CS552: Generative AI, Homework 4
Will Buchta
3/25/2025

1. **Diffusions: Unconditional Generation [13 pts]**: Using a trained diffusion model (you can either just use the pre-trained DDPM in `diff_unet_faces.cpt` or, if you prefer, train your own using the face data in `faces23k_48x48.npy.gz`), sample 100 faces (each is 48x48 pixels) and display them in a 10x10 collage. For DDPM, the sampling procedure is given in the Prince textbook (linked on Canvas), Algorithm 18.2. For σ_t, a common practice (according to the original DDPM paper by Ho et al, 2020) is to set $\sigma_t = \sqrt{\beta_t}$. The hyperparameters used for training, including the noise schedule $\{\beta_t\}_{t=1}^{T}$, are given in the starter code `homework4_template.py`.

See Figure 1 for my results and ddpm.py for the implementation.

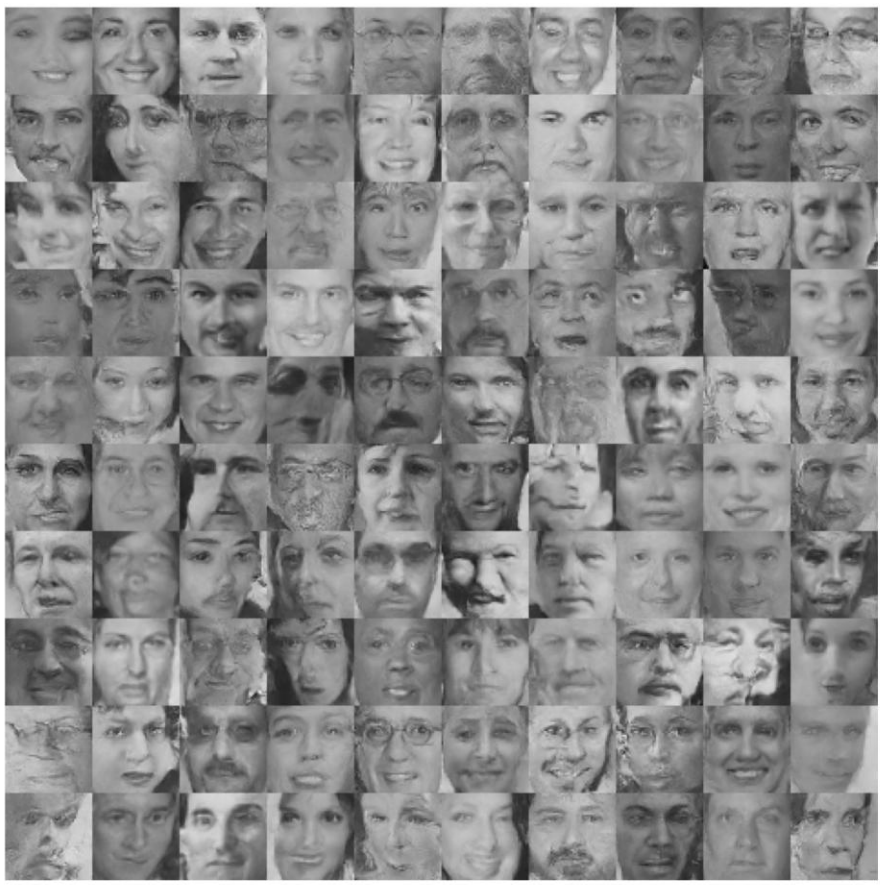

Figure 1: Unconditional face diffusion

2. **Diffusions: Inpainting/Merging [12 pts]**: Using the same pre-trained DDPM from the previous problem, "merge" two faces into one:

   (a) Select any two faces $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$.

   (b) Apply the diffusion kernel for some $t < T$ to compute $\mathbf{z}_t^{(1)}$ and $\mathbf{z}_t^{(2)}$.

   (c) Create $\mathbf{z}_t$ by combining the left half of $\mathbf{z}_t^{(1)}$ with the right half of $\mathbf{z}_t^{(2)}$.

   (d) Sample $\mathbf{x} \sim P(\mathbf{x} \mid \mathbf{z}_t)$.

   Apply this to 5 pairs of faces each for 4 different values of t each – show the results in a 4x15 collage (where the rows correspond to different t – 10 columns show the 5 pairs of $(\mathbf{x}^{(1)}, \mathbf{x}^{(2)})$, and 5 columns show the corresponding merged samples $\mathbf{x}$. Arrange the columns as you see fit but make sure to explain them clearly in your report.

See problem2() in ddpm.py. For each group of three columns, $\mathbf{x}^{(1)}$ is the first photo, $\mathbf{x}^{(2)}$ is the second photo, and $\mathbf{x}$ is the third photo. The best results can be seen when the diffusion kernel is applied 500 times in Figure 2:

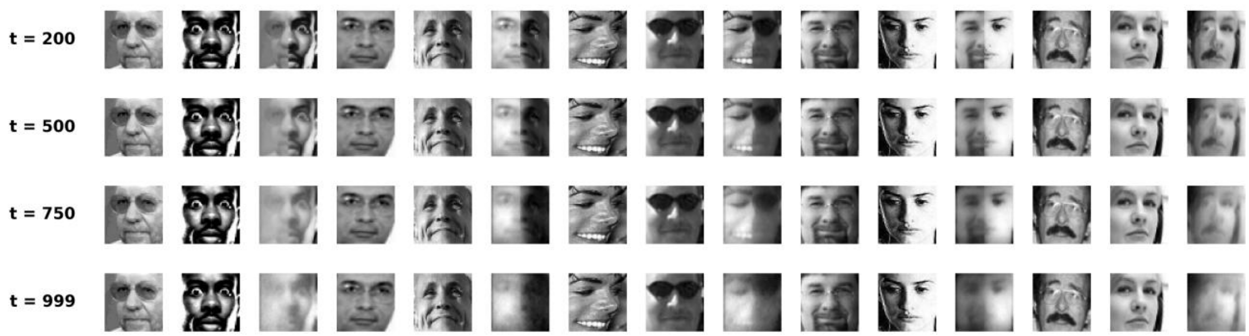

Figure 2: Merged faces

3. **Diffusions: Conditional Generation with Classifier Guidance [30 pts]**: Using the face data in `faces23k_48x48.npy` and associated age labels in `ages23k.npy`, train an age regressor f that maps from any $(\mathbf{z}_t, t)$ to an estimate $\hat{y}$ of how old the person in the corresponding $\mathbf{x}$ is. To create the training data for f, you will need to use the diffusion kernel to create $\mathbf{z}_t$ for various t and $\mathbf{x}$. Exclude any examples from training if either $y < 0$ (which indicates a missing label) or $y > 100$ (which is likely an incorrect label). f should output the mean of a Gaussian distribution with unit variance that characterizes the age of a specific noised image at a specific timestep t, i.e., $P(y \mid \mathbf{z}_t) = \mathcal{N}(f(\mathbf{z}_t, t), 1)$. As an architecture for the regressor, you can use the first half (i.e., downward convolution) of the same U-Net with time embeddings which implements the noise estimator E_θ in the pre-trained DDPM. However, the choice is up to you.

   After training the regressor, use classifier guidance (even though you actually have a regressor) to generate 4 faces each for target ages 18, 40, 60, and 80 years old. To compute $\frac{\partial \log P(y \mid \mathbf{z}_t)}{\partial \mathbf{z}_t} = -\frac{\partial (\hat{y}-y)^2}{\partial \mathbf{z}_t}$ for a target age value y, you can ask ChatGPT for help with the autograd expressions (you will need to enable `requires_grad=True` on the tensor representing $\mathbf{z}_t$). You may find it useful to multiply the gradient by a small number (e.g., 0.2) rather than using its full magnitude. Show these faces in a 4x10 collage.

See my implementation in ddpm.py. I had to use a very small 'guidance scale' for the classifier's gradient – see below for the difference between a scale of 0.001 and 0.0005. Any values higher than that resulted in "mush" for lack of a better word. The rows of the figures correspond to ages 18, 40, 60, and 80, which I think was represented quite well.

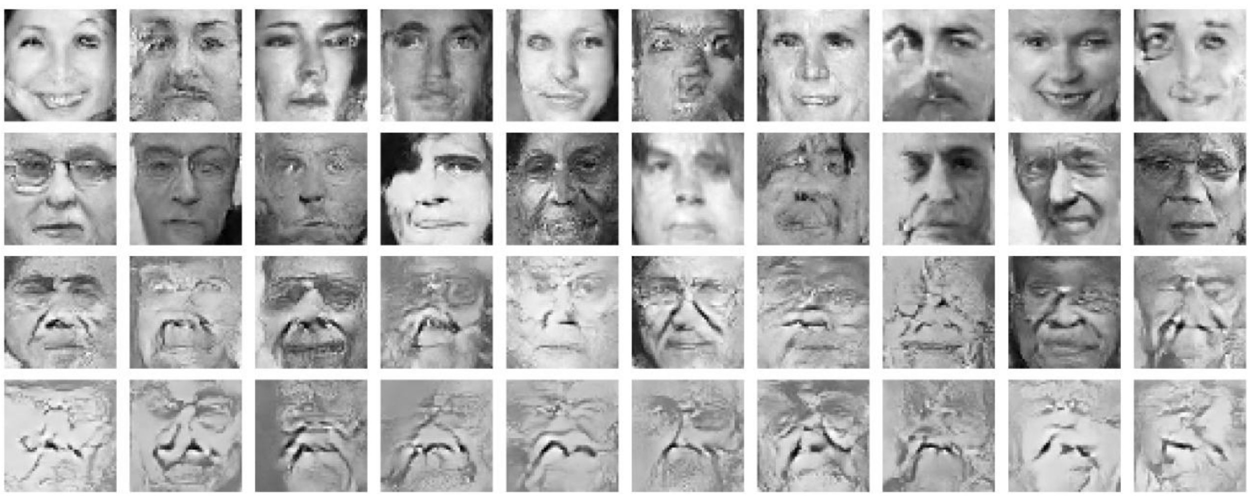

Figure 3: Classifier guidance with a scale of 0.001

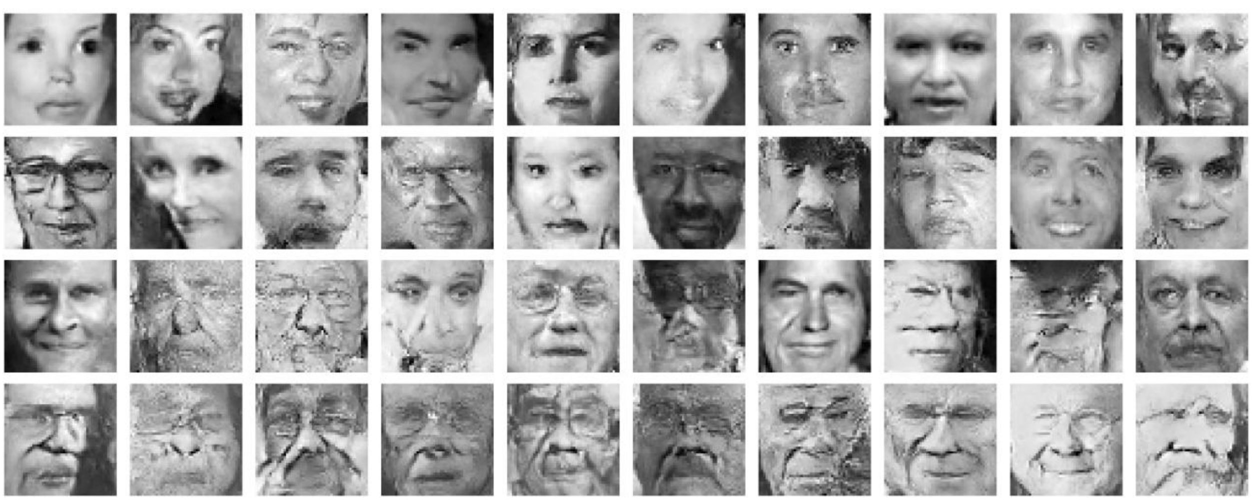

Figure 4: Classifier guidance with a scale of 0.0005