Assignment 9: Autoencoders & Representation Learning

1 Instructions

In this assignment, you will explore unsupervised and self-supervised learning by implementing various Autoencoder (AE) architectures. You will move beyond simply predicting labels to learning compressed, meaningful representations of high-dimensional data. You will implement Undercomplete, Sparse, Denoising, and Variational Autoencoders (VAEs) on the **CelebA dataset** (a large-scale face attributes dataset). You will evaluate these models not just on how well they reconstruct images, but on how useful their latent representations are for downstream classification tasks, interpretability (latent space arithmetic), and generative modeling.

1.1 Submission Requirements

You must submit two components to Gradescope:

1. **The Report:** A plaintext Markdown document containing the 5 sections described in Part 2.
   - **No Images:** Describe your findings with text and data tables.
   - **Character Limit:** 7,500 characters.
2. **Supplemental Material:** Your `.ipynb` notebook containing all code, plots, and raw results.

---

2 Part 1: Implementation (The Notebook)

You will author a Jupyter Notebook (`.ipynb`) to perform the following experiments. Using **Google Colab** with a GPU runtime is strictly required to process image data efficiently.

Global Setup & Data Acquisition:

- Fix the random seeds for reproducibility (e.g., `torch.manual_seed(42)`).
- Configure your device dynamically: `device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")`.

- **Dataset Management (Crucial for Speed):** The full CelebA dataset is massive.

To ensure your models train in a reasonable time, you must:

1. Use `torchvision.transforms` to resize the images to `64x64` and convert them to tensors.
2. Load the dataset using `torchvision.datasets.CelebA`.
3. **Subset the data:** Use `torch.utils.data.Subset` to create a training set of exactly 30,000 images and a test set of 5,000 images.
4. Create `DataLoader` instances for both subsets.

2.1 Experiment 1: Deterministic Autoencoders

Concept: Standard autoencoders consist of an **Encoder** that compresses an input x into a lower-dimensional latent representation z, and a **Decoder** that attempts to reconstruct the original input $\hat{x}$ from z. By forcing the network through this "bottleneck," it must learn the most salient features of the dataset.

2.1.1 Task 1.1: The Base Architecture

- Implement a `CNNEncoder` and `CNNDecoder` using `nn.Conv2d` and `nn.ConvTranspose2d` (or upsampling) layers.
- The Encoder should take a `3x64x64` image and output a flat latent vector z of size `latent_dim=128`.
- The Decoder should map z back to a `3x64x64` image. Use a `Sigmoid` or `Tanh` activation at the final layer (depending on your normalization strategy) to output valid pixel values.

2.1.2 Task 1.2: Model Variants

Wrap your base components into three distinct Autoencoder classes. Train each for a fixed number of epochs (e.g., 10-15) using Mean Squared Error (MSE) reconstruction loss.

1. **Undercomplete AE:** The standard formulation. The loss is purely $L = \text{MSE}(x, \hat{x})$.
2. **Sparse AE:** Encourage sparsity in the latent space by adding an L1 regularization penalty to the latent vector z. The loss becomes $L = \text{MSE}(x, \hat{x}) + \lambda \sum |z_i|$. **Tuning λ:** To tune λ, start with a very small value (e.g., 10^{-5}) and increase it logarithmically (e.g., 10^{-4}, 10^{-3}). A reasonable default to induce sparsity without collapsing the model's ability to reconstruct the image is often around 10^{-4} or 10^{-3}. You want to find the sweet spot where the L1 norm decreases visibly, but the MSE doesn't skyrocket.
3. **Denoising AE:** Modify the forward pass to add random Gaussian noise (or apply dropout) to the input x *before* passing it to the encoder. The loss is computed against the *clean*, original image: $L = \text{MSE}(x, \hat{x}_{\text{noisy\_input}})$.

Notebook Output:

- Final training and validation reconstruction losses for all three models.

- For the Sparse AE, explicitly track and plot the L1 norm of the latent representations across epochs.
- A plot displaying a grid of 5 original test images alongside their reconstructions from each of the three models.

2.2 Experiment 2: The Variational Autoencoder (VAE)

Concept: Deterministic AEs learn a discontinuous latent space, making them poor generative models. VAEs solve this by making the encoder output parameters of a probability distribution rather than a discrete point.

Specifically, the VAE optimizes the negative Evidence Lower Bound (ELBO), which balances how well the model reconstructs the data against how closely the learned latent distribution matches a prior (usually a standard normal distribution). The mathematical formulation of the loss is:

$$L_{\text{VAE}} = -\mathbb{E}_{q(z|x)}[\log p(x|z)] + D_{\text{KL}}(q(z|x)||p(z))$$

Where the first term is the Reconstruction Loss (e.g., MSE or BCE), and the second term is the Kullback-Leibler (KL) Divergence.

2.2.1 Task 2.1: Reparameterization and Loss

- Implement a `VAE` class. The encoder must branch into two linear layers at the end to output the mean vector μ and the log-variance vector $\log(\sigma^2)$. Note that both of these are vectors of size `latent_dim`.
- **Why Log-Variance?** Neural networks naturally output unconstrained real numbers $(-\infty, \infty)$. However, variance must be strictly positive. If we predicted variance directly, we would need to apply an activation function like ReLU or Softplus, which can suffer from dead gradients. By predicting $\log(\sigma^2)$ instead, the network's output remains unconstrained. You can recover the standard deviation for sampling by calculating $\sigma = \exp(0.5 \cdot \log(\sigma^2))$.
- Implement the **Reparameterization Trick**: Sample z using $z = \mu + \sigma \odot \epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$.
- **The Loss Function:** The full VAE loss minimizes the Reconstruction Loss plus the KL Divergence:
$$D_{\text{KL}} = -\frac{1}{2}\sum_{i=1}^{d}(1 + \log(\sigma_i^2) - \mu_i^2 - \exp(\log(\sigma_i^2)))$$
- Train the VAE on your CelebA subset.

Notebook Output:

- Plot the VAE training loss, explicitly plotting the Reconstruction loss and KL divergence as separate curves to observe their trade-off.
- A grid of 5 original vs. reconstructed images.

2.3 Experiment 3: Representation Learning & Interpretability

Concept: A good autoencoder doesn't just memorize pixels; it learns semantic features (e.g., lighting, pose, facial attributes) linearly separated in the latent space z.

2.3.1 Task 3.1: Linear Probing (Self-Supervised Evaluation)

We will test how useful the learned representations are for downstream tasks without updating the autoencoder itself.

- Using your trained **Undercomplete AE** and your **VAE**, iterate through your training and test data.
- **Freeze the Encoders:** Before training the classifier, iterate through the parameters of your encoders and set `param.requires_grad = False` so their weights are not updated.
- Attach a simple `nn.Linear` classifier on top of the latent space to predict the binary CelebA attribute `"Smiling"`. Dynamically pass images through the frozen encoder, take the resulting z vector, and pass it through the linear layer.
- Train only the linear layer using Cross Entropy Loss.

2.3.2 Task 3.2: Latent Space Arithmetic

- Using your VAE, compute the average latent vector for all images where `"Eyeglasses" == 1` (z_{glasses}) and where `"Eyeglasses" == 0` ($z_{\text{no\_glasses}}$).
- Calculate the attribute direction vector: $v_{\text{glasses}} = z_{\text{glasses}} - z_{\text{no\_glasses}}$.
- Take a test image without glasses, encode it to z, and create a modified latent vector: $z_{\text{new}} = z + \alpha v_{\text{glasses}}$.
- Decode z_{new} for varying values of α (e.g., -2, 0, 2, 4) to visually "add" glasses to the face.

Notebook Output:

- Test accuracy of the linear probe on the "Smiling" attribute for both the AE and VAE.
- A visual plot of the latent interpolation (adding glasses to a single image) across 4 values of α.

2.4 Experiment 4: Generative Modeling & Ex-Post Density Estimation

Concept: Because VAEs regularize their latent space toward a standard normal prior $\mathcal{N}(0, I)$, we should theoretically be able to sample random noise from $\mathcal{N}(0, I)$, pass it through the decoder, and generate brand new faces. However, VAEs often suffer from the "prior hole" problem, where the aggregated posterior of the training data doesn't perfectly match the $\mathcal{N}(0, I)$ prior, leading to poor samples when drawing strictly from the prior.

2.4.1 Task 4.1: Standard Prior Sampling

- Sample 16 random vectors directly from the standard normal prior $z \sim \mathcal{N}(0, I)$.
- Pass them through your VAE decoder to generate 16 novel faces.

2.4.2 Task 4.2: Ex-Post Density Estimation

To improve sample quality, we can post-hoc fit a distribution to the *actual* latent representations of our training data (as proposed in arXiv:1903.12436).

- Pass a large chunk of your training set through the VAE encoder to obtain their μ representations.
- Calculate the empirical mean ($\hat{\mu}$) and covariance matrix ($\hat{\Sigma}$) of these latent representations.
- Sample 16 new vectors from this fitted multivariate Gaussian: $z \sim \mathcal{N}(\hat{\mu}, \hat{\Sigma})$.
- Pass these through the VAE decoder.

Notebook Output:

- A grid of 16 faces generated via Standard Prior Sampling.
- A grid of 16 faces generated via Ex-Post Density Estimation.

---

3 Part 2: Content Requirements (The Report)

Your report must be organized into **exactly five sections** with Markdown headers.

3.1 Section 1: Executive Summary

- **One-Sentence Takeaway:** Summarize the core finding comparing the performance of deterministic AEs versus VAEs on either downstream linear probing or generative capabilities.
- **Summary Paragraph:** Outline the progression of the assignment, highlighting the transition from standard reconstruction and regularization to latent space interpretability and generative modeling.

3.2 Section 2: Autoencoder Regularization

- **Results Comparison:** Briefly discuss the visual differences in the reconstructions of the Undercomplete, Sparse, and Denoising AEs.
- **Mechanics:** Explain mechanically how the Denoising AE prevents the network from simply learning the identity function. How does this compare to the structural bottleneck of the Undercomplete AE and the L1 penalty of the Sparse AE?

3.3 Section 3: Representation Learning & Interpretability

- **Results Table:** Present a Markdown table comparing the Linear Probing accuracy of the Undercomplete AE vs. the VAE on the "Smiling" task.
- **Analysis:** Which model produced a more linearly separable representation for downstream tasks? Why might the VAE's KL-divergence penalty help or hurt its ability to learn these distinct semantic attributes compared to a standard AE?
- **Latent Arithmetic:** Discuss the success of your vector arithmetic task. Did adding the v_{glasses} vector successfully alter the image without destroying the original identity of the face?

3.4 Section 4: Generative Modeling & Prior Holes

- **Comparison:** Compare the qualitative visual quality of the faces generated by Standard Prior Sampling vs. Ex-Post Density Estimation.
- **Mechanics:** Explain the "prior hole" problem in VAEs. Why does calculating the empirical mean and covariance $(\hat{\mu}, \hat{\Sigma})$ of the training set's latent vectors yield more coherent generated faces than sampling blindly from $\mathcal{N}(0, I)$?

3.5 Section 5: Reflection

- **Training Dynamics:** Reflect on the difficulty of training the VAE. Did you encounter posterior collapse (where the KL divergence drops to zero and the model ignores the latent code)? How did you balance the Reconstruction loss vs the KL loss?
- **Hardware/Compute:** Discuss the importance of subsetting the dataset and resizing the images for this specific assignment.

---

4 Grading Rubric

Each of the five sections is weighted equally (**20% each**).

Criterion	Excellent (5)	Good (4)	Satisfactory (3)	Okay (2)	Poor (1)
Section 1: Summary	Takeaway is specific and insightful. Summary captures the architectural progression and task evaluations.	Takeaway is clear. Summary covers the main assignment arc.	Takeaway is generic. Summary is vague.	Summary misses key elements of the analysis.	Missing.

Criterion	Excellent (5)	Good (4)	Satisfactory (3)	Okay (2)	Poor (1)
Section 2: Regularization	Excellent explanation of identity function prevention, clearly connecting loss modifications to visual outcomes.	Good comparison. Explains the denoising mechanism well.	Compares variants but the mechanical "why" behind the regularization is weak.	Descriptions are incorrect or analysis is missing.	Missing.
Section 3: Interpretability	Table is complete. Deep, mechanical analysis of why VAEs vs standard AEs perform differently on linear probing and latent arithmetic.	Table present. Good explanation of linear separability and vector math results.	States results but lacks a discussion on the impact of the KL penalty.	Missing data table or failed to implement the latent interpolation.	Missing.
Section 4: Generative Models	Deep comparison of sample quality. Excellent, mathematically sound explanation of the "prior hole" problem and why ex-post estimation fixes it.	Good comparison. Explains the distribution mismatch well.	Mentions quality differences but lacks specific technical reasoning for the ex-post fix.	Evaluates tasks but ignores the distribution discussion entirely.	Missing.
Section 5: Reflection	Thoughtful reflection on VAE loss balancing, posterior collapse, and data management realities.	Good reflection on debugging and hyperparameters.	Generic reflection (e.g., "VAEs are hard to train").	Minimal effort.	Missing.