CS 230

Generating Terrain Elevation Models from Satellite Imagery using cGANs

Leon H. Kloker, James Swomley
Institute for Computational and Mathematical Engineering
Stanford University
`leonkl@stanford.edu, jswomley@stanford.edu`
`https://github.com/jswomley/cs230-project`

Abstract

The ability to understand the shape of a landscape is valuable in many ways, for tasks like agricultural land use surveying, natural resource identification, and environmental protection planning. To carry out the process of terrain modeling, we propose using a conditional Generative Adversarial Network (cGAN) to generate terrain elevation data from satellite visual imagery. We found that training a cGAN with L2 regularization on raw elevation data that is only shifted relative to the lowest point in an area creates a conditional generator that generates more accurate, sharp, and useful terrain models than other similar models attempting the same task.

1 Introduction

Creating terrain elevation models from single overhead images is an important task with applications ranging from flight planning to modelling water flow to precision farming and forestry. This task has been carried out for many years using LiDAR, which utilizes laser light reflectance to find the distance between a laser's source and its target. This process requires significant planning and expensive tools. Other ways to find the same information require large series of overlapping images and complicated trigonometric algorithms. With modern improvements in deep learning, simpler, cheaper, and more adaptable methods have become available. In this paper, we will present one of these deep learning methods to map an RGB satellite image pixel-by-pixel to a height map of the area.

2 Related Work

Surprisingly, there is not much previous work done in this field. Only Panagiotou et al. [2020] have investigated the described task using deep learning approaches. They employed a U-Net, CycleGAN and a conditional GAN, whereas a U-Net was used as a generator for the cGAN and also as a inverse generator in the CycleGan. The cGAN framework they used is similar to the pix2pix model introduced by Isola et al. [2017]. Their task was to predict elevation normalized with respect to the dataset they used. This means that an output of -1 corresponds to the smallest absolute elevation in the entire dataset and +1 corresponds to the highest absolute elevation. For this task, all three frameworks Panagiotou et al. [2020] employed performed similarly in terms of the L1 error of the prediction. The cGAN was the most promising method, however, to produce results that don't just

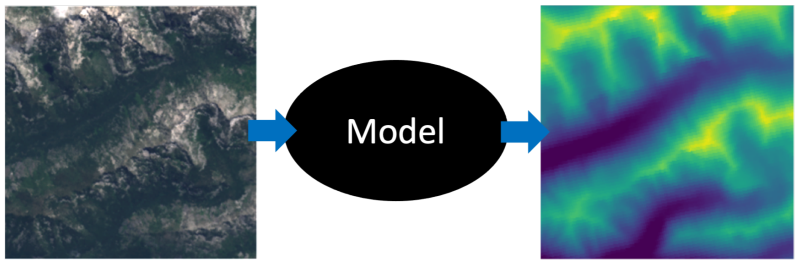


Figure 1: The model takes RGB images as input and, depending on the task, outputs a normalized or raw elevation map of the same size.

minimize the error by creating smooth elevation maps around the average height. Instead, due to the loss learned by the discriminator, it was able to produce qualitatively more realistic looking and sharper elevation maps.

A second approach to tackle the problem is to employ a diffusion model framework such as Palette by Saharia et al. [2022], as diffusion models have proven to be even more successful than GANs in some image-to-image translation tasks. As Dhariwal and Nichol [2021]. have shown, these tasks mainly consist of unconditional or class-conditional image generation. Moreover, the Palette framework has also already been applied to conditional image generation tasks such as colorization, where a U-Net architecture is used as the diffusion backbone predicting noise at a given timestep of the diffusion process. Even though diffusion models haven't been applied to elevation prediction yet, we believe that it should be possible to train Palette to create realistic elevation maps conditioned on RGB satellite images.

3 Dataset and Features

Our dataset consists of 25,000 pairs of 256 by 256 pixel images and their corresponding elevation data. The visual data is true color red-green-blue (RGB) imagery, while the elevation data takes the form of arrays representing the elevation in meters of each pixel from the RGB images. This elevation data is standardized by setting the lowest value of every image to zero, with each other value representing the relative elevation in meters above the minimal point of the same image. The median range of elevation that each image spans is 290 meters, with the most mountainous regions spanning over 2,000 meters.

Each data pair represents an approximately 10 by 10 kilometer area, with the entire dataset covering the entire western half of the United States and some of Canada.

The RGB images were drawn from the Landsat satellite, which provides a variety of spectral data including the visual wavelength activation for red, green, and blue, while the elevation data was drawn from a CGIAR satellite that focuses exclusively on elevation.

Potential limitations of our dataset include lack of diversity of image quality and time of year. Since all of our images are from the same satellite during the same time of year, it may have trouble generating elevation predictions from RGB images taken with different camera settings or at different times of the year. The strength of the dataset lies in its sheer size and diversity of terrain.

3.1 Data Preprocessing

We fetched the RGB and elevation data from the Google Earth Engine API. First, a set of larger images was generated with the intention of later splitting them up and formatting them for our needs. The RGB images were pulled from the summer of 2020 and filtered for lack of clouds. The elevation data was more difficult, since the only way to pull significant amounts of data from the API involves exporting grayscale images, which automatically casts the raw elevation data in meters to integer values between 0 and 255. With elevations spanning over 4,000 meters throughout the entire dataset, this step would create much information loss if done uniformly, and render the final smaller images with smaller elevation spans very undetailed.

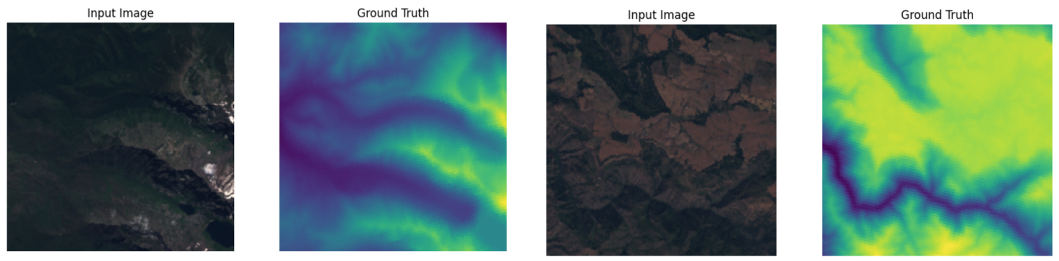


Figure 2: RGB image and corresponding height map, normalized to different colorscales for visualization purposes.

In order to get around this, we made the API-requested images as small as feasible runtime-wise, while scaling the bins to range from the minimum to maximum elevation within that requested image. By storing these scaling factors and then rescaling the image and converting it to an array post-generation, we were able to create sharp true-elevation arrays with minimal information loss.

After generation, we wrote a script to parse through all of the data and divide them into 256 by 256 pixel RGB images and their corresponding elevation arrays of the same size. During this process, the pairs were uniformly separated into training, test, and validation sets, with 23,000 pairs in training and 1,000 in both test and validation.

3.2 Problem Formulation

The way we formulated our problem was inspired largely by the poor approach taken in Panagiotou et al. [2020]. In their approach, they normalized their entire elevation dataset to range between -1 and 1, which has several drawbacks. First, the output created by the model is always in relation to the dataset it was trained on, so actual elevation can only be obtained by rescaling appropriately. Moreover, as the model output is bounded between -1 and 1 by a tanh activation function on the output layer, it is impossible for the model to accurately handle areas with elevations outside of the training data's range. Finally, the elevation data considered is elevation above sea level, meaning that a flat plain at sea level has a different elevation than a flat plain in Colorado. This creates a task with a fairly large Bayes error, as two nearly identical images correspond to two very different outputs.

Our approach solves all of these problems. Using unnormalized elevation data that remains in meters, but is standardized in each area such that the lowest point is zero, allows our model's output to be useful. The output gives an accurate and to-scale representation of the terrain of the area, regardless of where that area is. Elevation above sea level is not relevant for understanding a terrain's shape, which is the task our model aims to complete.

4 Model Architecture

While diffusion models may outperform GAN's in general image synthesis, they are very computationally expensive to train, have a much longer inference time, and haven't been shown to outperform GANs on conditional deterministic tasks. Because of this, we focused on the implementation of a cGAN for the scope of this project.

Herein, the generator architecture is equal to the U-Net depicted in figure 4. All the downsampling convolutional layers have kernels of size 4, a stride of 2 and same padding. Moreover, every convolution is followed by Batch Normalization and a leaky ReLU activation except for the first one where no batchnormalization is performed. All upsampling layers are also followed by Batch Normalization and a normal ReLU activation. Moreover the first 3 upsampling layers are subject to dropout with probability 50%. Only the last transpose convolution is not followed by batch normalization but also has a ReLU activation function as the output of the generator for the task as we defined it should be positive and unbounded. As depicted in figure 4, all corresponding down- and upsampling layers are connected by a skip connection. Furthermore, all normal and transpose convolutional layers except for the last one in both generator and discriminator don't use bias terms.

The discriminator is taken from the PatchGAN architecture introduced by Demir and Unal [2018]. It starts with three convolutional layers with kernel size 4, stride 2 and same padding. All of the first three layers are also followed by batchnorm and a leaky ReLU except for the first one where no batchnorm is performed. Then, zero padding is performed before another convolutional layer with kernel 4, stride 1, no padding, batchnorm and a leaky ReLU activation. Zero padding is done once more before a similar convolutional layer without an activation is used again to get an output of dimension 30x30. Hence, the output of the discriminator does not judge the entire elevation data at once but estimates the probability that a small patch of the given data is fake. It is also important to note that the discriminator takes the elevation data as well as the RGB image together, as a 256x256x4 sized input.

The loss function for this model is

$$\mathcal{L}(x,y) = -\log\left(D(x,y)\right) - \log\left(1 - D(x, G(x))\right) + \lambda_{L1}\|x - G(x)\|_1/(256^2) + \lambda_{L2}\|\theta_G\|_F^2$$

where y is a real elevation array, x is an RGB image, D is the discriminator, taking x and y as an input, and G is the generator.

The first term accounts for training the discriminator to classify real elevation data correctly. The second term simultaneously trains the generator to produce realistic looking outputs while the generator learns to not be fooled. The third term minimizes the L1 difference between the generated elevation and the true elevation, forcing the generator to create accurate results with respect to the ground truth. The final term applies L2 regularization to all generator parameters.

Hyperparameter λ_{L1} accounts for the relative importance of accurately capturing the elevation and λ_{L2} is the regularization parameter.

5 Experiments

As we changed the task as well as the dataset as previously described, we train a baseline model equal to the model by Panagiotou et al. [2020] on the new data. Here, λ_{L2} is set to zero and λ_{L1} to 0.7. As Panagiotou et al. [2020] used $\lambda_{L1} = 100$ but also had an output range of length 2, we start our baseline with $100 \cdot 2/290$ as our output should have a range of 290 meters on average. This model was trained on the training set for 100 epochs as we could see saturation of the training and validation error.

After training, the baseline model has an average L1 error of 32 meters on the training dataset meaning the error between the ground truth and the prediction is on average 32 meters for every pixel. The error on the validation and test dataset is 93, however, indicating overfitting. Moreover, the discriminator loss goes down to 0.01, meaning the discriminator is almost always able to distinguish real from generated elevation data. This alludes to the fact that λ_{L1} is chosen too large since the model mainly cares about fitting the training data instead of fooling the discriminator, thus almost degenerating to the case of only training a U-Net without the GAN framework.

As estimating relative elevation for some of the satellite images seems to be a difficult task, we start to focus on counteracting overfitting and thus reducing variance instead of working on bias reduction. In order to to so, we introduce L2 regularization with a varying parameter $\lambda_{L2} \in \{10^{-2}, 10^{-3}, 10^{-4}\}$.

Moreover, as the training of the baseline model has shown that λ_{L1} is too large giving too much weight to the minimization of the L1 error, we also ran experiments with $\lambda_{L1} \in \{0.01, 0.1, 0.2, 0.4\}$.

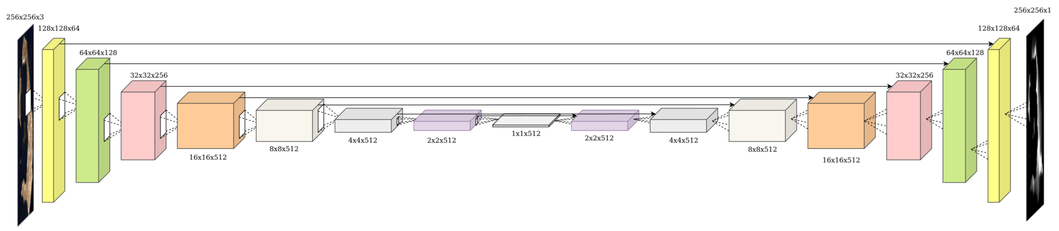


Figure 3: U-Net generator architecture (Panagiotou et al. [2020])

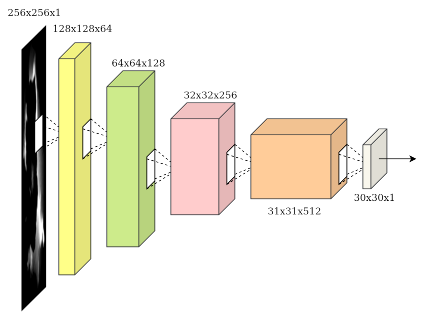


Figure 4: PatchGAN discriminator architecture (Panagiotou et al. [2020])

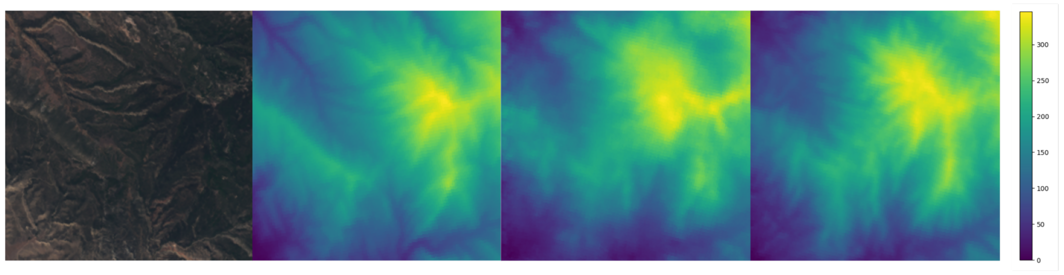


Figure 5: From left to right: RGB image, ground truth, model with $\lambda_{L1} = 0.7$, model with $\lambda_{L1} = 0.2$, both without regularization. It is visible that the right model output is more detailed and realistic as the model focuses more on creating realistic looking results than only minimizing the L1 error.

5.1 Evaluation

For all experiments except the final model, only one of the two parameters was changed at a time as training a model for 100 epochs is computationally expensive and occupies a g5.2xlarge AWS instance completely for more than one day.

Table 5.1 contains the L1 error on the test dataset for the different choices of hyperparameters. It is visible that the test error decreases until $\lambda_{L2} = 0.001$ where an optimum is reached.

Furthermore, decreasing λ_{L1} yields significantly larger test error if it is smaller than 0.2. When assessing the elevation images qualitatively, however, we can see for example in figure 5.1 that the elevation maps are more detailed and look more realistic than when the model focuses too much on just minimizing the L1 error. The same is visible but less pronounced in figure 7 in the appendix, where the output of the model with combined L2 regularization and $\lambda_{L1} = 0.4$ is compared to the baseline.

Table 1: L1 error on the test dataset for different hyperparameters.

λ_{L1}	λ_{L2}	L1 test error
0.7	0	88
0.7	10^{-4}	78
0.7	10^{-3}	66
0.7	10^{-2}	82
0.4	0	89
0.2	0	92
0.1	0	103
0.01	0	121
0.4	$5 \cdot 10^{-4}$	79

Overall, we can see that increasing λ_{L2} up to a threshold, while increasing the L1 train error, positively influences the L1 test error, thus, helping the model to generalize to unseen data. Changing λ_{L2}, however, has little to no influence on the qualitative evaluation of the produced elevation maps. Changing λ_{L1} on the other side, mainly influences the qualitative result of the model as a smaller λ_{L1} corresponds to giving more weight to the discriminator loss term of the generator. The more the model focuses on minimizing the discriminator loss, the more detailed and realistic the generated elevation maps look. When decreasing λ_{L1} below 0.2 however, the L1 error of the model starts to increase more rapidly as the model doesn't care enough about minimizing the distance to the ground truth.

Thus, for our final model, we decided to combine reasonable values for the hyperparameters in order to get a model that is able to generalize well to unseen data while simultaneously producing realistically looking elevation maps.

6 Conclusions

One of the most important parts of developing a deep learning model is understanding its utility and tailoring the model to serve its function as best possible. In this category we made the largest improvements over any existing model, by formulating our model architecture to export continuous and to-scale elevation data. In addition to this, altering the loss function by adding L2 regularization and changing the hyperparameters to improve model function allowed us generate a model that is superior to the baselinee both qualitately and quantitatively; we improved the sharpness of output images while decreasing L1 loss by nine meters. Further alterations could be made to our model by chaning input to accept for different and more types of image spectra, as well as applying image augmentation to make the model more generalizable.

7 Contributions

James Swomley lead in: Data Acquisition, Data Preprocessing
Leon Kloker lead in: Building and running the models
Shared: Visualizations, task reformulation

References

Emmanouil Panagiotou, Georgios Chochlakis, Lazaros Grammatikopoulos, and Eleni Charou. Generating elevation surface from a single rgb remotely sensed image using deep learning. *Remote Sensing*, 12(12):2002, 2020.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

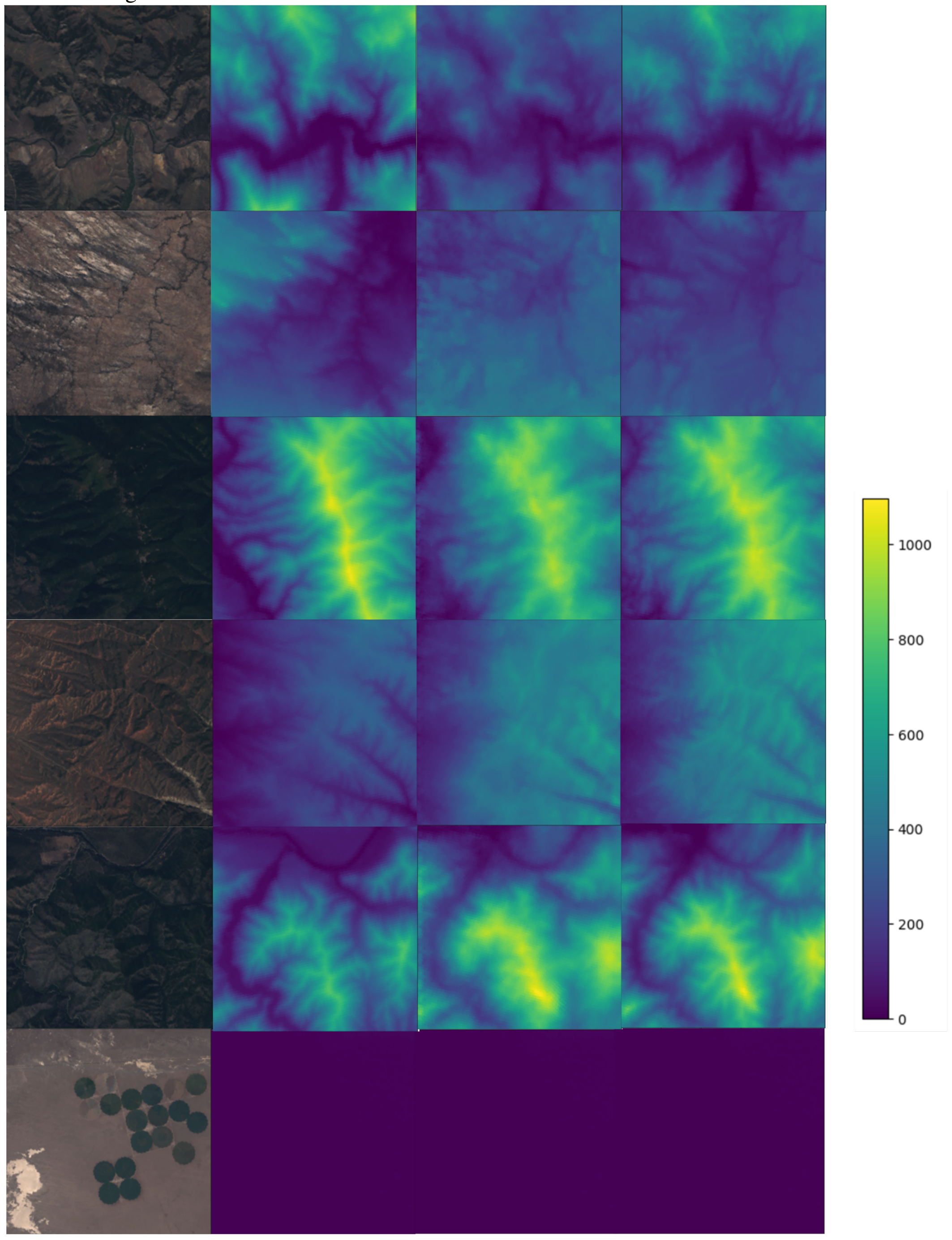
Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–10, 2022.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

Ugur Demir and Gozde Unal. Patch-based image inpainting with generative adversarial networks. *arXiv preprint arXiv:1803.07422*, 2018.

Appendices

Figure 6: From left to right: RGB image, ground truth, baseline model, final model with $\lambda_{L1} = 0.4, \lambda_{L2} = 5 \cdot 10^{-4}$. All elevation images have the colorscale depicted on the right in meters. The 6 samples were randomly drawn from the test dataset. It is visible that the elevations on the right show slightly more detail and sharpness than the predictions on the left. Moreover, the elevation values are closer to the ground truth.



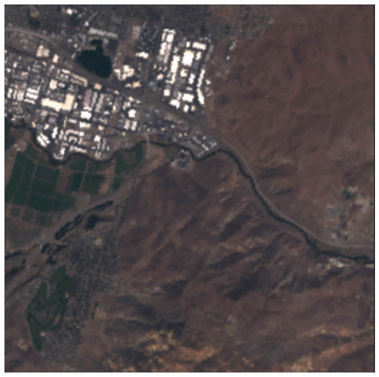


Figure 7: mountainous RGB image

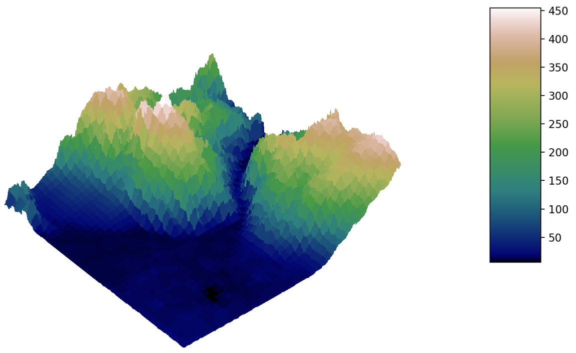


Figure 8: mountainous elevation model

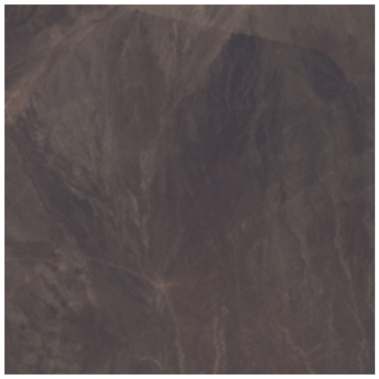


Figure 9: flat RGB image

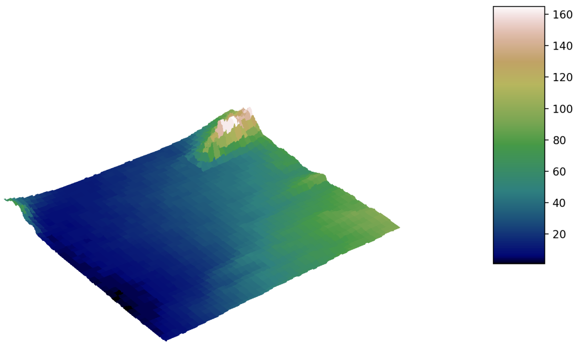


Figure 10: flat elevation model