EECS 182 Deep Neural Networks
Fall 2025 Anant Sahai and Gireeja Ranade Discussion 13

1. Denoising diffusion models for generation

In lecture, we talked about denoising diffusion models to get samples from a continuous distribution. This problem is about the potentially simpler binary case. We will assume that we have an unknown distribution of black-and-white images $P(\mathbf{x})$ together with a very large number of example images $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$. Formally, each image can be viewed as a binary vector of length m, i.e. $\mathbf{x} \in \{-1, +1\}^m$.

The first conceptual step in setting up a diffusion model is to choose the easy-to-sample distribution that we want to have at the end of the forward diffusion. For this, we choose m iid fair coin tosses (here we think of a fair coin as having a 50% chance of being +1 and a 50% chance of being -1) arranged into a vector.

Next, we need to choose a way to incrementally degrade the images. Let $\mathbf{Y}_0$ start with whatever image sample $\mathbf{x}$ we want to start with. At diffusion stage t, we generate $\mathbf{Y}_t$ from $\mathbf{Y}_{t-1}$ by randomly flipping each pixel of $\mathbf{Y}_{t-1}$ independently with probability δ where δ is a small positive number.

It turns out that this process of rare pixel-flipping can be reinterpreted for easier analysis. For the j-th pixel at diffusion stage t, this process can alternatively be viewed as first flipping an independent coin $R_t[j]$ with a probability 2δ of coming up $+1$ and then, if $R_t[j] = +1$ replacing $Y_{t-1}[j]$ with a freshly drawn independent fair coin $F_t[j]$ that is equally likely to be -1 or $+1$. If $R_t[j] \neq +1$, we leave that pixel alone i.e. $Y_t[j] = Y_{t-1}[j]$.

(a) We need to verify that if we do this and diffuse for sufficiently many stages T, that the resulting distribution is close to looking like m i.i.d. fair coins. **Show that the probability that pixel j has been replaced at some point by an independent fair coin by time T goes to 1 as $T \to \infty$.**

(HINT: It might be helpful to look at the probability that this has not happened…)

(b) To efficiently do diffusion training, we need a way to be able to quickly sample a realization of $\mathbf{Y}_t$ starting from $\mathbf{Y}_0 = \mathbf{x}_i$. **Give a procedure to sample a realization of $\mathbf{Y}_t$ given $\mathbf{Y}_0$ without having to generate $\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_{t-1}$.**

(HINT: This probability of whether each pixel is $+1$ or -1 can be obtained by flipping at most two (potentially biased) coins.)

(c) For the reverse diffusion process that will be used during image generation and is being learned during training, our goal is to approximate $P(\mathbf{Y}_{t-1}|\mathbf{Y}_t)$ with a neural net that has learnable parameters θ.

Suppose I give you a neural net whose input is a binary image $\mathbf{Y}_t$ and whose output is m real numbers that could each in principle be from $-\infty$ to $+\infty$ (for example, these could be the outputs of a linear layer). **Which of the following nonlinear activation functions would be most appropriate to convert them into a probability that we could use to sample whether the pixel in question should be a $+1$?**

- ◯ Sigmoid $\frac{1}{1+\exp(-x)}$
- ◯ ReLU $\max(0, x)$
- ◯ Tanh $\tanh(x) = \frac{\exp(2x)-1}{\exp(2x)+1}$

(d) The goal of training is to approximate a probability distribution for random denoising. However, we do not actually have access to $P(\mathbf{Y}_{t-1}|\mathbf{Y}_t)$ and decide to use $P(\mathbf{Y}_{t-1}|\mathbf{Y}_t, \mathbf{Y}_0)$ instead as a proxy. Note: this can be interpreted as finding an appropriate target distribution that corresponds to trying to predict $\mathbf{Y}_0$ itself at this stage.

What is $P(Y_{t-1}[j] = +1|\mathbf{Y}_t = \mathbf{y}, \mathbf{Y}_0 = \mathbf{x})$?

For simplicity, just do this calculation for the case $x[j] = +1$. To further help you save some time, you may use the following helper result that comes from Bayes' Rule. If A and B are both binary random variables where the prior probabilities are $P(A = +1) = \rho$ and $P(A = -1) = 1 - \rho$, with B being bit-flipped from A with independent probability δ — that is, $P(B = +1|A = +1) = 1 - \delta$, $P(B = -1|A = +1) = \delta$, $P(B = +1|A = -1) = \delta$, and $P(B = -1|A = -1) = 1 - \delta$ — then the conditional probabilities for A conditioned on B are given by:

$$P(A = +1|B = +1) = \frac{(1-\delta)\rho}{(1-\rho)\delta + (1-\delta)\rho} \tag{1}$$

$$P(A = -1|B = +1) = \frac{(1-\rho)\delta}{(1-\rho)\delta + (1-\delta)\rho} \tag{2}$$

$$P(A = +1|B = -1) = \frac{\delta\rho}{\rho\delta + (1-\delta)(1-\rho)} \tag{3}$$

$$P(A = -1|B = -1) = \frac{(1-\delta)(1-\rho)}{\rho\delta + (1-\delta)(1-\rho)} \tag{4}$$

(HINT: What is the distribution for $Y_{t-1}[j]$ given $Y_0[j]$?)

(e) Let the (conditional) probability distribution (on whether each pixel is $+1$) output by our neural net with nonlinearity be $Q_t(\mathbf{Y}_t)$. For training the denoising diffusion model $q(\mathbf{Y}_{t-1}|\mathbf{Y}_t)$, we choose to use SGD loss $D_{KL}(P(\mathbf{Y}_{t-1}|\mathbf{Y}_t, \mathbf{Y}_0 = \mathbf{x}_i)||Q_t(\mathbf{Y}_t))$ where $\mathbf{x}_i$ is the random training image drawn, t is the random time drawn from 1 to T, and $\mathbf{Y}_t$ is the randomly sampled realization of the forward diffusion at time t starting with the image $\mathbf{x}_i$ at time 0. This ends up being a loss on the vector of probabilities coming out of $Q_t(\mathbf{Y}_t)$ that can be written as a sum over the m entries in the vector of probabilities.

Given what you know about KL Divergence, what does this loss penalize most strongly? What does this loss look like at $t = 1$ in particular?

(f) **After training the model, how do we generate a new sample image from scratch? Describe the full sampling procedure, from initialization, to iterative denoising, to getting the final sample**

2. Latent Variable Models

(a) Let $q_\phi(z \mid x)$ denote the encoder and $p_\theta(x_i \mid z)$ denote the decoder. **Draw a block diagram for a VAE. In the diagram, show the encoding of input x, sampling of latent z, and decoding of z into reconstruction $\hat{x}$.**

(b) **Describe step-by-step what happens during a forward pass in Variational Autoencoder (VAE) training.**

(c) **Describe what the encoder and decoder of the VAE are *respectively* doing** to capture and encode this information into a latent representation of space z. **How is the information bottleneck created in VAE as opposed to a standard Autoencoder.**

(d) After training, you notice that when you run a held-out example through the VAE that the predictions were blurry. Please, discuss why that might be.

(e) Once the VAE is trained, **how do we use it to generate a new fresh sample from the learned approximation of the data-generating distribution?**

Contributors:

- Anant Sahai.
- Joey Hong.
- Jerome Quenum.
- Ryan Campbell.
- Past CS282 Staff.

