M2 Applications of Machine Learning

CRSiD: tmb76

University of Cambridge

Contents

Using a Diffusion Model on the MNIST Dataset

Chapter 1

Training a Diffusion Model

1.1 Denoising Diffusion Probabilistic Model (DDPM)

1.1.1 Diffusion Models

Diffusion models are a class of probabilistic latent variable models. They consist of an encoder and decoder. The encoder takes the input data and maps it to a latent space in a series of steps, resulting in a series of intermediate latent vairables. The encoder is similar to variational autoencoders (VAEs) in that it maps the input data to a latent space. However, the particularity of the encoder here is that the mappings it will apply at each time step are predetermnied. The key part is the decoder which is trained to learn what is the reverse process of the encoder, therefore being then able to produce samples [3, p.348].

1.1.2 Denoising Diffusion Probabilistic Model (DDPM)

In this report, the writing conventions of the Prince textbook will be followed [3]. The model used for this project is a DDPM. For this mode, the encoder takes in input data $\mathbf{x}$ and maps it to a latent space $\mathbf{z_T}$, of the same dimensionality as $\mathbf{x}$, in a series of steps: $\mathbf{z_0} \rightarrow \mathbf{z_1} \rightarrow \ldots \rightarrow \mathbf{z_T}$. This is defined by a Markov Chain that is known, which at each step adds Gaussian Noise following a Noise or Variance Schedule, $\beta_{1,\ldots,T}$. As it is a Markov Chain, and a type of variational autoencoder, the encoder can be described by an approximate probability distribution q such that [2]:

$$q(\mathbf{z}_{1:T}|\mathbf{x}) = \prod t = 1T q(\mathbf{z}_t|\mathbf{z}_{1:t-1})) \tag{1.1}$$

Where the individual step is given by:

$$q(\mathbf{z}_t|\mathbf{z}_{t-1}) = \mathcal{N}(\mathbf{z}_t; \sqrt{1-\beta_t}\mathbf{z}_{t-1}, \beta_t\mathbf{I}) \tag{1.2}$$

In other words, β_t describes how much noise is going to be added to the input data at each step t. Prince's textbook also provides a closed form expression which shows this more clearly [3]:

$$\mathbf{z}_1 = \sqrt{1-\beta_1}\mathbf{x} + \sqrt{\beta_1}\epsilon_1 \tag{1.3}$$

And it can be shown that after t steps, this gives:

$$\mathbf{z}_t = \sqrt{\alpha_t}\mathbf{x} + \sqrt{1-\alpha_t}\epsilon \tag{1.4}$$

where $\alpha_t = \prod_{s=1}^{t} 1 - \beta_s$ and $\epsilon \sim \mathcal{N}(0, \mathbf{I})$, is a sample from a standard normal distribution, and therefore is the actual noise added. The decoder is trained to learn the reverse process of the encoder, or simply how to go from $\mathbf{z}_T$ to $\mathbf{z}_{T-1}$, continuing back through the latent variables to the input data $\mathbf{x}$. Coming back to the approximate probability distribution q, the decoder is trained to learn the reverse distributions $q(\mathbf{z}_{t-1}|\mathbf{z}_t)$. Approximating them as normal distributions, they can be written:

$$Pr(\mathbf{z}_{t-1}|\mathbf{z}_t, \phi t) = \mathcal{N}_{z_{t-1}}(\mathbf{f}_t[\mathbf{z}_t, \phi_t], \sigma_t^2\mathbf{I}) \tag{1.5}$$

where $\mathbf{f}_t$ is a neural network that takes $\mathbf{z}_t$ as input and has parameters ϕ_t, which here is just the timestep t. The reason why the model predicts the mean of the normal distribution with the variance being fixed to $\sigma_t^2\mathbf{I}$ is discussed in greater detail in the Ho et al. (2020) paper [2]. The training algorithm can then be written as follows:

Training Algorithm for DDPM reverse process [2]

```
Input: Data $\mathbf{x}$
Output: $\mu_\mathbf{t} = \mathbf{f}_t[\mathbf{z}_t, t]$
repeat
for $i \in \mathcal{B}$ do                          ▷ For each training example index in batch
    $t \sim \mathcal{U}(1, \dots, T)$                ▷ Sample a random time step
    $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$    ▷ Sample noise
    $L = ||\epsilon - \epsilon_\phi(\sqrt{\alpha_t}\mathbf{x} + \sqrt{1-\alpha_t}\epsilon, t)||^2$    ▷ Compute individual noise
end for        ▷ Accumulate losses for batch and take gradient descent step
until convergence
```

Predicting the noise in the algorithm instead of the mean is done by modifying the parameterization of $\mathbf{f}_t[\mathbf{z}_t, t]$ [2].

1.2 Training the Model on the MNIST Dataset

Training the model is done

1.3 Fine-Tuning the Model

2 different sets of hyperparams, compare results, both good and bad samples from the model.

Chapter 2

Custom Degradation Function

In Bansal et al. (2022, [1]), a conceptual summary of degradation functions is given. Starting with image $\mathbf{x} \in \mathbb{R}$, the degradation of the image, or forward process of the encoder for the DDPM described in section 1.1.2, can be considered as follows: $\mathbf{x}_t = D(\mathbf{x}, t)$, where D is the degradation operator and t is the severity of the degradation. In other words, $D(\mathbf{x}, 0) = \mathbf{x}$. In Chapter 1, the D operator consisted of adding Gaussian noise with variance described by the Variance/Noise Schedule $\beta_{1,\ldots,T}$. In this chapter, a custom degradation function, or operator D, will be described and used to train the model on the MNIST dataset. The important part of the degradation function is that an inverse process, R, is required to invert D and satisfies: $R(\mathbf{x}_t, t) \approx \mathbf{x}$, and $R(\mathbf{x}_t, 1) \approx \mathbf{x}_{t-1}$. The degradation function will be described in the next section. And as was discussed in Chapter 1, this is implemented through a neural network parameterized by ϕ and trained to minimize the loss $L = ||\mathbf{x} - R_\phi(D(\mathbf{x}, t), t)||$, taking an l_1 norm [1].

2.1 A ... degradation function

Describe the custom degradation function.

2.2 Training the modified model on the MNIST dataset

Train model with degradation function and discuss results.

2.3 Comparing with the original model

Evaluate the fidelity of the samples generated by the two models, discussing any differences between them.

Bibliography

[1] Arpit Bansal, Eitan Borgnia, Hong-Min Chu, Jie S. Li, Hamid Kazemi, Furong Huang, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Cold diffusion: Inverting arbitrary image transforms without noise. *Journal of Computer Vision*, 2022.

[2] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*, 2020.

[3] Simon J.D. Prince. *Understanding Deep Learning*. MIT Press, 2023.