Generating Winged Horses With Score Based Models

Anonymous author

Abstract

Score based generative models have recently been able to exceed the state of the art for image generation benchmarks such as CIFAR-10. Unlike adversarial methods, these models are more stable to train and do not suffer from mode collapse. Recently these models have also been shown to interpolate between different data points in the distribution. This paper proposes sampling from a score based generative model to generate images of a Pegasus.

1 Methodology

A noise conditional score network (NCSN) [4] can be used to estimate the score of the data $\nabla_{\boldsymbol{x}} \log(p(\boldsymbol{x}))$ from its probability density $p(\boldsymbol{x})$. There are two possible ways to model the score: denoising score matching (equation 1) or sliced score matching. Both of these methods are able to achieve similar results however, dsm can be estimated four times faster than ssm and so we opted to use dsm to decrease the training time for our network. To estimate the dsm, we sample some random noise from a distribution $q(\tilde{\boldsymbol{x}}|\boldsymbol{x})$ and multiply this by a set of noise scales $\{\sigma\}_{i=0}^{L}$. We then add this noise to our input. The network then predicts the score for the input and we compute the mean squared error between the score and its target $\frac{\tilde{\boldsymbol{x}} - \boldsymbol{x}}{\sigma^2}$ where $\tilde{\boldsymbol{x}}$ are the perturbed inputs and $\boldsymbol{x}$ are the original inputs. The full loss function can be seen in equation 2. We decided to set $L = 232$ and use a geometric series for our σ values with $\sigma_1 = 50$. These values are based on suggestions from [5] which show that these are the optimal values for CIFAR-10. When trained, $s_{\boldsymbol{\theta}}(\boldsymbol{x}, \sigma_i) = \nabla_{\boldsymbol{x}} \log p_{\sigma_i}$.

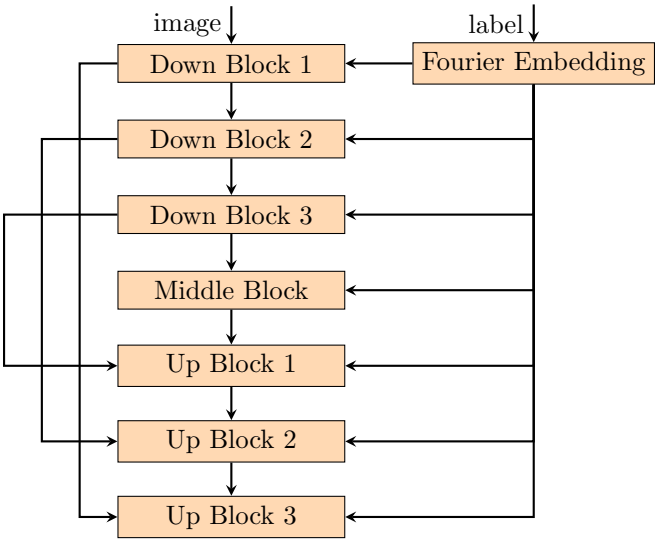
$$\mathcal{L}_{\text{DSM}} = \mathbb{E}_{q_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x})p_{data}(\boldsymbol{x})}[\, \| \boldsymbol{s}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}) - \nabla \tilde{\boldsymbol{x}} \log q_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) \|_2^2] \tag{1}$$

$$\mathcal{L}(\boldsymbol{\theta}; \sigma) = \frac{1}{2} \mathbb{E}_{p_{data}(\boldsymbol{x})} \mathbb{E}_{\tilde{x} \sim \mathcal{N}(\boldsymbol{x}, ^2)}[\, \| \boldsymbol{s}_\theta(\tilde{\boldsymbol{x}}, \sigma) + \frac{\tilde{\boldsymbol{x}} - \boldsymbol{x}}{\sigma^2} \|_2^2] \tag{2}$$

Once the network is trained, we can sample from it using Langevin Dynamics. This technique uses a fixed step size ϵ and an initial random value $\tilde{\boldsymbol{x}_0}$ to perform the sampling for T time steps. At each time step, we update perform half a step in the direction of the score of our sample and add a noise value $\sqrt{\epsilon}\boldsymbol{z}_t$ (equation 3). For our sampling, we use a variant called annealed Langevin Dynamics [4], which performs T updates for each noise scale in $\{\sigma\}_{i=0}^{L}$ where the final sample generated from the previous noise scale is used as the initial sample for the next noise scale. We also use a different step size ϵ for each noise scale σ_i where $\epsilon = \frac{\sigma_i^2}{\sigma_L^2}$. For this method, we set T=5 for each noise scale as in [5].

$$\tilde{\boldsymbol{x}}_t = \tilde{\boldsymbol{x}}_{t-1} + \frac{\epsilon}{2} \nabla_{\boldsymbol{x}} \log p(\tilde{\boldsymbol{x}}_{t-1}) + \sqrt{\epsilon} \boldsymbol{z}_t \tag{3}$$

1.1 Architecture



The model architecture is based on a U-NET inspired by the DDPM model from [2] as this architecture has been shown to also perform well for score matching networks [3].Instead of using the standard self-attention layers, We used a linear approximation of this layer for our model. In addition, we implement the noise conditioning technique in [5] where we divide the output by the noise scale. The U-NET has three down-scaling Resnet block layers each followed by an average pooling layer and linear self-attention. This is then followed by a middle block comprised of a resnet block, an attention layer and a second resnet block. Finally, there are three up-scaling layers consisting of a nearest neighbour interpolation followed by a Resnet and linear attention. We replaced the timestep embedding used in this architecture with the Fourier embedding used in [6]. This embedding takes as input a vector of randomly chosen noise scales for each image in the batch. Each Resnet block takes this embedding as an additional input. In addition, each block uses the InstanceNorm++ normalisations and ELU activations from [5] rather than the GroupNorm and Swish activations from [2] as these were more efficient for training but were able to achieve similar quality output images. We also decided to apply ReZero [1] to the Linear Attention layers to speed up model convergence.

1.2 Training

We used an Adam optimiser with standard beta values and a learning rate of 0.0001 as we found that larger learning rates caused greater fluctuations in the loss estimate (equation 2). The model was trained on the CIFAR-10 and STL-10 datasets using a batch size of 128 and 64 respectively. We chose to only train on the horse and bird classes from these datasets. Images were randomly horizontally flipped for both datasets and STL-10 was resized to 48 x 48 pixels to speed up training time. Additionally, the STL-10 training and test sets were combined to provide the network with more examples. We trained a model for 5000 epochs on CIFAR-10, STL-10 resized to 48 x 48 pixels and STL-10 resized to 64 x 64 pixels. In addition, we attempted to train a model on the full 96 x 96 STL-10 images however, this model was very slow to converge and the results were very blurry after 30 hours of training.

1.3 Interpolation

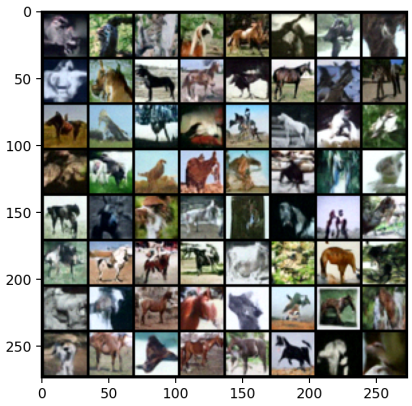
A method for interpolation between samples is presented in [5]. For each noise level σ in our geometric series $\{\sigma_i\}_{i=1}^{L}$, we run annealed Langevin Dynamics for T steps to produce $L \times T$ instances of Gaussian noise. For two samples $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ which produce corresponding Gaussian noise $\{\boldsymbol{z}^(1)_{ij}\}$ and $\{\boldsymbol{z}^(2)_{ij}\}$, where i is the ith noise level and j is the jth time step of annealed Langevin dynamics, we can use equation 4 to produce interpolated sample k where $1 \leq k \leq N$ from N interpolations. We can then set N and k to sample with a

ratio of $\frac{k}{N}$ from $\boldsymbol{x}_1$ and $\frac{N-k}{N}$ from $\boldsymbol{x}_2$. For our results, we used $k = 9$, $N = 15$ for the CIFAR-10 batch and $k = 11$, $N = 15$ for both STL-10 batches.

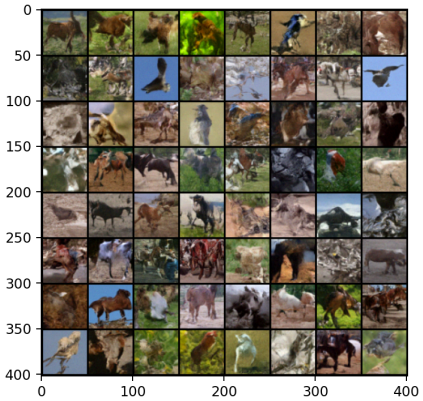
$$\{\cos\frac{k\pi}{2(N+1)}\boldsymbol{z}^{(1)}_{ij} + \sin\frac{k\pi}{2(N+1)}\boldsymbol{z}^{(2)}_{ij}\}_{1\leq i\leq L, 1\leq j\leq T} \tag{4}$$

2 Results

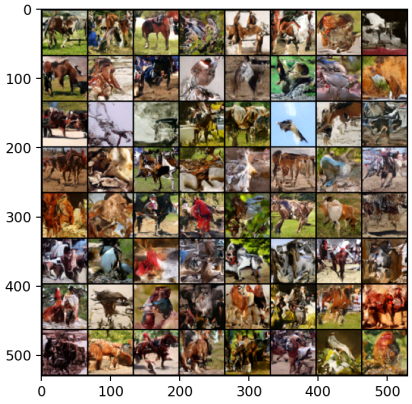
A batch of winged horses generated from CIFAR-10:



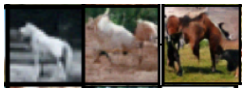
A batch of winged horses generated from STL-10 resized to 48 x 48 pixels:



A batch of winged horses generated from STL-10 resized to 64 x 64 pixels:

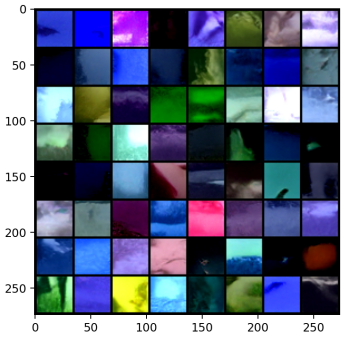


The best pegasus images from each batch (left to right CIFAR-10, STL-10 48x48, STL-10 64x64) were:



3 Limitations

Using annealed Langevin Dynamics, we initialise our samples randomly. As a result, when we interpolate between two samples, we sometimes will interpolate between two horses or two birds rather than between a horse and a bird which will not result in a Pegasus. [6] describes a method for class conditional sampling of score based models which could potentially be combined with the interpolation technique to solve this problem. Another limitation of this technique is the sampling time. Other methods such as GANs and VAEs are able to generate images very quickly however, NCSNs need to use annealing Langevin Dynamics each time a batch of images is sampled, which requires over 1000 total steps across all noise scales to generate good quality images. The model also experiences colour shift when trained for long periods of time which became a problem when attempting to train on images for long periods of time and can be seen in the following batch of images sampled from STL-10 at full size:



Bonuses

This submission has a total bonus of +2 marks as it is trained on STL-10 resized to 48x48 pixels and STL-10 resized to 64 x 64 pixels.

References

[1] Thomas Bachlechner et al. "ReZero is All You Need: Fast Convergence at Large Depth". In: *arXiv e-prints*, arXiv:2003.04887 (Mar. 2020), arXiv:2003.04887. arXiv: 2003.04887 [cs.LG].

[2] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising Diffusion Probabilistic Models". In: *arXiv e-prints*, arXiv:2006.11239 (June 2020), arXiv:2006.11239. arXiv: 2006.11239 [cs.LG].

[3] Alexia Jolicoeur-Martineau et al. "Adversarial score matching and improved sampling for image generation". In: *arXiv e-prints*, arXiv:2009.05475 (Sept. 2020), arXiv:2009.05475. arXiv: 2009.05475 [cs.LG].

[4] Yang Song and Stefano Ermon. "Generative Modeling by Estimating Gradients of the Data Distribution". In: *arXiv e-prints*, arXiv:1907.05600 (July 2019), arXiv:1907.05600. arXiv: 1907.05600 [cs.LG].

[5] Yang Song and Stefano Ermon. "Improved Techniques for Training Score-Based Generative Models". In: *arXiv e-prints*, arXiv:2006.09011 (June 2020), arXiv:2006.09011. arXiv: 2006.09011 [cs.LG].

[6] Yang Song et al. "Score-Based Generative Modeling through Stochastic Differential Equations". In: *arXiv e-prints*, arXiv:2011.13456 (Nov. 2020), arXiv:2011.13456. arXiv: 2011.13456 [cs.LG].