, Marie-Paule Cani, Pierre Poulin, and Bedrich Benes. 2015. Worldbrush: Interactive example-based synthesis of procedural virtual worlds. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 106.

Anastasia Feygina, Dmitry I Ignatov, and Ilya Makarov. 2018. Realistic post-processing of rendered 3D scenes. In *ACM SIGGRAPH 2018 Posters*. ACM, 42.

William T Freeman, Thouis R Jones, and Egon C Pasztor. 2002. Example-based super-resolution. *IEEE Computer graphics and Applications* 22, 2 (2002), 56–65.

Sakiko Fujieda, Yuki Morimoto, and Kazuo Ohzeki. 2017. An image generation system of delicious food in a manga style. In *SIGGRAPH Asia 2017 Posters, SA 2017*. Association for Computing Machinery, Inc, 28.

James Gain, Patrick Marais, and Wolfgang Straßer. 2009. Terrain sketching. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*. ACM, 31–38.

James Gain, Bruce Merry, and Patrick Marais. 2015. Parallel, realistic and controllable terrain synthesis. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 105–116.

Eric Galin, Eric Guérin, Adrien Peytavie, Guillaume Cordonnier, Marie-Paule Cani, Bedrich Benes, and James Gain. 2019. A Review of Digital Terrain Modeling. (2019).

Lin Gao, Jie Yang, Yi-Ling Qiao, Yu-Kun Lai, Paul L Rosin, Weiwei Xu, and Shihong Xia. 2018. Automatic unpaired shape deformation transfer. In *SIGGRAPH Asia 2018 Technical Papers*. ACM, 237.

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. 2015. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576* (2015).

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. 2016. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2414–2423.

Jean-David Génevaux, Eric Galin, Adrien Peytavie, Eric Guérin, Cyril Briquet, François Grosbellet, and Bedrich Benes. 2015. Terrain modelling from feature primitives. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 198–210.

Jiahao Geng, Tianjia Shao, Youyi Zheng, Yanlin Weng, and Kun Zhou. 2018. Warp-guided GANs for single-photo facial animation. In *SIGGRAPH Asia 2018 Technical Papers*. ACM, 231.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

Eric Guérin, Julie Digne, Eric Galin, and Adrien Peytavie. 2016. Sparse representation of terrains for procedural modeling. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 177–187.

Eric Guérin, Julie Digne, Eric Galin, Adrien Peytavie, Christian Wolf, Bedrich Benes, and Benoît Martinez. 2017. Interactive example-based terrain authoring with conditional generative adversarial networks. *ACM Transactions on Graphics (TOG)* 36, 6 (2017),

228.
Daniel Holden, Jun Saito, and Taku Komura. 2016. A deep learning framework for character motion synthesis and editing. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 138.
Alain Hore and Djemel Ziou. 2010. Image quality metrics: PSNR vs. SSIM. In *2010 20th International Conference on Pattern Recognition*. IEEE, 2366–2369.
Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1125–1134.
Tero Karras, Samuli Laine, and Timo Aila. 2018. A style-based generator architecture for generative adversarial networks. *arXiv preprint arXiv:1812.04948* (2018).
Tom Kelly, Paul Guerrero, Anthony Steed, Peter Wonka, and Niloy J Mitra. 2018. FrankenGAN: guided detail synthesis for building mass models using style-synchonized GANs. In *SIGGRAPH Asia 2018 Technical Papers*. ACM, 216.
Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).
Ares Lagae, Sylvain Lefebvre, George Drettakis, and Philip Dutré. 2009. Procedural noise using sparse Gabor convolution. *ACM Transactions on Graphics (TOG)* 28, 3 (2009), 54.
Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. 2017. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4681–4690.
Jerry Liu, Fisher Yu, and Thomas Funkhouser. 2017. Interactive 3D modeling with a generative adversarial network. In *2017 International Conference on 3D Vision (3DV)*. IEEE, 126–134.
Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep Learning Face Attributes in the Wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.
Benoit B Mandelbrot. 1982. *The fractal geometry of nature*. Vol. 1. WH freeman New York.
Benoit B Mandelbrot and John W Van Ness. 1968. Fractional Brownian motions, fractional noises and applications. *SIAM review* 10, 4 (1968), 422–437.
David M Mark and Peter B Aronson. 1984. Scale-dependent fractal dimensions of topographic surfaces: an empirical investigation, with applications in geomorphology and computer mapping. *Journal of the International Association for Mathematical Geology* 16, 7 (1984), 671–683.
Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).
Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. 2018. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957* (2018).
Takeru Miyato and Masanori Koyama. 2018. cGANs with projection discriminator. *arXiv preprint arXiv:1802.05637* (2018).
F Kenton Musgrave, Craig E Kolb, and Robert S Mace. 1989. The synthesis and rendering of eroded fractal terrains. In *ACM Siggraph Computer Graphics*, Vol. 23. ACM, 41–50.
Koki Nagano, Jaewoo Seo, Jun Xing, Lingyu Wei, Zimo Li, Shunsuke Saito, Aviral Agarwal, Jens Fursund, Hao Li, Richard Roberts, et al. 2018. paGAN: real-time avatars using dynamic textures. In *SIGGRAPH Asia 2018 Technical Papers*. ACM, 258.
Ryota Natsume, Tatsuya Yatagawa, and Shigeo Morishima. 2018. RSGAN: face swapping and editing using face and hair representation in latent spaces. In *ACM SIGGRAPH 2018 Posters*. ACM, 69.
Kamyar Nazeri, Eric Ng, Tony Joseph, Faisal Qureshi, and Mehran Ebrahimi. 2019. EdgeConnect: Generative Image Inpainting with Adversarial Edge Learning. *arXiv preprint arXiv:1901.00212* (2019).
NVIDIA. 2019. NVIDIA RTX Server Lineup Expands to Meet Growing Demand for Data Center and Cloud Graphics Applications. https://blogs.nvidia.com/blog/2019/03/18/rtx-server-lineup-expands/
Augustus Odena, Christopher Olah, and Jonathon Shlens. 2017. Conditional image synthesis with auxiliary classifier gans. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2642–2651.
Guim Perarnau, Joost Van De Weijer, Bogdan Raducanu, and Jose M Álvarez. 2016. Invertible conditional gans for image editing. *arXiv preprint arXiv:1611.06355* (2016).
Ken Perlin. 1985. An image synthesizer. *ACM Siggraph Computer Graphics* 19, 3 (1985), 287–296.
PlanetsideSoftwareLLC. 2019. Terragen. https://planetside.co.uk/
Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* (2015).
Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.
Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. In *Advances in neural information processing systems*. 2234–2242.
Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. 2018. It takes (only) two: Adversarial generator-encoder networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
Hao Wang, Nadav Schor, Ruizhen Hu, Haibin Huang, Daniel Cohen-Or, and Hui Huang. 2018. Global-to-local generative model for 3d shapes. In *SIGGRAPH Asia 2018 Technical Papers*. ACM, 214.
Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli, et al. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.
Lingyu Wei, Liwen Hu, Vladimir Kim, Ersin Yumer, and Hao Li. 2018. Real-Time Hair Rendering using Sequential Adversarial Networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 99–116.
Li-Yi Wei, Sylvain Lefebvre, Vivek Kwatra, and Greg Turk. 2009. State of the art in example-based texture synthesis. In *Eurographics 2009, State of the Art Report, EG-STAR*. Eurographics Association, 93–117.
Christopher Wojtan, Mark Carlson, Peter J Mucha, and Greg Turk. 2007. Animating Corrosion and Erosion.. In *NPH*. Citeseer, 15–22.
Steven Worley. 1996. A cellular texture basis function. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 291–294.
You Xie, Erik Franz, Mengyu Chu, and Nils Thuerey. 2018. tempogan: A temporally coherent, volumetric gan for super-resolution fluid flow. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 95.
Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. 2018. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318* (2018).
Howard Zhou, Jie Sun, Greg Turk, and James M Rehg. 2007. Terrain synthesis from digital elevation models. *IEEE transactions on visualization and computer graphics* 13, 4 (2007), 834–848.
Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017a. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*. 2223–2232.
Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017b. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*. 2223–2232.

A ARCHITECTURAL DETAILS

A.1 Pre-processing and Training Data

We prepared our training data using LIDAR images from the State of Oregon Department of Geology and Mineral Industries (DOGAMI).[1] The landscape in the State of Oregon is relatively diverse spanning mountains, hills, lakes, rivers, beaches, and deserts, which makes a rich dataset for creating different themes. Each source LIDAR image in the dataset is a heightfield with a resolution of one or two meters per pixel. To unify the dataset, we down-sampled them to 2m per pixel. The overall area spanned by the dataset is ~3,000km^2.

To extract different terrain themes to feed into our model, we assume that nearby locations have similar stylistic details. With this assumption, the LIDAR images are chopped up into sections of size 1024×1024 pixels (roughly 4km^2), resulting in 786 distinct themes. Note that some of the themes might be close to each other for neighboring patches but we do not consider these distinctions. As we will see in the experimental results (Section 4), the algorithm will find embeddings that respect these similarities reaffirming the effectiveness of embedding themes. Before we can use the terrains for our training, we need to normalize the inputs. To do so, we shift and re-scale the value of each pixel in the 1024×1024 pieces such that the maximum/minimum height of each patch generated from it is mapped into $+/-0.9$, respectively. During inference, we use the same method to scale the input lo-res terrain into the same range. We do not use the entire range of $[-1, 1]$ for our input because doing so will increase the chance of producing a clipped outputs after amplifying the terrain. After normalization, we further chop down each 1024×1024 section to patches of 256×256 with an offset of 128 pixels resulting in a total of 49 sample patches per each theme. We consider all of these 49 patches to be samples from the same theme. Table 2 summarizes the details of the dataset that we are using to train our model.

Number of themes	786
Samples/theme	49
Total samples	38,514
Sample size	256×256

Table 2. Details of the training dataset

In addition to training the model, we need a test dataset. We create the test dataset by leaving off some of the neighboring patches, adjacent to our theme regions. We use these adjacent patches that have no overlap with the training patches as our testing set, which is used for the experiments in Section 4. There are 8 patches per theme, 5940 in total, in this testing set.

In order to train the generator, we need a dataset containing tuples of lo- and hi-res terrains. To obtain such tuples for training, we applied 2D Butterworth lowpass filters of varying orders on the hi-res terrain to filter out the details in the height field map, producing a "blurred" version. We used these pairs as the dataset for training, i.e., each pair is composed of two 256×256 images.

[1] https://www.oregongeology.org/lidar/

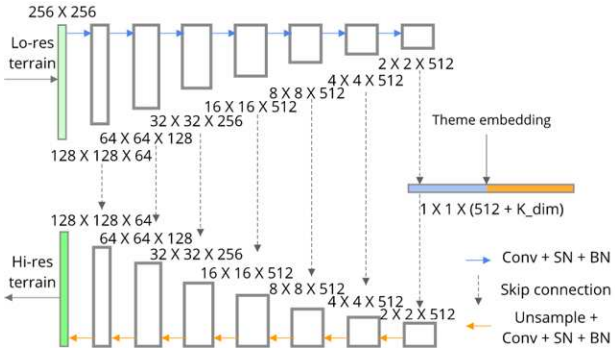

Fig. 12. Architecture of the generator. There are 14 convolutional layers each with non-linearity, batch normalization (BN), and SN. All of the non-linear elements are leaky ReLUs except the last layer, which has tanh as its non-linearity.

A.2 Amplification Factor

In this section, we discuss the amplification factor of GATA. Note that we are using the same patch size for both the input and the output to enable interactivity at the deployment. On the other hand, this makes quantifying the amplification factor for our method a delicate task. As a rough estimate of this amplification factor, we look at the cut-off frequency of the low-pass filter used in training (and also inference). Based on this, we estimate our area amplification factor to be ~50.

A.3 Generator

The generative model that we are training for our purpose is a conditional GAN (cGAN) [Isola et al. 2017; Mirza and Osindero 2014]. Our generator architecture is inspired by the U-net [Ronneberger et al. 2015], which is also used for achieving super-resolution [Ledig et al. 2017] and image to image translation [Isola et al. 2017]. The generator architecture is depicted in Fig. 12. U-net is comprised of two sub-networks, the contracting sub-network (upper side in the figure) and the expansive sub-network (lower side). While U-net looks similar to the popular encoder-decoder architectures, the main difference is the use of skip connections between corresponding layers in the contracting and expansive sub-networks [Ronneberger et al. 2015].

The contracting sub-network takes a lo-res terrain as input and distills a compressed representation of it in a high-dimensional vector. While there are many ways to condition the output of the network on the latent representations, i.e. theme embedding [Dumoulin et al. 2018], we choose to introduce this representation at the bottom of the contracting network. Thus, our proposed architecture differs from the other popular uses of the U-net architecture in that this compressed high-dimensional vector is also concatenated with the theme embedding vector and fed to the expansive sub-network. To stabilize training, we also apply Spectral Normalization (SN) [Miyato et al. 2018] on all of the convolutional layers, i.e., we scale down the weight matrix at each layer by its largest singular value. Although SN was originally proposed just for the discriminator network, recent works by [Nazeri et al. 2019; Zhang et al. 2018]

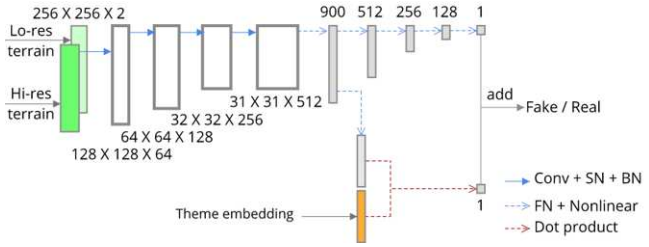

Fig. 13. Architecture of the discriminator. There are four convolutional layers with BN, SN and non-linearity. These layers are followed by four fully connected linear layers. The output of the convolutional layers is also projected linearly to the same dimension as theme embedding, where the inner product between these two vectors is computed and used in the last layer of the discriminator. All the non-linearities are leaky ReLUs except the last layer which uses a sigmoid.

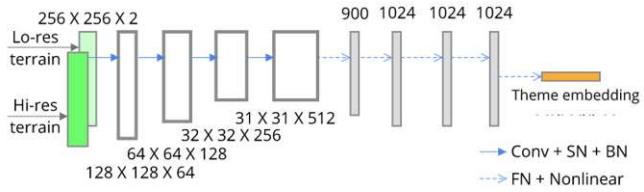

Fig. 14. Neural architecture of the encoder. Before feeding this neural architecture, we first apply a Butterworth filter on the hi-res input to obtain a lo-res version. We concatenate the lo- and hi-res terrains and feed them to this neural architecture. There are four convolutional layers with BN, SN and non-linearity. These layers are followed by four fully connected layers with non-linearity. The non-linearities are all leaky ReLUs.

have shown that the use of SN on the generator network leads to more stable training and improved results. This is also consistent with our observations in experiments.

A.4 Theme Embedding

Conditional GANs (cGANs) have achieved promising results on class-conditional image generation [Odena et al. 2017]. Recently, [Miyato and Koyama 2018] showed that class information, used as a one-hot encoded vector, along with projection discriminator can substantially improve the performance of cGANs. Moreover, the models that are trained in this way are capable of interpolating smoothly between classes [Miyato and Koyama 2018] by interpolating between the one-hot encoded representations. We use the same idea and discriminator architecture to condition the GAN training on the theme (analogous to class information in [Miyato and Koyama 2018]). But we extend this framework to the case where the themes are represented by trainable vectors and not one-hot encoded ones. We refer to this latent representation of the themes as *theme embedding*. In Section 4 we show that, similar to [Miyato and Koyama 2018], we can also interpolate between themes by interpolating the theme embedding vectors.

In Section A.1, we explained how we group different terrain patches into separate themes based on their proximity. Here we utilize this label directly to design our theme embedding. Similar to what is done in [Miyato and Koyama 2018], we build a look-up table to map each terrain group to an embedding vector *emb* and together they build a matrix *Emb*. We consider this matrix as part of the variables that are learned during training. We explain how to jointly train it with the other components in Section 3.3.

Finally, it is worth mentioning that the idea of using an embedding to represent semantic information is not new in the area of computer graphics. For example, [Holden et al. 2016] use an autoencoder architecture to represent natural motions as a high-dimensional manifold and use this representation for synthesizing and editing character motions in animations.

A.5 Discriminator

The discriminator network is designed to input a tuple of lo- and hi-res terrains as well as the corresponding theme embedding vector. It then produces a soft score in $[0, 1]$ that indicates the probability that the hi-res terrain is a fake amplification of the lo-res one. Hence, all the true hi-res terrains should get low scores while an unrealistic fake hi-res terrain should get a high score.

To feed the discriminator with the theme embedding, we use a projection architecture which was proposed by [Miyato and Koyama 2018] and proved to be effective on cGANs. Fig. 13 shows the architecture of our projection discriminator. We first concatenate the lo- and hi-res terrains as an input tuple and use multiple convolutional layers followed by down-sampling to obtain a vectorized representation of the tuple. We use SN on all these convolutional layers to stabilize the training. We then project the vectorized representation to a 1024-dimensional vector (which is the chosen size of the theme embedding vector) and take the inner product between this vector and the theme embedding vector. We use this inner product as the bias in the last layer, which is a soft-max. Note that we use fully connected layers with leaky ReLU non-linearity to map the vectorized representation into a scalar, which is used in the last layer.

A.6 Encoder

The encoder is a network that aims to recover the theme embedding from a hi-res terrain. The idea of adding the encoder to our overall training is inspired by the cycle GAN [Zhu et al. 2017b]. We want the information on the theme embedding to be extractable from the hi-res terrain. Therefore, we add a consistency term to our loss to ensure that the theme embedding of the corresponding hi-res terrain is close to the encoder's output (see Section 3.2). Note that at the inference time, one can also use this encoder to obtain the theme embedding for new terrains that are not part of the theme glossary used in the training process. This will enable generating outputs from new themes that have not been used in training.

The overall neural architecture of the encoder borrows from that of the discriminator, and is depicted in Fig. 14. Note that the input to the encoder is only the hi-res terrain. However, as we are interested in the embedding the semantic stylistic information, i.e., theme, that lies in the difference between the lo-res and hi-res version of the same heightmap, we first apply a lowpass filter on the hi-res input to obtain a lo-res version. Then, we feed the concatenation of the lo-

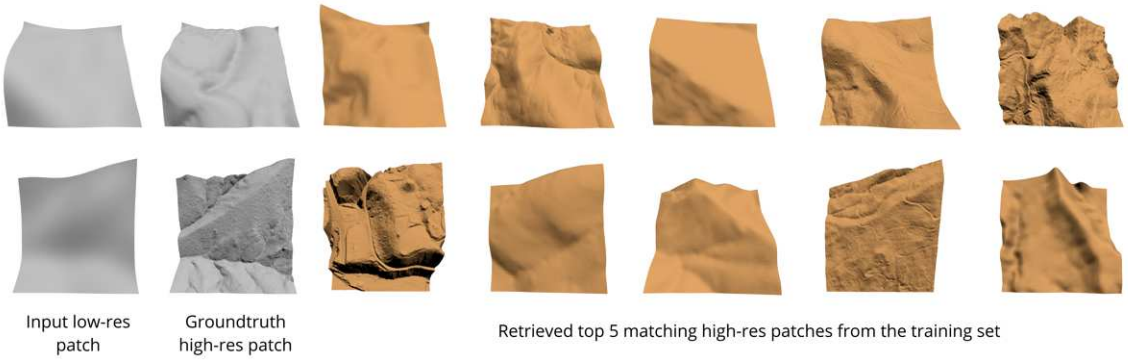

Fig. 15. From left to right: the lo-res input test patch, its ground truth hi-res version and the top-5 retrieved hi-res patches from the training set. We can see that the extracted hi-res patches do not present significant resemblance to the query input or its hi-res version.

Table 3. Comparison between the amplified terrain and the ground truth using different metrics. The comparison is performed on the test dataset. We present the results for different settings of the SCT hyperparameters. Note that SCT cannot outperform the lo-res, on average, in generating the results that are similar to original hi-res in terms of these metrics.

Method	SCT (patch size, offset, sparsity)						
	(64, 32, 1)	(128, 64, 1)	(192, 24, 1)	(192, 48, 1)	(192, 96, 4)	(192, 96, 2)	(192, 96, 1)
Avg. ℓ_1 distance	1028±677	1137±683	1210±710	1238±711	1150±696	1268±708	1432±733
SSIM	0.6692±0.2322	0.6315±0.2261	0.6529±0.2313	0.6384 ±0.2294	0.6201±0.2211	0.5972±0.2161	0.5743 ±0.2102
PSNR	34.63±4.68	33.63±4.24	33.15±4.20	32.93±4.11	33.57±4.22	32.61±3.94	31.47±3.70

and hi-res terrains to the neural architecture of the encoder. This architecture passes the input through a few convolutional layers followed by fully connected layers with leaky ReLU non-linearity. It finally generates a 1024-dimensional embedding vector. Please refer to our open-source code repository for all implementation details (https://github.com/electronicarts/siggraph-asia-2019-gata).

B VERIFYING TEST PATCHES AGAINST TRAINING DATA

To make sure that our training procedure does not overfit to the training, we perform a simple test in which we take random patches from the test set and compare their lo-res version to the closest 5 patches in the entire training set. As can be seen from Fig. 15, the patches that are retrieved with this method have little resemblance to the input lo-res patch or its original hi-res version.

C ABLATION STUDY

In this section we perform an ablation study on different loss terms in our training. In other words, we omit each one of the four terms, i.e., L_{adv}, L_{ℓ_1}, L_{fm} and L_{con}, and visually compare the output when training the model without the specific loss against the model that includes all of the training losses. Note that all models are trained for the same amount of time. Fig. 16 depicts the results of this comparison with the same input and the Canyon Theme (from Fig. 7). As can be seen, all the loss terms are necessary to produce the desired results that capture the stylistic details of the input theme.

D THE IMPACT OF HYPERPARAMETERS ON SCT

As we have mentioned earlier, SCT has three important hyperparameters: (1) patch size, (2) offset, and (3) sparsity level. In order to make sure our comparison is fair we have done a comprehensive sweep over different values of these hyperparameters for SCT. In this section we provide some visual and numerical results regarding different settings of these hyperparameters. Patch size, p, defines the size of the patches in the exemplar that are supposed to be matched to the input. In our experiments we try patch sizes $\{64, 128, 192\}$.

Offset defines how much overlap is between the input patches. The lower the offset, the more overlap there is between the adjacent patches. The offset has to be chosen with regards to the patch size [Guérin et al. 2016]. We used offsets equal to half and quarter of the patch size for our experiments.

Sparsity level, s, is the number of target patches (atoms) that are summed up to approximate the input patch. In our experiments we used sparsity levels $\{1, 2, 4\}$. We noticed that the sparsity levels greater than 4 tend to result in blurry images that resemble the input better, but do not incorporate the stylistic details.

Fig. 17 shows the visual performance of SCT with different settings of these hyperparameters. In our experiments we have found that patch size 192 with offset equal to 96 tends to generate the most appealing results. The sparsity levels 1, 2 and 4 generate very similar results that are very difficult to visually distinguish in terms of quality in most cases. Thus, SCT is not very sensitive to the choice of sparsity level. In our main experiments we chose sparsity level to be 1 as it performs robustly when applied on a wide range of

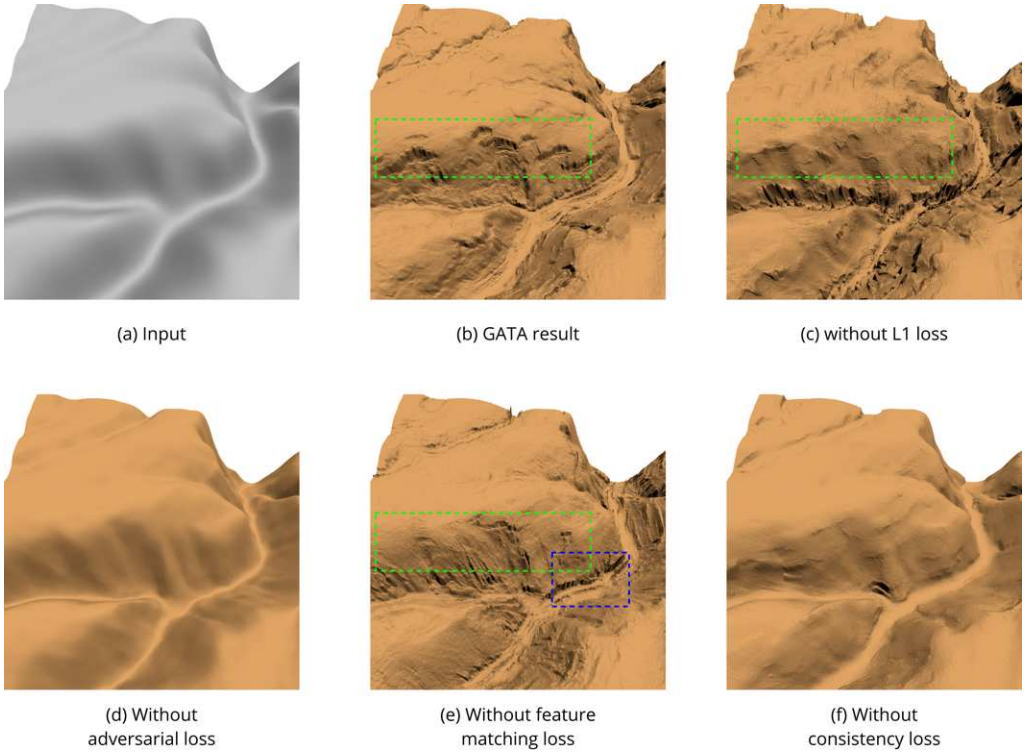

Fig. 16. This figure shows the results of the learned model with different loss configurations after the same number of iterations. Firstly, we see that without the adversarial loss the model fails to generate any stylistic details, which is expected. Secondly, without consistency loss, there is no mechanism to enforce the embedded space to capture the semantic information about the terrain themes. Hence, the result of the model in this case also lacks stylistic details that resemble the input theme. Thirdly, without the feature matching loss, although the model learns to generate more details, the trained model still fails to capture all stylistic details from the input theme. For example, no cliffs are generated in the output while the theme patch is chosen to be the Canyon. Finally, without the L_1 loss the model not only lacks cliffs in the output, the trained model generates some sharp noisy regions in the output. As can be seen, keeping all the loss terms gives the most appealing visual results with more stylistic details.

themes. It is worth noting that higher sparsity levels tend to generate smoother outputs on some themes which could be less visually appealing.

As an extension to Fig. 6, we provide the average ℓ_1 difference between overlapping areas of the output patches that are selected for different algorithms in Fig. 18 to include more hyperparameter settings for SCT. Although it seems that smaller patch sizes can get better results in terms of this metric, based on Fig. 18, the results of those patch sizes are not visually appealing (see Fig. 17). Such small patch sizes would fail to carry over stylistic features that are larger than the patch size. Hence, the output resembles the lo-res input without much added details. That is why they perform better with respect to average ℓ_1 difference, but do not generate visually appealing results.

We also applied SCT with different hyperparamter settings on our test set and compared the output with the original hi-res terrain in terms of the the three different metric that we introduced previously; see Section 4. We report the average and standard deviations of these results in Table 3. Comparing the numbers with the ones in Table 1, it is evident that SCT with any of the reported parameter settings is still outperformed by GATA.

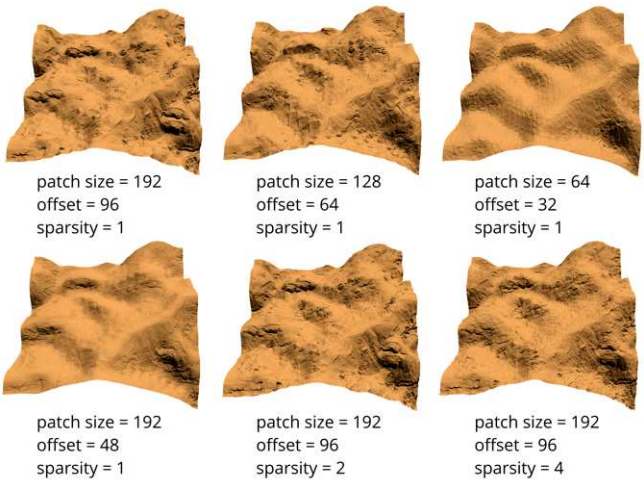

Fig. 17. The effect of different parameters on the output of SCT. Patch size 192 with offset equal to 96 tends to generate the most appealing results. The sparsity levels 1, 2 and 4 generate very similar results that are very difficult to distinguish.

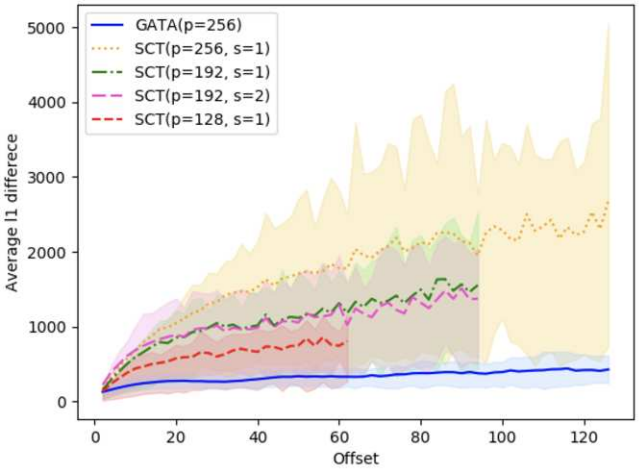

Fig. 18. We plot the average ℓ_1 difference on the overlapping area of the adjacent output patches for each method. For SCT we choose multiple sparsity levels, s, and patch sizes, p. We sweep all the offsets between 1 pixel and half of the patch size in each case. To give a better perspective in addition to the average, we also plot the 25% and 75% percentile values for each constellation of the methods. Although it seems that smaller patch sizes can get better results in terms of this metric, the results of those patch sizes are not visually appealing (see Fig. 17).

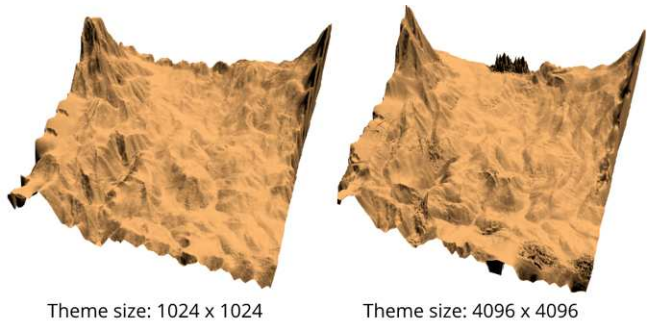

Fig. 19. We investigate the impact of the exemplar size on SCT. We want to see if using a possibly larger exemplar could automatically lead to better performance. In the left plot, we consider the 1024×1024 exemplar size that is used as the default option for SCT throughout the paper compared to a bigger exemplar size of 4096×4096.

Finally, we also investigate the impact of the exemplar size on SCT. Throughout the experiments in the paper, we have used a terrain of size 1024×1024 as the exemplar to create the atoms in the dictionary for SCT. As can be seen in Fig. 19, more artifacts and theme inconsistencies arise by increasing the size of this exemplar beyond this choice. This is due to the fact that by increasing the size of the exemplar, there is a higher chance that our exemplar would be a blend of multiple themes. In this case, SCT is not capable of controlling which theme it chooses for different patches. This could potentially lead to more incoherence as is evident from Fig. 19. Thus, if one wants to use a larger exemplar, they should manually ensure that this exemplar is stylistically coherent which could be challenging.