Generating an Image with a Trained Diffusion Model

```python
import torch
from unet import UNet  # same model architecture as training

model = UNet(in_channels=4, out_channels=3)
model.load_state_dict(torch.load("trained_diffusion_model.pth"))
model.eval()

num_noise_levels = 1000
step_size = 1.0 / num_noise_levels

# Start from pure noise (CIFAR-10 size: 32x32)
image = torch.randn(1, 3, 32, 32)

for step in reversed(range(num_noise_levels)):
    noise_amount = step / num_noise_levels

    # Tell the model the current noise level
    t_channel = torch.full_like(image[:, :1, :, :], noise_amount)
    model_input = torch.cat([image, t_channel], dim=1)

    # Predict the noise at this level and remove it
    with torch.no_grad():
        predicted_noise = model(model_input)
    image = (image - predicted_noise) / (1 - step_size)
```

Questions

Work with your group to study the code above.

1. What is `image` initialized to? What does it look like visually?

2. How many times does the model run during generation of one image?

3. In your own words, describe what happens at each iteration of the loop.

4. Why does each step only subtract a *small fraction* of the predicted noise, rather than all of it at once?

5. Why is image generation with diffusion models slow compared to other approaches?