FloodscapeDiffuser: Low-Rank Conditioning for Diffusion-Based Post-Flood Satellite Imagery Simulation

Khadijah Anwar Martin Pollack Xinqi Yu
Stanford University
450 Jane Stanford Way
{kanwar,pollackm,yxq}@stanford.edu

Abstract

We propose the FloodscapeDiffuser model for generating post-flood satellite images by conditioning on pre-flood images. This will help communities predict the effects of major flood events and plan mitigation strategies. We use as a base the generative foundation model DiffusionSat trained on satellite imagery which is capable of text-to-image generation. Its ability to produce post-flood images conditioned on pre-flood images is extended with further fine-tuning using a ControlNet on the xBD satellite dataset, which contains satellite images before and after floods. However, instead of using a full ControlNet that is computationally expensive to train, we introduce LoRA and DoRA alternatives. We show that these low-rank approximations can be trained with vastly fewer resources in much shorter amounts of time. These results are promising as it will allow more communities around the world to more easily fine-tune flood prediction models on their own data. Nonetheless, the LoRA and DoRA outputs are less detailed and fairly blurry, suggesting that these low-rank approximations are better suited to high-level approximation tasks than detailed simulations.

1. Introduction

Climate change is driving ever-more frequent and severe natural disasters including floods, wildfires, hurricanes, and more. Photorealistic visualizations of how a specific location might appear after such catastrophic and damaging events can help governments, NGOs, and residents communicate risk and prepare effectively. As a case study, we focus specifically on floods, whose risks are constantly growing in the face of rising sea levels and the proliferation of extreme weather patterns.

DiffusionSat [8] is a generative foundation model for satellite imagery that can generate realistic, high-resolution satellite imagery taking as input textual prompts and metadata like coordinates, timestamps, ground sampling distance, and cloud cover. However, we wanted to provide further image conditioning to this stable diffusion model to be able to perform **temporal inpainting**. In this process, we provide as input a satellite image of a location right before a flood and task the model with outputting an image of the same location right after a flood has occurred.

A popular model for extending the functionality of a foundation model to be able to condition on images is **ControlNet**[22]. However, these models can still be quite large and costly to train. Several alternative models have been introduced, most notably **ControlLoRA**[20], which combines ideas from ControlNet and low-rank adaptation (LoRA) for more efficient training and inference. There is also the possibility to use weight-decomposed LoRA (DoRA) instead when creating an simplified version of ControlNet.

Our proposed model **FloodscapeDiffuser** combines these ideas, taking the foundational DiffusionSat model as a base and finetuning it using the model architecture of ControlLoRA or ControlDoRA to allow for satellite imagery conditioning for flood data. We show that despite the smaller size of these low-rank alternatives, they are still able to perform adequately with much lower training and memory costs. We compare our results with those of DiffusionSat, which performed a similar task using a full ControlNet architecture.

The inputs and outputs are the same for our FloodscapeDiffuser model as for the full ControlNet architecture. We pass the model an RGB tile with a satellite image of a landscape pre-flood, metadata such as coordinates and distance measures, and the following static prompt: "a fmow satellite image after being affected by a flood natural disaster". The model then produces a simulated post-flood image that preserves realism and local geometry. Our model does this while requiring far fewer trainable parameters than the full-rank ControlNet baseline while still producing realistic and faithful images.

We find that ControlLoRA and ControlDoRA are indeed

faster and more efficient to train than the baseline, with the low-rank model taking roughly 10 minutes per epoch to train on an A100 GPU with 40GB of RAM. However, we also find that the outputs of the ControlLoRA model are far fuzzier and less similar to the groundtruth. This suggests that ControlLoRA may be better suited to tasks requiring high-level segmentation and approximation than detailed visual simulation.

2. Related Work

Latent-diffusion models (LDMs) are currently the state-of-the-art in producing high-quality generated images first proposed in 2022 [14]. Instead of processing the raw image pixel values through a **diffusion model** (DM) comprised of a forward diffusion process and a **denoising U-Net** [15], the pixel values are transformed into a latent space using a **variational autoencoder** (VAE). Before an input image is passed to the DM, it goes through the encoder of the VAE, and after denoising occurs in the DM the final output is passed through the decoder of the VAE to then be compared to the ground truth output. By converting images from pixel space to latent space, it has been shown to greatly reduce computational costs.

LDMs thus generate unconditioned images as output at inference time. However, researchers aimed to add conditioning in the form of either text or conditioning images to guide the generation of new images. A pivotal example is the foundation model **Stable Diffusion** (SD) which was created by the inventors of LDMs [14]. They trained an LDM using a subset of the LAION-5B database [16] containing an 860M parameter U-Net while using a frozen CLIP ViT-L/14 text encoder [12] with 123M parameters to condition the output images based on textual prompts. The textual conditioning is added in the form of cross-attention layers within the denoising U-Net of the LDM. This model is used as a base for an abundance of tasks that can be accomplished through fine-tuning, with new versions constantly being published publicly on GitHub and Huggingface for easy access. However, sometimes the results of SD can still be uncontrolled and unstable and be heavily reliant on a well-written textual prompt, something which models like **T2I-Adapter** have tried to address by adding additional textual conditioning to the pretrained SD model [11]. The T2I-Adapter model freezes the weights of the SD model while learning new external weights that transform conditional information that is concatenated to the inputs of every downsampling layer within the denoising U-Net. The model greatly improves textual conditioning while also allowing for the possibility of multiple conditions to be enforced.

However, textual prompts are not the only form of conditioning that may be required for a task. Images are another major modality used to condition LDMs and produce outputs aligned in a certain way. This conditioning has been accomplished in multiple ways. Firstly, in **InstructPix2Pix** [2] an additional image conditioning channel is added to the first convolutional layer in the denoising U-Net, where the conditioning image embedded in the latent space is concatenated with the noisy latent. This method is used mostly to edit or alter input images directly. However, for certain use cases this may be somewhat restrictive: instead of editing a picture we may want to more generally guide image generation using spatial control in the form of structural elements like edge maps or depth maps.

This brings us to the second main method for image conditioning: **ControlNet** [22]. Instead of adding additional conditioning channels like InstructPix2Pix, ControlNet creates two copies of the weights of the downsampling portion of the pretrained U-Net from Stable Diffusion. One set of weights is frozen to ensure information gained during pretraining on billions of images is not lost. The other set is initialized using the pre-trained weights and then learned by using a concatenation of the latent noise with the conditioning image in latent space. These two copied layers are then combined with zero convolutions, or 1×1 convolutions initialized to zero weights and biases which again helps protect the strong pre-trained backbone "by eliminating random noise as gradients in the initial training steps" [22]. ControlNets have shown great promise in guiding image generation in a more generic way besides directly editing images and have shown success in a variety of conditioning types like Canny Edge [3], Depth Map [13], Normal Map [18], M-LSD lines [5], HED soft edge [21], ADE20K segmentation [23], and Openpose [4].

However, ControlNet architectures can still contain many layers and have high complexities. For example, the original ControlNet model [22] had 361M parameters, which can be impractical and resource-intensive for a majority of use cases. A popular approach to fine-tune foundation models in a way that is more computationally and memory efficient while also not compromising on performance is low-rank adaption (**LoRA**). At first, the technique was exclusively applied to large language models, but recently their effectiveness for many different modalities have been discovered. LoRA involves freezing the pretrained weights and instead optimizing low-rank matrices that model dense layers' changes in the fine-tuning process [7]. Recent work has further extended LoRA to produce techniques like **DoRA**, or weight-decomposed low-rank adaptation [9]. Instead of fine-tuning a low-rank matrix for an entire weight matrix, DoRA first decomposes the weights into magnitude and direction, with LoRA being used to estimate the more complicated direction part. The magnitude is then estimated with a few parameters, with DoRA overall being more stable and having a greater learning capacity than LoRA.

The image conditioning ability of the ControlNet architecture has been combined with LoRA to form the **Control-LoRA model** introduced in 2024 [20]. Control-LoRA can take a 4.7GB ControlNet architecture and distill it to just 377MB [17]. Although there does tend to be slight decreases in performance related to metrics like Frechet Inception Distance (FID) as well as output clarity, Control-LoRA models take much less time to train, require far less memory, and do not require as many or as advanced GPUs. This makes it ideal for those without access to the largest computers or other resources but still want to harness the power of a foundation generative model that requires image conditioning.

3. Methods

Our proposed model, **FloodscapeDiffuser**, extends the Stable Diffusion base by introducing spatial and metadata conditioning in addition to the existing textual conditioning to simulate post-flood satellite imagery from pre-flood observations. The core architecture includes a frozen Stable Diffusion UNet backbone augmented with efficient ControlLoRA pathways that replace larger conventional ControlNet weights. We additionally incorporate a frozen CLIP encoder for textual prompts, sinusoidal timestep embeddings, structured embeddings for satellite metadata, and classifier-free guidance during inference. This design enables effective flood prediction while dramatically reducing computational requirements compared to full ControlNet approaches. For reference, our model architecture is very similar to the model architecture in Appendix A.1, except the green "DiffusionSat trainable copy" layers are replaced with either LoRA or DoRA adapters.

3.1. CLIP Text Encoder

We use the frozen CLIP ViT-L/14 text encoder with 123M parameters to process text prompts. All training examples use the same fixed prompt: "a fmow satellite image after being affected by a flood natural disaster." The tokenized prompt is encoded into a text embedding $E_t \in \mathbb{R}^{77\times768}$, which is injected into every level of the denoising UNet via cross-attention layers, consistent with the standard conditioning mechanism used in Stable Diffusion.

3.2. Timestep Embedding

Each diffusion timestep t is embedded using a sinusoidal positional encoding $\gamma(t)$, followed by a two-layer multilayer perceptron (MLP) to produce the final timestep embedding $E_{\text{time}} = \text{MLP}(\gamma(t))$. These embeddings are injected into each residual block of the UNet and Control-LoRA modules through addition, enabling the model to learn temporally-aware denoising behavior. In other words, this allows it to condition its denoising behavior on the current noise level.

3.3. Metadata Embedding

Each satellite image is associated with a JSON metadata file containing structured information such as sensor ID, coordinates, timestamp, and ground sampling distance. We extract seven numerical features from these files, normalize them, and process each feature independently through sinusoidal positional encoding followed by a TimestepEmbedding module. The final metadata embedding is computed as $E_m = \sum_{i=1}^{7} \text{TimestepEmbedding}_i(\gamma(m_i))$, where m_i represents the i-th normalized metadata feature and $\gamma(\cdot)$ denotes sinusoidal positional encoding. This embedding is added to the timestep embedding and injected into both U-Net and ControlLoRA blocks to provide spatial and temporal context for denoising.

3.4. VAE

We use the pretrained variational autoencoder (VAE) from Stable Diffusion to encode and decode RGB images into and out of the latent space. It has been shown that performing the forward and reverse diffusion processes within a latent space can drastically improve convergence and speed up training [14]. Given a $512 \times 512 \times 3$ pre-flood image y_{pre}, we compute its latent representation using the VAE Encdoer E to get $z_{0,pre} = E_{\text{VAE}}(y_{pre})$, where $z_{0,pre} \in \mathbb{R}^{4\times64\times64}$. This latent vector $z_{0,pre}$ is then considered fully denoised and begins the forward diffusion process. Then at the very end of our model pipeline when we aim to reconstruct an image consisting of pixels, we utilize the VAE Decoder D. At this final stage, our initial fully denoised image representing a pre-flood satellite image $z_{0,pre}$ has been converted to a fully denoised image of a post-flood scene $z_{0,post}$ by our forward and reverse processes. Thus, to get our predicted post-flood image $\hat{y}_{post}$ in the form of a $512 \times 512 \times 3$ RGB image, we perform the calculation $\hat{y}_{post} = D_{\text{VAE}}(z_{0,post})$. This is then considered the output of our overall model. The VAE weights remain frozen throughout training.

3.5. Forward Diffusion Process

Once our pre-flood image has been embedded into the latent space defined by our frozen VAE model to get the latent variable z_0, we perform the forward diffusion process which consists of T small noising steps. Using the stepwise variances β_t with $t \in \{1, \dots, T\}$, we get

$$z_t = \sqrt{1-\beta_t} z_{t-1} + \sqrt{\beta_t}\epsilon, \qquad \epsilon \sim N(0, I) \quad (1)$$

But by the properties of Gaussian distributions we get the closed form solution

$$z_t = \sqrt{\bar{\alpha}_t} z_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, \qquad \epsilon \sim N(0, I), \quad (2)$$

where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_t$.

If the values of the stepwise variances β_t are chosen correctly, the distribution of z_T should be indistinguishable from true Gaussian noise. No parameters are learned at this stage, but the noised latent variables z_t for $t \in \{1, \ldots, T\}$ are used in the denoising process carried out by the denoising UNet.

3.6. Stable Diffusion Denoising UNet

We use the pretrained UNet2DConditionModel from the DiffusionSat foundation model as the main denoising model. This UNet has an encoder-decoder structure with 860M parameters, with cross-attention layers for text conditioning and residual blocks for feature processing. The network takes as input a noisy latent z_t, timestep embedding E_{time}, text embedding E_t, metadata embedding E_m, as well as spatial residuals from the ControlNet modules. The output is then a new latent vector z_{t-1} representing the results of one step of denoising. Noisy latents are iteratively passed through the denoising UNet to produce the final cleaned latent vector. After being passed to the decoder of the VAE, this cleaned latent vector becomes the output of the model pipeline. Note that all UNet weights remain frozen when training our ControlLora and ControlDoRA models.

3.7. ControlNet with Frozen UNet Backbone

To enable conditioning on pre-flood imagery y_{pre}, we use a ControlNet-style pathway. For a pure ControlNet architecture, a copy of all UNet downsampling block weights are made, with one set remaining frozen and the other being trainable during fine-tuning. The conditioning image, in our case y_{pre}, is also transformed into the latent space using the VAE encoder. It is then processed by the trainable copy of the downsampling blocks, with the output being injected into the main UNet in the form of residuals. This injection is done via zero-initialized 1×1 convolutions ("zero convs") at each resolution. This architecture preserves the pretrained UNet's capacity while enabling fine-tuned spatial control.

3.8. LoRA for Efficient Adaptation

To reduce computational cost and the number of parameters needed, we replace full-rank ControlNet trainable matrices with Low-Rank Adaptation (LoRA) layers. LoRA modifies the ControlNet model by replacing the full trainable copies of the UNet downsampling block weights with rank-r matrices. Now if $W \in \mathbb{R}^{n \times m}$ are the frozen UNet weights, we introduce low-rank matrices A, B to model the residuals ΔW instead of a full residual matrix in the original ControlNet. Thus, using LoRA for ControlNet, the UNet downsampling weights become

$$W' = W + \Delta W, \tag{3}$$

where $\Delta W = AB^\top$ and $A \in \mathbb{R}^{n \times r}, B \in \mathbb{R}^{m \times r}$.

In our implementation, we use $r = 4$ for linear layers and $r = 0$ for convolutions, resulting in drastically fewer parameters than full ControlNet.

3.9. DoRA Alternative

We also experiment with Weight-Decomposed Low-Rank Adaptation (DoRA), which separates the magnitude and direction components of weight updates. DoRA rescales the low-rank update by a learned magnitude tensor, with the UNet downsampling weights for ControlDora becoming:

$$W' = \frac{\mathbf{m}}{\|W + AB^\top\|_c} \cdot (W + AB^\top) \tag{4}$$

where $\mathbf{m}$ is a learnable magnitude vector and $\| \cdot \|_c$ denotes column-wise normalization. This decomposition provides finer control over the adaptation process and can improve training stability, particularly beneficial given our small batch size constraints. Both LoRA and DoRA adapters are implemented using custom PyTorch modules and can be toggled via command-line flags during training.

3.10. Classifier-Free Guidance (CFG)

During inference, we use classifier-free guidance to adjust alignment with the conditioning inputs. We train the model to handle both conditional and unconditional inputs by randomly dropping conditioning information with 10% probability during training. At inference, we interpolate between conditional and unconditional predictions:

$$\hat{\epsilon} = \epsilon_\theta(z_t, \emptyset) + w \cdot (\epsilon_\theta(z_t, c) - \epsilon_\theta(z_t, \emptyset)) \tag{5}$$

where w is the guidance scale (we use $w = 6$), c represents all conditioning information (text, image, and metadata), and ϵ_θ is the predicted noise. CFG improves sample fidelity and conditioning adherence without requiring model retraining.

3.11. Loss Function

Our main objective is minimizing the standard diffusion objective using denoising score matching with mean squared error (MSE):

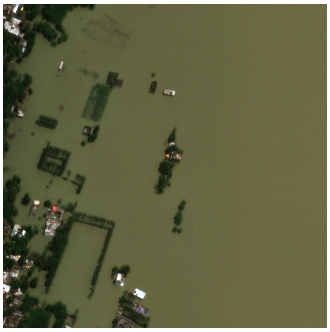
$$\mathcal{L}_{\text{MSE}} = \mathbb{E}_{z_0,t,\epsilon}\left[\|\epsilon_\theta(z_t, t, c) - \epsilon\|^2\right] \tag{6}$$

where c encompasses all conditioning information (text, pre-flood imagery, and metadata).

3.12. Our contributions

Our implementation builds on the HuggingFace Diffusers package, the DiffusionSat repository [8], and the ControlLora repository [20]. Our key contributions included implementing DoRA layers and integrating the ControlLora layers with the DiffusionSat Unet architecture for

(a) Before flood



(b) After flood

Figure 1: Data example

proper satellite imagery conditioning. In addition, we optimized the timestep and metadata embedding layers for our use case. New scripts were written to fit this new model as well as perform inference with it. In this work, we compare our LoRA- and DoRA-based models against the full ControlNet baseline from DiffusionSat and analyze the effectiveness of our more efficient conditioning approach.

4. Data

We fine-tune our model and evaluate its performance on the **xBD** disaster-response dataset [6]. It contains images of 19 natural disasters worldwide, with all images having ground-sample distances of 0.3–0.5m. This dataset was initially created to support the xView2 modeling competition for assessing building damage caused not just by floods but also hurricanes and earthquakes. From this dataset, we extracted before and after images related to three different flood events: the Nepal Monsoon Floods in August 2017, the U.S. Midwest Floods in March 2019, and the Palu Tsunami in September 2018. See 1 below for an example of a satellite image before (1a) and after (1b) a flood.

Each flooding event provides geo-aligned pre-flood and post-flood RGB tiles in 16-bit GeoTIFF format at 1024×1024 pixels. These are accompanied by JSON metadata (sensor readings, latitude/longitude, timestamp, etc.).

Using these raw images we performed multiple pre-processing steps. First we clamped each pixel value to transform it from 16-bit to 8-bit. We then center-cropped/resized every tile to resolution 512×512 to match the input resolution of DiffusionSat. We then grouped these pre-processed images, JSON metadata, and our constant text prompt "a fmow satellite image after being affected by a flood natural disaster" together for each event. Lastly, we packaged samples into the standard WebDataset format. This consists of putting 100 events together in a single TAR shard, with a shard-manifest text file referencing each TAR file. This format allows for efficient streaming I/O during training and inference.

We split our dataset into three disjoint sets in the form of train, validation, and test datasets. We made sure to stratify these datasets by flood event, so each dataset has at least one event from Nepal, the U.S., and Palu. The train dataset contained roughly 1200 pairs of images, and the both the validation and testing datasets contained 120 pairs. Then before passing the images to the model for training and inference, we performed an additional processing step in the form of normalization, assuring that the RGB values within each channel have mean zero and standard deviation one. No data augmentation was performed.

5. Results

5.1. Experimental Setup

For training our FloodscapeDiffuser model, we trained the layers of the ControlLora conditioner for 10 epochs with a learning rate of $5e{-}4$. By only training for 10 epochs we saw less overfitting when evaluating on the validation set, and the loss already had began to plateau at that stage. This was fewer epochs than what DiffusionSat used for their full ControlNet model [22], but it seemed appropriate for our case. Our learning rate was larger than that used by DiffusionSat as a higher value allowed loss values to decrease more quickly at the beginning of training. This fact that LoRA benefits from the highest possible learning rate that facilitates stability has been proven rigorously previously [1]. In addition, we used the AdamW optimizer because of its proven superior performance for ControlNet satellite applications [8]. Our loss plots for training are included in Appendix A.2 and A.3

Other experimental design choices were limited by the computational complexity and memory requirements of diffusion models. We could only use batch sizes as large of 8 without our training sequence terminating due to memory issues. To compensate for this, we used gradient accumulation steps of 2 to get effective batch sizes of 16 while still managing our memory usage. Also, for hyperparameter selection we used our training and validation data splits instead of multi-fold cross validation given the long training times and computational cost of our models.

5.2. Qualitative Evaluation

We start by evaluating our models qualitatively. We visually inspected our model outputs which were produced using 50 inference timesteps and a classifier-free guidance scale of $w = 6$. In Figure 2 we provide a flood event that occurred next to a body of water, and Figure 3 represents an example fully on land. In both of these figures, on the far left is the conditioning image in the form of the landscape before the flood. On the far right is the target, or a satellite image of the landscape after the flood occurred. Then in the middle left is the output of the ControlLoRA model and the middle right is the output of the ControlDoRA model. The two middle images aim to mirror the image on the far right. In addition, above the images for the true pre- and post-flood events are metadata information passed to our model.

Overall, we see that our model outputs are somewhat blurry and unclear. They also tend not to be very colorful, with most of the images being beige or brown or white. The one exception is the ControlLora model near water in Figure 2 which correctly makes the water blue, albeit not the correct shade of blue from the ground truth satellite images. This blurriness and lack of color is probably due to the fact that we are using low-rank adaptations. With fewer parameters to estimate, our models are more likely to underfit. The lack of parameters make it difficult for our model to produce small details in a satellite output, and it also makes it hard to have rich information in the color channels, thus defaulting to somewhat average colors like beige and white. As a result, we suggest using our results for more higher-level analysis and large-scale damage assessment, as details are hard to notice in our images.

However, the structures of our produced outputs does seem rich and accurate for flood situations. The outlines of roads, buildings, trees, landscapes, and coastlines are clearly visible and faithful to the groundtruth in our outputs. We also see clear signs of flood effects. For example, in Figure 2 the ControlLora and ControlDoRA coastlines have clearly receded from the conditioning pre-flood image, although for ControlLora it is highly exaggerated. This mirrors the fact that the groundtruth target has less coastline than the pre-flood image. Similarly, in Figure 3 the groundtruth target has wider gaps between trees and more flat sections, and this is clearly seen in our outputs. Looking in the bottom left of these images, we see larger gray patches than what existed pre-flood, showing clear signs of flood damage.

Overall, our ControlLora and ControlDora outputs seem to be of relatively similar caliber. The coloring of the ControlLora model seems to be slightly richer in Figures 2 and 3, and we also note that the lines seem to be slightly more distinct in the ControlLora images. This was unexpected given the current popularity of DoRA over LoRA in the literature.

5.3. Quantitative Evaluation

Next, we report numeric metrics such as the popular Peak Signal to Noise Ration (PSNR) and Structural Simularity (SSIM), which we compare to our baseline.

The first quantitative measure we use is PSNR. Given an input image f and output image g and a maximum possible pixel value of L, the equation for calculating PSNR between the two images [10] is

$$PSNR(f, g) = 10 \cdot \log_{10} \left(\frac{L^2}{MSE(f, g)} \right), \quad (7)$$

where for images of height H and width W and channel dimension C we get

$$MSE(f, g) = \frac{1}{HWC} \sum_{i=1}^{H} \sum_{j=1}^{W} \sum_{k=1}^{C} (f_{ijk} - g_{ijk})^2$$

PSNR is one of the most popular metrics for assessing the quality of generated images. However, it has been shown that it does not always align well with human perception [10]. It mostly considers pixel differences, which may not be the most important factor in generation and align with human-subjective value. Also, a slight shift in pixel values can cause a dramatic decrease in PSNR while making no difference in how humans perceive it.

These issues have led to the development of other metrics such as SSIM, first developed in 2004 [19]. SSIM still focuses on differences between images, but does so by comparing luminance, contrast, and structure. The luminance of an image Y is calculated as the mean of the pixel intensity values μ_y, and contrast is calculated as the standard deviation of these values σ_y. So if we suppose that image y has C channels for an image size $H \times W$, and an individual pixel value is y_{ijk}, then

$$\mu_y = \frac{1}{HWC} \sum_{i=1}^{H} \sum_{j=1}^{W} \sum_{k=1}^{C} y_{ijk} \quad (8)$$

$$\sigma_y = \frac{1}{HWC - 1} \sum_{i=1}^{H} \sum_{j=1}^{W} \sum_{k=1}^{C} [y_{ijk} - \mu_y]^2 \quad (9)$$

Then SSIM utilizes a similarity function S to compare values between two images, with

$$S(x, y, c) = \frac{2 \cdot x \cdot y + c}{x^2 + y^2 + c} \quad (10)$$

for scalar values x and y to be compared and $c = (k \cdot L)^2$, where $0 < k \ll 1$ is a constant for numerical stability and L as above is the maximal pixel value.

Now say that we have input image f and output image g. We then have that the luminance comparison is

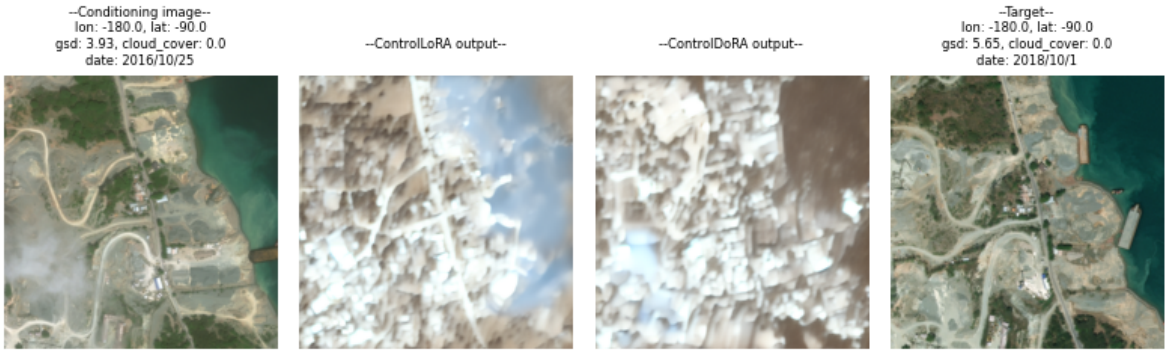


Figure 2: Model Output with Body of Water

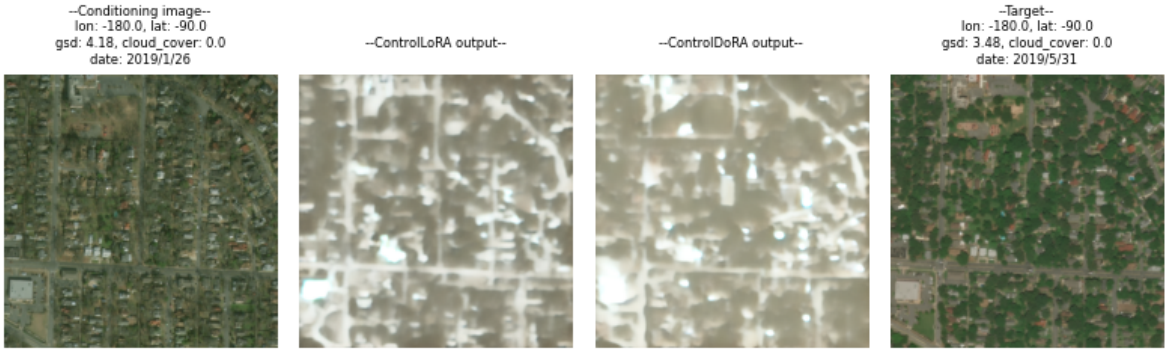


Figure 3: Model Output with Land

$C_\ell(f,g) = S(\mu_f, \mu_g, c_1)$ and the contrast comparison is $C_c(f,g) = S(\sigma_f, \sigma_g, c_2)$ for chosen scalars $c_1, c_2 > 0$.

Lastly, the structure comparison takes into account the pixel standard deviations σ_f and σ_g as well as the empirical covariance of pixel values defined as

$$\sigma_{f,g} = \frac{1}{HWC-1}\sum_{i=1}^{H}\sum_{j=1}^{W}\sum_{k=1}^{C}(f_{ijk}-\mu_f)(g_{ijk}-\mu_g). \quad (11)$$

We then get that the structure comparison C_s is similar to a correlation coefficient where for constant $c_3 > 0$ we have

$$C_s = \frac{\sigma_{f,g} + c_3}{\sigma_f \cdot \sigma_g + c_3}. \quad (12)$$

In the end, we get that the final SSIM value is

$$SSIM(f,g) = [C_\ell(f,g)]^\alpha \cdot [C_c(f,g)]^\beta \cdot [C_s(f,g)]^\gamma, \quad (13)$$

where $\alpha > 0$, $\beta > 0$, and $\gamma > 0$ are tunable parameters to adjust the relative importance of the various comparisons.

For our purposes, we chose $\alpha = \beta = 0.5$ and $\gamma = 2$. This is because for satellite imagery, the luminance and contrast can vary greatly depending on weather, cloud coverage, time of year, camera quality, etc. Thus, these two comparisons should have less effect on the overall SSIM. Instead, we want structure to have the largest effect on our comparison, allowing us to focus on the changing of bodies of water, structural damage that occurred, and landscapes affected as a result of flooding. In addition, we chose our stabilizing constants as

$$c_1 = c_2 = c_3 = (0.01 * L)^2 = (0.01 * 255)^2 = 6.5025.$$

Larger values would have taken away from the expressiveness of the comparisons, and smaller values led to small fractions that were more difficult to compare.

We compare our generated PSNR and SSIM values to that of the baseline DiffusionSat with ControlNet in Table 1 by performing inference over images in our testing dataset. However, the authors of DiffusionSat did not make publicly available their choices of α, β, γ and c_1, c_2, c_3 for SSIM in their work [8], making it more difficult to compare our metrics.

Model	**PSNR↑**	**SSIM↑**
Baseline	**18.31**	**0.39**
ControlLoRA	12.95	0.33
ControlDoRA	13.07	0.32

Table 1: Image Quality Assessment Metrics

Inspecting Table 1 and using the fact that higher PSNR and SSIM values indicate better image quality, we see that our ControlLoRA and ControlDoRA methods do significantly under-perform

the baseline with ControlNet. The PSNR for the baseline is 41.4% and 40.1% higher than the LoRA and DoRA alternatives, respectively. Then the difference is less for SSIM, where the baseline has a value 18.2% and 21.9% higher. This difference in quality metrics was expected, however, given the greatly reduced size of our model. Also, compared to ControlLora, ControlDora does better in terms of SSIM but worse in terms of PSNR.

The smaller difference in the SSIM metric compared to the PSNR metric was probably caused by our choice to focus on structure over luminance and contrast, and as we saw in our Qualitative Evaluation subsection our model is best at producing accurate structures. Another caveat is that we do not know the SSIM weighting chosen by the baseline, which could be affecting this comparison.

5.4. Efficiency Evaluation

Lastly, we highlight the efficiency of our approach with details on hardware used, training times, and the number of parameters.

Model	GPUs	Training Hours	Parameters
Baseline	4	20	378,582,160
ControlLoRA	**1**	**4**	**900,864**
ControlDoRA	**1**	**4**	1,004,480

Table 2: Efficiency Metrics

For our model training we only used a single A100 GPU through GoogleColab. However, the authors of DiffusionSat used four A100 GPUs for their ControlNet fine-tuning. See the first column of Table 2. For many researchers, meteorologists, or others interested in flooding, it can be difficult or expensive to get access to so many powerful GPUs. Thus, our methods provide a more accessible and cost-effective alternative to producing strong flood generation models.

In addition, Table 2 highlights how much faster our model training was compared to DiffusionSat. Both the LoRA and DoRA versions of ControlNet only took about 4 hours for each training run consisting of 10 epochs. This is only one fifth of the time it took to train the complete DiffusionSat ControlNet model for a similar number of iterations, which turned out to be 20 hours. Thus, for extremely critical scenarios where a flood prediction model is needed in a shorter time frame, our model would be preferred. Our shorter training time also allows more models and hyperparameter combinations to be iterated upon, greatly speeding up the total time needed to produce a final model.

As seen in the last column of Table 2, our models contained about a quarter of the number of parameters as the full baseline model based on a ControlNet architecture, going from almost 380M parameters to either 900K or 1M parameters for ControlLora and ControlDora, respectively. This means that only around one quarter of the number of parameters were needed to fit our models. Fewer parameters meant less computation and memory usage, but also helped assure that our models did not overfit during the fine-tuning phase for flood prediction. In fact, our models likely underfit. Our frozen base weights from DiffusionSat already contained very rich satellite image representations, so having fewer learned parameters meant we were better able to preserve that information gained during pre-training and avoid overfitting.

Despite the lower performance in terms of image quality, our models have many efficiency benefits. They can be trained much more cheaply and with fewer GPUs, making them more accessible. Training times are much lower as well, expediting the entire process if results are needed quickly. Lastly, with fewer parameters there was less computation, memory usage, and risk of overfitting.

6. Conclusion

In this paper, we introduced FloodScapeDiffuser, which is a lightweight, low-rank diffusion model for generating post-flood satellite imagery simulations. The model takes in a conditional input image, as well as a text prompt asking the model to simulate what the given location would look like flooded. Building on the original baseline architecture of a DiffusionSat model with a full ControlNet, we replaced the ControlNet used for image conditioning with LoRA and DoRA alternatives.

As hypothesized, we find that these low-rank adaptations are significantly more efficient to train. They require less memory and have lower training times compared to the baseline ControlNet model. Nonetheless, the tradeoff for this computational efficiency is the performance and detail that the model can accommodate. Generally, the outputs produced by the low-rank adaptations are fuzzier and less detailed, as demonstrated by the declines in SSIM and PSNR compared to the baseline model. While the model does preserve overall structure, such as building blocks and the relative positioning of structures relative to each other, it does not capture low-level detail of the structures or degree of flood damage. It tends to generate grey and fuzzy outputs.

One possible explanation for this lower performance is that the low-rank nature of this adaptation is not able to capture the complex details between the structures in the image. Thus, the model cannot show granular flooding damage and structural components. Another limitation is the limited number of training images (1110 pairs), and the low-detail nature of satellite imagery. as we only had 1100 training pairs. We may improve on this by using data augmentation or collecting more data. Another limitation included the limited batch size due to the memory limitations of the GPU (A100, 40 GB RAM). This is indicated by the general lack of stability seen in the loss curves (see Appendix A.2 and A.3), with the loss fluctuating significantly for both DoRA and LoRA. This could be alleviated by using a lower learning rate and increasing gradient accumulation.

Overall, we find that low-rank adaptations provide a computationally efficient alternative to the full ControlNet model. These might be suitable in cases where compute is limited and only a high-level approximation or segmentation is needed such as a segmentation of flooding zones or structures. However, in order to generate a detailed simulation, we likely need to leverage full-rank layers and higher-dimensional representations in the early-middle layers of the model, which often encode granular structural details.

A. Appendix

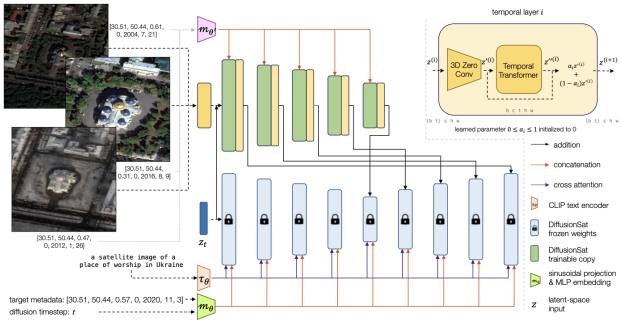


Figure A.1: DiffusionSat ControlNet Architecture

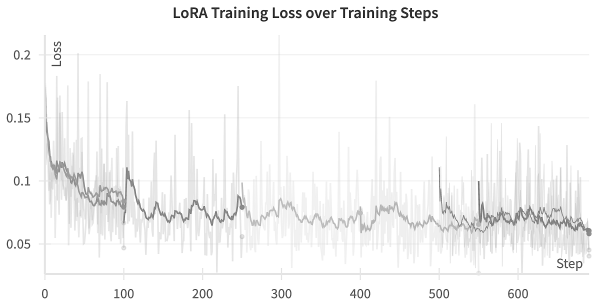


Figure A.2: LoRA Training Loss over Training Steps

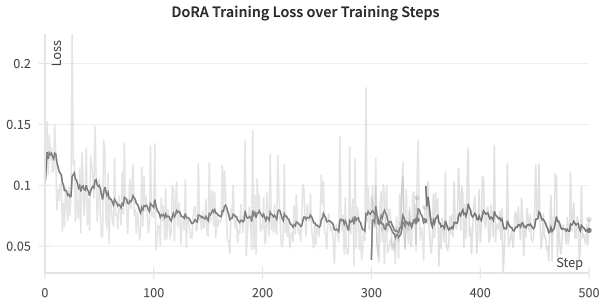


Figure A.3: DoRA Training Loss over Training Steps

B. Contributions & Acknowledgments

Martin found the source code of HuggingFace Diffusers and DiffusionSat and reconfigured all the code to make the two packages align. He wrote all the code for connecting the LoRA layers with the DiffusionSat denoising UNet and also implemented the building of the DoRA layers. He wrote all scripts related to the training and inference of the final FloodscapeDiffuser model. In addition, he wrote all sections of the final report except for the methods and conclusion.

Khadijah extracted and prepared the data, writing data preparation scripts to restructure data into shards for efficient processing. She also was in charge of the full GPU setup, Google Colab configuration, and 10 hours of training runs and loss tracking for both the LoRA and DoRA models. She wrote the conclusion section and produced loss plots.

Xinqi was in charge of reproducing the baseline ControlNet model and fine-tuning the pretrained weights on our dataset. He also refined the data pipeline to use .npy files rather than .png files to enable more efficient runs and prevent data loss. For the final report, he wrote the methods section.

References

[1] D. Biderman, J. Portes, J. J. G. Ortiz, M. Paul, P. Greengard, C. Jennings, D. King, S. Havens, V. Chiley, J. Frankle, C. Blakeney, and J. P. Cunningham. Lora learns less and forgets less, 2024.

[2] T. Brooks, A. Holynski, and A. A. Efros. Instructpix2pix: Learning to follow image editing instructions, 2023.

[3] J. Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(6):679–698, Nov. 1986.

[4] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(1):172–186, 2021.

[5] G. Gu, B. Ko, S. Go, S.-H. Lee, J. Lee, and M. Shin. Towards light-weight and real-time line segment detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(1):726–734, Jun. 2022.

[6] R. Gupta, R. Hosfelt, S. Sajeev, D. Goodman, J. Ventura, C. Prakash, B. A. Seligman, C. R. Shipley, and J. H. Wood. Creating xbd: A dataset for assessing building damage from satellite imagery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 10–17, 2019.

[7] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models, 2021.

[8] S. Khanna, R. Y. Du, I. Butsky, M. D. Hoffman, and S. Ermon. Diffusionsat: Generative foundation models for high-resolution satellite imagery. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.10205.

[9] S.-Y. Liu, C.-Y. Wang, H. Yin, P. Molchanov, Y.-C. F. Wang, K.-T. Cheng, and M.-H. Chen. Dora: Weight-decomposed low-rank adaptation, 2024.

[10] B. B. Moser, A. S. Shanbhag, F. Raue, S. Frolov, S. Palacio, and A. Dengel. Diffusion models, image super-resolution, and everything: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, page 1–21, 2024.

[11] C. Mou, X. Wang, L. Xie, Y. Wu, J. Zhang, Z. Qi, Y. Shan, and X. Qie. T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models, 2023.

[12] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark,

G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision, 2021.

[13] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3):1623–1637, 2022.

[14] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. *CoRR*, abs/2112.10752, 2021.

[15] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[16] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, P. Schramowski, S. Kundurthy, K. Crowson, L. Schmidt, R. Kaczmarczyk, and J. Jitsev. Laion-5b: An open large-scale dataset for training next generation image-text models, 2022.

[17] StabilityAI. Control-lora model card, 2023.

[18] I. Vasiljevic, N. Kolkin, S. Zhang, R. Luo, H. Wang, F. Z. Dai, A. F. Daniele, M. Mostajabi, S. Basart, M. R. Walter, and G. Shakhnarovich. Diode: A dense indoor and outdoor depth dataset, 2019.

[19] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[20] H. Wu, W. Kong, Y. Zhang, T. Namikawa, R. Panda, and R. Feris. Controllora: Low-rank adaptation for efficient image-conditioned diffusion control. *arXiv preprint arXiv:2401.12950*, 2024.

[21] S. Xie and Z. Tu. Holistically-nested edge detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[22] L. Zhang, A. Rao, and M. Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. arXiv:2302.05543.

[23] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.