MLMI17 Advanced Computer Vision:

Coursework #1: Representation Learning and Generative Models

Ayush Tewari

Due: Feb 23rd, 2026 online via moodle

Your answers should contain an explanation of what you do. You must also give an *interpretation* of what the numerical values and graphs you provide *mean* – why are the results the way they are? Each question should be labelled and answered separately and distinctly. Total combined length of answers must not exceed 5 sides of a4 (plus cover page), minimum 11pt font, 1 inch margins. Please submit the pdf and all your code in a zip.

Part 1: Representation Learning This part of the coursework uses two files: `representations.py`, which contains the model definitions and training logic, and `part1.py`, which is the main script for running experiments. You are required to complete the missing code blocks marked with `### YOUR CODE STARTS HERE ###` in `representations.py`. All code is hosted at `https://github.com/CambridgeCVCourses/CW1/tree/main`.

(a) **Feature Extraction and t-SNE Analysis:** Extract features from the provided dataset using CLIP, DINO, and MAE backbones. Visualize these using t-SNE. Compare how these three pre-training objectives (contrastive, self-distillation, and masked auto-encoding) cluster the fruit classes in the embedding space.

(b) **Linear Probing:** Train a linear head on top of the frozen backbones. Report validation accuracies and provide an interpretation of the training loss curves. Does the ranking of models in the t-SNE visualization correlate with their linear probing performance? (HINT: Accuracy with CLIP should be around 90%.)

(c) **Fine-tuning:** Initialise a model with your best linear probe and fine-tune the entire backbone. Report validation accuracies and provide an interpretation of the training loss curves. Compare these results with the linear probe results. Explain the performance gain (or lack thereof).

Part 2: Diffusion Models This part of the coursework uses two files: `DIT.py`, which contains model definitions, and `part2.py`, which is the main script for running experiments. The code follows DDPM [1], where the network predicts noise from noisy images. Diffusion Transformers [2] are used as the denoising neural network. All code is hosted at `https://github.com/CambridgeCVCourses/CW1/tree/main`.

(a) **Forward and Reverse Diffusion:** Complete the missing logic in `forward_diffusion()` to implement the closed-form sampling of $\mathbf{x}_t$ given $\mathbf{x}_0$ and a timestep $\mathbf{t}$. In the training loop, the network is optimized to predict the noise ϵ added to the image. Subsequently, implement the iterative reverse diffusion sampling function, `sample_ddpm()`, to generate new images from pure Gaussian noise. You should refer to Algorithm 1 (Training) and Algorithm 2 (Sampling) in the DDPM paper [1] to ensure your implementation is correct. Include the following:

i. Images generated by your diffusion model after training
ii. Visualisation of the denoising process during inference
iii. Plot of the training loss
iv. A short (max 10 sentence, including equations) description of the denoising network architecture.

Figure 1: Samples from an unconditional diffusion model trained on MNIST

A successful implementation should give you samples that look like those in Figure 1.

(b) **Conditional Generation:** MNIST includes labels for each image. Update the training code to train a conditional diffusion model and generate images conditioned on the digit label. Include the following:

i. Plot of the training loss. Comment on differences with unconditional training

ii. 5 generated images for each digit

iii. A short (max 10 sentence, including equations) description of how the denoising network architecture changes from the unconditional case.

(c) **Classifier-free Guidance:** Implement classifier-free guidance [3], using the `forward_with_cfg()` function in `DIT.py`. Train the diffusion model with classifier-free guidance, and generate samples with different guidance values. Include the following:

i. Generated samples with difference guidance values (1, 5, 10, 15, 20)

ii. Analysis of the impact of guidance values on the generation quality

iii. A short (max 10 sentence, including equations) description of how classifier-free guidance was implemented.

References

[1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[2] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[3] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.