Semester Project 1

Master of Science in Engineering

Data Science

Visualization of Landscape Changes using Synthesized Aerial Imagery

---

Author
Felix Matthias Saaro

Supervisor
Prof. Dr. Mark Cieliebak

Secondary Supervisor
Prof. Dr. Adrienne Grêt-Regamey

January 31th, 2025

Abstract

Land use and land cover (LULC) are in constant flux, driven by both planned interventions and unplanned dynamics. While research and planning institutions actively analyze and project landscape transformations, effectively communicating these changes to non-expert stakeholders remains a challenge.

This project addresses this gap by developing a system to synthesize aerial imagery directly from LULC data, enabling intuitive visualization of landscape scenarios. Within the project different synthesizing methods are examined and implemented. This includes a Generative Adversarial Network (GAN), a diffusion based U-NET, and a Masked Auto Encoder (MAE).

The synthesized imagery is integrated with existing or modified elevation maps to generate realistic 3D scenes, offering a tangible representation of potential or projected land changes. Additionally, the system provides interactive tooling for manual adjustments to LULC maps, empowering users to explore alternative configurations.

The results demonstrate the feasibility of this approach, though further refinement, including enlarged training datasets and computational resources are required to improve image realism. Future work will expand the capabilities of the system by incorporating anthropogenic elements, such as buildings and infrastructure, to better reflect the interaction between natural and human-made environments. This tool aims to bridge the communication divide between technical experts and broader audiences, supporting participatory planning, policy development, and public participation in sustainable land management.

Contents

1 Introduction **1**

1.1 Literature . . . 2

1.2 Outline . . . 3

2 Methods **4**

2.1 Problem . . . 4

2.2 Data . . . 4

2.2.1 SWISSIMAGE . . . 5

2.2.2 swissALTI3D . . . 5

2.2.3 Arealstatistik Schweiz . . . 6

2.3 Generative Adversarial Networks . . . 9

2.3.1 pix2pix . . . 10

2.3.1.1 Generator . . . 11

2.3.1.2 Discriminator . . . 11

2.3.1.3 Loss . . . 12

2.3.1.4 Optimization . . . 12

2.4 Diffusion Models . . . 12

2.4.1 U-NET . . . 13

2.4.1.1 Encoder . . . 13

2.4.1.2 Bottleneck . . . 14

2.4.1.3 Decoder . . . 14

2.4.1.4 Noise Scheduler . . . 14

2.4.1.5 Loss Function . . . 14

2.5 Masked Auto Encoders . . . 15

2.5.0.1 Encoder . . . 15

2.5.0.2 Decoder 17

3 Results **18**

3.1 Data Loader 18

3.2 pix2pix 19

3.3 U-NET 21

3.4 Conditional U-NET 22

3.5 ViTMAE 24

3.6 Tooling 29

3.6.1 Adjust Land use / Land cover 30

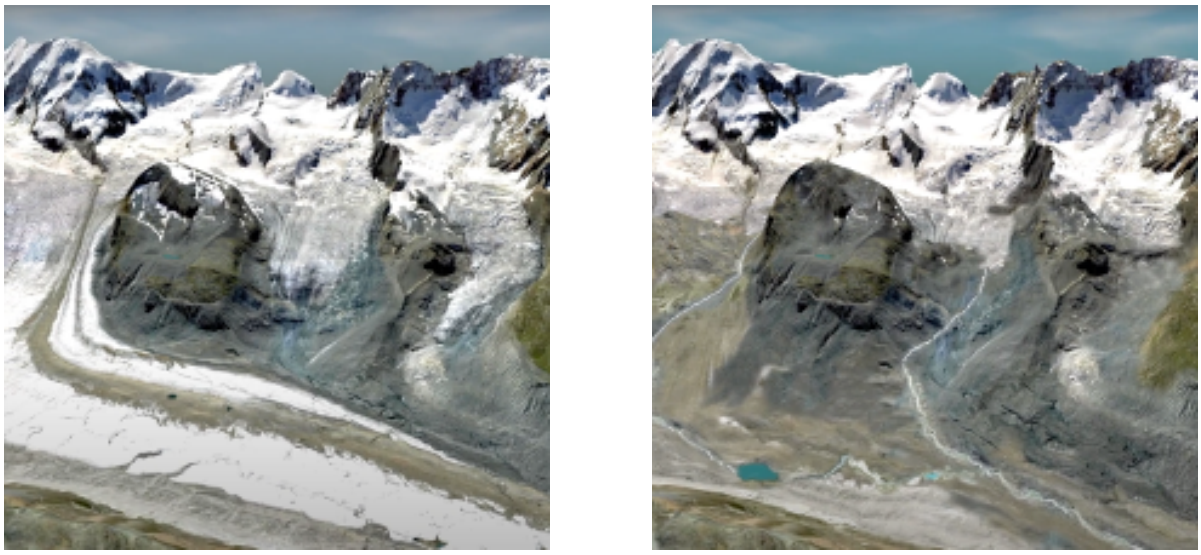
3.6.2 3D scene creation 32

4 Discussion **34**

4.1 Future Work 34

1 Introduction

Communicating climate scenarios effectively is a significant challenge, particularly when it comes to visualizing landscape changes over time. Traditional methods of illustrating these changes include 3D modeling or creating aerial images with image manipulation tools. These can be extremely time consuming and finicky. A recent visualization of how our landscape would change with an average temperature increase of 4 °C by the Swiss Federal Institute for Forest, Snow and Landscape Research (WSL) [1] is a successful example of climate communication. Another instance of such a comprehensive visualization is a video created by Julien Anet in 2023 [2].



(a) Today (b) Estimated 2060

Figure 1: Frame out of the visualization by Julien Anet [2] showing the Gorner glacier region in a side by side comparison between today and 2060.

Figure 1 was created using the SWISSIMAGE orthophoto dataset in a 10 cm resolution from 2018 [3], the swissALTI3D elevation model [4], a glacier bed model created by Grab et al. [5], and the runoff projections by Farinotti et al. [6]. The different datasets and information were combined manually, textures were adjusted, and a 3D model was created using Blender.

The aim of this project is to create an application that can visualize landscape changes similar to [2] by leveraging public data such as [3, 4, 5] while reducing the necessary manual interaction.

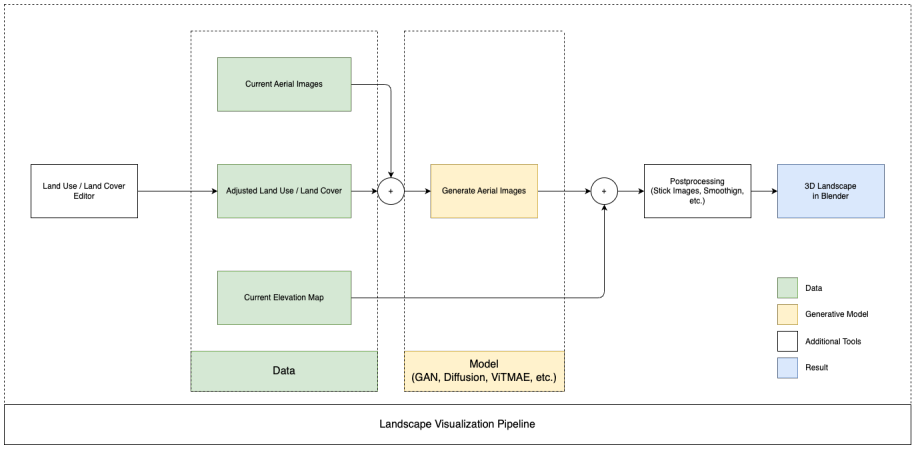


Figure 2: Overview of what the process pipeline could look like. It consists of a tool to adjust LULC maps, datasets, a model to synthesize aerial images and a post-processing tool to combine the generated images and the elevation maps into a 3D scene.

The pipeline proposed in figure 2 to create the visualization consists of three stages. (1) Manually adjust the land use and land cover (LULC) maps, based on climate models, environmental factors, current LULC, and expert knowledge. (2) Synthesizing the aerial images using the adjusted LULC maps as well as existing aerial images. (3) Combine the synthesized images and elevation maps into a 3D scene, which also includes some post processing and normalization steps.

This application has potential in several fields. In science communication, it can greatly improve how climate models are presented, making complex data easier to understand through clear visualizations thus making them accessible to a broad audience. Policymakers can use these visual tools to make better decisions on landscape planning, and educators can enhance lessons to engage students with simulated environmental changes. Additionally, the application can be valuable in exhibitions, enabling artists to depict future or past landscapes. This can help illustrate the impact of different climate actions, raising public awareness and understanding of environmental issues.

1.1 Literature

Zhu & Kelly et al. [7] introduces a method called Seamless Satellite-image Synthesis System (SSSS), which uses two neural networks to first generate satellite image tiles from map data, and then remove seams between the generated tiles. In November of 2024 Lütjens et al. [8] proposed a physically consistent method build upon a version of the pix2pix model. Arguing that to use generative models in climate change communication is not applicable because of hallucinations. In their work they demonstrated the generation of future flooding and reforestation events where physic-based models are used to create segmentation maps to condition the generation.

The introduction of Masked Auto Encoder (MAE) by He et al. [9] led to different approaches [10, 11, 12] of pre-training transformer models to generate satellite images. These pre-trained models can be used to fine tune on different downstream tasks, such as semantic segmentation or pixel regression tasks. The first being SatMAE created by Cong et al. [11] proposing a pre-training framework for temporal and multi-

spectral satellite imagery based on MAE. A few months later Reed et al. [10] proposes Scale-MAE, a pre-training method that improves multiscale representations for remote sensing imagery, outperforming standard MAE and SatMAE [11]. During the same time Jakubik et al. [12] introduced a novel framework for the efficient pre-training and fine-tuning of foundational models on extensive geospatial data. The pre-training is based on the work from [9] and resulted in the creation of Prithvi, a Vision Transformer (ViT) based model, trained on 1TB of multispectral satellite imagery from the Harmonized Landsat-Sentinel 2 (HLS) dataset. Most recently in December 2024 the team developing Prithvi has updated their model, named Prithvi-EO-2.0 [13] which offer significant improvements over its predecessor. In the newest model temporal (Year and Day of Year) and locations (Latitude and Longitude) embeddings are used to enhance the performance. A possible downstream task proposed by the authors include LULC classification.

Khanna et al. [14] introduces DiffusionSat, a generative foundation model for satellite imagery trained on large, high-resolution remote sensing datasets. It is a diffusion based approach that is proven in image generation tasks.

1.2 Outline

This project attempts to answer the following questions:

- How can advancements in image generation be used to visualize environmental changes?
- How can different image generation methods be conditioned on LULC existing data?

With these questions answered the following goals are set out to be achieved:

- Train a model to synthesize aerial images from LULC maps
- Build a demo application that includes:
  - Adjusting LULC maps
  - Generate new aerial images from the adjusted LULC maps
  - Merge the generated images with existing elevation maps to create 3D scenes

2 Methods

In this project three generative deep learning architectures are utilized: (1) Generative Adversarial Network (GAN), (2) Diffusion Models (U-NET and Conditional U-NET), and (3) Masked Auto Encoder (MAE) with Vision Transformer (ViT). These approaches and how they work are detailed below including a description of the datasets used for this project.

2.1 Problem

The generation of aerial images from LULC maps belongs to the Image-to-Image translation problems. It is a common task in computer vision where the goal is to learn a mapping between an input image and an output image, such that the output image. Specific sub tasks include style transfer, data augmentation or image restoration.

Mathematically it can be described as as follows. Let X and Y be the source and target image domains. The goal is to learn a mapping function $G : X \rightarrow Y$ such that the translated images $G(x)$ for $x \in X$ satisfy desired properties (e.g., semantic consistency or perceptual quality).

The objective function is then defined as:

$$G = \arg\min_{G} \mathbb{E}_{x \sim p_{data}(x)}[\mathcal{L}_{\text{task}}(G(x), y)]$$

Where $\mathcal{L}_{\text{task}}(G(x), y)$ is a specific loss depending on the chosen method and learning strategy. This essentially means that the average loss over the entire data distribution for the objective function is to be minimized.

2.2 Data

The selection of appropriate datasets is critical for any deep learning-based project. For this project, three datasets are required: *(1) base aerial imagery* to evaluate the reconstruction and generation capabilities of the model, *(2) classifications of LULC* to condition the generation process, and *(3) current elevation maps* to allow creation of 3D scenes.

The spatial and temporal alignment of these datasets is essential. A common challenge is the projection problem, where the curvature of the Earth necessitates projecting images and maps onto a 2D plane. Misaligned projections prevent accurate overlay of datasets from the same region. For this project, it is imperative that the LULC maps aligns precisely with the base imagery and elevation data. Thus, all datasets must share a common projection system.

The datasets are provided as GeoTIFF files, which include metadata specifying the Coordinate Reference System (CRS), spatial resolution, and horizontal and vertical datums. All files use the Swiss coordinate system LV95 and the CH1903+ crs.

2.2.1 SWISSIMAGE

SWISSIMAGE [3] is an orthophoto mosaic comprising high-resolution digital color aerial imagery covering all of Switzerland. It provides a ground resolution of 10 cm in lowland areas and main Alpine valleys, and 25 cm in the high Alpine regions. The resolution refers to the size of the small visible objects. Meaning that in a 10 cm resolution image objects of the size larger than 10 cm can be identified. The dataset can also be downloaded in a lower 2 m resolution, to reduce computational resources the lower resolution was used in this project. This dataset is updated on a three-year cycle from west to east.

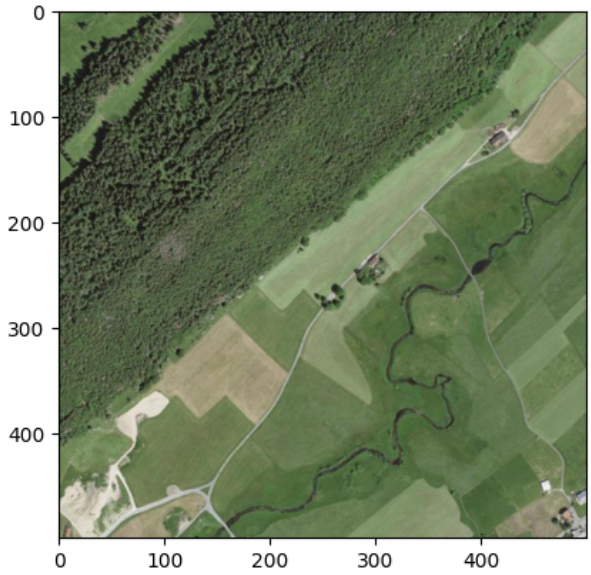


Figure 3: Aerial image from the SWISSIMAGE dataset showing parts of the Orbe river within the Vallée de Joux in the canton of Vaud. The image has a 2 meter resolution is 500 by 500 pixels large and depicts 1 square kilometer.

The geographic bounds of each tile (e.g., coordinates, length, and width) are calculated and utilized to extract corresponding tiles from the other datasets.

2.2.2 swissALTI3D

swissALTI3D [4] is a highly precise digital elevation model representing Switzerland's surface without vegetation and artificial structures. It is updated on a six year cycle to ensure accuracy and relevance.

The dataset is generated through Light Detection and Ranging (LiDAR) up to an elevation of 2000 m.a.s.l., supplemented by photogrammetric methods and stereo-correlation. This process involves capturing high-density elevation points, which are then processed and refined to produce a seamless and detailed model.

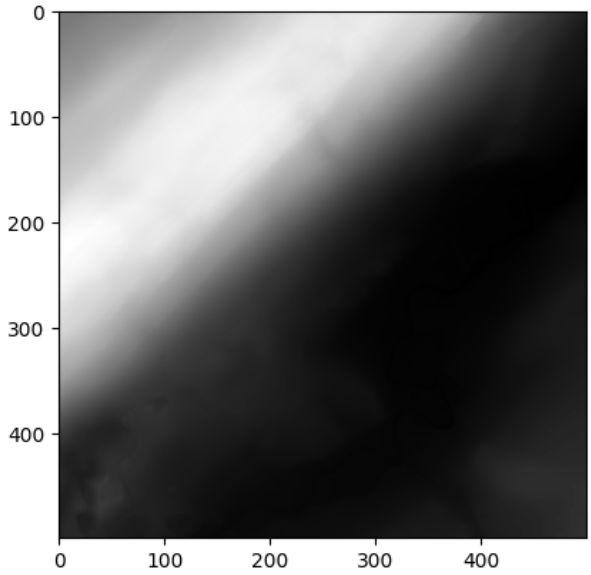


Figure 4: Elevation map from the swissALTI3D dataset showing parts of the Orbe river within the Vallée de Joux in the canton of Vaud the same as in figure 3. The image has a 30 cm resolution is 500 by 500 pixels large and depicts 1 square kilometer.

The current swissALTI3D model is based on data collected from 2012 onward, with periodic updates integrating more recent scans. Depending on the year of the scans different accuracies exist. In the newest generation the collected data is accurate to about 0.3 m in regions bellow 2000 m.a.s.l. and 1 to 3 m in regions above.

2.2.3 Arealstatistik Schweiz

The LULC segmentation maps derive from Swiss land use statistics, called Arealstatistik (AS), [15], produced by the Bundesamt für Statistik (BFS) between 2013 and 2018. A revised version was released in 2024, though this project utilizes the earlier dataset for consistency with downstream processing workflows. Classification is performed on a 100-meter grid, with each cell assigned a dominant category from the 72 classes.

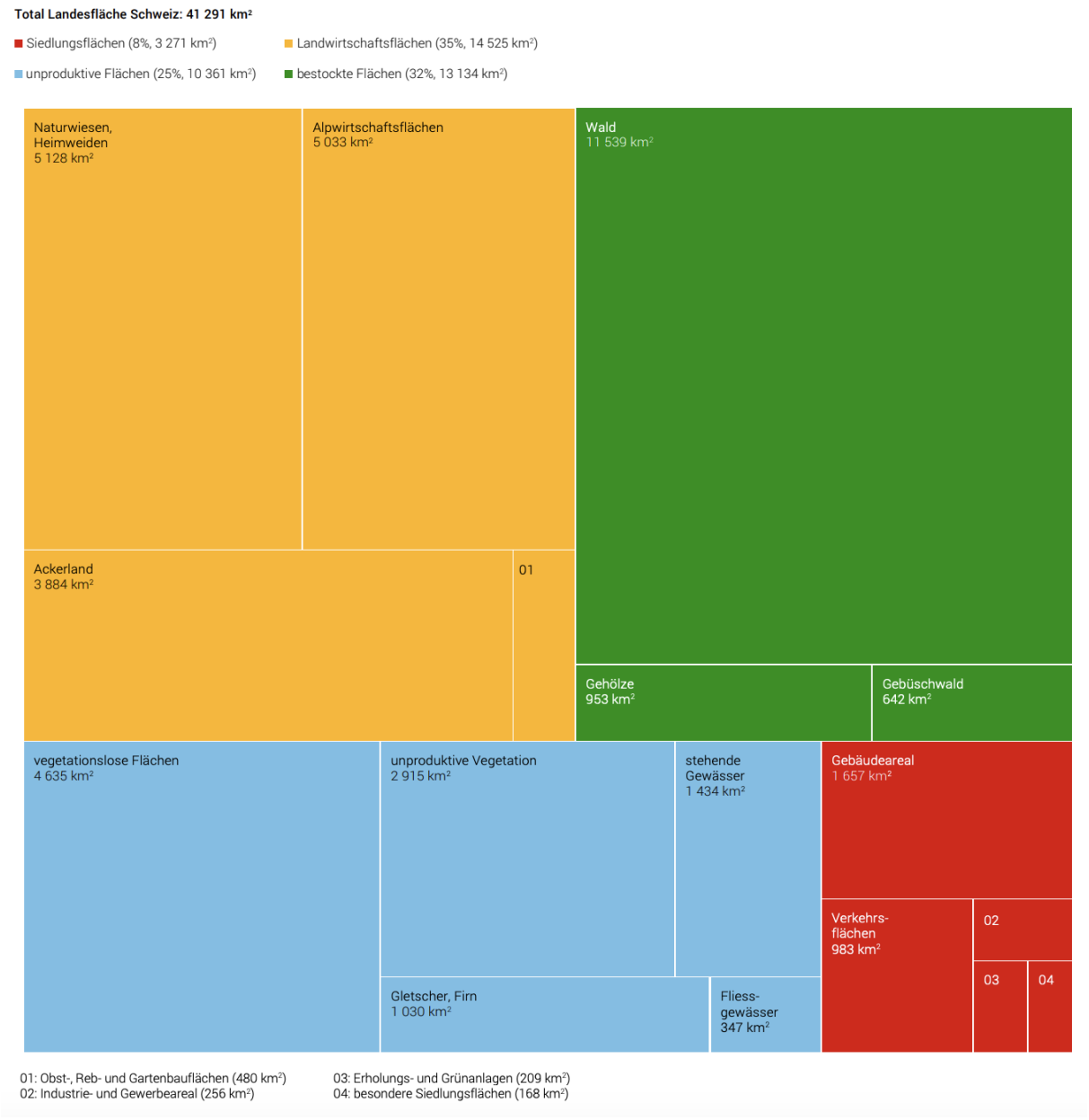


Figure 5: Distribution of LULC taken from the AS 2018 [15].

The distribution of the LULC shown in figure 5 highlights the imbalanced class distribution. It shows that the land cover class forest resembles almost a third of the entire surface of Switzerland and settlement area (Siedlungsfläche) only accounts for about 8% of the LULC. This has to be accounted for when training any model on this data as the model will most likely perform better on the well represented classes.

The four main categories in figure 5 can be subdivided into 17 classes (AS17) with 27 intermediate classes

(AS27) and 72 basic classes (AS72) as shown in figure 6

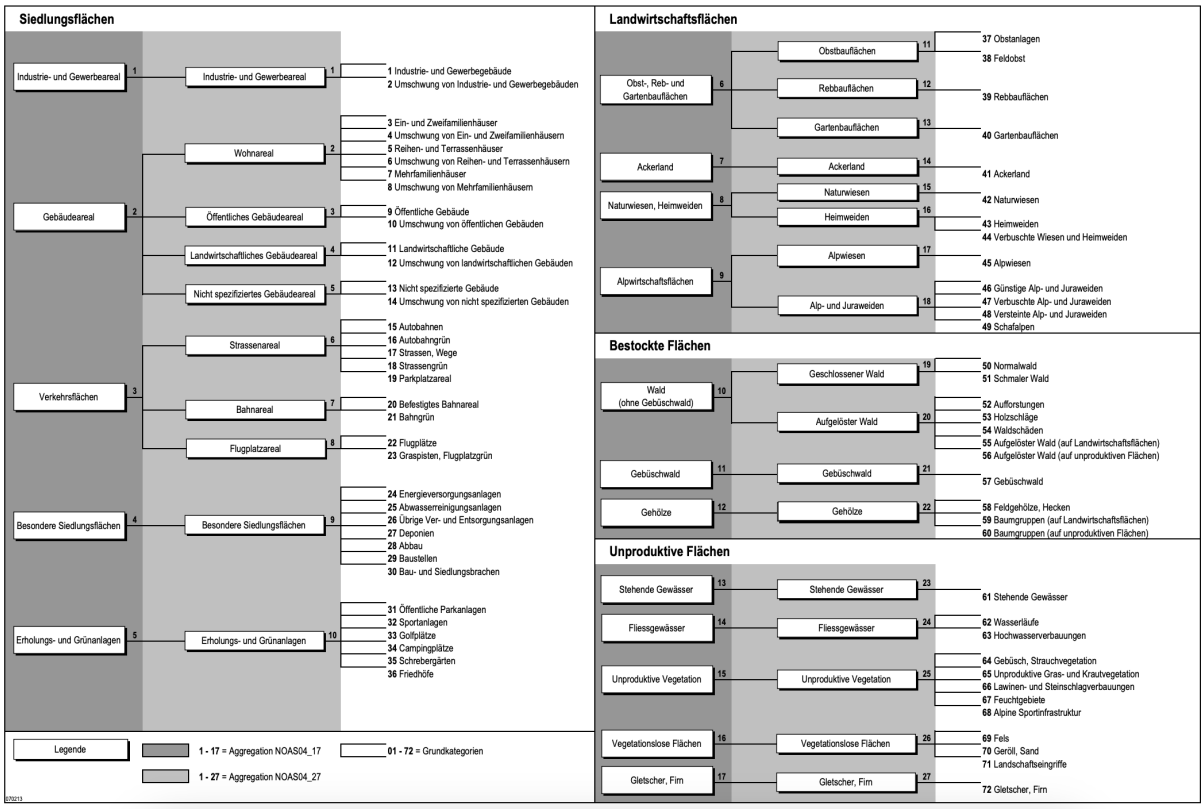


Figure 6: Standard NOAS04 nomenclature showing the hierarchical structure of the LULC classes.

In 2022, Giuliani et al. [16] introduced a down scaling method to enhance spatial resolution. Their approach probabilistically assigns LULC classes to a 25-meter grid (16× resolution increase) using neighborhood coherence analysis constrained by an expert-derived compatibility matrix. The method additionally integrates linear landscape features such as roads, rivers, and railways. While applicable to newer AS releases, the computational intensity of this workflow necessitates reliance on preprocessed 2018 data for this project.

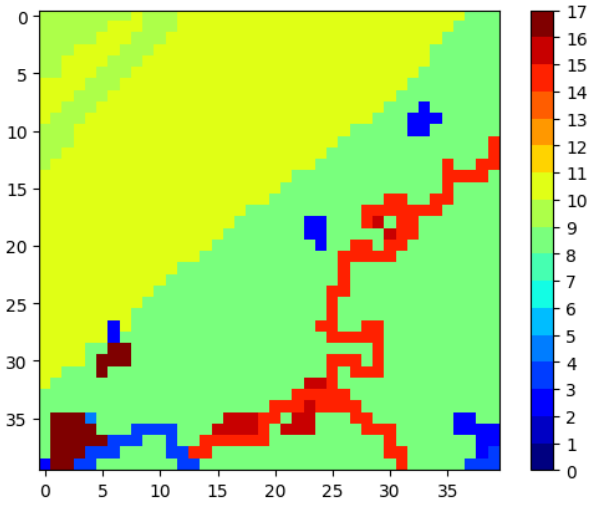


Figure 7: LULC from the down scaled land use statistics dataset on the AS17 classes [16] showing parts of the Orbe river shown in red within the Vallée de Joux in the canton of Vaud the same as in figure 3. The image is 40 by 40 pixels large and depicts 1 square kilometer.

For the experiments in this project the down scaled version AS72 and AS17 classes are used.

2.3 Generative Adversarial Networks

Generative Adversarial Network (GAN) were first introduced in 2014 by Goodfellow et al. [17]. The core idea is to train two neural networks. A generator G that synthesizes images and a discriminative model D that estimates the probability that an image comes from the training data rather than the generator. The architectural overview is shown in figure 8.

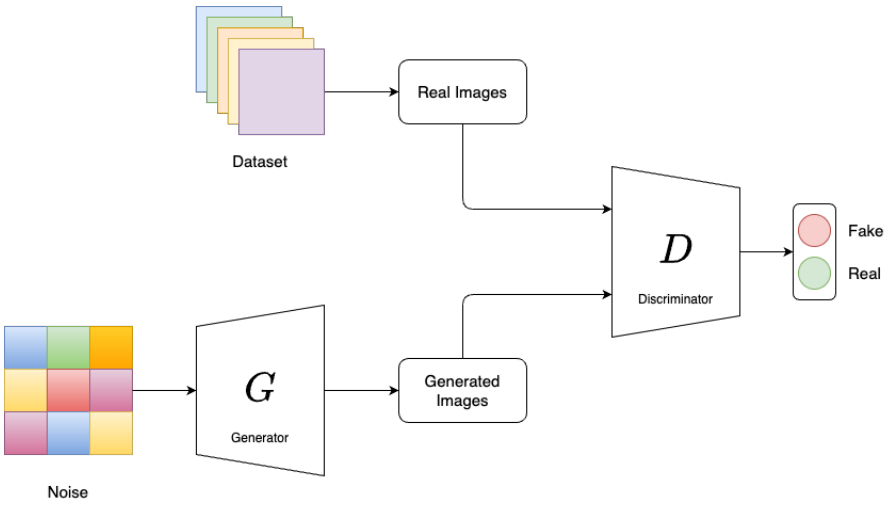


Figure 8: Overview of the training setup proposed by Goodfellow et al. [17]. Includes a generator G to produce images and a discriminator d to decide if an image is real or fake. The goal of the generator is to fool the discriminator.

The goal of the generator is to maximize the probability of the discriminator making a mistake. This is a corresponds to a minimax two-player optimization algorithm. Goodfellow et al. define this as follows:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \backsim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \backsim p_z(z)}[\log(1 - D(G(z)))]$$

Where $V(D,G)$ is the value function where the probability of discriminator D of assigning the correct label to both training samples and generated samples from G is maximized. Simultaneously G is trained to minimize $\log(1 - D(G(z)))$.

2.3.1 pix2pix

In 2018 Isola et al. [18] proposed a general purpose solution, called pix2pix, to generate images based on a conditioning image. This is usually refereed as the image-to-image translation problem. Instead of generating images from noise it is generated from low dimensional representation of an input image. For this Isola et al. proposed the use of a U-NET (explained in section 2.4.1) styled network as the generator G. The idea is to not only learn a mapping from the input to the output image but also learn a loss function to train this mapping. Typically a information scares image is used as the input image, for example a sketch of an object, to generate a high fidelity image of what that object would look like in real life. Both the input and the output image have the same shape (number of channels, width, and height).

Figure 9 shows how the approach Isola et al. could look like applied to this project. The input of the U-NET styled generator network is the LULC map.

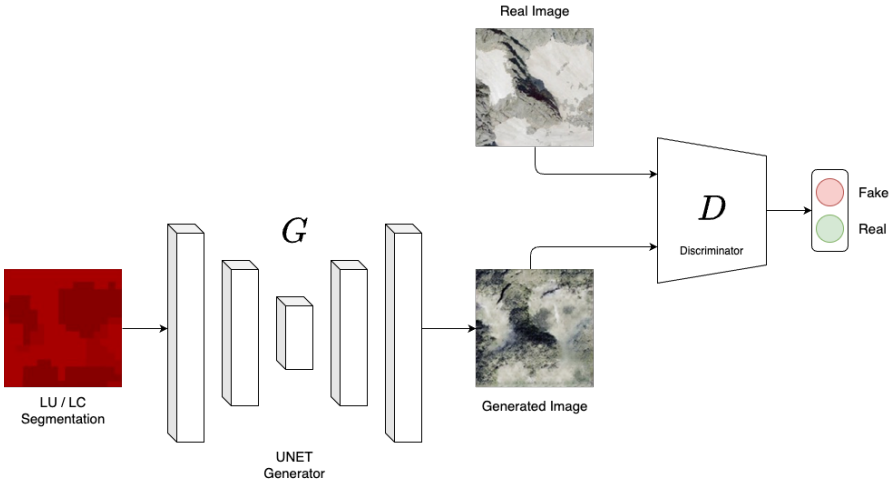


Figure 9: Overview of the implementation based on pix2pix for for this project. The input of the generator is the LULC segmentation, the output is the generated aerial image. The generator tries to fool the discriminator to predict the generated image as real.

2.3.1.1 Generator

The generator is suggested to be an encoder decoder network in the style of a U-NET. The encoder part of the U-NET has eight down blocks. Each block consists of a convolution, batch normalization, and a leaky Rectified Linear Unit (ReLU) activation function. Before the encoder the image has shape of $(256, 256, 3)$, this means the image is 256 pixels wide and tall and has three color channels. Each block in the encoder compresses the image until it has the shape $(1, 1, 512)$. This can be considered the latent space embedding of the input image.

The decoder is symmetrical, meaning it has seven up blocks plus one extra transposed convolution and a tanh activation function as the output. Each block consists of a transposed convolution, batch normalization, optional dropout of 50% and a ReLU activation. The dropout only applies to the first three up blocks. After the decoder the image has the original shape of $(256, 256, 3)$.

Additional skip connections are used to pass information from the encoder blocks to the decoder. This means that on each up block the output of the previous block is concatenated with the output of the encoder block on the same level.

2.3.1.2 Discriminator

The discriminator is a convolutional PatchGAN classifier, proposed by Li & Wand in 2016 [19]. It attempts to classify each image patch whether it is real or not. The input to the discriminator is the input image and the target image. Both have the same shape $(256, 256, 3)$, image that is 256 pixel wide and tall and has three color channels. The images are concatenated and down samples using similar down blocks as used in the generator (explained in Section 2.3.1.1). The output of the discriminator has the shape (30, 30, 1), this 30 x 30 matrix is the discriminators focus on local patches. Each output unit

corresponds to a 70 x 70 region in the input image, enabling a fine-grained evaluation of the image.

2.3.1.3 Loss

During the training for each step the loss is computed using the generator output, the discriminator output and the target image. To update the parameters, a separate loss for the generator and discriminator is computed. The generator loss is a combination of Binary Cross Entropy (BCE) as well as a weighted L1 loss. The discriminator loss is the sum of real and generated loss. Where the real loss is a BCE loss of the real images and an array of ones (ones resemble the label for real images) and the generated loss is a BCE loss of the generated images and zeros (zeros resemble the label for fake images).

Binary cross entropy loss is defined as follows:

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(p_i) + (1-y_i)\log(1-p_i)\right],$$

It is typically used for binary classification tasks, in this case deciding if an image is real or fake. So the possible classes are 0 for fake images and 1 for real images. It quantifies the distance between the predicted probability p and the true label y. The loss penalizes confident incorrect prediction heavily. Meaning that, when $y = 1$ the term $-\log(p)$ grows larger if p approaches 0, and conversely, when $y = 0$, $-\log(1-p)$ penalizes predictions where p nears 1. The total loss is averaged over all samples in a batch called N.

2.3.1.4 Optimization

As the optimizer for both the generator and the discriminator Adam is used. Adam is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order derivatives, first introduced by Kingma & Ba in 2017 [20]. According to the original paper, it is computationally efficient, has little memory requirements, invariant to diagonal rescaling of gradients, and is well suited for problems that are large in terms of data and parameters.

2.4 Diffusion Models

Diffusion models are generative models that learn to create data by gradually removing noise from an initial random distribution. They are named after the diffusion process, a thermodynamic concept where particles spread from high to low concentration. In machine learning, this translates to iteratively corrupting training data with noise during the forward process and training a neural network to reverse this corruption, called the reverse process. Diffusion models can generating high-quality images due to their ability to model complex distributions.

2.4.1 U-NET

The U-NET architecture, introduced by Ronneberger et al. in 2015 [21], was originally designed for biomedical image segmentation. Its symmetric encoder-decoder structure with skip connections, shown in figure 10, enables precise localization while retaining contextual information, making it ideal for tasks requiring pixel-wise predictions.

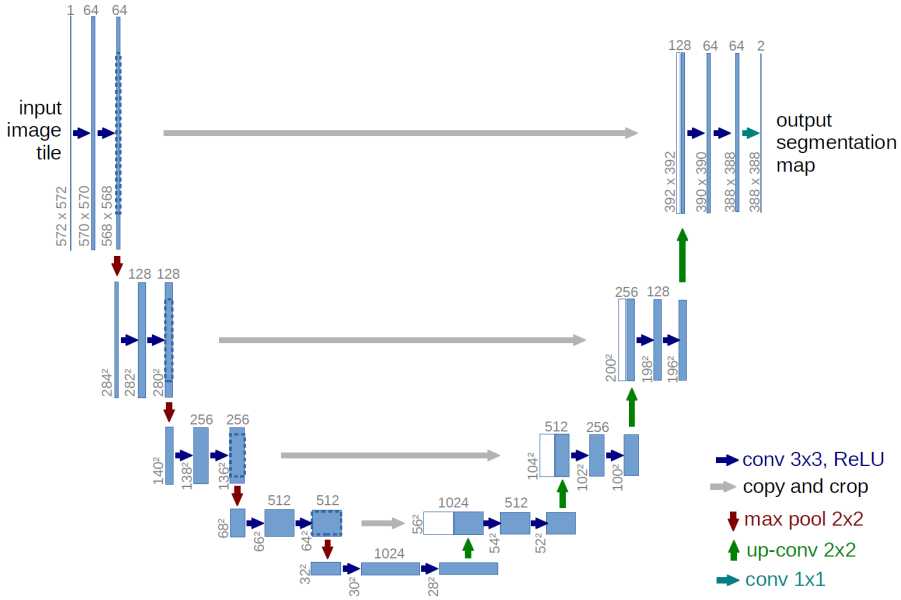


Figure 10: U-NET architecture as proposed by Ronneberger et al. in 2018 [21]. Each blue box corresponds to a tensor with the number of channels denoted on top of the box. The width and height of the tensor is provided on the lower left side of the box. White boxes represent tensors copied by using skip connections. The arrows denote the different operations.

The exact architecture described in figure 10 is used in this project.

2.4.1.1 Encoder

The encoder compresses input images into lower-dimensional latent representations through hierarchical feature extraction. It is build in similar fashion to the encoder in the pix2pix model described in section 2.3.1.1. The main difference is that this U-NET uses two consecutive convolutional layers (e.g., $3x3$ kernels) with activation functions (e.g., ReLU) and batch normalization. Instead of using leaky ReLU this implementation uses regular ReLU activations. Also each of the down blocks utilize batch normalization. Max pooling (typically $2x2$) is used to down samples feature maps after each down block, reducing spatial dimensions while increasing channel depth.

2.4.1.2 Bottleneck

The bottleneck sits between the encoder and decoder, containing the most compressed representation of the input. This layer acts as an information bottleneck, forcing the network to prioritize essential features. In other applications, bottlenecks can reduce computational complexity or enable latent space manipulations.

2.4.1.3 Decoder

The decoder reconstructs the original resolution from the latent representation using transposed 2D convolutions for up sampling. Each decoder block combines up sampled features with skip connections from the encoder via concatenation, followed by double 2D convolution block. A final $1x1$ convolution maps channels to the desired output dimensions (e.g., RGB channels).

2.4.1.4 Noise Scheduler

In 2020 Ho et al. [22] proposed the concept of Denoising Diffusion Probabilistic Model (DDPM). A class of latent variable models inspired by considerations from non-equilibrium thermodynamics, shown in figure 11.

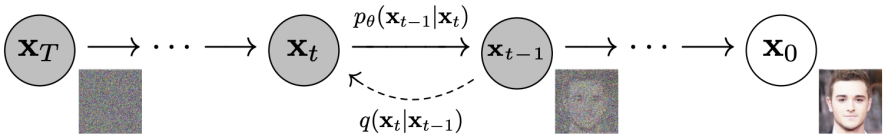


Figure 11: Overview of the denoising process proposed by Ho et al. in 2020 [22]. During the forward process noise is added gradually for each time step, going from right to left. During generation the noise is subtracted for each time step.

DDPM operate through two phases: a forward diffusion process and a learned reverse process. During the forward phase, input data is incrementally corrupted by adding Gaussian noise over multiple time steps until it becomes pure noise. The reverse process trains a neural network, in this project a U-NET, to predict and subtract the noise at each step, effectively reconstructing the original data from randomness. In this project the original DDPM proposed by Ho et al. [22] is used. During training and generation 1'000 times steps are typically used.

2.4.1.5 Loss Function

The loss function is Mean Squared Error (MSE) between the predicted and actual noise:

$$\mathcal{L} = \mathbb{E}_{\mathbf{x}_0,\epsilon,t}\left[\|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|^2\right],$$

where ϵ is ground-truth noise and ϵ_θ is the model's prediction. Pixel-wise MSE ensures fine-grained fidelity, critical for image generation.

2.5 Masked Auto Encoders

The Masked Auto Encoder (MAE), proposed by He et al. in 2021 [9], is a self-supervised learning framework for visual representation learning.

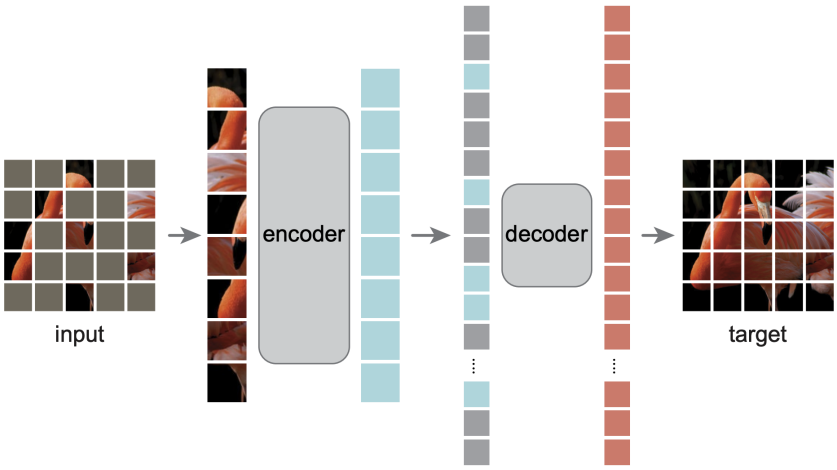


Figure 12: Overview showing the pre-training of the MAE architecture proposed by He et al. in 2021 [9]. A large random subset of image patches is masked out. Then the visible patches are given to the encoder which produces the embeddings shown as blue boxes. To the other patches mask tokens are assigned, depicted as the gray boxes. Then a small decoder reconstructs all patches.

MAE is a useful architecture to pre-train vision models. Figure 12 gives an overview of how the pre-training is supposed to work. Inspired by masked language modeling in natural language processing, MAE randomly masks patches of an input image and trains an encoder to produce high quality embeddings from the visible patches. These can be used by a small decoder to reconstruct the input image.

2.5.0.1 Encoder

In this project the encoder part of the MAE was chosen to be a Vision Transformer (ViT). Introduced by Dosovitskiy et al. in 2020 [23] ("An Image is Worth 16x16 Words"), adapt the transformer architecture—originally designed for Natural Language Processing (NLP), to process images. ViT are commonly used as encoders because they utilize self attention to capture long-range dependencies between image patches, enabling global context modeling from the start.

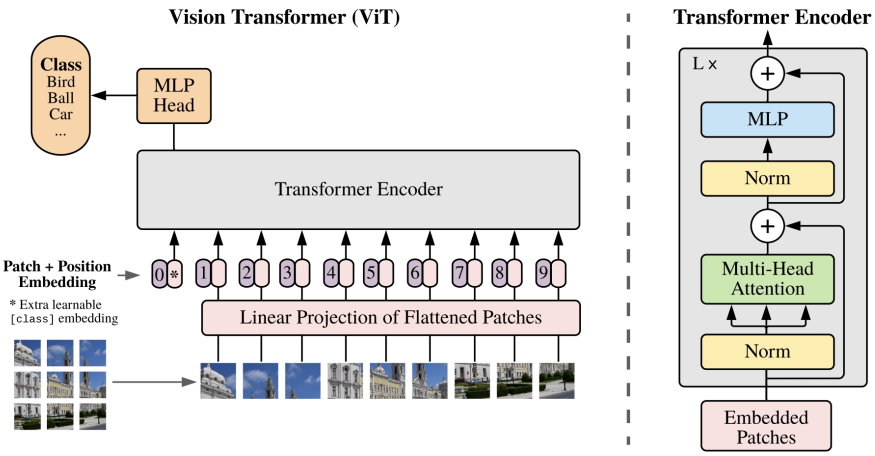


Figure 13: Overview of how a ViT is build. On the bottom left the input image is depecited, this is split into patches where a linear projection is applied as well as a patch and positional embedding before sending it to a traditional transformer encoder.

Figure 13 show the core building blocks include patch embeddings (splitting images into fixed-size patches, linearly projected into tokens), positional embeddings (learned 1D vectors added to patches to encode spatial relationships), and transformer blocks (multi-head self-attention + feed-forward networks). Positional embeddings compensate for the transformers lack of inherent spatial awareness by preserving patch order.

Each transformer block in ViT processes tokens via multi-head self-attention, where each patch dynamically attends to others to compute relevance-weighted feature combinations, followed by a feed-forward network, with layer normalization and residual connections for stability. Multi-head self-attention is powerful because it allows the model to contextualize information globally, weighing patch interactions based on content, which enhances feature representation for tasks like classification.

In a transformer block, multi-head attention is computed as follows:

$$\text{Multi-Head}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = [\text{head}_1, \ldots, \text{head}_h]\mathbf{W}^O,$$

Where each attention head is defined as:

$$\text{head}_i = \text{Attention}\left(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V\right),$$

Here, Q, K, V are the query, key, and value matrices, and $Attention(QW_i^Q, KW_i^K, VW_i^V)$ are learnable projection weights for the i-th head. The attention function is:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}.$$

Where d_k is the dimension of keys. The final projection W^O maps concatenated outputs back to the original dimension.

Multi-head attention is a mechanism in transformer architectures that enables the model to jointly focus on information from different representation subspaces at multiple positions. Instead of computing attention once, the input queries (Q), keys (K), and values (V) are linearly projected h times (for h heads) into distinct lower-dimensional spaces. Each head independently computes scaled dot-product attention, which measures pairwise similarity between queries and keys, scales by $\sqrt{d_k}$ to stabilize gradients, and uses the softmax to generate weights for aggregating values. By concatenating all heads and projecting with W^O, the model integrates diverse attention patterns (e.g., local vs. global, syntactic vs. semantic). This parallelized design enhances expressiveness, allowing the transformer to capture complex dependencies in sequences while maintaining computational efficiency.

2.5.0.2 Decoder

The decoder receives the encoder output, the latent features of the unmasked patches, the mask tokens, and the positional embeddings for all patches to retain the spatial information. In the proposed MAE the decoder is only used during pre-training, meaning that for downstream tasks such as segmentation or classification it is discarded. However in this project the decoder plays a vital role as it is the key component to generate aerial images. Despite the importance the simple implementation proposed by Dosovitskiy et al. is used.

It consists of a linear layer to expand the latent features, a series of standard transformer block with multi-head self attention and feed-forward networks. And finally a layer normalization and an output projection to map the output to pixel values.

3 Results

The project is split into three main work packages: *(1) create a dataset* depending on the specific data requirements; *(2) model development* includes implementation and testing of different models; *(3) additional tooling such* as LULC manipulation, rendering pipeline, etc.

3.1 Data Loader

An automated pipeline was developed to concurrently retrieve 2 meter resolution SWISSIMAGE tiles (Section 2.2.1) and corresponding swissALTI3D elevation map (Section 2.2.2) via the Swiss Federal Geoportal's web services[1]. The down scaled LULC maps from Giuliani et al. [16] were accessed via the University of Geneva's public repository[2].

For each SWISSIMAGE tile, processing included:

- Extraction of map projections and tile boundaries (lower-left/upper-right coordinates)
- Spatial alignment of corresponding LULC segmentation masks and elevation values
- Upscale the LULC as well as the elevation map to match the base tile size
- Saving the extracted segmentation and elevation maps to new files

This generated three outputs per tile: (1) aerial imagery, (2) LULC segmentation, and (3) elevation data. The dataset is then organized as:

```
/dataset
    /base
        /2504_1159.png
        /{east}_{north}.png
    /lulc
        /2504_1159.png
        /{east}_{north}.png
    /alti
        /2504_1159.png
        /{east}_{north}.png
```

This folder structure can then easily be utilized using the PyTorch Datasets and DataLoader classes[3].

For this purpose, a custom class inheriting from PyTorch's `Dataset` was implemented as shown in the code example 1.

[1] https://www.geo.admin.ch/
[2] https://yareta.unige.ch/archives/6ab4b715-904f-4cb9-961c-6a25b4c1116b
[3] https://pytorch.org/tutorials/beginner/basics/data$_t$*utorial.html*

Listing 1: Example implementation showcasing how to create a custom PyTorch dataset.

```
class SwissImageDataset(Dataset):
    def __init__(self, cfg, directory, transform=None):
        self.cfg = cfg
        self.directory = directory
        self.transform = transform
        self.image_names = sorted(os.listdir(directory))

    def __len__(self):
        return len(self.image_names)

    def __getitem__(self, index):
        # Load Base Tile
        # Load Segmentation
        # Load Elevation

        # Apply Transformation

        return {
            'base': base_tile,
            'lulc': lulc_tile,
            'alti': alti_tile
        }
```

For most of the experiments the following transformations were applied to the images.

- Resizing depending on the expected input shape of the model
- Converting the image to a PyTorch Tensor
- Normalizing the tensor using the mean and standard deviation

3.2 pix2pix

In this preliminary work, a pix2pix conditional Generative Adversarial Network (GAN) was implemented to generate aerial imagery from LULC maps, following the TensorFlow foundational tutorial[4]. The script was adapted to the dataset comprising of 400 base aerial images with each having the corresponding LULC segmentation. The images were resized to 256×256 pixels as suggested by the reference implementation. The generator architecture is a U-NET structure with symmetric eight down sampling and up sampling blocks, as well as skip connections to preserve fine-grained spatial details. The LULC segmentation is used as the input to the generator model. Training was conducted over 40 epochs with each 1000 steps. Binary Cross Entropy (BCE) loss was used to optimize the generator's ability to minimize discrepancies between generated outputs and real aerial imagery. The discriminator was trained independently to evaluate the plausibility of generated images against the base aerial photographs.

[4]https://www.tensorflow.org/tutorials/generative/pix2pix

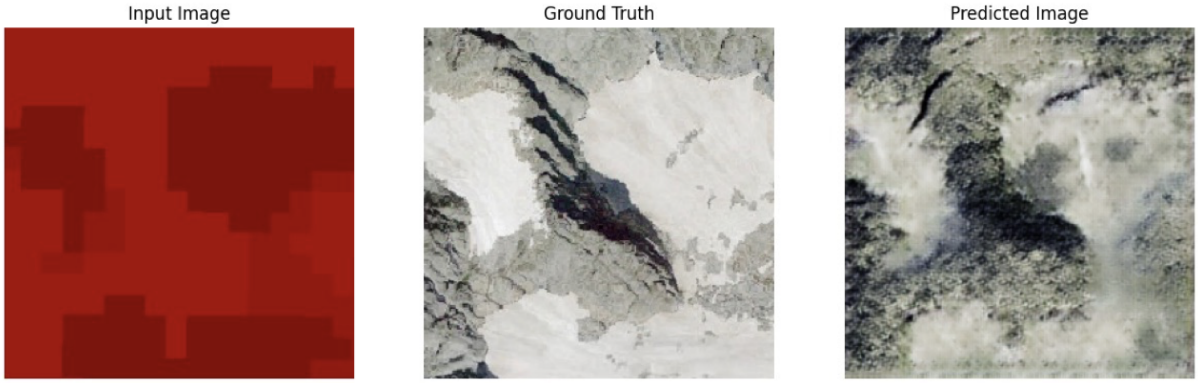


Figure 14: A pix2pix model trained for 40 epochs using 400 samples on the down scaled LULC classes by Giuliani et al. [16] with a resolution of 25 m and the orthophoto aerial images from SWISSIMAGE [3] with a resolution of 10 cm used as the ground truth during training.

Figure 14 shows the model output after 40 training epochs. In the generated image the segmentation can be recognized indicating that the conditioning works. Interestingly the shadow of the peak in the center of the base image was also generated indicating overfitting.

Figure 15: Without any method to continuously synthesizing the aerial images the predicted image borders don't align and the grid of input images is shown in the final prediction.

Figure 15 highlights the problem that each tile is generated individual and that when aligning the tiles the border between them is easily recognizable. It also shows that the generation seems to be deterministic indicating that the model is overfitted on the specific class.

Preliminary results demonstrate the feasibility of generating aerial images conditioned on LULC maps using the pix2pix framework. However, challenges such as overfitting and inconsistencies in tile alignment require further investigation. To enhance the output, future work should focus on optimizing the U-NET architecture potentially refining skip connections or layer configurations—and exploring alternative conditioning mechanisms to improve coherence in generated imagery.

3.3 U-NET

Building on preliminary results demonstrating the feasibility of LULC conditioned aerial image synthesis. A simplified U-NET architecture to address overfitting and data efficiency was explored. The idea is to reduce the reliance on larger datasets by omitting the discriminator model.

Inputs were restructured to combine the original aerial RGB image (3 channels) with the LULC segmentation map (1 channel) as a 4-channel composite, enhancing spatial context while retaining the LULC segmentation. Multiple implementations were tested including an implementation from scratch. Due to the poor results it was decided to use the reference implementation from HuggingFaces Diffusers Library which offers the UNet2DModel class[5].

The implementation uses six down blocks with one of them being an attention down block that will suppress activations in irrelevant regions at the skip connections in an attempt to reduce the number of redundant features brought across. Then six up blocks are used for the up sampling with one of them being an attention up block.

A series of experiments was conducted with a combination of the following hyper parameters:

- Number of Epochs: 1, 10, 20, 50, 100, 140, 200
- Sample Size: 1, 10, 100, 500, 1'000, 3'000, 10'000

Unfortunately, due to unknown reasons most of the experiments resulted in poorly generated outputs unusable to solve the task of generating aerial images from LULC segmentation. This is most likely due to a mistake in the implementation of the training pipeline, model usage or evaluation method. A mistake in the external dependencies used such as the Diffusers or PyTorch library can be excluded as proven results from these tools exist.

For three of the experiments the results are shown and an attempt at explaining them has been done.

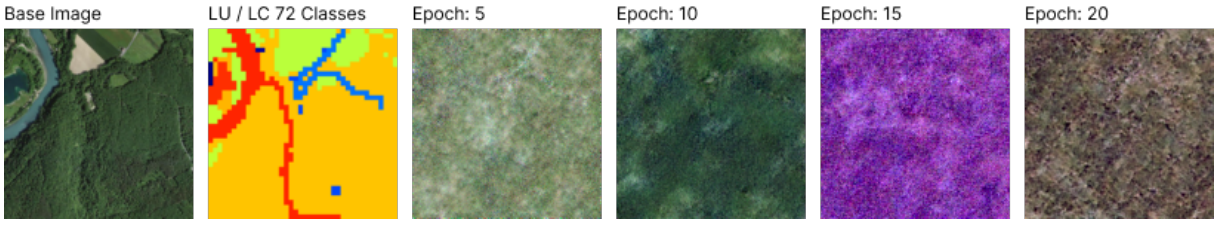


Figure 16: Training results of a U-NET after every 5th epoch. Trained for 20 epochs on 500 sample images, each generation used 1000 time steps in the diffusion process. Base image and segmentation during training into a 4 channel image.

One of the first experiments yielding somewhat promising results is shown in figure 16. A UNet2DModel trained for 20 epochs on a subsample of 500 images. Each containing four channels, three for the visible colors, and one for the LULC segmentation. After the 5th epoch it looks like a grassland texture with

[5] https://huggingface.co/docs/diffusers/main/en/api/models/unet2d

few linear features. On the subsequent epochs the texture stays similar but the color hue is changing. No feature from the LULC segmentation can be found in any of the generated images.

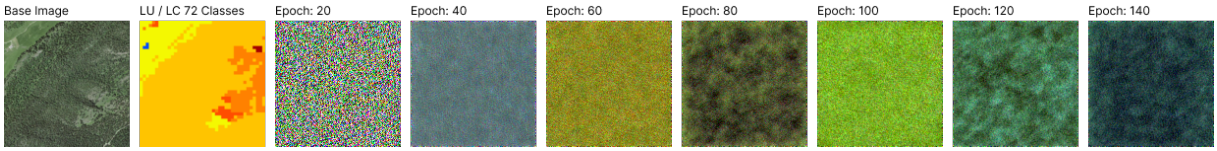


Figure 17: Training results of a U-NET after every 20th epoch. Trained for 140 epochs on 1000 sample images, each generation used 1000 time steps in the diffusion process. Base image and segmentation during training into a 4 channel image.

In a later attempt, more images were used and training was extended to 140 epochs. The same model implementation and configuration was used as explained in the previous experiment. The results shown in figure 17 reveal that it took longer to reach a state where some textures are visible. It also shows that the color hue changes between different epochs. Still, no feature from the LULC can be found in the generated images. Indicating that the conditioning might not work by adding the segmentation as a separate channel to the input image.

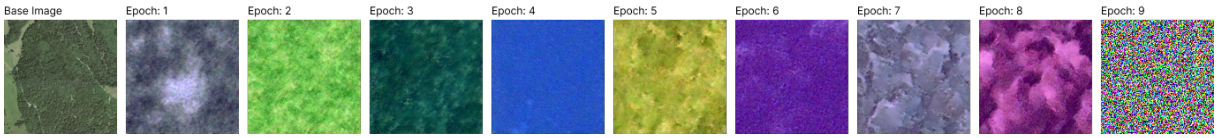


Figure 18: Training results of a U-NET after every epoch. Trained for 10 epochs on 3000 sample images, each generation used 1000 time steps in the diffusion process. No conditioning was applied during the training.

To see if the training would work without the LULC segmentation it was omitted. A new model was trained for 10 epochs on 3'000 images and on each epoch an image was generated. The changing color hue is still present in the outputs. Already on the first epoch some textures were visible and the textures changed throughout the different epochs. Training seemed to collapse after the 8th epoch. The reason for this is not known.

To conclude, it was not possible to create any usable results by choosing a very simple U-NET implementation. The reason for this is not known. Conditioning by applying the LULC segmentation as a separate channels does not seem to work with this implementation. Thus further investigating is required to find a way to include the LULC segmentation as a condition to the image generation process.

3.4 Conditional U-NET

In an attempt to improve the conditioning mechanism an enhanced version of the original U-NET implementation was used. The UNET2DConditionModel[6] from the Diffusers library provides an additional method to conditioning mechanism by providing an embedded version of the conditional image. The

[6]https://huggingface.co/docs/diffusers/en/api/models/unet2d-cond

LULC segmentation is embedded using a trainable embedding layer. The embedding is trained at the same time as the model. In the forward pass the embedding is received and passed to each down sample step.

The architecture stays mostly the same with six down blocks with one of them being an attention down block and six up sampling blocks with one of them being an attention up block.

Again a series of experiments was conducted with a combination of the following hyper parameters:

- Number of Epochs: 1, 10, 20, 50, 100
- Sample Size: 500, 2'000, 20'000, 40'000

Similar to the results in section 3.3 the results from the experiments using the conditional U-NET yielded better but still unusable results. In this section a selection of experiments is examined.

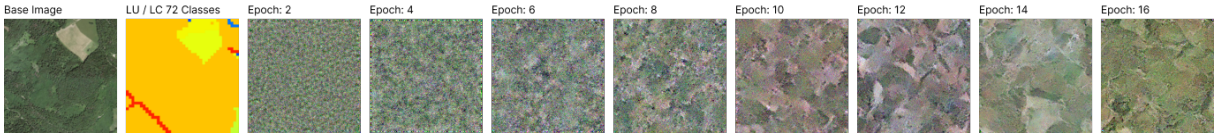


Figure 19: Results for a conditional U-NET Trained for 16 epochs on 2000 samples. The segmentation contains 72 classes.

In one of the first experiments the conditional was trained for 16 epochs on 2000 samples using the segmentation with the AS72 classes. Compared to the results in section 3.3, the changing color hue between the different epochs was reduced as shown in figure 19. Different textures were generated that could resemble agricultural lands, however the dominant segmentation class was forest.

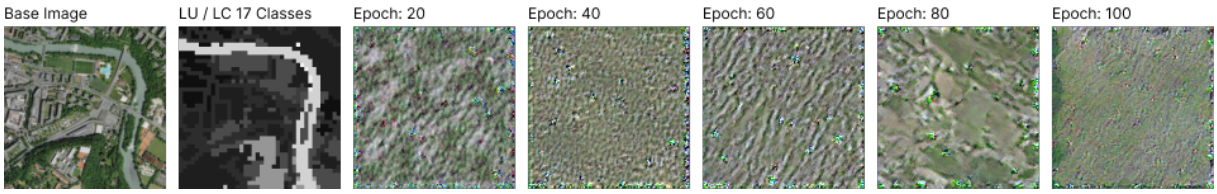


Figure 20: Results for a conditional U-NET Trained for 100 epochs on 500 samples. The segmentation contains 17 classes.

In an attempt to improve the results the number of classes was reduced to 17 (AS17) as explained in section 2.2.3. The model was trained for 100 epochs on 500 samples. The generated tiles in figure 20 reveal different textures and no chaning color hue. However on the 100th epoch not much of the textures were left. The generated images did not resemble the base aerial image and no features from the segmentation could be recognized.

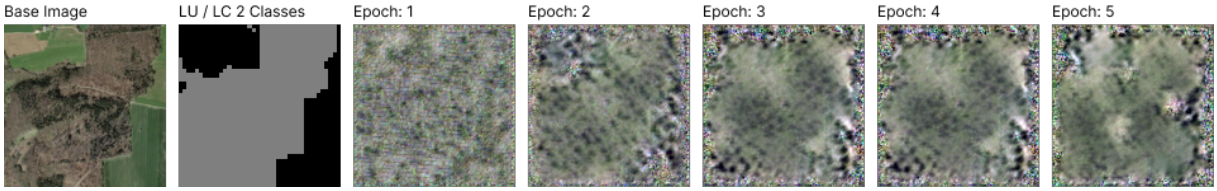


Figure 21: Results for a conditional U-NET Trained for 5 epochs on 40'000 samples. The segmentation contains 2 classes.

Further reducing the number of classes to 2 (Forrest and Not Forest) the generation yielded again different results with the segmentation being distinctly visible. The model shown in figure 21 was trained for 5 epochs on 40'000 samples. This indicates that the conditioning mechanism either requires fewer classes or needs to be trained differently to provide better embeddings. This could be achieved by training a separate Vision Transformer model to produce the embeddings.

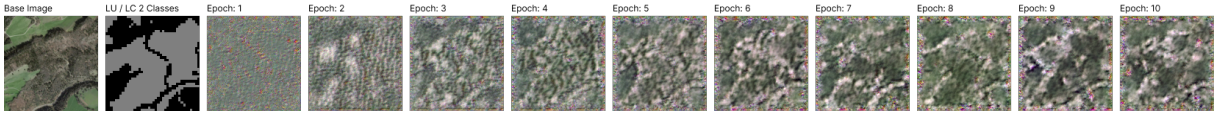


Figure 22: Results for a conditional U-NET Trained for 10 epochs on 20'000 samples. The segmentation contains 2 classes.

In a last attempt the number of epochs were raised to 10 while the number of samples was reduced to 20'000 samples. Figure 22 shows that the segmentation can be recognized in the generated outputs but the colors resemble a rocky surface instead of the desired forest.

To conclude, the results using a conditional U-NET improved, but are not yet usable for this project. Significant time and effort has been invested in attempts to produce usable results with no clear path forward being found. Suggestions for future attempts include:

- Use larger sample size
- Use few epochs (1 to 10)
- Evaluate after each epoch to quickly identify errors or training collapse
- Use fewer classes and select the samples based on if the class is contained in it

For the sake of this project, a new method had to be found.

3.5 ViTMAE

To evaluate the MAE approach an implementation by Hugginfaces transformers library was used that utilizes Vision Transformer (ViT) called VitMAEForPretraining[7]. The use of this implementation ensuring architectural reliability by leveraging existing components. Mirroring the U-NET approach (Section

[7] https://github.com/huggingface/transformers/blob/main/docs/source/en/model$_doc/vit_mae.md$

3.3), segmentation maps were concatenated as an additional input channel to RGB aerial imagery for conditioning. Experiments tested scalability across different sample sizes (10,000–30,000) using the 17-class AS17 LULC classes. An overfitting experiment was conducted to validate that the implementation of the model can learn something.

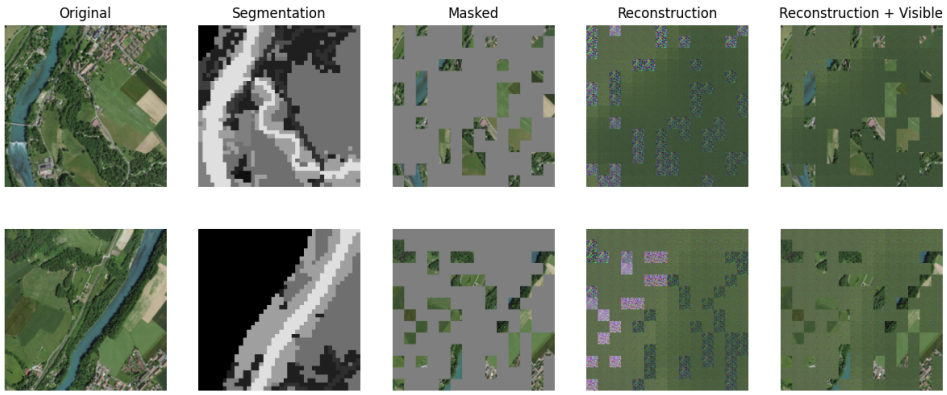


Figure 23: Evaluation output showing the original aerial image, the corresponding LULC segmentation. The masked model input, the models reconstruction as well as a merged version of the non masked parts and the reconstruction.

The output file created during training after each evaluation is shown in figure 23. In the first two columns the original aerial image is shown with the corresponding LULC segmentation. In the center the masked image is shown which is given as an input to the ViTMAE model. The next column shows the raw reconstruction from the model output and in the last column the patches that are not masked are merged together with the reconstruction.

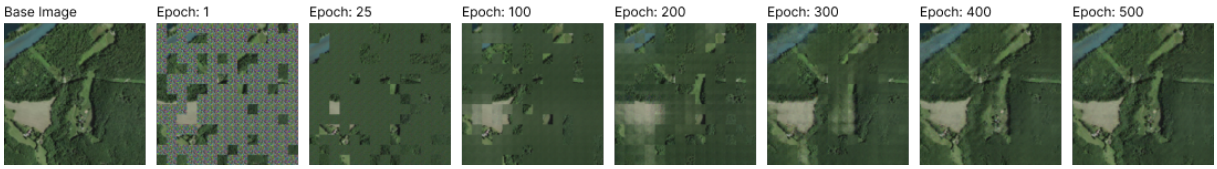


Figure 24: Overfitting test with one sample to validate the training and model implementation. First image shows the base aerial image that is to be reconstructed. Within 500 epochs the model is overfitted to reconstruct the image exactly. No conditioning was applied during this training.

In order to verify the implementation of the model and the surrounding training pipeline a model was trained on a single sample for 1'000 epochs. After 500 epochs it was able to almost exactly reconstruct the input image as shown in figure 24.

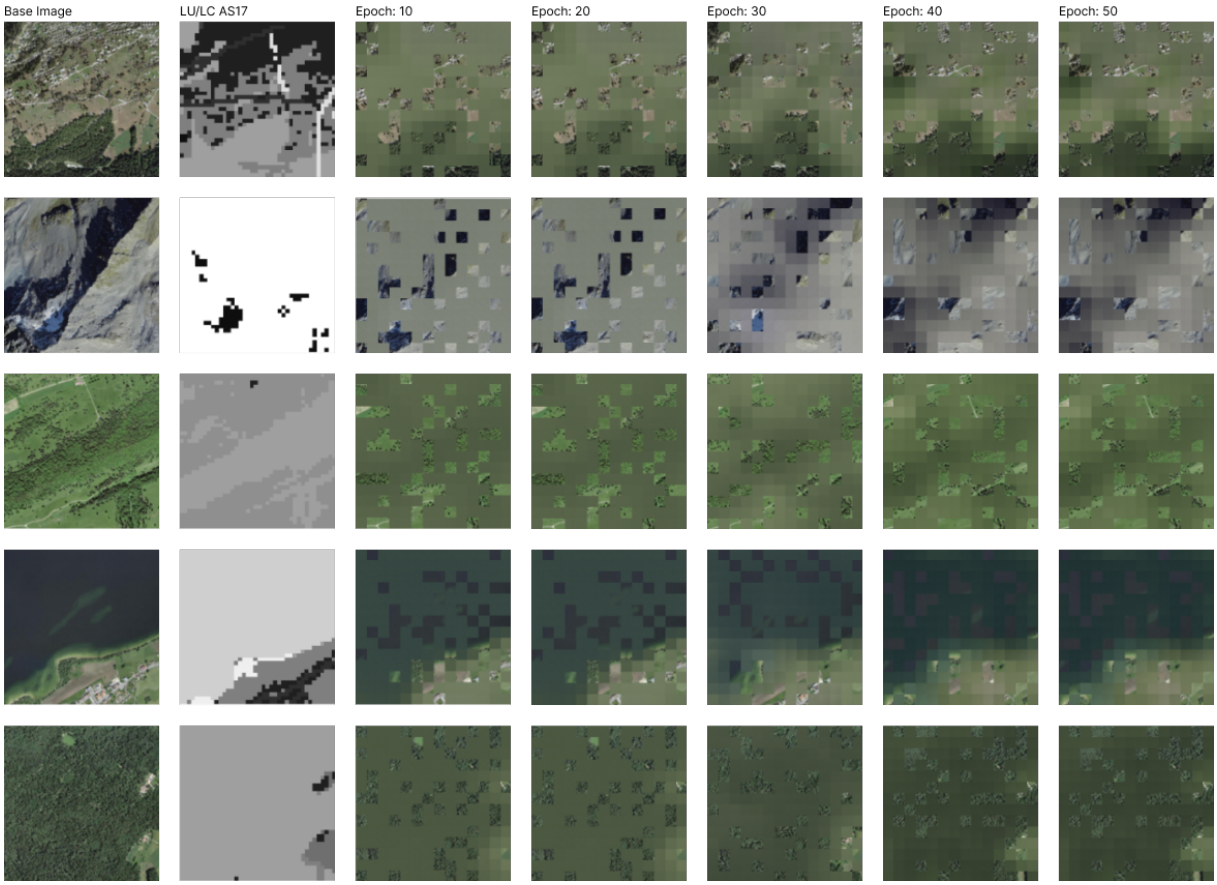
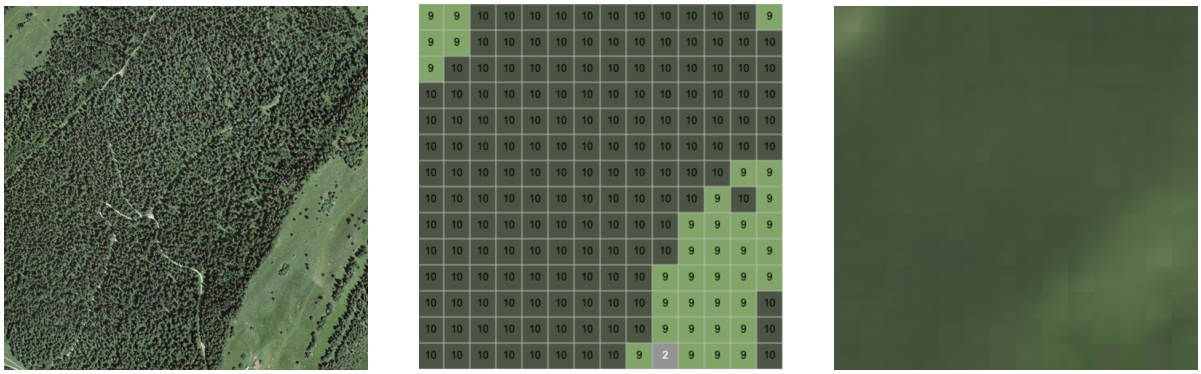


Figure 25: Training on 50 epochs with the LULC segmentation AS17 with 10'000 samples. The segmentation is added as an additional channel to the base image.

Scaling it up to a larger sample size of 10'000 and training for 50 epochs the results reveal that conditioning the model seems to work and that the results (Figure 25, although seemingly pixilated, resemble input images closely. Important to know is that no other hyper parameters were adjusted besides the sample size and training epochs. This indicates that the model could scale very well. From this experiment it becomes clear that reconstructing populated areas with building and infrastructure is very difficult if the model cannot be trained for a long time due to the low resolution output. The conducted experiment took about 10 hours while utilizing 32 GB of memory on a single NVIDIA Tesla V100 GPU[8].

Training on 30'000 samples for 1000 epochs resulted in the best performing model of this project. This took approximately 120 GPU hours to train. The results from this model are examined in figure 26 and 27.

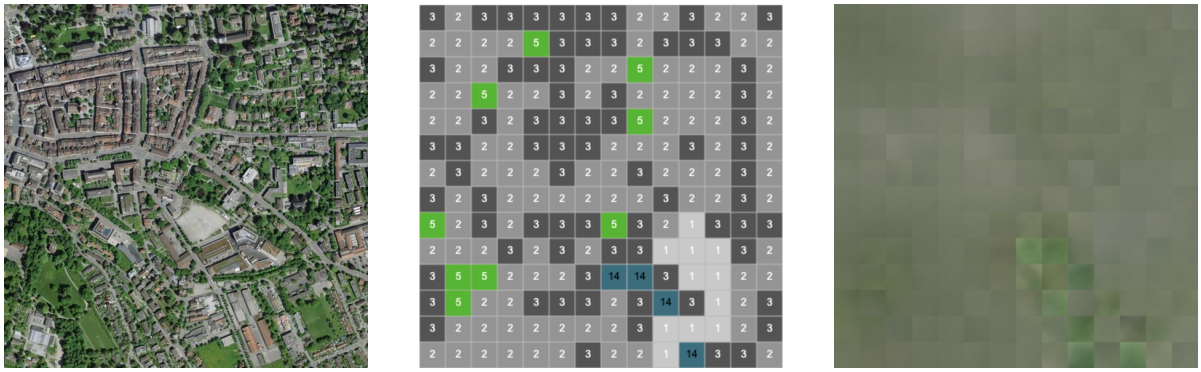
[8]https://www.nvidia.com/en-gb/data-center/tesla-v100/



(a) Original (b) Land use / Land cover Segmentation (c) Reconstructed

Figure 26: Comparison between the original aerial image, the LULC segmentation, as well as the reconstructed image.

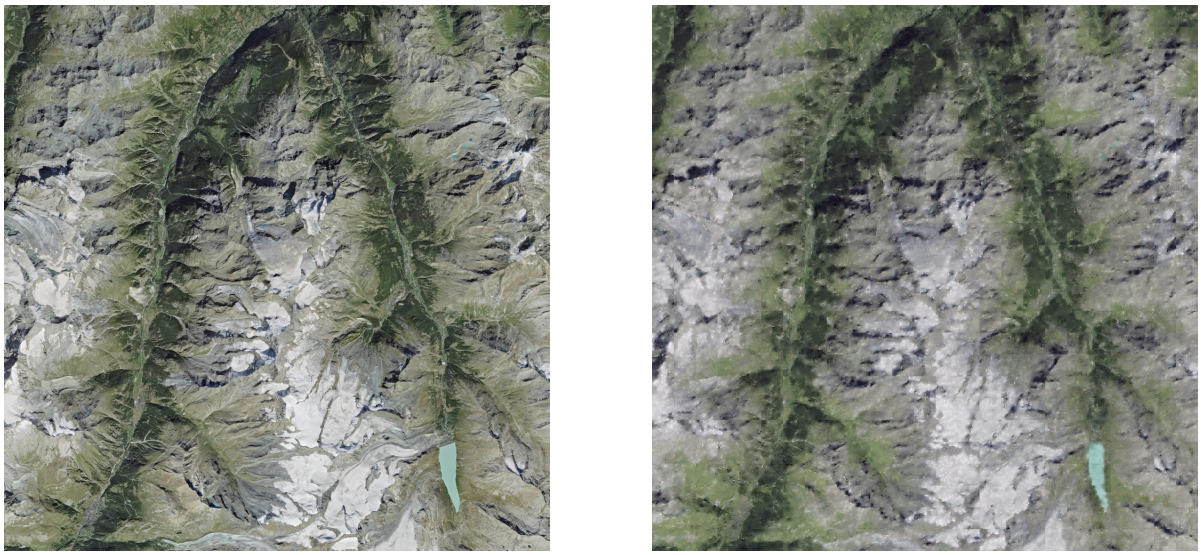
Aerial image showing forest in the municipality of Bassins located in the canton of Vaud. Most of the LULC is classified as forest with some region on the top left and bottom right being alpine agricultural area. Within the reconstructed image the different classes are visible and the color is somewhat matching. However the resolution is quite low indicating that the model is not sufficiently trained. The evaluation on this image is also favorable since it contains mostly the class, forest, which is represented the most in the dataset.



(a) Original (b) Land use / Land cover Segmentation (c) Reconstructed

Figure 27: Comparison between the original aerial image, the LULC segmentation, as well as the reconstructed image.

Examining results that include under represented classes in figure 27 highlights the limitations of the current model. The reconstruction clearly lacks detail and the color is not matching very well. Further training is necessary to perform better on these classes.

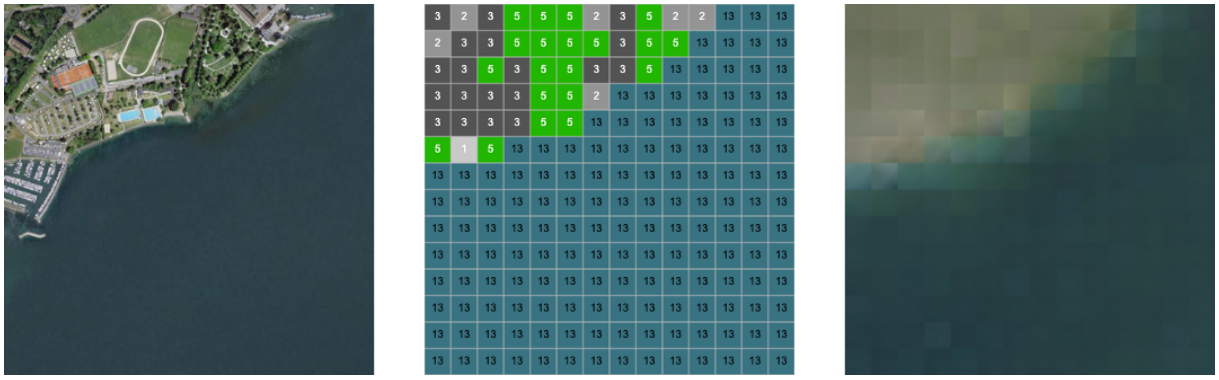


(a) Original (b) Reconstructed

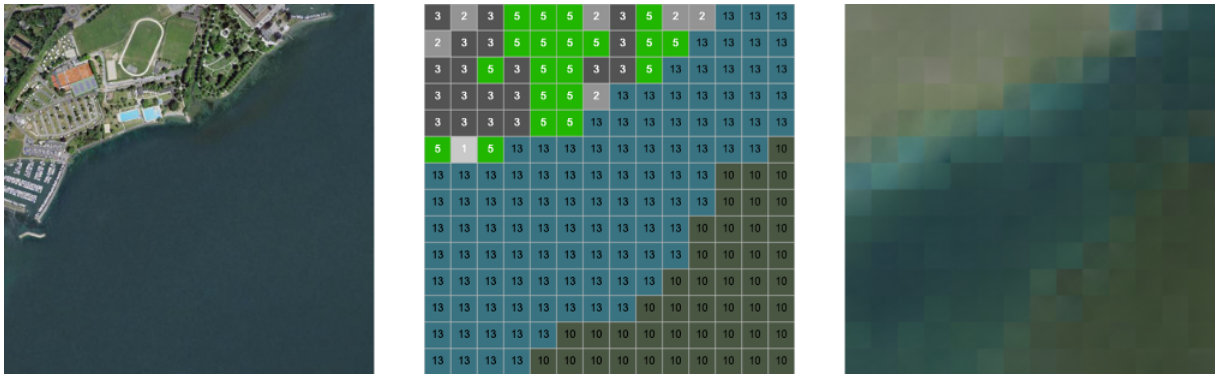
Figure 28: Image comparison of the original and reconstructed area of the Matter- and Saastal. The size of the area is about $25km^2$ and consists of 625 individual tiles.

By reconstructing a larger area the effect of low resolution is attempted to be mitigated. Figure 28 shows a large $25km^s$ area of the Matter- and Saastal. In the reconstructed image the contours of the mountainous area visible as well as the forested area. The Mattmark reservoir is also clearly visible. This highlights that the model is able to reconstruct images from the LULC segmentation, albeit at a very low resolution.

The reconstruction is only one part in achieving the results desired for this project. More importantly the model should generate new images from update LULC segmentation.



(a) Original Land use / Land Cover



(b) Adjusted Land use / Land Cover

Figure 29: Generation with the original LULC compared to the generation with adjusted LULC.

In figure 29 the reconstructed image is compared to the generated image after adjusting the LULC. In the adjusted segmentation the lower right of the image was changed to contain forest. This example shows that the model can generate new aerial images from the updated segmentation.

Overall this approach shows promising results, being able to generate aerial images from LULC maps. However the low resolution prevents the generated images from being used. Only when generating for a very large area are the results somewhat usable. Further training is required to improve the generated output. Additionally, it should be investigated how to quantitatively measure the performance for certain LULC classes since it is known that the classes are not equally balanced.

3.6 Tooling

During this project a few different tools have been developed to:

- Adjust the LULC maps by loading the segmentation of a location, selecting a new class, and drawing on the map to change the class assignment

- A web service that exposes the generation of the model through an API to generate new images after adjusting the LULC maps
- A tool to combine the newly generated images and import them into Blender[9] in order to project them onto an existing elevation map and create 3D scenes.

With these tools new aerial images can be generated from updated LULC data and showcased in a 3D scene.

3.6.1 Adjust Land use / Land cover

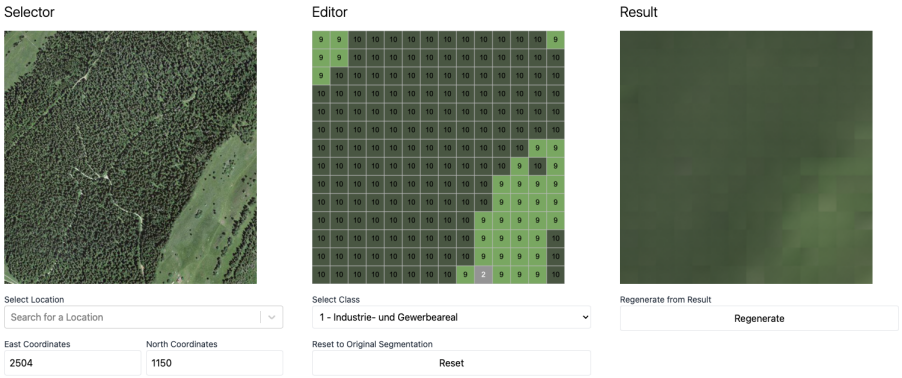


Figure 30: Overview of the LULC editor. The left side shows the aerial image from a selected location. In the center the editor is placed and on the left is a preview of the generated output using the LULC segmentation from the editor.

Figure 30 shows the interface of the LULC editor. In a first step the user can select a location to load the aerial image and the current LULC maps. Then the user can select a new segmentation class and draw in the map. While drawing the model generates updated images from the new LULC map.

[9] https://www.blender.org/

Figure 31: Process overview of the location selection. First the user can search for a place, street or city in the search bar. Once the item that is searched for has been selected the tool calculates the LV95 coordinates and loads the aerial image as well as the LULC segmentation. Once these are loaded the output image is generated.

If the coordinates of a tile are not known or a location is to be selected by its address or name the user can simply search into location selection search bar. While typing possible locations from the search inputs are continuously loaded from the Mapbox API[10]. When a location is found and clicked on the tool calculates the LV95 coordinates which are used to load the correct aerial image and the corresponding LULC map. Once the aerial image and the segmentation are loaded the model generates an an image to show the users the potential output.

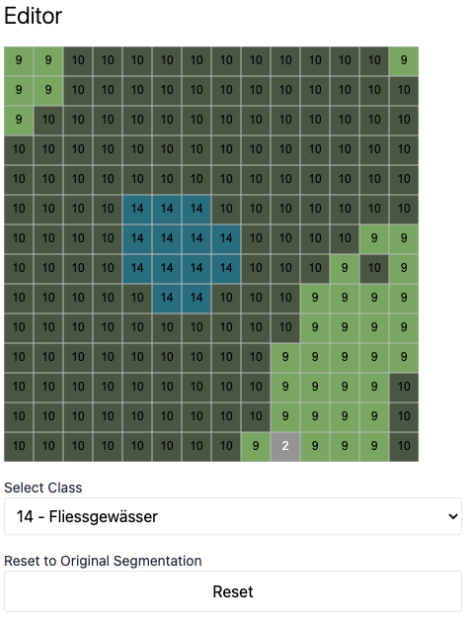


Figure 32: LULC editor. By clicking and dragging the mouse across the different tiles the user can draw the selected class. A reset button sets the classes to the original segmentation map.

[10]https://docs.mapbox.com/api/overview/

When editing the LULC map the user first has to select the new class that is to be drawing. Then by clicking and dragging the mouse across the tiles the crossed tiles are set to the new class. Once the mouse button is lifted the updated segmentation is saved and a new aerial image is generated.

3.6.2 3D scene creation

To combine the newly generated images and create a 3D scene, similar to the one in figure 33, the individually generated tiles first have to be stitched together. This works by including the east and north coordinates into the generated file names. From the location of the tiles the current elevation maps is loaded and stitched together as well. Then a new blender file is created and a separate script is loaded.

Within blender the following steps are executed:

1. Remove all objects from the scene (Lights, Camera, etc.)
2. Add a plane
3. Add the generated image as a texture
4. Split the plane into numerous subsection (necessary to apply the displacement)
5. Apply the elevation map as a displacement modifier
6. Apply smoothing to the plane
7. Add sun object
8. Adjust the sun position to match the correct aximuth and elevation angles depending on the location and a given date time

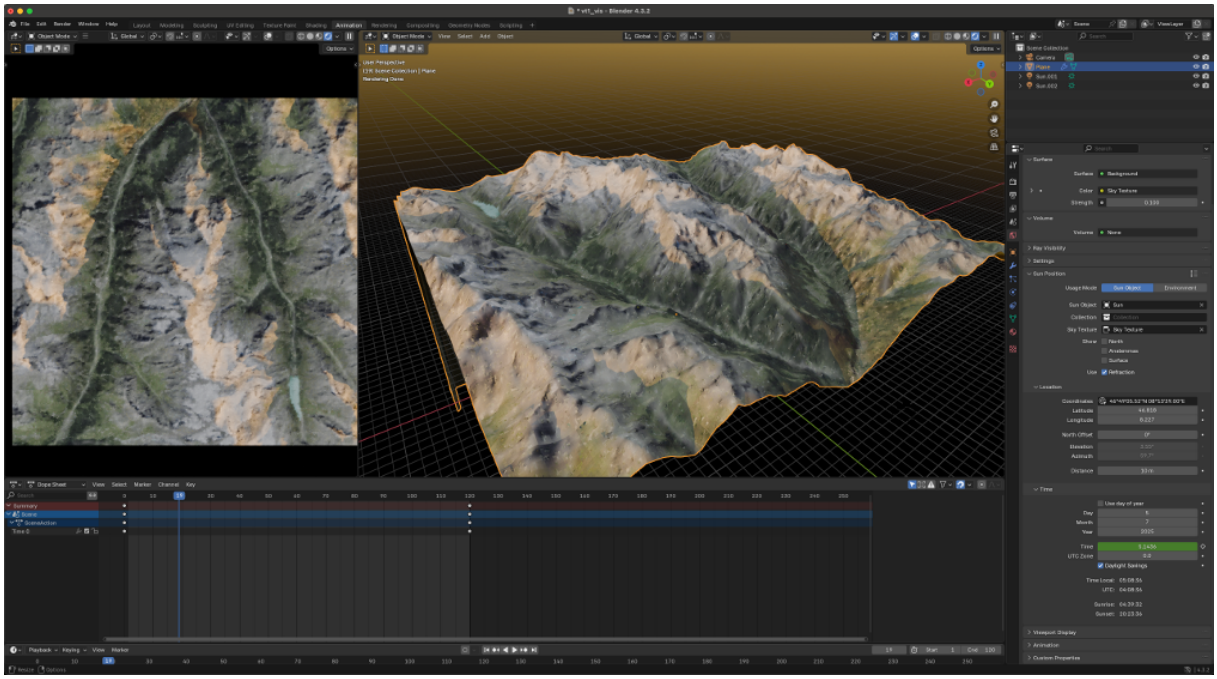


Figure 33: Screenshot of the generated tiles combined and projected onto the elevation surface. A global light source is added with the matching azimuth and elevation angle depending on the location of the tiles and a set date and time.

4 Discussion

Aerial image synthesis from LULC maps using a MAE has shown promising result. However the low resolution makes the generated images unsuitable for most usages except when generating images for a larger area. Generating images on LULC classes that are underrepresented is unusable at the time. Further training is required to improve the output of this model. It is also shown that MAE are scalable but require a lot of resources (e.g. sample size, compute, memory, etc.) to yield good results.

4.1 Future Work

To address the afore mentioned limitations future work should focus on:

- Increase dataset, by using the 10 cm resolution from the SWISSIMAGE dataset. The higher resolution tiles are 10'000 by 10'000 pixels which can be spitted into 1'521 smaller tiles. This will create roughly 65 Mio. image tiles to be used during training.
- Data augmentation to create a more balanced dataset. For example in cities the image tiles could be rotated or flipped to create additional image tiles.
- Find a way to quantitatively measure the performance for individual LULC classes. This could help identify low performing classes which could be used to efficiently augment the dataset
- Train with more memory to utilize larger batch sizes and reduce the number of epochs

Additionally preprocess the images to remove existing shadows using methods similar to the one proposed by Kwatra et al. [24]. Shadow region could be identified based on the elevation map as well as the location and date time the image was taken.

It might also be worth to invest more time into the conditional U-NET approach. Ideally this could help reduce the computational effort.

To make the final 3D scene more immersive a way of including settlements and infrastructure models should be investigated.

And finally, future models should be used to synthesis aerial images from forecasts such as the one provided by Bütikofer et al. [25] in order to underline the importance and usefulness of such a tool.

List of Figures

1 Frame out of the visualization by Julien Anet [2] showing the Gorner glacier region in a side by side comparison between today and 2060. 1

2 Overview of what the process pipeline could look like. It consists of a tool to adjust LULC maps, datasets, a model to synthesize aerial images and a post-processing tool to combine the generated images and the elevation maps into a 3D scene. 2

3 Aerial image from the SWISSIMAGE dataset showing parts of the Orbe river within the Vallée de Joux in the canton of Vaud. The image has a 2 meter resolution is 500 by 500 pixels large and depicts 1 square kilometer. 5

4 Elevation map from the swissALTI3D dataset showing parts of the Orbe river within the Vallée de Joux in the canton of Vaud the same as in figure 3. The image has a 30 cm resolution is 500 by 500 pixels large and depicts 1 square kilometer. 6

5 Distribution of LULC taken from the AS 2018 [15]. 7

6 Standard NOAS04 nomenclature showing the hierarchical structure of the LULC classes. . 8

7 LULC from the down scaled land use statistics dataset on the AS17 classes [16] showing parts of the Orbe river shown in red within the Vallée de Joux in the canton of Vaud the same as in figure 3. The image is 40 by 40 pixels large and depicts 1 square kilometer. . . 9

8 Overview of the training setup proposed by Goodfellow et al. [17]. Includes a generator G to produce images and a discriminator d to decide if an image is real or fake. The goal of the generator is to fool the discriminator. 10

9 Overview of the implementation based on pix2pix for for this project. The input of the generator is the LULC segmentation, the output is the generated aerial image. The generator tries to fool the discriminator to predict the generated image as real. 11

10 U-NET architecture as proposed by Ronneberger et al. in 2018 [21]. Each blue box corresponds to a tensor with the number of channels denoted on top of the box. The width and height of the tensor is provided on the lower left side of the box. White boxes represent tensors copied by using skip connections. The arrows denote the different operations. . . . 13

11 Overview of the denoising process proposed by Ho et al. in 2020 [22]. During the forward process noise is added gradually for each time step, going from right to left. During generation the noise is subtracted for each time step. 14

12 Overview showing the pre-training of the MAE architecture proposed by He et al. in 2021 [9]. A large random subset of image patches is masked out. Then the visible patches are given to the encoder which produces the embeddings shown as blue boxes. To the other patches mask tokens are assigned, depicted as the gray boxes. Then a small decoder reconstructs all patches. 15

13 Overview of how a ViT is build. On the bottom left the input image is depecited, this is split into patches where a linear projection is applied as well as a patch and positional embedding before sending it to a traditional transformer encoder. 16

14 A pix2pix model trained for 40 epochs using 400 samples on the down scaled LULC classes by Giuliani et al. [16] with a resolution of 25 m and the orthophoto aerial images from SWISSIMAGE [3] with a resolution of 10 cm used as the ground truth during training. . . 20

15 Without any method to continuously synthesizing the aerial images the predicted image borders don't align and the grid of input images is shown in the final prediction. 20

16 Training results of a U-NET after every 5th epoch. Trained for 20 epochs on 500 sample images, each generation used 1000 time steps in the diffusion process. Base image and segmentation during training into a 4 channel image. 21

17 Training results of a U-NET after every 20th epoch. Trained for 140 epochs on 1000 sample images, each generation used 1000 time steps in the diffusion process. Base image and segmentation during training into a 4 channel image. 22

18 Training results of a U-NET after every epoch. Trained for 10 epochs on 3000 sample images, each generation used 1000 time steps in the diffusion process. No conditioning was applied during the training. 22

19 Results for a conditional U-NET Trained for 16 epochs on 2000 samples. The segmentation contains 72 classes. 23

20 Results for a conditional U-NET Trained for 100 epochs on 500 samples. The segmentation contains 17 classes. 23

21 Results for a conditional U-NET Trained for 5 epochs on 40'000 samples. The segmentation contains 2 classes. 24

22 Results for a conditional U-NET Trained for 10 epochs on 20'000 samples. The segmentation contains 2 classes. 24

23 Evaluation output showing the original aerial image, the corresponding LULC segmentation. The masked model input, the models reconstruction as well as a merged version of the non masked parts and the reconstruction. 25

24 Overfitting test with one sample to validate the training and model implementation. First image shows the base aerial image that is to be reconstructed. Within 500 epochs the model is overfitted to reconstruct the image exactly. No conditioning was applied during this training. 25

25 Training on 50 epochs with the LULC segmentation AS17 with 10'000 samples. The segmentation is added as an additional channel to the base image. 26

26 Comparison between the original aerial image, the LULC segmentation, as well as the reconstructed image. 27

27 Comparison between the original aerial image, the LULC segmentation, as well as the reconstructed image. 27

28 Image comparison of the original and reconstructed area of the Matter- and Saastal. The size of the area is about $25km^2$ and consists of 625 individual tiles. 28

29 Generation with the original LULC compared to the generation with adjusted LULC. . . 29

30 Overview of the LULC editor. The left side shows the aerial image from a selected location. In the center the editor is placed and on the left is a preview of the generated output using the LULC segmentation from the editor. 30

31 Process overview of the location selection. First the user can search for a place, street or city in the search bar. Once the item that is searched for has been selected the tool calculates the LV95 coordinates and loads the aerial image as well as the LULC segmentation. Once these are loaded the output image is generated. 31

32 LULC editor. By clicking and dragging the mouse across the different tiles the user can draw the selected class. A reset button sets the classes to the original segmentation map. 31

33 Screenshot of the generated tiles combined and projected onto the elevation surface. A global light source is added with the matching azimuth and elevation angle depending on the location of the tiles and a set date and time. 33

Glossary

AS Arealstatistik. 6–8, 35

BCE Binary Cross Entropy. 12, 19

BFS Bundesamt für Statistik. 6

CRS Coordinate Reference System. 4

DDPM Denoising Diffusion Probabilistic Model. 14

GAN Generative Adversarial Network. i, 4, 9, 19

LiDAR Light Detection and Ranging. 5

LULC land use and land cover. i, 2–4, 6–11, 18–23, 25–32, 34–37

MAE Masked Auto Encoder. i, 2–4, 15, 17, 24, 34, 35

MSE Mean Squared Error. 14, 15

NLP Natural Language Processing. 15

pix2pix pix2pix. ii, iii, 2, 10, 11, 13, 19, 20, 35, 36

ReLU Rectified Linear Unit. 11, 13

SSSS Seamless Satellite-image Synthesis System. 2

U-NET U-Net. i–iii, 4, 10, 11, 13, 14, 19–24, 34–36

ViT Vision Transformer. 3, 4, 15, 16, 24, 36

ViTMAE Vision Transformer Masked Auto Encoder. iii, 24–28

WSL Swiss Federal Institute for Forest, Snow and Landscape Research. 1

References

[1] S. Tobias, E. G. Siegrist, L. Bütikofer, M. Bürgi, K. Liechti, E. Reynard, A. Guisan, D. Urbach, and C. Randin, "+4 °C und mehr: Schweizer Landschaften im Klimawandel," 2023.

[2] Julien Anet, "Gorner-Region: Evolution from today to 2060 - Side by Side," July 2023.

[3] "SWISSIMAGE 10 cm, digital orthophotomosaic of Switzerland - opendata.swiss."

[4] "swissALTI3D."

[5] M. Grab, "Swiss Glacier Thickness – Release 2020," 2020. Accepted: 2021-05-14T05:39:05Z Publisher: ETH Zurich.

[6] D. Farinotti, S. Usselmann, M. Huss, A. Bauder, and M. Funk, "Runoff evolution in the Swiss Alps: projections for selected high-alpine catchments based on ENSEMBLES scenarios," *Hydrological Processes*, vol. 26, no. 13, pp. 1909–1924, 2012. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/hyp.8276.

[7] J. Zhu and T. Kelly, "Seamless Satellite-image Synthesis," Nov. 2021. arXiv:2111.03384 [cs].

[8] B. Lütjens, B. Leshchinskiy, O. Boulais, F. Chishtie, N. Díaz-Rodríguez, M. Masson-Forsythe, A. Mata-Payerro, C. Requena-Mesa, A. Sankaranarayanan, A. Piña, Y. Gal, C. Raïssi, A. Lavin, and D. Newman, "Generating physically-consistent satellite imagery for climate visualizations," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 62, pp. 1–11, 2024.

[9] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked Autoencoders Are Scalable Vision Learners," Dec. 2021. arXiv:2111.06377 [cs].

[10] C. J. Reed, R. Gupta, S. Li, S. Brockman, C. Funk, B. Clipp, K. Keutzer, S. Candido, M. Uyttendaele, and T. Darrell, "Scale-MAE: A Scale-Aware Masked Autoencoder for Multiscale Geospatial Representation Learning," Sept. 2023. arXiv:2212.14532 [cs].

[11] Y. Cong, S. Khanna, C. Meng, P. Liu, E. Rozi, Y. He, M. Burke, D. B. Lobell, and S. Ermon, "SatMAE: Pre-training Transformers for Temporal and Multi-Spectral Satellite Imagery," Jan. 2023. arXiv:2207.08051 [cs].

[12] J. Jakubik, S. Roy, C. E. Phillips, P. Fraccaro, D. Godwin, B. Zadrozny, D. Szwarcman, C. Gomes, G. Nyirjesy, B. Edwards, D. Kimura, N. Simumba, L. Chu, S. K. Mukkavilli, D. Lambhate, K. Das, R. Bangalore, D. Oliveira, M. Muszynski, K. Ankur, M. Ramasubramanian, I. Gurung, S. Khallaghi, Hanxi, Li, M. Cecil, M. Ahmadi, F. Kordi, H. Alemohammad, M. Maskey, R. Ganti, K. Weldemariam, and R. Ramachandran, "Foundation Models for Generalist Geospatial Artificial Intelligence," Nov. 2023. arXiv:2310.18660 [cs].

[13] D. Szwarcman, S. Roy, P. Fraccaro, orsteinn Elí Gíslason, B. Blumenstiel, R. Ghosal, P. H. de Oliveira, J. L. de Sousa Almeida, R. Sedona, Y. Kang, S. Chakraborty, S. Wang, A. Kumar, M. Truong, D. Godwin, H. Lee, C.-Y. Hsu, A. A. Asanjan, B. Mujeci, T. Keenan, P. Arevalo, W. Li, H. Alemohammad, P. Olofsson, C. Hain, R. Kennedy, B. Zadrozny, G. Cavallaro, C. Watson, M. Maskey, R. Ramachandran, and J. B. Moreno, "Prithvi-eo-2.0: A versatile multi-temporal foundation model for earth observation applications," 2024.

[14] S. Khanna, P. Liu, L. Zhou, C. Meng, R. Rombach, M. Burke, D. Lobell, and S. Ermon, "DiffusionSat: A Generative Foundation Model for Satellite Imagery," Dec. 2023. arXiv:2312.03606 [cs].

[15] *Die Bodennutzung in der Schweiz*. No. 19365051, Neuchâtel: Bundesamt für Statistik (BFS), Nov 2021.

[16] G. Giuliani, D. Rodila, N. Külling, R. Maggini, and A. Lehmann, "Downscaling Switzerland Land Use/Land Cover Data Using Nearest Neighbors and an Expert System," *Land*, vol. 11, p. 615, May 2022. Number: 5 Publisher: Multidisciplinary Digital Publishing Institute.

[17] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[18] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks," Nov. 2018. arXiv:1611.07004 [cs].

[19] C. Li and M. Wand, "Precomputed real-time texture synthesis with markovian generative adversarial networks," 2016.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

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

[22] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," 2020.

[23] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[24] V. Kwatra, Mei Han, and Shengyang Dai, "Shadow removal for aerial imagery by information theoretic intrinsic image analysis," in *2012 IEEE International Conference on Computational Photography (ICCP)*, (Seattle, WA, USA), pp. 1–8, IEEE, Apr. 2012.

[25] L. Bütikofer, A. Adde, D. Urbach, S. Tobias, M. Huss, A. Guisan, and C. Randin, "High resolution land use forecasts for switzerland in the 21st century," 2023.