PMLDL Assignment 3: Diffusion

In this assignment, your task is to build and train your own diffusion model. You have **two options** for the diffusion model task:

1. **Text-to-Image Generation**
2. **Image Inpainting**

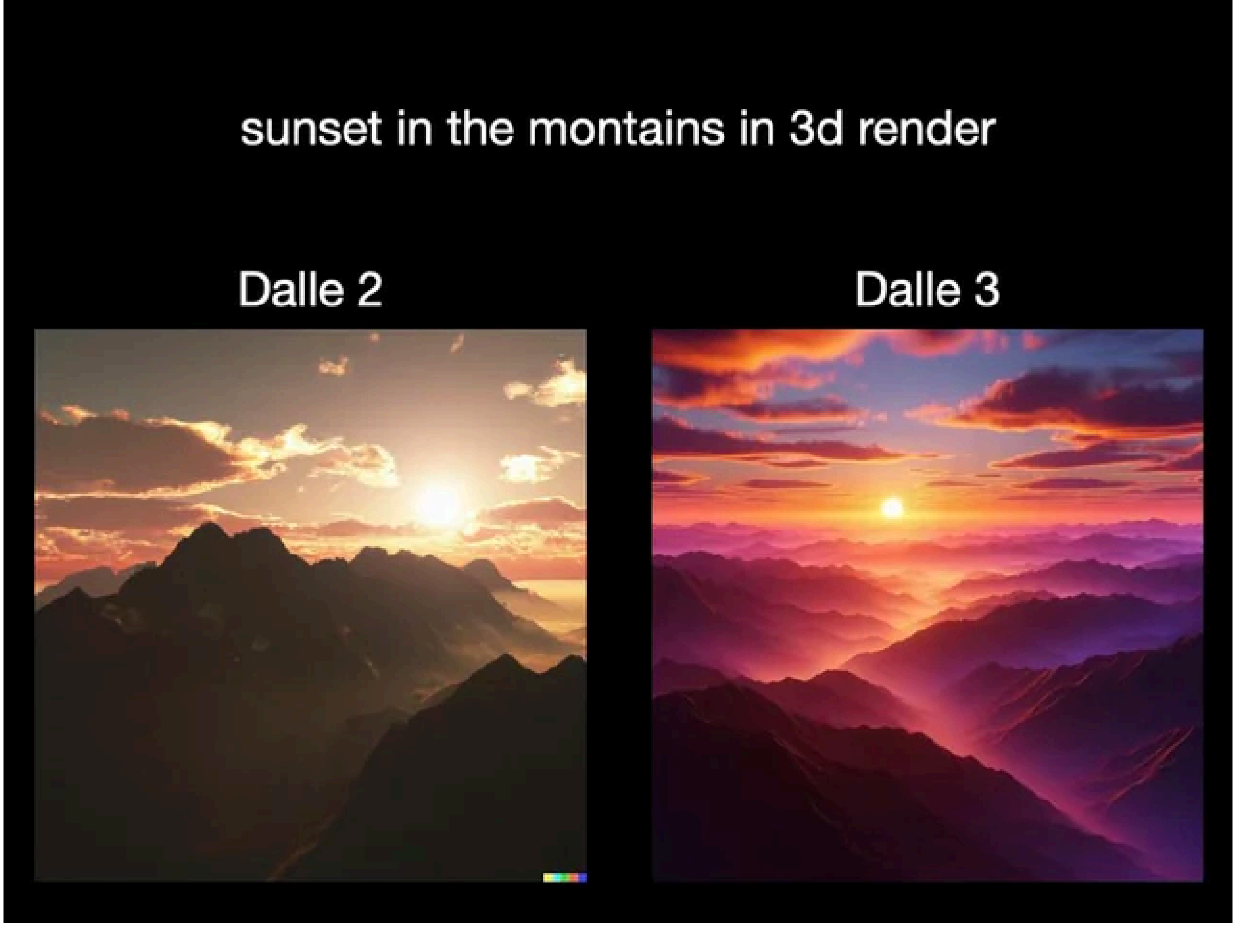
You can select one task from these options and implement and train a diffusion model for the task. But if you wish to implement both options, then you will get bonus points.

Now, let us describe the tasks in details.

Option 1: Text-to-Image Generation

Task Decription

Text-to-Image Generation is a type of image generation process where the model creates images based on specific input textual prompts. These prompts guide the output, allowing control over the visual characteristics or content of the generated images.



For example, the model generates images based on descriptive text (e.g., “a sunset over a mountain range”), commonly used in text-to-image generation models like DALL-E and Stable Diffusion.

Recommended Implementation Steps

To implement such a model, you can follow these recommended steps (but you can follow your own steps, if you wish):

1. Gather a **dataset** of paired text-image data, such as the COCO dataset or Open Images Dataset, where each image has a corresponding descriptive text prompt. For large-scale models, high-quality, diverse datasets (e.g., LAION) are useful.
2. **Resize** images to a uniform size (e.g., 256x256 pixels). **Tokenize** the text prompts using a suitable tokenizer (e.g., BERT, CLIP, or a custom tokenizer) that matches the model input requirements.
3. Use a **U-Net architecture** to learn denoising at different time steps, as it captures multi-scale features essential for generating complex images.
4. Integrate a text **encoder** (like CLIP or a Transformer-based encoder) to convert the text prompts into embeddings. Inject these embeddings into the U-Net model at each step via cross-attention or concatenation to condition the image generation on the text prompt.
5. Define the forward (adding noise) and reverse (denoising) **diffusion steps**. The forward process progressively adds Gaussian noise to an image over multiple time steps, while the reverse process attempts to reconstruct the image from noise conditioned on the prompt.
6. Generate and **visualize** the images for some prompts. Start with random noise and use the learned reverse diffusion process to iteratively denoise the image while

conditioning on a given prompt.

Grading Criteria for Text-to-Image Generation

- **Data preprocessing** is done correctly - 1 point
- **Model** is implemented correctly - 1 point
- **Training** is implemented correctly - 1 point
- The model generates **meaningful images**, but not just noise - 1 point
- The model generated the image that **follows the prompt** - 1 point

Option 2: Image Inpainting

Task Description

Image inpainting is the process of filling in missing, damaged, or unwanted parts of an image in a way that looks seamless and natural. This technique is widely used in fields such as image restoration, editing, and computer vision, where it's often applied to remove objects, repair damaged images, or reconstruct occluded areas.

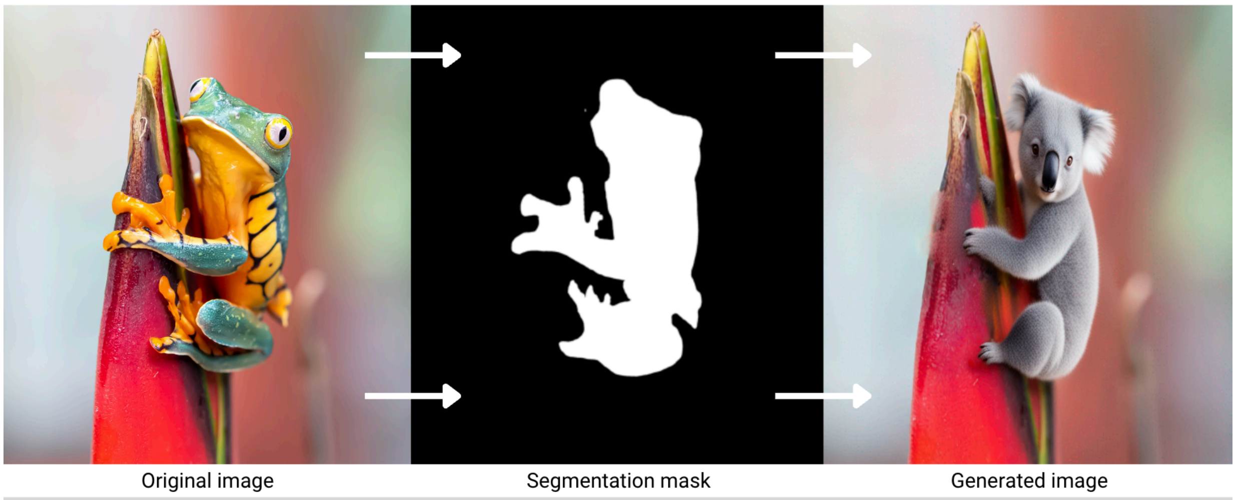


Image inpainting

Recommended Implementation Steps

To implement such a model, you can follow these recommended steps (but you can follow your own steps, if you wish):

1. Gather a **dataset** of images that you want to inpaint. This could range from generic datasets (e.g., ImageNet, CelebA) to domain-specific data (e.g., medical images, landscapes).
2. Create **binary masks** to define the regions that will be inpainted. Masks can vary in shape and size, such as irregular shapes (for natural occlusions) or rectangular masks (for object removal). These masks will be used to hide parts of the images during training.
3. Use a **U-Net architecture** as the backbone of the diffusion model. This model will predict the noise at each timestep during the denoising process and can effectively handle different scales and resolutions.
4. To **perform inpainting**, the model must learn to condition on the unmasked, visible regions around the masked areas. You can implement this by setting the masked

regions to zero or noise and passing the image, mask, and time step as inputs to the model.

5. Set up a forward and reverse **diffusion process**. In the forward diffusion process, noise is progressively added to an image over a series of time steps until it becomes nearly random noise. The reverse process is where the model learns to gradually remove this noise to reconstruct the image.
6. Generate and **visualize** the images with missing regions. Start the inpainting process with an image that has missing (masked) regions filled with noise. Run the reverse diffusion process. At each step, the model predicts the noise to remove based on the context around the masked area, gradually refining the inpainted region.

Grading Criteria for Image Inpainting

- **Data preprocessing** is done correctly - 1 point
- **Model** is implemented correctly - 1 point
- **Training** is implemented correctly - 1 point
- The model generates **meaningful images**, but not just noise - 2 point

Technical Notes

In this assignment, you are not restricted to any specific tools and frameworks to implement and train the model. That is why we do not provide specific source code that you can use as a basis. However, if you do not have an idea of what framework to use, we recommend to use PyTorch.

As the backbone model, you can also use the model of your choice (but the recommended one is a U-Net). To achieve the best performance of the diffusion model, do not train the model from scratch but try to use a pre-trained model. Still, training the model from scratch is not prohibited and you can do it, but keep in mind that it is a resource-demanding process and the result may be poor.

If you feel that you need more computational resources for your model, you can book these in InnoDataHub.

Submission

Submit the solution of the assignment as an **.ipynb** file to Moodle. It is highly recommended to keep the notebook for submission clean and structured.

Cheating Policy

According to the regulations of the university, the assignment should be performed by your own. You can use the resources from the Internet, but the final solution must be compiled by your own. In case we suspect a student in cheating, the student will be invited to the **in-person defence**.

Useful Links

- DDPM Paper

- Inpainting Tutorial
- Conditional Diffusion Model Tutorial
- Stabe Diffusion from Scratch in PyTorch