Part 1: DIY Diffusion - Grading Rubric

Total Points: 50

Task 1: Forward Diffusion Implementation (12 points)

Item	Criteria	Points
1.1	Calls prepare_noise_schedule() to obtain beta and alpha schedules	3
1.2	Correctly loops through timesteps from 0 to T-1	2
1.3	Samples Gaussian noise with the same shape as input image at each timestep	2
1.4	Applies correct forward diffusion equation: x_t = sqrt(α_t) * x_0 + sqrt(1-α_t) * ε	3
1.5	Returns a list containing all noised images (one per timestep)	2

Task 2: UNet Architecture Reflection (9 points)

Item	Criteria	Points
2.1a	**Skip Connections - What is concatenated:**	
2.1a.1	Correctly identifies that encoder feature maps are concatenated with decoder representations	1
2.1a.2	Identifies that concatenation occurs at matching resolution levels (e.g., x2 with upsampled features)	1
2.1b	**Skip Connections - Purpose:**	
2.1b.1	Explains that skip connections preserve fine-grained spatial details or structural information	1
2.2	**Time Embeddings - Importance:**	
2.2.1	Explains that the model needs to know which timestep/noise level it is operating on	2
2.2.2	Explains that different timesteps require different denoising strategies or behaviors	1
2.3	**Input/Output Shape - Importance:**	
2.3.1	Explains that the UNet predicts noise with the same dimensions as the input	2
2.3.2	Explains that matching shapes allows proper subtraction of predicted noise from noisy input	1

Task 3: Reverse Diffusion Sampling (15 points)

Item	Criteria	Points
3.1	Initializes x as random Gaussian noise with shape (3, 32, 32)	2
3.2	Loops backwards from timestep T-1 to 0 (reverse order)	2
3.3	Adds batch dimension to x before passing to model (using unsqueeze)	2
3.4	Creates timestep batch with shape (batch_size,) for model input	1
3.5	Removes batch dimension from model output (using squeeze)	1
3.6	Computes pred_x0 correctly: (x - sqrt(1-α_t) * noise_pred) / sqrt(α_t)	3
3.7	Clamps pred_x0 to [-1, 1] range	1
3.8	Correctly adds noise back when t > 0 using: sqrt(α_{t-1}) * pred_x0 + sqrt(1-α_{t-1}) * noise	2
3.9	Sets x = pred_x0 for final timestep (t=0)	1

Task 4: Training Loop & Model Evaluation (14 points)

Training Implementation (8 points)

Item	Criteria	Points
4.1	Samples random timesteps with shape (batch_size,) for each batch	1
4.2	Generates random noise matching batch shape	1
4.3	Reshapes alpha values for broadcasting (with .view(-1, 1, 1, 1))	1
4.4	Applies forward diffusion to create noisy images: x_t = sqrt(α_t) * batch + sqrt(1-α_t) * noise	2
4.5	Obtains model prediction by passing noisy images and timesteps to model	1
4.6	Computes MSE loss between predicted noise and actual noise	1
4.7	Performs complete gradient update: zero_grad(), backward(), optimizer.step()	1

Model Quality Evaluation (6 points)

Item	Criteria	Points
4.8	Training loss decreases to ≤0.15 (Pikachus) OR ≤0.20 (diverse Pokemon)	3

Item	Criteria	Points
4.9	Generated samples are qualitatively recognizable as Pokemon sprites (have discernible shapes, colors, and features similar to training data)	3

Grading Notes:

- Each criterion is evaluated as either met (full points) or not met (0 points)
- Code must run without errors for the relevant task to receive points
- For Task 4.8: Loss threshold is 0.15 for Pikachus dataset, 0.20 for pokemon_simple dataset
- For Task 4.9: Grader will visually assess whether generated images show Pokemon characteristics (appropriate colors, shapes resembling creatures, recognizable features)
- Partial credit is not awarded within individual criteria; each criterion is all-or-nothing

Part 2: Guided Diffusion (with CLIP) - Grading Rubric

Total Points: 50

Task 1: CLIP Similarity Search (16 points)

Implementation (12 points)

Item	Criteria	Points
1.1	Encodes text query using CLIP text encoder (get_text_features)	3
1.2	Normalizes text embedding by dividing by its L2 norm	2
1.3	Computes similarities between text embedding and all image embeddings (using matrix multiplication)	3
1.4	Finds index of highest similarity score	2
1.5	Returns both the most similar image and its similarity score	2

Results Quality (4 points)

Item	Criteria	Points
1.6	At least 4 test queries produce results that make intuitive qualitative sense given the query description	2
1.7	At least 4 test queries achieve similarity scores > 0.25	2

Task 2: Guidance Scale Exploration (9 points)

Item	Criteria	Points
2.1a	**Question 1 - Low guidance scale behavior:**	

Item	Criteria	Points
2.1a.1	Explains that low guidance scale (0) causes the model to ignore the text prompt or produce random/incoherent images	2
2.1b	**Question 1 - High guidance scale behavior:**	
2.1b.1	Explains that high guidance scale (12) makes the model overly constrained by the prompt, producing overly literal or rigid images	1
2.2a	**Question 2 - Mathematical definition:**	
2.2a.1	Provides correct guidance formula: guided_noise = noise_uncond + guidance_scale * (noise_cond - noise_uncond)	2
2.2b	**Question 2 - Explanation:**	
2.2b.1	Explains how guidance_scale=0 reduces to unconditional generation (ignoring text)	1
2.2b.2	Explains how high guidance_scale amplifies the conditional component, making the model heavily favor the text prompt	1
2.3	**Question 3 - Common defaults and applications:**	
2.3.1	Explains why scales like 7-8 are common (balance between text adherence and image quality)	1
2.3.2	Identifies appropriate use cases for smaller or larger guidance scale values	1

Task 3: Guided Diffusion Implementation (25 points)

Implementation (18 points)

Item	Criteria	Points
3.1	Calls UNet with latents, timestep, and unconditional embeddings to get unconditional noise prediction	3
3.2	Extracts noise prediction using .sample attribute from UNet output (for unconditional)	2
3.3	Calls UNet with latents, timestep, and text embeddings to get conditional noise prediction	3
3.4	Extracts noise prediction using .sample attribute from UNet output (for conditional)	2
3.5	Applies classifier-free guidance formula correctly: noise_uncond + guidance_scale * (noise_cond - noise_uncond)	3
3.6	Uses scheduler.step() with guided noise prediction to update latents	3

Item	Criteria	Points
3.7	Extracts updated latents using .prev_sample attribute from scheduler output	2

Reflection Questions (7 points)

Item	Criteria	Points
3.8a	**Question 1 - Qualitative comparison:**	
3.8a.1	States that implementation produces qualitatively similar results to Task 2's StableDiffusionPipeline	2
3.8a.2	Confirms that behavior across guidance scales matches expectations (scale 0 ignores prompt, higher scales increase prompt adherence)	1
3.8b	**Question 2 - VAE encoder usage:**	
3.8b.1	Explains that encoder is not needed because text-to-image generation starts from random noise in latent space	2
3.8b.2	Identifies scenarios requiring the encoder (image-to-image generation, style transfer, or editing existing images)	2

Grading Notes:

- Each criterion is evaluated as either met (full points) or not met (0 points)
- Code must run without errors for the relevant task to receive points
- For Task 1.6-1.7: Grader will evaluate whether returned images match query semantics and check similarity scores
- For Task 2 reflection questions: Answers must demonstrate understanding of the concepts, not just repeat the prompt
- For Task 3.8a: Visual comparison should verify similar guidance scale behaviors between implementations
- Partial credit is not awarded within individual criteria; each criterion is all-or-nothing