IMAGE GENERATION OF BEDROOMS USING DIFFUSION MODELS.

000866628 cghh83

ABSTRACT

This paper demonstrates the usability of an improved DDPM model for unconditional image generation of bedrooms using the LSUN dataset at a 128×128 resolution. The proposed solution uses a variety of techniques to improve the quality and diversity of images sampled.

1 INTRODUCTION

Denoising Diffusion Probabilistic Models (DDPM) (Ho et al., 2020) are a type of diffusion probabilistic model for the purpose of producing image samples at a given timestep with regards to a forward noise process. Eventually, allowing the model to start with random gaussian noise and iterate backwards to predict an unseen image. OpenAI's "Improved Denoising Diffusion Probabilistic Models" (Nichol & Dhariwal, 2021) builds on this to optimise the noise schedule, improve the log-likelihood of the model, and reduce gradient noise. Further building on this, "Diffusion Models Beat GANs on Image Synthesis" (Dhariwal & Nichol, 2021) explores further architectural changes to improve unconditional image generation. This paper is based on that model, using the LSUN "bedrooms" dataset (Yu et al., 2015) to generate images of bedrooms.

2 METHODOLOGY

2.1 FORWARD PROCESS

The forward process uses a distribution of real images $q(0)$, which we can sample from to get the image $x_0 \sim q(x_0)$ (the non-noisy image). The forward process $q(x_t|x_{t-1})$, adds gaussian noise from a predefined variance schedule β where $0 < \beta_0 \leq \beta_1 \leq \cdots \leq \beta_{T-1} \leq \beta_T < 1$ and $T = 1000$, $\beta_0 = 0.0001$ and $\beta_T = 0.02$, as proposed by Ho et al. (2020).

$$\beta = (\beta_0, \beta_0 + (\frac{\beta_T - \beta_0}{T-1}), \cdots, \beta_0 + (T-2) \cdot (\frac{\beta_T - \beta_0}{T-1}), \beta_T)$$
$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \boldsymbol{I})$$

The noise schedule $\hat{\alpha}$ is initialised as the cumulative product of all α value where $\alpha_t = 1 - \beta_t$. These are used to save calculations when sampling from $q(x_0)$. Using this noise schedule and the normal distribution $\epsilon \sim \mathcal{N}(0, \mathbf{I})$, values $\mathbf{x}_0, \mathbf{x}_1, \cdots \mathbf{x}_T$ can be sampled from q as shown below.

$$\hat{\alpha} = (1-\beta_1, (1-\beta_1)(1-\beta_2), \cdots, \prod_{t=1}^{T-1}(1-\beta_t), \prod_{t=1}^{T}(1-\beta_t))$$
$$\mathbf{x}_t = \mathbf{x}_0 \cdot \sqrt{\hat{\alpha}_t} + \epsilon \cdot \sqrt{1-\hat{\alpha}_t}$$

2.2 REVERSE PROCESS

The reverse process constitutes the conditional distribution $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, which seeks to reverse this noising process, by learning the mean μ_θ and the variance Σ_θ of the Gaussian noise added to the images. This is done using a neural network with model parameters θ, which are optimised by

gradient descent. Both μ_θ and Σ_θ are dependent on the noise time-step t, giving the parameterised process below.

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$$

As stated by Ho et al. (2020), the combination of $\mathbf{p}_\theta$ and $\mathbf{q}$ is a variational auto-encoder (VAE) Kingma & Welling (2013), which gives the variational lower bound L_{vlb} as the sum of all losses at each time-step.

$$\mathbf{L}_{vlb} := \mathbf{L}_0 + \mathbf{L}_1 + \cdots + \mathbf{L}_T$$

Every term of $\mathbf{L}_{vlb}$, excluding $\mathbf{L}_0$, is a Kullback–Leibler (KL) divergence between two Gaussian distributions, and therefore the loss can be taken as the mean-squared-error between the means. As $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ can be sampled with $\mathbf{x}_t$ without the previous $\mathbf{x}_{t-1}$, a random t value can be used to optimise $\mathbf{L}_t$. The model predicts the noise, $\epsilon_\theta(\mathbf{x}_t, t)$, that has been added to $\mathbf{x}_t$.

This can be used to give the final loss function $\mathbf{L}_t$ with $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and the estimate of this noise given by the model.

$$\mathbf{L}_t = \| \epsilon - \epsilon_\theta(\mathbf{x}_t, t) \|^2$$

Sampling images from the model is done by taking the mean of the predicted noise. This is done by reparameterising the noise as shown.

$$\mu_\theta(\mathbf{x}, t) = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\hat{\alpha}_t)}}\epsilon_\theta(\mathbf{x}_t, t))$$

2.3 Model Architecture

Ho et al. (2020) originally proposed a U-Net (Ronneberger et al., 2015) that uses downsampling and upsampling layers. The downsampling layers consisted of residual blocks followed by downsampling convolutions, and the upsampling layers consisted of residual blocks followed by upsampling convolutions. Across both of these, attention layers were used at the 16×16 resolution. For each residual block, the time-step embedding was projected to specify the diffusion time t. Dhariwal & Nichol (2021) improves this architecture by using attention blocks not just at the 16×16 resolution. Within the model, a dropout probability was used for all layers to stop overfitting. This model architecture was implemented in PyTorch (Paszke et al., 2019).

The optimiser used for the model parameters was an Adam optimiser. The learning rate was able to increase with the use of scaling and shifting in the residual layers. This allowed the residual blocks to multiply and move the normalised values to give better predictions on a batch basis.

2.4 Model Parameters

Ho et al. (2020) uses a learning rate of 0.0001, but due to the scaling and shifting of the normalised values, a higher learning rate of 0.0003 was usable without model parameters exploding. As proposed by Dhariwal & Nichol (2021), for image generation at 128×128 attention blocks were used at 32, 16 and 8. The dropout probability used was 0.1.

2.5 Training

A large problem during training was the overfitting to small portions of the dataset. The first step to overcome this was using gradient accumulation. The limited amount of GPU memory available during training meant batch sizes were limited to 8 images, which caused the model to overfit to each batch massively. A mini-batch size of 8 was used with an actual batch size of 128. Optimiser steps were only taken once the mini-batches accumulated to the total batch size.

Another addition to this was using the exponential moving average (EMA) for the model parameters with a decay of $\lambda = 0.9999$. This was used to take averages over the model parameters for use in sampling, to negate the effects of overfitting in the model. The results of this are shown in Fig. 1. Without ema sampling, the model does not create consistent results, with patterns and objects missing and some edges being rounded and unclear.

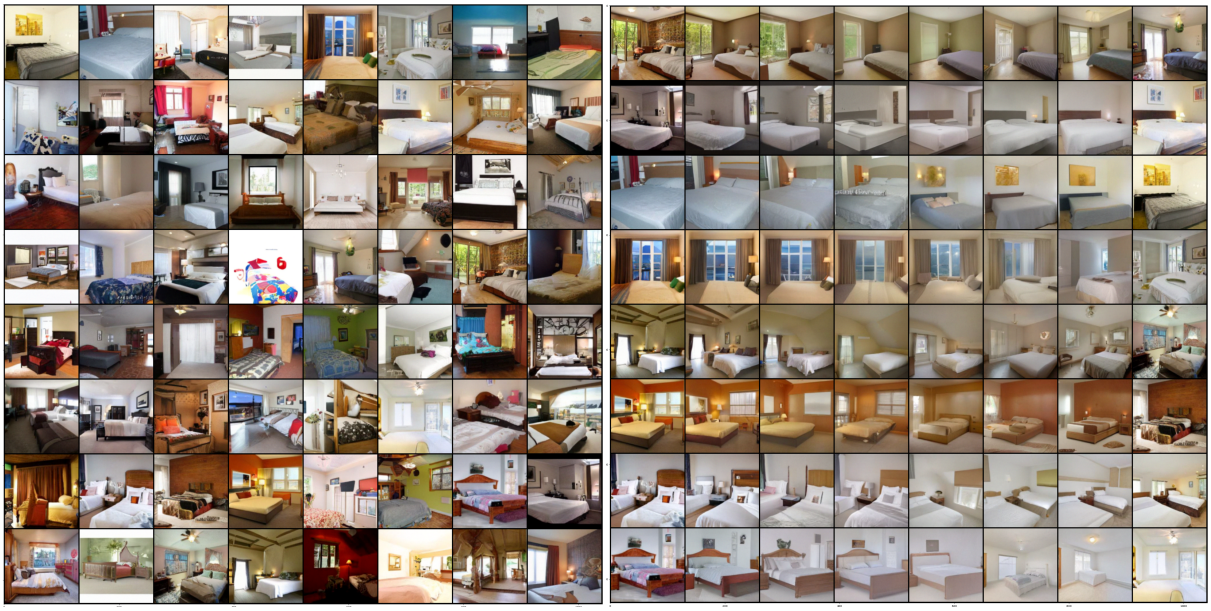
$$\theta_{ema} \leftarrow \lambda \cdot \theta_{ema} + (1 - \lambda) \cdot \theta$$

Figure 1: Results from the model with non-EMA sampling (left) and EMA sampling (right).

3 RESULTS

Taking a sample of 64 images (Fig. 2a) shows the model consistently generating realistic images of bedrooms in the 128×128 resolution. The samples have a lot of variety, with different lighting and orientation of the bedrooms. However, some images do not resemble full bedrooms due to too much white or are missing details that make them look unrealistic. There is a large diversity amongst the sample, with the very little similarity between images. The scenes in the images vary, as well as the lighting.

By using pairs of images from this sample, the images were partially noised and then linearly interpolated. The model sampled from these noisy images to give the results in Fig. 2b. The model creates realistic images, that fit the expected gradual change between the pairs. However, in some places, the model creates similar, but seemingly random images rather than interpolations.



(a) Random sample of 64 images.

(b) Interpolations between 8 pairs of images from the sample.

Figure 2: Samples from the model.

Fig. 3 shows some of the cherry-picked samples that the model generated.

Figure 3: Some of the best selected samples from the model.

The uniqueness of the samples is shown in Fig. 4, with the 5 nearest neighbours of the images using the LPIPS metric (Zhang et al., 2018). Only a portion of the dataset was compared to, due to the computation time required for comparison between each image.

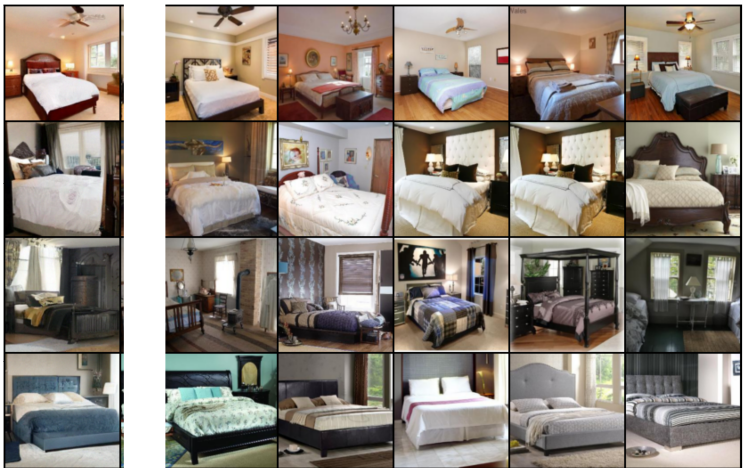


Figure 4: Samples on the left, compared to their 5 nearest neighbours in a portion of the LSUN bedrooms dataset.

4 LIMITATIONS

Due to the nature of the assignment, training time was limited, as well as computational limitations. In the future to improve image realism, longer training with larger GPU memory would benefit the quality of samples as the model can iterate over the dataset many more times. On the other hand, a smaller portion of the dataset could have been used rather than the entirety. Whilst this may reduce the diversity in sampled images, it would allow for the model to iterate over the training set many more times.

Many of the samples from the model resemble bedrooms, however, not all of them look realistic. This could be due to the down-scaling of the images in the train set, as detail is lost, or the lack of training time. The interpolations from the model look like real images across a gradient, however, in one case, the interpolation looks faded and blurry. An improvement to help solve this problem would be to manually configure what time-step to use for each pair of images when adding noise to them, as too low of a time-step creates what look like linear alpha blends, whereas too high of a time-step creates images that do not fit the interpolation.

5 BONUSES

Training was carried out on LSUN at the 128×128 resolution (+8 marks).

REFERENCES

Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021. URL https://arxiv.org/abs/2105.05233.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013. URL https://arxiv.org/abs/1312.6114.

Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models, 2021. URL https://arxiv.org/abs/2102.09672.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi (eds.), *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pp. 234–241, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24574-4.

Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018. URL https://arxiv.org/abs/1801.03924.