AS4 Student Assignment Guide
Due: Tuesday, Dec 9 @5:30pm

Overview

In this assignment you will design, train, and evaluate image generation models inside an NVIDIA container executed through Apptainer on a Slurm based GPU cluster.
You will:

1. Set up a reproducible GPU environment using an NVIDIA NGC container wrapped by Apptainer and launched via Slurm.
2. Implement and train a Deep Convolutional GAN (DCGAN) style image generator on a small dataset (for example MNIST, Fashion-MNIST, or CIFAR-10), using transpose convolutions as introduced in the lecture slides.
3. Run and analyze a diffusion model using an existing Python library (for example a UNet based denoising or text conditioned latent diffusion model).
4. Compare GAN based image generation to diffusion based image generation in terms of image quality, stability, and computational cost.
5. Document all work in a well structured report and provide all scripts, job files, and configuration artifacts.

The assignment is designed so that a focused student can complete it in about 10 days.

Learning Objectives

By the end of this assignment you should be able to:

1. Explain how convolution and transpose convolution are used to process and generate images in CNN based architectures.
2. Describe the training dynamics of Generative Adversarial Networks, including generator and discriminator losses, mode collapse, and instability.
3. Run a DCGAN style model for image generation at scale using Python and GPUs inside an NVIDIA container on a Slurm cluster.
4. Describe the forward and reverse diffusion processes and explain the role of UNet architectures in diffusion models.
5. Use a prebuilt diffusion model (for example a latent diffusion model) for image generation on a GPU and analyze its behaviour relative to a GAN.
6. Package your work in a reproducible manner using Apptainer, including all Python code, configuration, and Slurm job scripts.

Assignment Structure

1. Task: Environment and Container Setup

Goal: Run Python with GPU acceleration inside an NVIDIA NGC container managed by Apptainer, launched through Slurm.

1. Select a base NVIDIA container
   - Choose a deep learning container from NVIDIA NGC that includes PyTorch (or your preferred framework).
   - Record the container image identifier and version in a README file.

2. Create an Apptainer image definition or configuration
    - Wrap the NGC container using Apptainer so that it can run on the cluster.
    - Ensure that:
        - GPU access is enabled.
        - Your project directory on the cluster is visible inside the container.
        - All required Python libraries can be installed or are already present.
3. Create a Slurm batch configuration for running the container
    - Create a batch submission file that:
        - Requests at least one GPU.
        - Allocates sufficient memory and wall time.
        - Launches the Apptainer container.
        - Runs a simple Python test script that prints the detected GPUs and basic CUDA information.
4. Verify the setup
    - Submit a short test job.
    - Confirm that:
        - The job runs to completion.
        - The output log confirms that a GPU is visible inside the container.
        -

2. Task: DCGAN Style Image Generator

Goal: Implement and train a DCGAN style model for image generation.

1. Data preparation
    - Write a Python data pipeline that:
        - Downloads or loads the dataset.
        - Applies necessary preprocessing (for example scaling pixel values, resizing, converting to tensors).
    - Decide on batch size and number of epochs that are realistic within cluster limits.
2. Model architecture
    - Design a generator based on transpose convolutions as described in the lecture slides on DCGAN and transpose convolution.
    - Design a discriminator based on regular convolutions.
    - Use normalization and activation functions consistent with DCGAN best practices (for example batch normalization, nonlinearity choices).
3. Training procedure
    - Implement the coupled training loop where:
        - The discriminator distinguishes real images from generated images.
        - The generator is updated to fool the discriminator.
    - Log:
        - Generator and discriminator losses per epoch.
        - Representative generated image grids at regular intervals.
4. Experiments
    - Run at least two different training configurations, for example:
        - Baseline architecture and learning rate.
        - Modified architecture (for example different latent dimension, changed depth, or additional regularization).
    - For each configuration:

        - Save model checkpoints.
        - Save generated samples at multiple stages of training.
5. Analysis
    - Examine generated images for signs of:
        - Mode collapse (for example repeated or nearly identical samples).
        - Training instability (for example diverging losses).
    - Discuss how the lecture notes on GAN issues relate to your observations.

3. Task: Diffusion or Latent Diffusion Model Experiment

Goal: Use a diffusion based model to generate images and compare behaviour with your GAN. You are not required to implement the diffusion model from scratch. You may use a well established Python library.

1. Model selection
    - Choose one of the following:
        - A standard denoising diffusion probabilistic model with a UNet backbone.
        - A text conditioned latent diffusion model (for example Stable Diffusion) for simple prompts.
2. Configuration inside the container
    - Install the required library in your Apptainer / NVIDIA container environment.
    - Confirm that the model can be loaded and that generation runs on the GPU.
3. Generation experiments
    - For a denoising diffusion model:
        - Add synthetic Gaussian noise to a small set of images and run the model to denoise them, relating to the lecture description of the forward and reverse diffusion processes.
    - For a latent diffusion model:
        - Choose at least three simple natural language prompts.
        - Generate multiple images per prompt to explore variation in the latent space.
4. Performance and stability observations
    - Record:
        - Run times per sample or per batch.
        - Number of denoising steps.
        - Any qualitative artefacts or failures (for example incomplete objects, misaligned attributes).
    - Relate these to the pros and cons of diffusion models discussed in the slides (training stability, computational cost, rich sampling process).

4. Task: Comparative Analysis and Scaling Considerations

Goal: Provide a graduate level analysis of GAN versus diffusion based image generation.

1. Qualitative comparison
    - Place representative result grids from:
        - The best DCGAN model.
        - The diffusion or latent diffusion model.
    - Compare:
        - Diversity of samples.

        - Sharpness and realism.
        - Types of artefacts.
2. Quantitative or proxy metrics
    - If possible, compute at least one quantitative metric for each model (for example a simple classifier based score, reconstruction error, or an approximate FID).
    - Discuss limitations of the chosen metric.
3. Training and inference cost
    - Compare:
        - Wall time to train the DCGAN to a reasonable quality.
        - Wall time to generate a fixed number of images with each approach.
        - GPU utilization and memory demands based on your Slurm logs.
4. Theoretical reflection
    - Use concepts from the lectures to explain:
        - Why GANs can suffer from mode collapse and unstable dynamics.
        - Why diffusion models are generally more stable but computationally intensive.
        - How latent diffusion reduces cost by operating in a lower dimensional space.
5. Scaling and engineering perspective
    - Discuss how your workflow would need to change to:
        - Train on larger datasets.
        - Use multiple GPUs or nodes.
        - Integrate more advanced models such as StyleGAN or conditional GANs.

5. Task: Submission and Final Report

Goal: Ensure that your work is reproducible and well documented.

Written report (primary deliverable)

- Length guideline: 6–8 pages (excluding figures and appendices).
- Recommended structure:
    - Introduction and Objectives
        - Motivation for image generation.
        - Brief overview of GANs and diffusion models.
    - Methods
        - Dataset and preprocessing.
        - DCGAN architecture and training regime.
        - Diffusion model configuration.
        - Cluster, container, and Slurm setup.
    - Results
        - Visual examples from both models.
        - Quantitative metrics or proxy measures.
        - Training and inference performance.
    - Discussion
        - Analysis of mode collapse, stability, sample diversity.
        - Interpretation of diffusion model behaviour.
        - Limitations of your approach.
    - Conclusion

- Summary of key findings.
- Suggestions for improving architectures, training strategies, or scaling to larger problems.

- Appendix
  - Environment details, container versions, and any additional figures.

Deliverables

Submit a compressed archive or repository containing:

1. Report (PDF)
   - Graduate level analysis following the structure above.
   -