Guided diffusion process

Exercise 4

Advanced Deep Learning in Computer Vision

February 2024

In this exercise, you are asked to guide the diffusion process in the sprites dataset. This dataset consists of 5 classes: {human, non-human, food, spell, side-facing}.

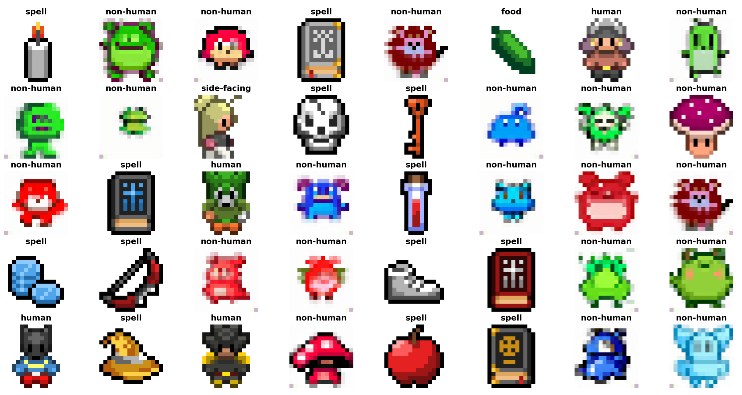

Figure 1: Samples from the sprites dataset

You should write a pdf report for the tasks below:

1. **Classifier Guidance (cg):** You will implement a modified version of Algorithm 1 from the paper: Diffusion Models Beat GANs on Image Synthesis. The only modification is that we use constant variance instead of learnable. In particular, you will guide the generation process of your DDPM model from Exercise 3 using your own classifier. To achieve this:

   (a) Build your own classifier to classify noisy images of the dataset in `model.py` (line 175). Don't forget to add the timestep as input. HINT: use the UNet encoder.

   (b) Train your classifier (use the template file `classifier_train.py`) and save the weights in `weights/classifie/model.pth`. Evaluate it using the script: `classifier_eval.py`. What do you observe?

(c) Have a look at how the function `classifier_conditioning` in the file `ddpm.py` is used to get gradients from the classifier and guide the sampling process. We trained for you a standard DDPM model and saved the weights in `weights/DDPM/model.pth`.

(d) Run the script `sample_cg.py` to generate samples for each class. Include your results and explain the sampling process in your report.

2. **Classifier-Free Guidance (cFg):** Now, your task is to train and implement a new UNet that takes an optional additional input, which is the one-hot encoded class label. You will implement a modified version of Algorithm 1 from this paper. Here is the algorithm that you should implement:

Algorithm 1 Train a diffusion model with classifier-free guidance

```
Require: $p_{\text{uncond}}$                                   ▷ probability of unconditional training
1: repeat
2:     $\mathbf{x}, \mathbf{y} \sim p(\mathbf{x}, \mathbf{y})$                                   ▷ for each batch of data
3:     $\mathbf{y} \leftarrow \varnothing$ with probability $p_{\text{uncond}}$          ▷ Randomly discard labels to train unconditionally
4:     $t \sim \text{Uniform}(\{1, \dots T\})$
5:     $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
6:     Take gradient descent step on
           $\nabla_\theta \left\| \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t, \mathbf{y}) \right\|^2$     ▷ $\mathbf{y}$ is input to the UNet.
7: until converged
```

To implement it you should:

(a) Add a layer in UNet (line 144, `model.py`) that projects the input labels **y** to the time embedding dimension. Note that for this exercise **y** should be one-hot encoded and not an integer.

(b) In the forward function (line 154) add to the positional encoding of time the projection of **y**.

(c) Implement Algorithm 1 in `ddpm_train.py`. To train the classifier-free guidance model, activate the flag. Run: python ddpm_train.py –cfg. The weights will be saved in `weights/DDPM-cFg/model.pth`. Include your results in the report.

3. **Quantitative comparison:** Your final task is to compare classifier guidance and classifier-free guidance with the well-known metric FID for image generation. Implement FID score in `ddpm_eval.py`. It will take some time to run the evaluation. Try to debug your FID implementation before running it by comparing the FID score measured on the exact same test data (which should be extremely small) and with pure noise images (which should be extremely large). Explain how FID is calculated and report your results in the report.

4. (OPTIONAL) Classifier free-guidance mixing: Since you trained the classifier free-guidance model using one-hot encoded vectors you can mix the classes during sampling. Play around and come up with weird generations.