E4: Multimodal Diffusion Models

Prof. Dr. Anna Rohrbach,
Prof. Dr. Marcus Rohrbach,
Hector Garcia Rodriguez, M. Sc.
Jonas Henry Grebe, M. Sc.

TECHNISCHE UNIVERSITÄT DARMSTADT

Deadline Wednesday 9th July, 2025

General Information

This exercise is about Multimodal Diffusion Models (DMs). The coding part will focus on Textual Inversion (TI), a special method to fine-tune an existing text-to-image DM.

To hand in your solutions to this exercise, you will have to submit the following:

- Your report as a PDF named `E4_GROUP_<YOUR_GROUP_NUMBER>.pdf` containing your answers to the question part and any additional explanations, descriptions, or discussions required for the coding part. You will also be asked to provide code snippets for several of the coding tasks in your report. Parts of your solution that are not clearly linked to a task won't be graded.

  Please use the E4 LaTeX template that we provide for this exercise. You must answer within the `solutionbox` environment, and not change the size of the boxes. You must place your textual answers using:
  ```
  \begin{solutionbox}[N cm]
  <YOUR ANSWER>
  \end{solutionbox}.
  ```
  Your answers should be precise and fit within the existing `solutionbox`. Parts of your answer outside of this box may not be taken into account. If you want to add a table, use only the `tabular` environment. **The font and any other formatting cannot be changed.**

  Additionally, **you must also add to the report your code snippets, using the `solutioncode` environment provided**. Here's how to use it:
  ```
  \begin{solutioncode}[language=Python]
  <YOUR CODE>
  \end{solutioncode}.
  ```
  You only need to add the code in between `# ---- YOUR CODE STARTS HERE ----` and `# ---- YOUR CODE ENDS HERE ----`, since you should only edit that.

- Additionally, a Google Colab link (with edit access to everyone) to your copy of the provided notebook has to be submitted to Moodle. Ensure that the results mentioned in your report are reproducible from the submitted checkpoint of your notebook.

  Important: Before you share it, you have to save the checkpoint of your notebook, which gives it a time stamp and shows it in the revision history (`File` $\rightarrow$ `Save and checkpoint`), where you can rename it to `Final Submission (Group <YOUR_GROUP_NUMBER>)`. Every change after this submission checkpoint won't be graded. If there is no pinned revision that is marked as your submission, we will just grade the last existing checkpoint from before the deadline. *If you used Kaggle, please upload your notebook to Google Colab again to provide this link.*

  Any use of ChatGPT is not allowed, and it will be penalized.

In case of questions regarding the submission process, please use the **Admin Q&A** forum on Moodle.

Exercises

This exercise has a maximum score of 25 points: up to 10 points for the question part and 15 points for the coding part.

Question Part (10 Pts.)

Answer the following questions about Diffusion Models (DMs) before and during your work on the coding part of this exercise:

1. ✍ **Energy-based Models** (4 Pts.): A less obvious but important perspective on Diffusion Models (DMs) is their close connection to Energy-based Models (EBMs). While DMs are often described through stochastic processes and denoising steps, they can also be interpreted as implicitly learning an energy landscape over the data space, where the model guides samples toward regions of higher data density. This perspective offers a unifying view that connects diffusion models to a broader class of generative modelling approaches. Learn about EBMs and how they relate to DMs. Start your exploration with this helpful blog by Yang Song.

   a) Summarize the core ideas and intuition of EBMs. Explain the connection between EBMs and DMs. (2 Pts.)

   b) Derive the mathematical relationship between the score function $s_\theta(x)$ and the energy function $e_\theta(x)$ from the energy-based formulation of $p_\theta(x)$. (2 Pts)

2. ✍ **Inference Efficiency** (2 Pts.): In contrast to approaches like Generative Adversarial Networks (GANs), DMs often require hundreds to thousands of denoising steps T. At the same time, text-to-image DMs today often utilize Classifier-free Guidance (CFG) (Ho et al. 2022) that typically combines an unconditional and a prompt-conditioned prediction for more control on the diversity-fidelity trade-off via a new inference-time hyperparameter.

   State how CFG impacts the computational complexity at inference for a batch of B prompts and T inference timesteps. Then, explain when and how DMs can pre-compute and cache the unconditional scores to minimize compute requirements for CFG inferences under the B prompts.

3. ✍ **Learn about Textual Inversion** (4 Pts.):

   Begin by learning about Textual Inversion (TI), a special fine-tuning technique for pre-trained text-to-image diffusion models (DMs). Then explain TI in detail. Focus on its intuition, goals, technical details, and limitations. Assume SD v1.4 is the model that TI is applied to.

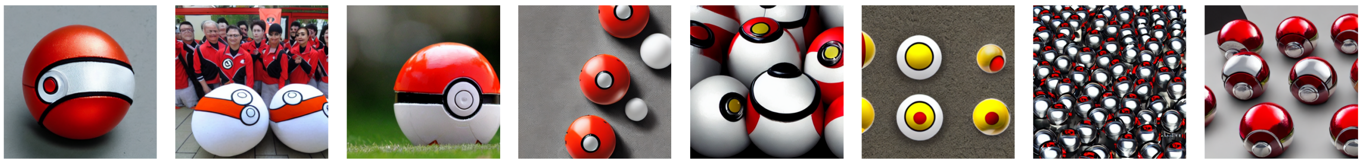

Figure 1: Reference samples that were generated using the default configuration and a prompt containing only the placeholder token.

Coding Part (15 Pts.)

This coding exercise is about implementing Textual Inversion for SD v1.4. In Moodle, we provide a E4 Google Colab Starter Kit that contains some initial code framework. In the notebook, we already provide a collected set $\mathcal{X}$ of images that will be downloaded and saved under `"content/images"` when executing the respective cell. For this task, you might want to get familiar with the underlying `StableDiffusionPipeline` implementation from Hugging Face.

1. **Loss Function** (2 Pts.): The TI method internally relies on a standard DM loss. Complete the loss function in the code.

   This function expects a `TrainingWrapper` instance, the original image `latents`, the `timestep_idx` of the current timestep, and the CLIP `embeddings`. The function starts by generating noise, and then adds it appropriately to the latents (based on the noise schedule $\rightarrow$ `wrapper.scheduler`). Then, it predicts the score with the model and computes the final loss based on the target and the prediction. It returns the loss in the end.

2. **Optimization Setup** (5 Pts.): TI requires a certain setup of the DM before a correct training can start. This is the most complex task and requires diving into the documentation of the Hugging Face components. The most important parts of this setup are:

   - A new set of placeholder tokens is added to the `Tokenizer` and the `Text Encoder` is adapted accordingly. Ensure to properly add those tokens to the `Tokenizer` and properly extend the `Text Encoder`'s embeddings. There are existing methods of the Hugging Face classes.
   - You should use the `initializer_token` attribute of the `TextualInversionConfig` to find token embeddings for the initialization.
   - The token embedding layer of the `Text Encoder` is the one that is trained and updated, nothing else! This is important, so you might want to make appropriate assertions (`assert` in Python) to ensure that this is the case in your implementation.
   - *The tricky part*: We only want to optimize the new embedding vectors, not the entire embedding matrix. That requires a simple trick to ensure that only the new vectors are changing over the iterations. For this, you are allowed to add additional class attributes to the `TrainingWrapper` class if you need to: (1) Store a copy of the original embeddings. (2) Implement a function that overwrites the old part of the text encoder embeddings with this copy. (3) Then, patch the optimizer's step method to always apply this overwrite after an optimization step. (`optimizer.step = patched_step`)

   Implement this necessary setup in the `prepare_for_textual_inversion` function. This method returns the prepared optimizer (you may use PyTorch's `AdamW`) for the textual inversion.

3. **Main Algorithm** (8 Pts.)

   a) **Implementation**: Implement TI in the provided `textual_inversion` function. Use the default configuration to apply it to the given image set $\mathcal{X}$ and fix the batch size for TI fine-tuning to 1. You may use the provided methods for the image transformation and for validation sampling. (4 Pts.)

      For reference, we provide a set of samples (see Figure 1) that were generated with the default configuration and a basic prompt that only contains the placeholder token.

   b) **Evaluation** (4 Pts.)

i. Create a set of 3 custom diverse prompt templates featuring other visual concepts/attributes (e.g., “comic book illustration” or “black and white”). These prompts should test how well the diffusion model can integrate your learned pseudo-word with varied textual contexts.

ii. Select your own unique visual concept that cannot be easily described using ordinary text prompts (e.g., a specific object from your environment). Collect and upload 5–10 representative images of your chosen concept. Adapt the configuration as needed and re-run TI to learn a new pseudo-word for your custom concept. Test it with the same 3 prompt templates as before.

iii. Create 2 more interesting prompts that combine both of your learned pseudo-words and test them with your TI model. Include the important parts of your evaluation code in the report!

c) ✍ **Write-up**: Generate at least 4 samples for each prompt. Add the resulting $(3+3+2)\times 4 = 32$ images to your report and include an analysis discussing:

- How well each of the two learned concepts interact with other visual concepts/attributes
- How well the two learned concepts interact with each other